

List Scheduling in Order of α -Points on a Single Machine

Martin Skutella

Fachbereich Mathematik, Universität Dortmund, D-44221 Dortmund, Germany
martin.skutella@uni-dortmund.de
<http://www.mathematik.uni-dortmund.de/~skutella/>

Abstract. Many approximation results for single machine scheduling problems rely on the conversion of preemptive schedules into (preemptive or non-preemptive) solutions. The initial preemptive schedule is usually an optimal solution to a combinatorial relaxation or a linear programming relaxation of the scheduling problem under consideration. It therefore provides a lower bound on the optimal objective function value. However, it also contains structural information which is useful for the construction of provably good feasible schedules. In this context, list scheduling in order of so-called α -points has evolved as an important and successful tool. We give a survey and a uniform presentation of several approximation results for single machine scheduling with total weighted completion time objective from the last years which rely on the concept of α -points.

1 Introduction

We consider the following single machine scheduling problem. There is a set of n jobs $J = \{1, \dots, n\}$ that must be processed on a single machines. Each job j has a non-negative integral *processing time* p_j , that is, it must be processed during p_j time units. The machine can process at most one job at a time. Each job j has a non-negative integral *release date* r_j before which it cannot be processed. In *preemptive* schedules, a job may repeatedly be interrupted and continued later. In *non-preemptive* schedules, a job must be processed in an uninterrupted fashion. There may be *precedence constraints* between jobs. If $j \prec k$ for $j, k \in J$, it is required that j is completed before k can start. We denote the completion time of job j by C_j and seek to minimize the *total weighted completion time*: A non-negative *weight* w_j is associated with each job j and the goal is to minimize $\sum_{j \in J} w_j C_j$. In order to keep the presentation simple, we will almost always assume that the processing times p_j are positive integers and that $w_j > 0$, for all jobs $j \in J$. However, all results can be carried over to the case with zero processing times and weights.

In scheduling, it is quite convenient to refer to the respective problems using the standard classification scheme of Graham, Lawler, Lenstra, and Rinnooy Kan [20]. The problems that we consider are special variants of the single machine scheduling problem $1|r_j, prec, (pmtn) | \sum w_j C_j$. The entry “1” in the first field indicates that the scheduling environment provides only one machine. The second field can be empty or contains some of the job characteristics $r_j, prec,$ and $pmtn$, indicating whether there

are nontrivial release dates or precedence constraints, and whether preemption is allowed. We put an item in brackets to indicate that we consider both variants of the problem, with and without the corresponding feature. The third field refers to the objective function. We are interested in minimizing the total weighted completion time $\sum w_j C_j$ or, for the special case of unit weights, the *total completion time* $\sum C_j$.

Scheduling with the total weighted completion time objective has recently achieved a great deal of attention, partly because of its importance as a fundamental problem in scheduling, and also because of new applications, for instance, in compiler optimization [9] and in parallel computing [6]. In the last years, there has been significant progress in the design of approximation algorithms for various special cases of the general single machine problems $1|r_j, prec(, pmtn) | \sum w_j C_j$; see, e.g., [29, 21, 16, 11, 35, 17, 18, 36, 10, 1, 3].

Recall that a ρ -*approximation algorithm* is a polynomial-time algorithm guaranteed to deliver a solution of cost at most ρ times the optimal value; the value ρ is called *performance guarantee* or *performance ratio* of the algorithm. A *randomized* ρ -approximation algorithm is a polynomial-time algorithm that produces a feasible solution whose *expected* objective function value is within a factor of ρ of the optimal value.

For problems without precedence constraints, we also consider the corresponding *on-line setting* where jobs arrive over time and the number of jobs is unknown in advance. Each job $j \in J$ becomes available at its release date, which is not known in advance; at time r_j , we learn both its processing time p_j and its weight w_j . Even in the on-line setting, the value of the computed schedule is compared to the optimal (off-line) schedule. The derived bounds are called *competitive ratios*. While all randomized approximation algorithms discussed in this paper can be derandomized in the off-line setting without loss in the performance guarantee, there is a significant difference in the on-line setting. It is well-known that randomized on-line algorithms often yield better competitive ratios than any deterministic on-line algorithm can achieve (see, e.g., the positive and negative results on the problem $1|r_j | \sum C_j$ in Table 1).

The conversion of preemptive schedules to nonpreemptive schedules in order to get approximation algorithms was introduced by Phillips, Stein and Wein [29] and was subsequently also used in [7, 11], among others. Phillips et al. [29], and Hall, Shmoys, and Wein [22] introduced the idea of list scheduling in order of α -points to convert preemptive schedules to non-preemptive ones. Even earlier, de Sousa [46] used this idea heuristically to turn ‘preemptive’ solutions to a time-indexed linear programming relaxation into feasible nonpreemptive schedules. For $\alpha \in (0, 1]$, the α -point of a job with respect to a preemptive schedule is the first point in time when an α -fraction of the job has been completed. Independently from each other, Goemans [16] and Chekuri, Motwani, Natarajan, and Stein [11] have taken up this idea, and showed that choosing α randomly leads to better results. Later, α -points with individual values of α for different jobs have been used by Goemans, Queyranne, Schulz, Skutella, and Wang [17].

Table 1 summarizes approximation and on-line results based on the concept of α -points which were obtained for the single machine scheduling problems under consideration. These results are compared with the best known approximation and on-line results and also with corresponding hardness results. In the absence of precedence

Table 1. Summary of approximation results and non-approximability results. In all cases, $\varepsilon > 0$ can be chosen arbitrarily small. The bounds marked with one and two stars refer to randomized and deterministic on-line algorithms, respectively. The second column contains the results discussed in this paper while the third column gives the best currently known results. The last column contains the best known lower bounds, that is, results on the hardness of approximation.

Approximation and hardness results for single machine scheduling problems			
problem	α -points	best known	lower bounds
$1 r_j \sum C_j$	1.5820* [11]	$1 + \varepsilon$ [1] 2** [29]	1.5820* [48, 50] 2** [23]
$1 r_j \sum w_j C_j$	1.6853* [17] 2.4143** [16]	$1 + \varepsilon$ [1] 2** [3]	1.5820* [48, 50] 2** [23]
$1 r_j, pmtn \sum w_j C_j$	$\frac{4}{3}$ * [36]	$1 + \varepsilon$ [1] 2** [36]	1.038* [14] 1.073** [14]
$1 prec \sum w_j C_j$	$2 + \varepsilon$ [35]	2 [21]	?
$1 r_j, prec \sum w_j C_j$	$e + \varepsilon$ [35]		?
$1 r_j, prec, pmtn \sum w_j C_j$	$2 + \varepsilon$ [35]	2 [21]	?

constraints, polynomial-time approximation schemes have recently been obtained by Afrati et al. [1]; however, list scheduling in order of α -points still leads to the best known randomized on-line algorithms for these problems. For problems with precedence constraints, the concept of α -points either yields the best known approximation result (for the problem $1|r_j, prec|\sum w_j C_j$) or only slightly worse results. It is one of the most interesting open problems in the area of machine scheduling to obtain non-approximability results for these problems [38, 51].

We mention further work in the context of α -points: Schulz and Skutella [37] apply the idea of list scheduling in order of α -points to scheduling problems on identical parallel machines, based on a single machine relaxation and random assignments of jobs to machines. Savelsbergh, Uma, and Wein [32] give an experimental study of approximation algorithms for the problem $1|r_j|\sum w_j C_j$. In particular, they analyze list scheduling in order of α -points for one single α and for individual values of α for different jobs. They also test the approximation algorithms within a branch and bound scheme and apply it to real world scheduling instances arising at BASF AG in Ludwigshafen. Uma and Wein [49] extend this evaluation and study the relationship between several linear programming based lower bounds and combinatorial lower bounds for the problems $1|r_j|\sum w_j C_j$. The heuristic use of α -points for resource constrained project scheduling problems was also empirically analyzed by Cavalcante, de Souza, Savelsbergh, Wang, and Wolsey [5] and by Möhring, Schulz, Stork, and Uetz [26].

In this paper, we give a survey and a uniform presentation of the approximation results listed in the second column of Table 1. We start with a description and analysis of simple list scheduling heuristics for non-preemptive and preemptive single machine

scheduling in Section 2. The concept of α -points is introduced in Section 3. In Section 4 we review the result of Chekuri et al. for the problem $1|r_j|\sum C_j$. Time-indexed LP relaxations for general constrained single machine scheduling together with results on their quality are discussed in Section 5. In Section 6 we review related results of Schulz [34] and Hall, Schulz, Shmoys, and Wein [21] on list scheduling in order of LP completion times. The approximation results of Goemans et al. [17] and Schulz and Skutella [36] for single machine scheduling with release dates are discussed in Section 7. The results of Schulz and Skutella [35] for the problem with precedence constraints are presented in Section 8. Finally, the application of these results to the on-line setting is discussed in Section 9.

2 Non-preemptive and Preemptive List Scheduling

In this section we introduce and analyze non-preemptive and preemptive list scheduling on a single machine for a given set of jobs with release dates and precedence constraints. We consider both the non-preemptive and the preemptive variant of the problem and argue that the class of list schedules always contains an optimal schedule. Finally we discuss simple approximations based on list scheduling. Many results presented in this section belong to the folklore in the field of single machine scheduling.

Consider a list representing a total order on the set of jobs which extends the partial order given by the precedence constraints. A straightforward way to construct a feasible non-preemptive schedule is to process the jobs in the given order as early as possible with respect to release dates. This routine is called LIST SCHEDULING and a schedule constructed in this way is a (non-preemptive) *list schedule*.

The first result for LIST SCHEDULING in this context was achieved by Smith [45] for the case without release dates or precedence constraints and is known as *Smith's ratio rule*:

Theorem 1. LIST SCHEDULING in order of non-decreasing ratios p_j/w_j gives an optimal solution for $1|\sum w_j C_j$ in $O(n \log n)$ time.

Proof. Since all release dates are zero, we can restrict to schedules without idle time. Consider a schedule S with two successive jobs j and k and $p_j/w_j > p_k/w_k$. Exchanging j and k in S leads to a new schedule S' whose value differs from the value of S by $w_j p_k - w_k p_j < 0$. Thus, S is not optimal and the result follows. The running time of LIST SCHEDULING in order of non-decreasing ratios p_j/w_j is dominated by the time needed to sort the jobs, and is therefore $O(n \log n)$. \square

For the special case of unit weights ($w_j \equiv 1$), Smith's ratio rule is sometimes also referred to as the *shortest processing time rule (SPT-rule)*. We always assume that jobs are numbered such that $p_1/w_1 \leq \dots \leq p_n/w_n$; moreover, whenever we talk about LIST SCHEDULING in order of non-decreasing ratios p_j/w_j , we refer to this sequence of jobs.

Depending on the given list and the release dates of jobs, one may have to introduce idle time when one job is completed but the next job in the list is not yet released. On the other hand, if preemptions are allowed, it does not make sense to leave the machine

idle while another job at a later position in the list is already available (released) and waiting. We would better start this job and preempt it from the machine as soon as the next job in the list is released. In **PREEMPTIVE LIST SCHEDULING** we process at any point in time the first available job in the list. The resulting schedule is called *preemptive list schedule*. Special applications of this routine have been considered, e. g., by Hall et al. [21] and by Goemans [15, 16]. 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.

Since we can assume without loss of generality that $j \prec k$ implies $r_j \leq r_k$, and since the given list is a linear extension of the precedence order, the preemptive list schedule respects precedence constraints and is therefore feasible. The following result on the running time of **PREEMPTIVE LIST SCHEDULING** has been observed by Goemans [16].

Lemma 1. *Given a list of n jobs, the corresponding list schedule can be created in linear time while **PREEMPTIVE LIST SCHEDULING** can be implemented to run in $O(n \log n)$ time.*

Proof. The first part of the lemma is clear. In order to construct a preemptive list schedule in $O(n \log n)$ time we use a priority queue that always contains the currently available jobs which have not been finished yet; the key of a job is equal to its position in the given list. At each point in time we process the top element of the priority queue or leave the machine idle if the priority queue is empty. There are two types of events that cause an update of the priority queue: Whenever a job is completed on the machine, we remove it from the priority queue; when a job is released, we add it to the priority queue. This results in a total of $O(n)$ priority queue operations each of which can be implemented to run in $O(\log n)$ time. \square

As a consequence of the following lemma one can restrict to (preemptive) list schedules.

Lemma 2. *Given a feasible non-preemptive (preemptive) schedule, (**PREEMPTIVE**) **LIST SCHEDULING** in order of non-decreasing completion times does not increase completion times of jobs.*

Proof. The statement for non-preemptive schedules is easy to see. **LIST SCHEDULING** in order of non-decreasing completion times coincides with shifting the jobs in the given non-preemptive schedule one after another in order of non-decreasing completion times as far as possible to the left (backwards in time).

Let us turn to the preemptive case. 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 $t \geq 0$ 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 $C_k^P \leq C_j^P$ are processed. We denote the set of these jobs by K . By the definition of t , we know that $r_k \geq t$, for all $k \in K$. Hence, $C_j^P \geq t + \sum_{k \in K} p_k$. On the other hand, the definition of K implies $C_j = t + \sum_{k \in K} p_k$ and therefore $C_j \leq C_j^P$. \square

In PREEMPTIVE LIST SCHEDULING a job is only preempted if another job is released at that time. In particular, there are at most $n - 1$ preemptions in a preemptive list schedule. Moreover, since all release dates are integral, preemptions only occur at integral points in time. Therefore we can restrict to schedules meeting this property. Nevertheless, we will sometimes consider schedules where preemptions can occur at arbitrary points in time, but we always keep in mind that those scheduled can be converted by PREEMPTIVE LIST SCHEDULING without increasing their value.

2.1 Simple Bounds and Approximations

One of the most important techniques for approximating NP-hard machine scheduling problems with no preemptions allowed is the conversion of preemptive schedules to non-preemptive ones. The first result in this direction has been given by Phillips et al. [29]. It is a 2-approximation algorithm for the problem $1|r_j|\sum C_j$.

Lemma 3. Consider LIST SCHEDULING according to a list j_1, j_2, \dots, j_n . Then the completion time of job j_i , $1 \leq i \leq n$, is bounded from above by

$$\max_{k \leq i} r_{j_k} + \sum_{k \leq i} p_{j_k}.$$

Proof. For fixed i , modify the given instance by increasing the release date of job j_1 to $\max_{k \leq i} r_{j_k}$. The completion time of j_i in the list schedule corresponding to this modified instance is equal to $\max_{k \leq i} r_{j_k} + \sum_{k \leq i} p_{j_k}$. \square

It is well known that the preemptive problem $1|r_j, pmtn|\sum C_j$ can be solved in polynomial time by the *shortest remaining processing time rule (SRPT-rule)* [4]: Schedule at any point in time the job with the shortest remaining processing time. The value of this optimal preemptive schedule is a lower bound on the value of an optimal non-preemptive schedule. Together with the following conversion result this yields the 2-approximation algorithm of Phillips et al. [29] for $1|r_j|\sum C_j$.

Theorem 2. Given an arbitrary feasible preemptive schedule P , LIST SCHEDULING in order of non-decreasing completion times yields a non-preemptive schedule where the completion time of each job is at most twice its completion time in P .

Proof. The proof follows directly from Lemma 3 since both $\max_{k \leq i} r_{j_k}$ and $\sum_{k \leq i} p_{j_k}$ are lower bounds on C_j^P (we use the notation of Lemma 3). \square

An instance showing that the job-by-job bound given in Theorem 2 is tight can be found in [29].

2.2 A Generalization of Smith's Ratio Rule to $1|r_j, pmtn|\sum w_j C_j$

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 non-decreasing ratios p_j/w_j . Schulz and Skutella [36] give the following lower bound on the performance of this simple heuristic.

Lemma 4. *The performance guarantee of PREEMPTIVE LIST SCHEDULING in order of non-decreasing ratios p_j/w_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 $1 \leq j \leq n$. Preemptive list scheduling in order of non-decreasing ratios of p_j/w_j preempts job j at time r_{j-1} , for $j = 2, \dots, n$, and finishes it only after all other jobs $j - 1, \dots, 1$ have been completed. The value of this schedule is therefore $n^4 - \frac{1}{2}n^3 + \frac{1}{2}n$. The SRPT-rule, which solves instances of $1|r_j, pmtn | \sum C_j$ optimally, sequences the jobs in order $n, \dots, 1$. 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. \square

On the other hand, one can give a matching upper bound of 2 on the performance of this algorithm. As pointed out in [36], the following observation is due to Goemans, Wein, and Williamson.

Lemma 5. *PREEMPTIVE LIST SCHEDULING in order of non-decreasing ratios p_j/w_j is a 2-approximation for $1|r_j, pmtn | \sum w_j C_j$.*

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

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

This concludes the proof. \square

In spite of the negative result in Lemma 4, Schulz and Skutella [36] present an algorithm that converts the preemptive list schedule (in order of non-decreasing ratios p_j/w_j) into another preemptive schedule and achieves performance guarantee $4/3$ for $1|r_j, pmtn | \sum w_j C_j$; see Section 7. The analysis is based on the observation of Goemans [15] that the preemptive list schedule represents an optimal solution to an appropriate LP relaxation.

3 The Concept of α -Points

In this section we discuss a more sophisticated technique that converts feasible preemptive schedules into non-preemptive ones and generalizes the routine given in Theorem 2. The bounds discussed in this section are at the bottom of the approximation results presented below. Almost all important structural insights are discussed here. We are given a single machine and a set of jobs with release dates and precedence constraints. Besides, we consider a fixed feasible preemptive schedule P and the completion time of job j in this schedule is denoted by C_j^P .

For $0 < \alpha \leq 1$ the α -point $C_j^P(\alpha)$ of job j with respect to P is the first point in time when an α -fraction of job j has been completed, i. e., when j has been processed on the machine for αp_j time units. In particular, $C_j^P(1) = C_j^P$ and for $\alpha = 0$ we define $C_j^P(0)$ to be the starting time of job j .

The average over all points in time at which job j is being processed on the machine is called the *mean busy time* of j . In other words, the mean busy time is the average over all α -points of j , i. e., $\int_0^1 C_j^P(\alpha) d\alpha$. This notion has been introduced by Goemans [16]. If j is being continuously processed between $C_j^P - p_j$ and C_j^P , its mean busy time is equal to $C_j^P - \frac{1}{2}p_j$. Otherwise, it is bounded from above by $C_j^P - \frac{1}{2}p_j$.

We will also use the following notation: For a fixed job j and $0 \leq \alpha \leq 1$ we denote the fraction of job k that is completed by time $C_j^P(\alpha)$ by $\eta_k(\alpha)$; in particular, $\eta_j(\alpha) = \alpha$. The amount of idle time that occurs before time $C_j^P(\alpha)$ on the machine is denoted by $t_{\text{idle}}(\alpha)$. The α -point of j can be expressed in terms of $t_{\text{idle}}(\alpha)$ and $\eta_k(\alpha)$:

Lemma 6. *For a fixed job j and $0 \leq \alpha \leq 1$ the α -point $C_j^P(\alpha)$ of job j can be written as*

$$C_j^P(\alpha) = t_{\text{idle}}(\alpha) + \sum_{k \in J} \eta_k(\alpha) p_k \geq \sum_{k \in J} \eta_k(\alpha) p_k ,$$

where $\eta_k(\alpha)$ denotes the fraction of job k that is completed by time $C_j^P(\alpha)$.

3.1 List Scheduling in Order of α -Points

In Theorem 2 we have analyzed LIST SCHEDULING in order of non-decreasing completion times, i. e., in order of non-decreasing 1-points. Phillips et al. [29], and Hall et al. [22] introduced the idea of LIST SCHEDULING in order of non-decreasing α -points $C_j^P(\alpha)$ for some $0 \leq \alpha \leq 1$. This is a more general way of capturing the structure of the given preemptive schedule; it can even be refined by choosing α randomly. We call the resulting schedule α -schedule and denote the completion time of job j in this schedule by C_j^α .

Goemans et al. [17] have further extended this idea to individual, i. e., job-dependent α_j -points $C_j^P(\alpha_j)$, for $j \in J$ and $0 \leq \alpha_j \leq 1$. We denote the vector consisting of all α_j 's by $\alpha = (\alpha_1, \dots, \alpha_n)$. Then, the α_j -point $C_j^P(\alpha_j)$ is also called α -point of j and the α -schedule is constructed by LIST SCHEDULING in order of non-decreasing α -points; the completion time of job j in the α -schedule is denoted by C_j^α .

For the feasible preemptive schedule P , the sequence of the jobs in order of non-decreasing α -points respects precedence constraints since $j \prec k$ implies $C_j^P(\alpha_j) \leq C_k^P \leq C_k^P(0) \leq C_k^P(\alpha_k)$. Therefore the corresponding α -schedule is feasible.

To analyze the completion times of jobs in an α -schedule, we also consider schedules that are constructed by a slightly different conversion routine which is called α -CONVERSION [17]. A similar procedure is implicitly contained in [11, proof of Lemma 2.2].

α -CONVERSION:

Consider the jobs $j \in J$ in order of non-increasing α -points $C_j^P(\alpha_j)$ and iteratively change the preemptive schedule P to a non-preemptive schedule by applying the following steps:

- i) remove the $\alpha_j p_j$ units of job j that are processed before $C_j^P(\alpha_j)$ and leave the machine idle during the corresponding time intervals; we say that this idle time *is caused by* job j ;
- ii) delay the whole processing that is done later than $C_j^P(\alpha_j)$ by p_j ;
- iii) remove the remaining $(1 - \alpha_j)$ -fraction of job j from the machine and shrink the corresponding time intervals; *shrinking* a time interval means to discard the interval and move earlier, by the corresponding amount, any processing that occurs later;
- iv) process job j in the released time interval $(C_j^P(\alpha_j), C_j^P(\alpha_j) + p_j]$.

Figure 1 contains an example illustrating the action of α -CONVERSION. Observe that in the resulting schedule jobs are processed in non-decreasing order of α -points and no job j is started before time $C_j^P(\alpha_j) \geq r_j$. The latter property will be useful in the analysis of on-line α -schedules in Section 9.

Lemma 7. *The completion time of job j in the schedule computed by α -CONVERSION is equal to*

$$C_j^P(\alpha_j) + \sum_{\substack{k \\ \eta_k(\alpha_j) \geq \alpha_k}} (1 + \alpha_k - \eta_k(\alpha_j)) p_k .$$

Proof. Consider the schedule constructed by α -CONVERSION. The completion time of job j is equal to the idle time before its start plus the sum of processing times of jobs that start no later than j . Since the jobs are processed in non-decreasing order of their α -points, the amount of processing before the completion of job j is





$$\sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} p_k . \tag{1}$$

The idle time before the start of job j can be written as the sum of the idle time $t_{\text{idle}}(\alpha_j)$ that already existed in the preemptive schedule P before $C_j^P(\alpha_j)$ plus the idle time before the start of job j that is caused in steps i) of α -CONVERSION; notice that steps iii) do not create any additional idle time since we shrink the affected time intervals. Each job k that is started no later than j , i. e., such that $\eta_k(\alpha_j) \geq \alpha_k$, contributes $\alpha_k p_k$ units of idle time, all other jobs k only contribute $\eta_k(\alpha_j) p_k$ units of idle time. As a result, the total idle time before the start of job j can be written as

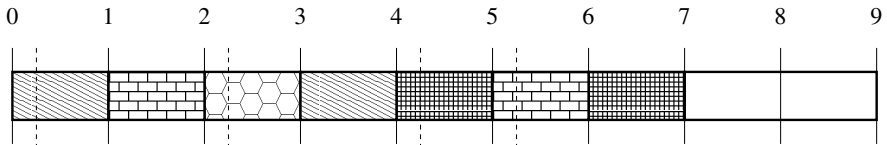
$$t_{\text{idle}}(\alpha_j) + \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} \alpha_k p_k + \sum_{\substack{k \\ \alpha_k > \eta_k(\alpha_j)}} \eta_k(\alpha_j) p_k . \tag{2}$$

The completion time of job j in the schedule constructed by α -CONVERSION is equal to the sum of the expressions in (1) and (2); the result then follows from Lemma 6. \square

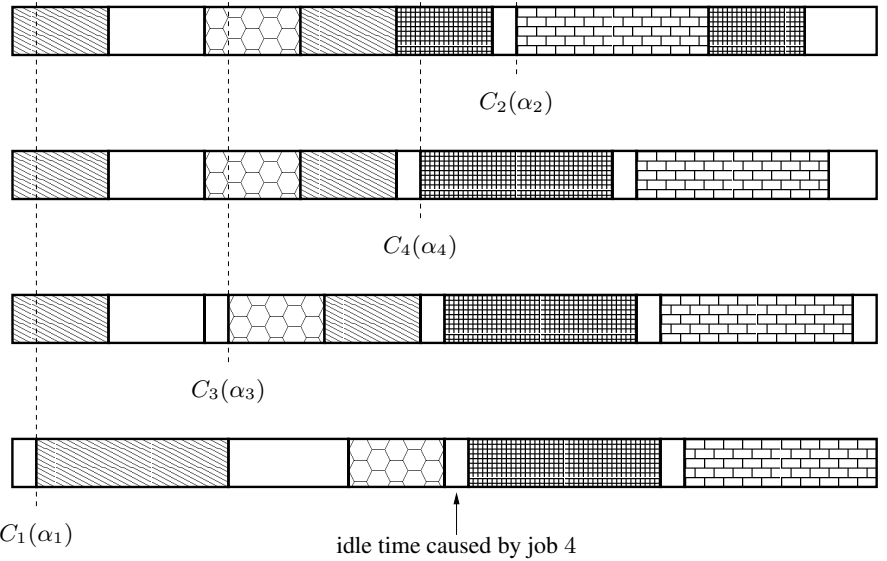
Instance:

job j	r_j	p_j	α_j
1 	0	2	$\frac{1}{8}$
2 	1	2	$\frac{5}{8}$
3 	2	1	$\frac{1}{4}$
4 	4	2	$\frac{1}{8}$

preemptive schedule P :



α -CONVERSION:



α -schedule:

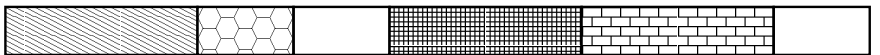


Fig. 1. The conversion of an arbitrary preemptive schedule P to a non-preemptive one by α -CONVERSION and by LIST SCHEDULING in order of non-decreasing α -points

It follows from Lemma 7 that the completion time C_j of each job j in the non-preemptive schedule constructed by α -CONVERSION satisfies $C_j \geq C_j^P(\alpha_j) + p_j \geq r_j + p_j$, hence is a feasible schedule. We obtain the following corollary.

Corollary 1. *The completion time of job j in an α -schedule can be bounded by*

$$C_j^\alpha \leq C_j^P(\alpha_j) + \sum_{\substack{k \\ \eta_k(\alpha_j) \geq \alpha_k}} (1 + \alpha_k - \eta_k(\alpha_j)) p_k .$$

Proof. Since the schedule constructed by α -CONVERSION processes the jobs in order of non-decreasing α -points, the result follows from Lemma 7 and Lemma 2. \square

3.2 Preemptive List Scheduling in Order of α -Points

Up to now we have considered non-preemptive LIST SCHEDULING in order of α -points. For a vector α we call the schedule that is constructed by PREEMPTIVE LIST SCHEDULING in order of non-decreasing α -points *preemptive α -schedule*. The completion time of job j in the preemptive α -schedule is denoted by $C_j^{\alpha-pmtn}$.

The reader might wonder why we want to convert a preemptive schedule into another preemptive schedule. Later we will interpret solutions to time-indexed LP relaxations as preemptive schedules. However, the value of these schedules can be arbitrarily bad compared to their LP value. The results derived in this section will help to turn them into provably good preemptive schedules.

Again, to analyze preemptive α -schedules we consider an alternative conversion routine which we call PREEMPTIVE α -CONVERSION and which is a modification of α -CONVERSION. The difference is that there is no need for causing idle time by removing the α_j -fraction of job j from the machine in step i). We rather process $\alpha_j p_j$ units of job j . Thus, we have to postpone the whole processing that is done later than $C_j^P(\alpha_j)$ only by $(1 - \alpha_j) p_j$, because this is the remaining processing time of job j , which is then scheduled in $[C_j^P(\alpha_j), C_j^P(\alpha_j) + (1 - \alpha_j) p_j]$.

This conversion technique has been introduced by Goemans, Wein, and Williamson [18] for a single α . Figure 2 contains an example illustrating the action of PREEMPTIVE α -CONVERSION. Observe that, as in the non-preemptive case, the order of completion times in the resulting schedule coincides with the order of α -points in P . Moreover, since the initial schedule P is feasible, the same holds for the resulting schedule by construction.

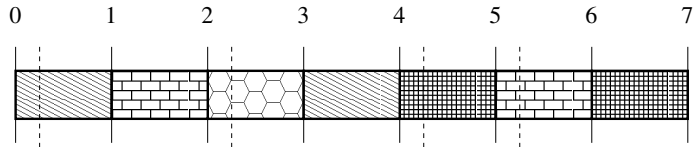
Lemma 8. *The completion time of job j in the preemptive α -schedule can be bounded by*

$$C_j^{\alpha-pmtn} \leq C_j^P(\alpha_j) + \sum_{\substack{k \\ \eta_k(\alpha_j) \geq \alpha_k}} (1 - \eta_k(\alpha_j)) p_k . \tag{3}$$

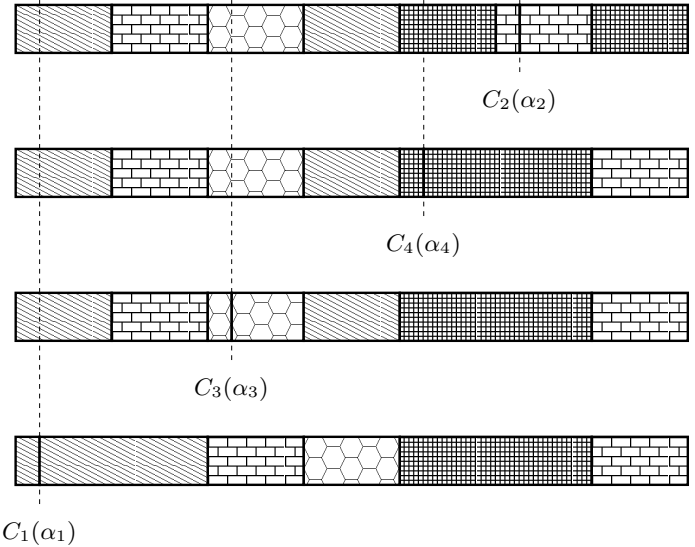
In the absence of nontrivial release dates, the same bound holds for the completion time C_j^α of job j in the non-preemptive α -schedule.

Proof. Using the same ideas as in the proof of Lemma 7 it can be seen that the right hand side of (3) is equal to the completion time of job j in the schedule constructed by PREEMPTIVE α -CONVERSION. The only difference to the non-preemptive setting is that no additional idle time is caused by the jobs. Since the order of completion times in the schedule constructed by PREEMPTIVE α -CONVERSION coincides with the order of α -points, the bound in (3) follows from Lemma 2.

preemptive schedule P :



PREEMPTIVE α -CONVERSION:



preemptive α -schedule:

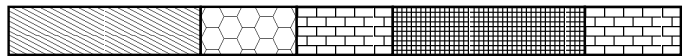


Fig. 2. The conversion of the preemptive schedule P by PREEMPTIVE α -CONVERSION and by PREEMPTIVE LIST SCHEDULING in order of non-decreasing α -points for the instance given in Figure 1

In the absence of nontrivial release dates, PREEMPTIVE LIST SCHEDULING always constructs a non-preemptive schedule that coincides with the non-preemptive list schedule. In particular, $C_j^\alpha = C_j^{\alpha-pmtn}$ and the bound given in (3) holds for C_j^α too. \square

Lemma 6, Corollary 1, and Lemma 8 contain all structural insights in (preemptive) α -schedules that are needed to derive the approximation results presented below.

3.3 Scheduling in Order of α -Points for Only One α

Corollary 1 and Lemma 6 yield the following generalization of Theorem 2 that was first presented in [11]:

Theorem 3. For each fixed $0 < \alpha \leq 1$, the completion time of job j in the α -schedule can be bounded by

$$C_j^\alpha \leq \left(1 + \frac{1}{\alpha}\right) C_j^P(\alpha) \leq \left(1 + \frac{1}{\alpha}\right) C_j^P .$$

Proof. Corollary 1 and Lemma 6 yield

$$\begin{aligned} C_j^\alpha &\leq C_j^P(\alpha) + \sum_{\substack{k \\ \eta_k(\alpha) \geq \alpha}} (1 + \alpha - \eta_k(\alpha)) p_k \\ &\leq C_j^P(\alpha) + \sum_{\substack{k \\ \eta_k(\alpha) \geq \alpha}} p_k \\ &\leq C_j^P(\alpha) + \sum_{\substack{k \\ \eta_k(\alpha) \geq \alpha}} \frac{\eta_k(\alpha)}{\alpha} p_k \\ &\leq \left(1 + \frac{1}{\alpha}\right) C_j^P(\alpha) . \end{aligned}$$

The second bound in Theorem 3 follows from the definition of α -points. □

It is shown in [11] that the bound given in Theorem 3 is tight. The following lemma is an analogue to Theorem 3 for preemptive α -schedules.

Lemma 9. For each fixed $0 < \alpha \leq 1$, the completion time of job j in the preemptive α -schedule can be bounded by

$$C_j^{\alpha\text{-pmtn}} \leq \frac{1}{\alpha} C_j^P(\alpha) \leq \frac{1}{\alpha} C_j^P .$$

In the absence of nontrivial release dates the same bound holds for the completion time C_j^α of job j in the non-preemptive α -schedule.

Proof. Lemma 8 and Lemma 6 yield

$$\begin{aligned} C_j^{\alpha\text{-pmtn}} &\leq C_j^P(\alpha) + \sum_{\substack{k \\ \eta_k(\alpha) \geq \alpha}} (1 - \eta_k(\alpha)) p_k \\ &\leq C_j^P(\alpha) + (1 - \alpha) \sum_{\substack{k \\ \eta_k(\alpha) \geq \alpha}} p_k \\ &\leq C_j^P(\alpha) + \frac{1 - \alpha}{\alpha} \sum_{\substack{k \\ \eta_k(\alpha) \geq \alpha}} \eta_k(\alpha) p_k \\ &\leq \frac{1}{\alpha} C_j^P(\alpha) . \end{aligned}$$

In the absence of nontrivial release dates the same bound holds for C_j^α since the result of Lemma 8 can be applied in this case too. □

4 An $e/(e - 1)$ -Approximation Algorithm for $1 | r_j | \sum C_j$

In this section we present the result of Chekuri et al. [11] on the conversion of preemptive to non-preemptive single machine schedules. Moreover, we discuss a class of instances found by Goemans et al. [17] which shows that this result is tight, that is, it yields a tight worst case bound on the value of an optimal non-preemptive schedule in terms of the optimal value with preemptions allowed.

In Theorem 3 the best choice of α seems to be $\alpha = 1$ yielding the factor 2 bound that has already been determined in Theorem 2. The key insight of Chekuri et al. is that one can get a better (expected) bound by drawing α randomly from $[0, 1]$ instead of using only one fixed value of α . The underlying intuition is that the completion time C_j^α of a given job j in the α -schedule cannot attain the worst case bound $(1 + \frac{1}{\alpha}) C_j^P(\alpha)$ for all possible choices of α . Moreover, although the worst case bound is bad for small values of α , the overall α -schedule might be good since the total amount of idle time caused by jobs in the first step of α -CONVERSION is small and the schedule is therefore short.

Theorem 4. *For each job j , let α_j be chosen from a probability distribution over $[0, 1]$ with density function $f(x) = e^x/(e - 1)$. Then, the expected completion time $E[C_j^\alpha]$ of job j in the α -schedule can be bounded from above by $\frac{e}{e-1} C_j^P$, where C_j^P denotes the completion time of job j in the given preemptive schedule P .*

Proof. To simplify notation we denote $\eta_k(1)$ by η_k . For any fixed α , Corollary 1 and Lemma 6 yield

$$\begin{aligned} C_j^\alpha &\leq C_j^P(\alpha_j) + \sum_{\substack{k \\ \eta_k \geq \alpha_k}} (1 + \alpha_k - \eta_k(\alpha_j)) p_k \\ &= C_j^P(\alpha_j) + \sum_{\substack{k \\ \eta_k \geq \alpha_k}} (\eta_k - \eta_k(\alpha_j)) p_k + \sum_{\substack{k \\ \eta_k \geq \alpha_k}} (1 + \alpha_k - \eta_k) p_k \\ &\leq C_j^P + \sum_{\substack{k \\ \eta_k \geq \alpha_k}} (1 + \alpha_k - \eta_k) p_k . \end{aligned}$$

In order to compute a bound on the expected completion time of job j , we integrate the derived bound on C_j^α over all possible choices of the random variables $\alpha_k, k \in J$, weighted by the given density function f . Since

$$\int_0^\eta f(x) (1 + x - \eta) dx = \frac{\eta}{e - 1}$$

for each $\eta \in [0, 1]$, the expected completion time of job j can be bounded by

$$\begin{aligned} E[C_j^\alpha] &\leq C_j^P + \sum_k p_k \int_0^{\eta_k} f(\alpha_k) (1 + \alpha_k - \eta_k) d\alpha_k \\ &= C_j^P + \frac{1}{e - 1} \sum_k \eta_k p_k \leq \frac{e}{e - 1} C_j^P . \end{aligned}$$

This concludes the proof. □

Theorem 4 is slightly more general than the original result in [11]. The only condition on the random choice of the vector α in Theorem 4 is that each of its entries α_j has to be drawn with density function f from the interval $[0, 1]$. Chekuri et al. only consider the case where $\alpha_j = \alpha$ for each $j \in J$ and α is drawn from $[0, 1]$ with density function f . However, the analysis also works for arbitrary interdependencies between the different random variables α_j , e. g., for jointly or pairwise independent α_j . The advantage of using only one random variable α for all jobs is that the resulting randomized approximation algorithm can easily be derandomized. We discuss this issue in more detail below.

Corollary 2. *Suppose that α is chosen as described in Theorem 4. Then, the expected value of the α -schedule is bounded from above by $\frac{e}{e-1}$ times the value of the preemptive schedule P .*

Proof. Using Theorem 4 and linearity of expectations yields

$$E\left[\sum_j w_j C_j^\alpha\right] = \sum_j w_j E[C_j^\alpha] \leq \frac{e}{e-1} \sum_j w_j C_j^P .$$

This concludes the proof. □

Since the preemptive problem $1|r_j, pmtn|\sum C_j$ can be solved in polynomial time by the shortest remaining processing time rule, computing this optimal solution and converting it as described in Theorem 4 is a randomized approximation algorithm with expected performance guarantee $\frac{e}{e-1}$ for $1|r_j|\sum C_j$.

The variant of this randomized algorithm with only one random variable α for all jobs instead of individual α_j 's is of special interest. In fact, starting from a preemptive list schedule P (e. g., the one constructed by the SRPT-rule), one can efficiently compute an α -schedule of least objective function value over *all* α between 0 and 1; we refer to this schedule as the *best α -schedule*. The following proposition, which can be found in [17], yields a deterministic $\frac{e}{e-1}$ -approximation algorithm with running time $O(n^2)$ for the problem $1|r_j|\sum C_j$.

Proposition 1. *For a fixed preemptive list schedule P , there are at most n different (preemptive) α -schedules; they can be computed in $O(n^2)$ time.*

Proof. As α goes from 0 to 1, the α -schedule changes only whenever an α -point, say for job j , reaches a time at which job j is preempted. Thus, the total number of changes in the (preemptive) α -schedule is bounded from above by the total number of preemptions. Since a preemption can occur in the preemptive list schedule P only whenever a job is released, the total number of preemptions is at most $n - 1$, and the number of (preemptive) α -schedules is at most n . Since each of these (preemptive) α -schedules can be computed in $O(n)$ time, the result on the running time follows. □

For the weighted scheduling problem $1|r_j|\sum w_j C_j$, even the preemptive variant is strongly NP-hard, see [24]. However, given a ρ -approximation algorithm for one of the preemptive scheduling problems $1|r_j, (prec,) pmtn|\sum w_j C_j$, the conversion technique of Chekuri et al. can be used to design an approximation with performance

ratio $\rho \frac{e}{e-1}$ for the corresponding non-preemptive problem. Unfortunately, this does not directly lead to improved approximation results for these problems. In Section 8, however, we present a more sophisticated combination of a 2-approximation algorithm for $1|r_j, prec, pmtn|\sum w_j C_j$ with this conversion technique by Schulz and Skutella [35], which yields performance guarantee e for the resulting algorithm.

Another consequence of Corollary 2 is the following result on the power of preemption which can be found in [17].

Theorem 5. *For single machine scheduling with release dates and precedence constraints so as to minimize the weighted sum of completion times, the value of an optimal non-preemptive schedule is at most $\frac{e}{e-1}$ times the value of an optimal preemptive schedule and this bound is tight, even in the absence of precedence constraints.*

Proof. Suppose that P is an optimal preemptive schedule. Then, the expected value of an α -schedule, with α chosen as described in Theorem 4, is bounded by $\frac{e}{e-1}$ times the value of the optimal preemptive schedule P . In particular, there must exist at least one fixed choice of the random variable α such that the value of the corresponding non-preemptive α -schedule can be bounded from above by $\frac{e}{e-1} \sum_j w_j C_j^P$.

To prove the tightness of this bound we consider the following instance with $n \geq 2$ jobs. There is one large job, denoted job n , and $n - 1$ small jobs denoted $j = 1, \dots, n - 1$. The large job has processing time $p_n = n$, weight $w_n = 1/n$ and release date $r_n = 0$. Each of the $n - 1$ small jobs j has zero processing time, weight $w_j = \frac{1}{n(n-1)}(1 + \frac{1}{n-1})^{n-j}$, and release date $r_j = j$.

The optimal preemptive schedule has job n start at time 0, preempted by each of the small jobs; hence its completion times are: $C_j = r_j$ for $j = 1, \dots, n - 1$ and $C_n = n$. Its objective function value is $(1 + \frac{1}{n-1})^n - (1 + \frac{1}{n-1})$ and approaches $e - 1$ for large n .

Now consider an optimal non-preemptive schedule C^* and let $k = \lfloor C_n^* \rfloor - n \geq 0$, so k is the number of small jobs that can be processed before job n . It is then optimal to process all these small jobs $1, \dots, k$ at their release dates, and to start processing job n at date $r_k = k$ just after job k . It is also optimal to process all remaining jobs $k + 1, \dots, n - 1$ at date $k + n$ just after job n . Let C^k denote the resulting schedule, that is, $C_j^k = j$ for all $j \leq k$, and $C_j^k = k + n$ otherwise. Its objective function value is $(1 + \frac{1}{n-1})^n - \frac{1}{n-1} - \frac{k}{n(n-1)}$. Therefore the optimal schedule is C^{n-1} with objective function value $(1 + \frac{1}{n-1})^n - \frac{1}{n-1} - \frac{1}{n}$. Thus, as n grows large, the optimal non-preemptive cost approaches e . \square

Unfortunately, the result of Theorem 5 cannot be easily extended to the case of unit weights. The instance discussed in the proof of the theorem relies on the large number of jobs with processing time zero whose weights are small compared to the weight of the large job.

5 Time-Indexed LP Relaxations

As already mentioned in the last section, even the preemptive variants of the scheduling problems $1|r_j|\sum w_j C_j$ and $1|prec|\sum w_j C_j$ are NP-hard. In order to give approximation results for these problems, we consider a time-indexed LP relaxation of

$1|r_j, prec, pmtn | \sum w_j C_j$ whose optimal value serves as a surrogate for the true optimum in our estimations. Moreover, we interpret a solution to this relaxation as a so-called *fractional preemptive schedule* which can be converted into a non-preemptive one using the techniques discussed in the previous sections.

For technical reasons only, we assume in the following that all processing times of jobs are positive. This is only done to keep the presentation of the techniques and results as simple as possible. Using additional constraints in the LP relaxation, all results presented can be carried over to the case with zero processing times.

The following LP relaxation is an immediate extension of a time-indexed LP proposed by Dyer and Wolsey [13] for the problem without precedence constraints. Here, time is discretized into the periods $(t, t + 1], t = 0, 1, \dots, T$ where $T + 1$ is the planning horizon, say $T + 1 := \max_{j \in J} r_j + \sum_{j \in J} p_j$. We have a variable y_{jt} for every job j and every time period $(t, t + 1]$ which is set to 1 if j is being processed in that time period and to 0 otherwise. The LP relaxation is as follows:

$$\text{minimize } \sum_{j \in J} w_j C_j^{LP}$$

$$\text{subject to } \sum_{t=r_j}^T y_{jt} = p_j \quad \text{for all } j \in J, \tag{4}$$

$$\sum_{j \in J} y_{jt} \leq 1 \quad \text{for } t = 0, \dots, T, \tag{5}$$

$$(LP) \quad \frac{1}{p_j} \sum_{\ell=r_j}^t y_{j\ell} \geq \frac{1}{p_k} \sum_{\ell=r_k}^t y_{k\ell} \quad \text{for all } j < k \text{ and } t = 0, \dots, T, \tag{6}$$

$$C_j^{LP} = \frac{p_j}{2} + \frac{1}{p_j} \sum_{t=r_j}^T y_{jt} (t + \frac{1}{2}) \quad \text{for all } j \in J, \tag{7}$$

$$y_{jt} = 0 \quad \text{for all } j \in J \text{ and } t = 0, \dots, r_j - 1,$$

$$y_{jt} \geq 0 \quad \text{for all } j \in J \text{ and } t = r_j, \dots, T.$$

Equations (4) say that all fractions of a job, which are processed in accordance with its release date, must sum up to the whole job. Since the machine can process only one job at a time, the machine capacity constraints (5) must be satisfied. Constraints (6) say that at any point $t + 1$ in time the completed fraction of job k must not be larger than the completed fraction of its predecessor j .

Consider an arbitrary feasible schedule P , where job j is being continuously processed between $C_j^P - p_j$ and C_j^P on the machine. Then, the expression for C_j^{LP} in (7) corresponds to the real completion time C_j^P of j , if we assign the values to the LP variables y_{jt} as defined above, that is, $y_{jt} = 1$ if j is being processed in the time interval $(t, t + 1]$. If j is not being continuously processed but preempted once or several times, the expression for C_j^{LP} in (7) is a lower bound on the real completion time. A more precise discussion of this matter is given in Lemma 10 a) below. Hence, (LP) is a relaxation of the scheduling problem under consideration.

A feasible solution y to (LP) does in general not correspond to a feasible preemptive schedule. Consider the following example:

Example 1. Let $J = \{1, \dots, n\}$ with $p_j = 1$ for all $j \in J$, $w_j = 0$ for $1 \leq j \leq n - 1$, $w_n = 1$ and $j \prec n$ for $1 \leq j \leq n - 1$. We get a feasible solution to (LP) if we set $y_{jt} = \frac{1}{n}$ for all $j \in J$ and $t = 0, \dots, T = n - 1$.

In the solution to (LP) given in Example 1 several jobs share the capacity of the single machine in each time interval. Moreover, the precedence relations between job n and all other jobs are not obeyed since fractions of job n are processed on the machine before its predecessors are completed. However, the solution fulfills constraint (6) such that at any point in time the completed fraction of job n is not larger than the completed fraction of each of its predecessors.

Phillips et al. [29] introduced the term *fractional preemptive schedule* for a schedule in which a job can share the capacity of one machine with several other jobs. We extend this notion to the case with precedence constraints and call a fractional preemptive schedule P feasible if no job is processed before its release date and, at any point in time, the completed fraction of job j is not larger than the completed fraction of each of its predecessors. Carrying over the definition of α -points to fractional preemptive schedules, the latter condition is equivalent to $C_j^P(\alpha) \leq C_k^P(\alpha)$ for all $j \prec k$ and $0 \leq \alpha \leq 1$.

Corollary 3. *The results presented in Section 3, in particular Lemma 6, Corollary 1, and Lemma 9, still hold if P is a fractional preemptive schedule.*

Proof. The proofs of the above-mentioned results can directly be carried over to the more general setting of fractional preemptive schedules. □

We always identify a feasible solution to (LP) with the corresponding feasible fractional preemptive schedule and vice versa. We mutually use the interpretation that seems more suitable for our purposes. The expression for C_j^{LP} in (7) is called the *LP completion time* of job j .

The following lemma highlights the relation between the LP completion times and the α -points of jobs in the corresponding fractional preemptive schedule for a feasible solution to (LP) . The observation in a) is due to Goemans [16]; it says that the LP completion time of job j is the sum of half its processing time and its mean busy time. An analogous result to b) was given in [22, Lemma 2.1] for a somewhat different LP.

Lemma 10. *Consider a feasible solution y to (LP) and the corresponding feasible fractional preemptive schedule P . Then, for each job j :*

$$a) C_j^{LP} = \frac{p_j}{2} + \frac{1}{p_j} \sum_{t=r_j}^T y_{jt} \left(t + \frac{1}{2}\right) = \frac{p_j}{2} + \int_0^1 C_j^P(\alpha) d\alpha \leq C_j^P$$

and equality holds if and only if $C_j^P = C_j^P(0) + p_j$;

$$b) C_j^P(\alpha) \leq \frac{1}{1-\alpha} C_j^{LP} \quad \text{for any constant } \alpha \in [0, 1) \text{ and this bound is tight.}$$

Proof. For a job $j \in J$, we denote by ζ_j^t , $t = r_j, \dots, T + 1$, the fraction of j that is finished in the fractional preemptive schedule P by time t . Since $0 = \zeta_j^{r_j} \leq \zeta_j^{r_j+1} \leq \dots \leq \zeta_j^{T+1} = 1$, we get

$$\begin{aligned} \int_0^1 C_j^P(\alpha) d\alpha &= \sum_{t=r_j}^T \int_{\zeta_j^t}^{\zeta_j^{t+1}} C_j^P(\alpha) d\alpha \\ &= \sum_{t=r_j}^T (\zeta_j^{t+1} - \zeta_j^t) (t + \frac{1}{2}) \end{aligned}$$

since $C_j^P(\alpha) = t + \frac{\alpha - \zeta_j^t}{\zeta_j^{t+1} - \zeta_j^t}$ for $\alpha \in (\zeta_j^t, \zeta_j^{t+1}]$,

$$= \sum_{t=r_j}^T \frac{y_{jt}}{p_j} (t + \frac{1}{2}) .$$

The machine capacity constraints (5) yield $C_j^P(\alpha) \leq C_j^P - (1 - \alpha) p_j$ for $\alpha \in [0, 1]$, thus

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

Equality holds if and only if $C_j^P(\alpha) = C_j^P - (1 - \alpha) p_j$ for $0 \leq \alpha \leq 1$, which is equivalent to $C_j^P(0) = C_j^P - p_j$. This completes the proof of a). As a consequence we get for $\alpha \in [0, 1]$

$$(1 - \alpha) C_j^P(\alpha) \leq \int_{\alpha}^1 C_j^P(x) dx \leq \int_0^1 C_j^P(x) dx \leq C_j^{LP} .$$

In order to prove the tightness of this bound, we consider a job j with $p_j = 1$, $r_j = 0$ and an LP solution satisfying $y_{j0} = \alpha - \epsilon$ and $y_{jT} = 1 - \alpha + \epsilon$, where $\epsilon > 0$ small. This yields $C_j^{LP} = 1 + (1 - \alpha + \epsilon) T$ and $C_j^P(\alpha) \geq T$. Thus, for ϵ arbitrarily small and T arbitrarily large, the given bound gets tight. \square

As a result of Lemma 10 b) the value of the fractional preemptive schedule given by a solution to (LP) can be arbitrarily bad compared to its LP value. Nevertheless, one can use the information that is contained in the structure of an LP solution in order to construct a feasible schedule whose value can be bounded in terms of the LP value, as will be shown in the following sections.

Notice that we cannot solve (LP) in polynomial time, but only in pseudo-polynomial time, since T and therefore the number of variables y_{jt} is only pseudo-polynomial in the input size of the problem. Schulz and Skutella [37] describe a closely related and only slightly worse LP relaxation of polynomial size in the general context of unrelated parallel machines. The idea is to change to new variables which are not associated with exponentially many time intervals of length 1, but rather with a polynomial number of intervals of geometrically increasing size. We omit the technical details in this survey.

In the absence of precedence constraints, it was indicated by Dyer and Wolsey [13] that (LP) can be solved in $O(n \log n)$ time. Goemans [15] developed the following result (see also [17]).

Theorem 6. *For instances of the problems $1|r_j(\cdot, pmtn)|\sum w_j C_j$, the linear programming relaxation (LP) can be solved in $O(n \log n)$ time and the preemptive list schedule in order of non-decreasing ratios p_j/w_j corresponds to an optimal solution.*

Proof. In the absence of precedence relations, constraints (6) are missing in (LP) . Moreover, we can eliminate the variables C_j^{LP} from the relaxation by plugging (7) into the objective function. What remains is a transportation problem and, as a result, y_{jt} can be assumed to be integral.

The remaining part of the proof is based on an interchange argument: Consider any optimal 0/1-solution y to (LP) and suppose that $j < k$ (such that $p_j/w_j \leq p_k/w_k$), $r_j \leq t < \tau$, and $y_{kt} = y_{j\tau} = 1$. Then, replacing $y_{kt} = y_{j\tau}$ by 0 and $y_{k\tau} = y_{jt}$ by 1 gives another feasible solution to (LP) with an increase in the objective function of $(\tau - t) \left(\frac{w_k}{p_k} - \frac{w_j}{p_j} \right) \leq 0$. Thus, the new solution is also optimal and through iterative application of this interchange argument we arrive at the solution that corresponds to the preemptive list schedule in order of non-decreasing ratios p_j/w_j . \square

It follows from Theorem 6 that in the absence of precedence constraints and release dates an optimal single machine schedule corresponds to an optimal solution to (LP) . Moreover, if we allow release dates and preemption, the optimal solution to (LP) described in Theorem 6 is a feasible schedule that minimizes the weighted sum of mean busy times. It is also shown in [17] that, in the absence of precedence constraints, the LP relaxation (LP) is equivalent to an LP in completion time variables which defines a supermodular polyhedron.

5.1 Another Time-Indexed LP Relaxation

For the non-preemptive problem $1|r_j|\sum w_j C_j$ Dyer and Wolsey [13] also proposed another time-indexed LP relaxation that can easily be extended to the setting with precedence constraints; see Figure 3. Again, for each job j and each integral point in time $t = 0, \dots, T$, we introduce a decision variable x_{jt} . However, now the variable is set to 1 if j starts processing at time t and to 0 otherwise. Note that the x_{jt} variables do not have the preemptive flavor of the y_{jt} variables and therefore lead to an LP relaxation for non-preemptive single machine scheduling only.

In the absence of precedence constraints, Dyer and Wolsey showed that this relaxation is stronger than (LP) . In fact, even for instances with precedence constraints, every feasible solution to (LP') can easily be transformed into a solution to (LP) of the same value by assigning

$$y_{jt} := \sum_{\ell=\max\{0, t-p_j+1\}}^t x_{j\ell} \quad \text{for } j \in J \text{ and } t = 0, \dots, T.$$

In particular, we can interpret a feasible solution to (LP') as a feasible fractional preemptive schedule and the results in Lemma 10 also hold in this case.

$$\begin{aligned}
 &\text{minimize} && \sum_{j \in J} w_j C_j^{LP} \\
 &\text{subject to} && \sum_{t=r_j}^T x_{jt} = 1 && \text{for all } j \in J, \\
 &&& \sum_{j \in J} \sum_{\ell=\max\{0, t-p_j+1\}}^t x_{j\ell} \leq 1 && \text{for all } t = 0, \dots, T, \tag{8} \\
 (LP') &&& \sum_{\ell=r_j}^t x_{j\ell} \geq \sum_{\ell=r_k}^{t+p_j} x_{k\ell} && \text{for all } j \prec k \text{ and } t = r_j, \dots, T - p_j, \tag{9} \\
 &&& C_j^{LP} = p_j + \sum_{t=r_j}^T t x_{jt} && \text{for all } j \in J, \\
 &&& x_{jt} = 0 && \text{for all } j \in J \text{ and } t = 0, \dots, r_j - 1, \tag{10} \\
 &&& x_{jt} \geq 0 && \text{for all } j \in J \text{ and } t = r_j, \dots, T.
 \end{aligned}$$

Fig. 3. The time-indexed LP relaxation (LP')

5.2 Combinatorial Relaxations

In this subsection we show that the LP relaxation (LP') is equivalent to a combinatorial relaxation of $\lfloor r_j, prec \rfloor \sum w_j C_j$. We interpret an instance of this scheduling problem as a game for one person who is given the set of jobs J and the single machine and wants to find a feasible schedule of minimum value.

We consider the following relaxation of this game: Assume that there are k players $1, \dots, k$ instead of only one. Each player i is given one machine and a set of jobs $J_i = J$. Moreover, the players are allowed to cooperate by exchanging jobs. To be more precise, a job of player i can be scheduled on an arbitrary machine rather than only on i 's machine. However, each player has to respect release dates and precedence constraints of his jobs. The aim is to minimize the average over all players of the weighted sum of completion times of their jobs. This relaxation is called *k-player relaxation*, a feasible solution is a *k-player schedule*. Moreover, if k is not fixed but can be chosen, we call the resulting relaxation *multi-player relaxation* and a feasible solution is a *multi-player schedule*.

As an example, consider the following single machine instance without precedence constraints consisting of four jobs:

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

Note that job 1 is the most important job. The instance is constructed such that either at least one unit of idle time has to be introduced before the start of job 1 or this job has to be delayed until time 7. Both alternatives are optimal and give a schedule with value 87. However, there exists a 2-player schedule of value 83.5: Both players start their first job at time 6. Player 1 starts his second job at time 0 and then, at time 3, the second job of player 2 on his machine. Player 2 processes the four remaining jobs of length 2 in the intervals $(0, 6]$ and $(8, 10]$ on his machine. Notice that the two players neither delay their first jobs nor introduce idle time before their start. As a result of this example we get the following corollary.

Corollary 4. *The 2-player relaxation is not better than a $\frac{174}{167}$ -relaxation for the problem $1|r_j|\sum w_j C_j$.*

The negative result in Corollary 4 can be slightly improved to a bound of $1 + 1/(2\sqrt{55} + 9) > \frac{174}{167}$ by increasing the processing time of job 1 to $(\sqrt{55} - 3)/2$ and its weight to $\sqrt{55} - 1$. The main result of this subsection is the following theorem:

Theorem 7. *The LP relaxation (LP') of the problem $1|r_j, prec|\sum w_j C_j$ is equivalent to the multi-player relaxation.*

Proof. We first argue that each k -player schedule corresponds to a feasible solution of (LP') with the same value: For all $j \in J$ and $t = 0, \dots, T$ let x_{jt} be the number of players whose job $j \in J_i$ is started at time t divided by k . It follows from the definition of the relaxation that x satisfies all constraints of (LP') and is thus a feasible solution. To understand that constraints (9) are fulfilled, observe that each player has to respect the precedence constraints of his jobs. Moreover, the value of x is equal to the value of the k -player schedule by definition of x .

On the other hand, we can show that for each feasible solution x whose values x_{jt} are all rational numbers there exists a k such that x corresponds to a feasible solution of the k -player relaxation. Let k be the least common multiple of the denominators of the rational values x_{jt} . For each job $j \in J$ and each player $i = 1, \dots, k$, let t_{ji} the smallest integer such that $\sum_{\ell=0}^{t_{ji}} x_{j\ell} \geq \frac{i}{m}$. The value t_{ji} is the starting time of i 's job j in the k -player schedule that we construct by induction over $t = 0, \dots, T$: At time $t = 0$ start all jobs $j \in J$ of players i with $t_{ji} = 0$ on the k machines. Notice that the number of these job-player pairs is bounded by k since $\sum_{j \in J} x_{j0} \leq 1$ by constraints (8). Assume that for all players i all jobs j with $t_{ji} < t$ have been started at time t_{ji} . Then, the number of idle machines at time t is equal to

$$k \left(1 - \sum_{j \in J} \sum_{\ell=\max\{0, t-p_j+1\}}^{t-1} x_{j\ell} \right).$$

Therefore, by constraints (8), there are sufficiently many idle machines at time t such that all jobs j of players i with $t_{ji} = t$ can be started. This k -player schedule respects release dates by constraints (10). Moreover, for a pair of jobs $j \prec k$ and each player i we get $t_{ji} + p_j \leq t_{ki}$ by the definition of the t_{ji} 's and constraints (9). \square

Similar interpretations of time-indexed LP relaxations have been given by van den Akker [2, Proof of Theorem 3.1] and Schulz [33]. Chan, Muriel, and Simchi-Levi [8] used this idea to prove results on the quality of those time-indexed LP relaxations.

It is an interesting direction for future research to investigate how the structure of an optimal k -player schedule can be used to construct provably good schedules. Another interesting question is if one can restrict to special values of k in order to get an optimal multi-player schedule, e. g., $k \leq n$ or $k \in O(n)$, or values of k depending on the given instance. This would also yield results on the structure of the polyhedron corresponding to (LP') , i. e., results on an optimal vertex or on general vertices (like $\frac{1}{n}$ -integrality etc.).

Another interesting topic is the complexity of the k -player relaxation for special values of k . Of course, it is NP-hard for $k = 1$. We conjecture that it is NP-hard to find an optimal k -player schedule for each fixed k . We also guess that it is NP-hard to compute an optimal multi-player schedule when k is not fixed but bounded by the number of jobs n .

5.3 On the Quality of Time-Indexed LP Relaxations

We conclude this section with negative and positive results on the quality of the relaxations (LP) and (LP') . The positive results follow from the LP-based approximations discussed in the following sections. The lower bound for (LP) and $1|r_j|\sum w_j C_j$ is due to Goemans et al. [17]. A summary of the results is given in Table 2.

Theorem 8.

- a) *The LP relaxations (LP) and (LP') are 2-relaxations for the scheduling problem $1|prec|\sum w_j C_j$ and this bound is tight.*
- b) *(LP) is a 2-relaxation for the problem $1|r_j, prec, pmtn|\sum w_j C_j$ and this bound is tight.*
- c) *(LP) and (LP') are e -relaxations for the problem $1|r_j, prec|\sum w_j C_j$ and not better than 2-relaxations.*
- d) *(LP) is a $\frac{4}{3}$ -relaxation and not better than an $\frac{8}{7}$ -relaxation for the scheduling problem $1|r_j, pmtn|\sum w_j C_j$.*
- e) *(LP) is a 1.6853-relaxation and not better than an $\frac{e}{e-1}$ -relaxation for the problem $1|r_j|\sum w_j C_j$.*
- f) *(LP') is a 1.6853-relaxation and not better than a $\frac{174}{167}$ -relaxation for the problem $1|r_j|\sum w_j C_j$ ($\frac{174}{167} \approx 1.0419$).*

Proof. It is shown in Section 8 that (LP) is a 2-relaxation for $1|prec|\sum w_j C_j$ and $1|r_j, prec, pmtn|\sum w_j C_j$, and an e -relaxation for the problem $1|r_j, prec|\sum w_j C_j$. Moreover, as mentioned above, (LP') is a stronger relaxation than (LP) for the non-preemptive problems. To prove the negative results in a), b), and c) we use the instance given in Example 1 and consider the feasible solutions to (LP) and (LP') given by $y_{jt} = x_{jt} = \frac{1}{n}$ for $j = 1, \dots, n - 1$ and $t = 0, \dots, n - 1$; moreover, we set $y_{nt} = x_{nt} = \frac{1}{n}$ for $t = 1, \dots, n$. The value of an optimal schedule is n whereas the value of the given solutions to (LP) and (LP') is $\frac{n+3}{2}$. When n goes to infinity, the ratio of these two values converges to 2.

Table 2. Summary of results on the quality of the time-indexed LP relaxations (LP) and (LP') given in Theorem 8

Results on the quality of (LP) and (LP')				
problem	(LP)		(LP')	
	lower bound	upper bound	lower bound	upper bound
$1 r_j \sum w_jC_j$	1.5820	1.6853	1.0419	1.6853
$1 r_j, pmtn \sum w_jC_j$	$\frac{8}{7}$	$\frac{4}{3}$	—	—
$1 prec \sum w_jC_j$	2	2	2	2
$1 r_j, prec \sum w_jC_j$	2	e	2	e
$1 r_j, prec, pmtn \sum w_jC_j$	2	2	—	—

The positive results in d), e), and f) are derived in Section 7. To prove the negative result in d), 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 $\frac{n}{2}$, and all weights are $\frac{1}{n^2}$. The last job has processing time $p_n = n$, weight $w_n = \frac{1}{2n}$, and is released at time 0. This instance is constructed such that every reasonable preemptive schedule without idle time on the machine has value $2 - \frac{3}{2n}$. However, an optimal solution to (LP) given in Theorem 6 has value $\frac{7}{4} - \frac{5}{4n}$ such that the ratio goes to $\frac{8}{7}$ when n gets large.

We use Theorem 5 in order to prove the negative result in e). There we have argued that the ratio between the value of an optimal non-preemptive schedule and the value of an optimal preemptive schedule can be arbitrarily close to $\frac{e}{e-1}$. The optimal LP value is a lower bound on the value of an optimal preemptive schedule; this completes the proof of e).

The negative result in f) follows from Corollary 4 and Theorem 7. □

It is an open problem and an interesting direction for future research to close the gaps between the lower and the upper bounds highlighted in Table 2. We conjecture that the lower bound of $\frac{e}{e-1}$ for the relaxation (LP) of the problem $1|r_j|\sum w_jC_j$ is not tight.

For the relaxation (LP') of the same problem we strongly believe that the precise ratio is closer to the lower than to the upper bound. We hope that the combinatorial interpretation given in Subsection 5.2 will lead to improved approximation results and upper bounds on the quality of the relaxation (LP').

Remark 1. For the problem with precedence constraints the relaxation (LP) can be slightly strengthened by replacing (6) with the stronger constraints

$$\frac{1}{p_j} \sum_{\ell=r_j}^t y_{j\ell} \geq \frac{1}{p_k} \sum_{\ell=r_k}^{t+\min\{p_j, p_k\}} y_{k\ell} \quad \text{for all } j \prec k \text{ and } t = 0, \dots, T - \min\{p_j, p_k\} .$$

However, the relaxation is not stronger than (LP') by constraints (9), and therefore not better than a 2-relaxation.

Potts [30] introduced a linear programming relaxation of $1|prec|\sum w_j C_j$ in linear ordering variables. Hall et al. [21] showed that this relaxation is a 2-relaxation. Chekuri and Motwani [10] provided a class of instances showing that this result is tight.

6 Scheduling in Order of LP Completion Times

Schulz [34] (see also [21]) introduced the idea of LIST SCHEDULING in order of LP completion times and applied it to get approximation algorithms for quite a few problems to minimize the weighted sum of completion times. In this section we briefly review his results for the scheduling problems under consideration in order to give the reader the opportunity to compare the underlying ideas and intuition with the concept of α -points.

Schulz used a strengthened version of an LP relaxation in completion time variables that was introduced by Wolsey [52] and Queyranne [31]. However, the technique and analysis can also be applied to the stronger relaxation (LP) in time-indexed variables. The following lemma contains the key insight and is a slightly weaker version of [34, Lemma 1].

Lemma 11. *Consider a feasible solution y to (LP). Then, for each job $j \in J$*

$$\sum_{\substack{k \\ C_k^{LP} \leq C_j^{LP}}} p_k \leq 2 C_j^{LP} .$$

Proof. Let $K := \{k \in J \mid C_k^{LP} \leq C_j^{LP}\}$ and $p(K) := \sum_{k \in K} p_k$. This yields

$$\begin{aligned} p(K) C_j^{LP} &\geq \sum_{k \in K} p_k C_k^{LP} \\ &\geq \sum_{k \in K} \sum_{t=0}^T y_{kt} \left(t + \frac{1}{2}\right) && \text{by Lemma 10 a)} \\ &= \sum_{t=0}^T \left(t + \frac{1}{2}\right) \sum_{k \in K} y_{kt} ; \end{aligned}$$

using the constraints (5) and (4) the last term can be bounded by

$$\geq \sum_{t=0}^{p(K)-1} \left(t + \frac{1}{2}\right) = \frac{1}{2} p(K)^2 .$$

Dividing the resulting inequality by $p(K)$ yields the result. □

With Lemma 11 at hand one can prove the following theorem:

Theorem 9. *Given a feasible solution to (LP), LIST SCHEDULING in order of non-decreasing LP completion times yields a feasible schedule where the completion time of each job j is bounded by $2 C_j^{LP}$ in the absence of nontrivial release dates, and by $3 C_j^{LP}$ else.*

Proof. We denote the fractional preemptive schedule corresponding to the given feasible LP solution by P . To check the feasibility of the computed schedule observe that $j \prec k$ implies $C_j^P(\alpha) \leq C_k^P(\alpha)$ for $0 \leq \alpha \leq 1$, and thus $C_j^{LP} \leq C_k^{LP}$ by Lemma 10 a). Therefore, if ties are broken in accordance with the precedence constraints the sequence is feasible.

In the absence of nontrivial release dates, the completion time of each job j is the sum of the processing times of jobs that start no later than j . Thus, the bound of 2 is a direct consequence of Lemma 11.

Since $\max_{k: C_k^{LP} \leq C_j^{LP}} r_k$ is a lower bound on C_j^{LP} , the bound of 3 follows from Lemma 11 and Lemma 3. \square

The LP relaxation in completion time variables that is used in [34] can be solved in polynomial time. Therefore, computing an optimal solution to the LP relaxation and then applying LIST SCHEDULING in order of non-decreasing LP completion times is a 2-approximation algorithm for $1|prec|\sum w_j C_j$ and a 3-approximation algorithm for $1|r_j, prec|\sum w_j C_j$. For the problem without nontrivial release dates, Schulz proves a slightly better performance guarantee of $2 - \frac{2}{n+1}$.

Hall et al. [21] show that PREEMPTIVE LIST SCHEDULING in order of non-decreasing LP completion times for an appropriate LP relaxation in completion time variables is a 2-approximation algorithm for the problem $1|r_j, prec, pmtn|\sum w_j C_j$.

Möhring, Schulz, and Uetz [27] study the problem of minimizing the total weighted completion time in stochastic machine scheduling. Job processing times are not known in advance, they are realized on-line according to given probability distributions. The aim is to find a scheduling policy that minimizes the average weighted completion time in expectation. They generalize results from deterministic scheduling and derive constant-factor performance guarantees for priority rules which are guided by optimal LP solutions in completion time variables. Skutella and Uetz [43, 44] generalize this approach and give approximation algorithms with constant performance guarantee for precedence-constrained scheduling problems.

7 Approximations for Single Machine Scheduling with Release Dates

In this section we present approximation algorithms for the problems $1|r_j|\sum w_j C_j$ and its preemptive variant $1|r_j, pmtn|\sum w_j C_j$ that have been obtained by Goemans et al. [17] and Schulz and Skutella [36], respectively. The first constant-factor approximation algorithms to minimize the total weighted completion time on a single machine subject to release dates are due to Phillips et al. [29]. They consider an LP relaxation that is also based on time-indexed variables which have a different meaning however. Based on an optimal solution to this relaxation they apply an idea which is somehow related to PREEMPTIVE LIST SCHEDULING in order of α -points for $\alpha = \frac{1}{2}$. This leads to an approximation algorithm with performance guarantee $8 + \varepsilon$ for the generalization of the problem $1|r_j, pmtn|\sum w_j C_j$ to the setting of unrelated parallel machines. Together with the conversion technique described in Theorem 2 this yields a $(16 + \varepsilon)$ -approximation algorithm for $1|r_j|\sum w_j C_j$.

Hall et al. [22] use LIST SCHEDULING in order of α -points for several problems and different but fixed values of α based on the LP relaxation (LP') in order to compute provably good schedules. However, their definition of α -points slightly differs from the one discussed here. Based on a different approach that relies on the results of Shmoys and Tardos [40] for the generalized assignment problem, they give a 4-approximation algorithm for $1|r_j|\sum w_j C_j$, which has subsequently been improved to performance guarantee 3, see Section 6 and [21].

Independently from each other, Chekuri et al. [11] (see Section 4) and Goemans [16] have taken up the idea of converting preemptive schedules into non-preemptive ones by LIST SCHEDULING in order of α -points (as introduced in [29, 22]), and enriched it by the use of randomness. Recently, Afrati et al. [1] gave polynomial-time approximation schemes for the problems considered in this section and generalizations thereof.

We analyze the following simple randomized algorithm for single machine scheduling with release dates. We consider both the non-preemptive and the preemptive variant of the algorithm which only differ in the last step:

Algorithm: RANDOM- α

- 1) Construct the preemptive list schedule P in order of non-decreasing ratios p_j/w_j .
- 2) For each job $j \in J$, draw α_j randomly from $[0, 1]$.
- 3) Output the resulting (preemptive) α -schedule.

Since (PREEMPTIVE) LIST SCHEDULING can be implemented to run in $O(n \log n)$ time by Lemma 1, the running time of Algorithm RANDOM- α is $O(n \log n)$. It follows from Theorem 6 that the first step of Algorithm RANDOM- α implicitly computes an optimal solution to the relaxation (LP). This observation is used in the analysis. Note, however, that the algorithm is purely combinatorial and can be formulated and implemented without even knowing the LP relaxation.

While the total number of possible orderings of jobs is $n! = 2^{O(n \log n)}$, it is shown in [17] that the maximal number of different α -schedules which can be computed by Algorithm RANDOM- α is at most 2^{n-1} and this bound can be attained. In particular, Algorithm RANDOM- α could also be formulated over a discrete probability space. Due to the possibly exponential number of different α -schedules, we cannot afford to derandomize Algorithm RANDOM- α by enumerating all (α_j) -schedules. One can instead use the method of conditional probabilities [28] and the derandomized version can be implemented to run in $O(n^2)$ time; we refer to [17] for details.

As in Section 4, the variant of Algorithm RANDOM- α with only one random variable α for all jobs instead of individual α_j 's is of special interest. We denote this variant by RANDOM- α and it follows from Proposition 1 that it can be derandomized yielding a deterministic algorithm with running time $O(n^2)$. This deterministic algorithm is called BEST- α . The proof of the following results on the performance of Algorithm RANDOM- α (and thus of Algorithm BEST- α) can be found in [17].

Theorem 10.

- a) For fixed α the performance guarantee of Algorithm RANDOM- α is $\max\{1 + \frac{1}{\alpha}, 1 + 2\alpha\}$. In particular, for $\alpha = 1/\sqrt{2}$ the performance guarantee is $1 + \sqrt{2}$.

b) If α is drawn uniformly at random from $[0, 1]$, then the expected performance guarantee of RANDOM- α is 2.

The same performance ratio of 2 is also achieved by Algorithm RANDOM- α with independent and uniformly distributed random variables α_j . However, in this case the analysis is noticeably simpler and more general. In particular it does not make use of the special structure of the preemptive list schedule P but works for an arbitrary fractional preemptive schedule P corresponding to a feasible solution to (LP); see Lemma 12 below.

Theorem 11. *Let the random variables α_j be pairwise independently and uniformly drawn from $[0, 1]$. Then, Algorithm RANDOM- α achieves expected performance guarantee 2 for the scheduling problems $1|r_j|\sum w_j C_j$ and $1|r_j, pmtn|\sum w_j C_j$.*

Theorem 11 is a consequence of the stronger result in Lemma 12. Starting with an arbitrary fractional preemptive schedule in the first step of RANDOM- α we can relate the value of the schedule computed by the algorithm to the corresponding LP value.

Lemma 12. *Let y be a feasible solution to (LP) and denote the corresponding fractional preemptive schedule by P . Suppose that the random variables α_j are pairwise independently and uniformly drawn from $[0, 1]$. Then, for each job $j \in J$, its expected completion time in the corresponding α -schedule is at most $2C_j^{LP}$.*

Proof. We consider an arbitrary, but fixed job $j \in J$. To analyze the expected completion time of j , we first keep α_j fixed, and consider the conditional expectation $E_{\alpha_j}[C_j^\alpha]$. Since the random variables α_j and α_k are independent for each $k \neq j$, Corollary 1 and Lemma 6 yield

$$\begin{aligned} E_{\alpha_j}[C_j^\alpha] &\leq C_j^P(\alpha_j) + p_j + \sum_{k \neq j} p_k \int_0^{\eta_k(\alpha_j)} (1 + \alpha_k - \eta_k(\alpha_j)) d\alpha_k \\ &= C_j^P(\alpha_j) + p_j + \sum_{k \neq j} p_k \left(\eta_k(\alpha_j) - \frac{\eta_k(\alpha_j)^2}{2} \right) \\ &\leq C_j^P(\alpha_j) + p_j + \sum_{k \neq j} p_k \eta_k(\alpha_j) \leq 2 \left(C_j^P(\alpha_j) + \frac{1}{2} p_j \right). \end{aligned}$$

To get the unconditional expectation $E[C_j^\alpha]$ we integrate over all possible choices of α_j :

$$E[C_j^\alpha] = \int_0^1 E_{\alpha_j}[C_j^\alpha] d\alpha_j \leq 2 \left(\int_0^1 C_j^P(\alpha_j) d\alpha_j + \frac{p_j}{2} \right) = 2C_j^{LP};$$

the last equation follows from Lemma 10 a). □

Proof (of Theorem 11). Algorithm RANDOM- α starts with an optimal solution to (LP) whose value is a lower bound on the value of an optimal (preemptive) schedule. We

compare the expected value of the (preemptive) α -schedule to this lower bound. Using Lemma 12 and linearity of expectations we get

$$E\left[\sum_j w_j C_j^\alpha\right] = \sum_j w_j E[C_j^\alpha] \leq 2 \sum_j w_j C_j^{LP} .$$

Since for each fixed α and each job j the completion time of j in the preemptive α -schedule is always less than or equal to its completion time in the non-preemptive α -schedule, the preemptive variant of Theorem 11 follows from the result for the non-preemptive case. We have even shown that the non-preemptive variant of Algorithm RANDOM- α computes a schedule whose expected value is bounded by twice the value of an optimal preemptive schedule. Thus, even this algorithm constitutes a randomized 2-approximation algorithm for $1|r_j, pmtn|\sum w_j C_j$. \square

The expected performance guarantee of Algorithm RANDOM- α can be improved beyond 2 by using more intricate density functions and by exploiting the special structure of the preemptive list schedule P in the first step of Algorithm RANDOM- α .

We start with an analysis of the structure of the preemptive list schedule P . Consider any job j , and assume that, in the preemptive schedule P , job j is preempted at time s and its processing resumes at time $t > s$. Then all jobs which are processed between s and t have a smaller index; as a result, these jobs will be completely processed between times s and t . Thus, in the preemptive list schedule P , between the start time and the completion time of any job j , the machine is constantly busy, alternating between the processing of portions of j and the complete processing of groups of jobs with smaller index. Conversely, any job preempted at the start time $C_j^P(0)$ of job j will have to wait at least until job j is complete before its processing can be resumed.

We capture this structure by partitioning, for a fixed job j , the set of jobs $J \setminus \{j\}$ into two subsets J_1 and J_2 : Let J_2 denote the set of all jobs that are processed between the start and completion of job j . All remaining jobs are put into subset J_1 . Notice that the function η_k is constant for jobs $k \in J_1$; to simplify notation we write $\eta_k := \eta_k(\alpha_j)$ for those jobs. The same holds for the function t_{idle} since no idle time occurs between the start and the completion of job j in P ; we thus write t_{idle} instead of $t_{\text{idle}}(\alpha_j)$. For $k \in J_2$, let $0 < \mu_k < 1$ denote the fraction of job j that is processed before the start of job k ; the function η_k is then given by

$$\eta_k(\alpha_j) = \begin{cases} 0 & \text{if } \alpha_j \leq \mu_k, \\ 1 & \text{if } \alpha_j > \mu_k, \end{cases} \quad \text{for } k \in J_2.$$

We can now rewrite the equation in Lemma 6 as

$$C_j^P(\alpha_j) = t_{\text{idle}} + \sum_{k \in J_1} \eta_k p_k + \sum_{\substack{k \in J_2 \\ \alpha_j > \mu_k}} p_k + \alpha_j p_j = C_j^P(0) + \sum_{\substack{k \in J_2 \\ \alpha_j > \mu_k}} p_k + \alpha_j p_j . \tag{11}$$

Plugging (11) into Lemma 10 a) yields

$$C_j^{LP} = C_j^P(0) + \sum_{k \in J_2} (1 - \mu_k) p_k + p_j . \tag{12}$$

Moreover, Corollary 1 can be rewritten as

$$C_j^\alpha \leq C_j^P(0) + \sum_{\substack{k \in J_1 \\ \alpha_k \leq \eta_k}} (1 + \alpha_k - \eta_k) p_k + \sum_{\substack{k \in J_2 \\ \alpha_j > \mu_k}} (1 + \alpha_k) p_k + (1 + \alpha_j) p_j , \tag{13}$$

where, for $k \in J_2$, we have used the fact that $\alpha_k \leq \eta_k(\alpha_j)$ is equivalent to $\alpha_j > \mu_k$. Similarly, Lemma 8 can be rewritten as

$$C_j^{\alpha-pmtn} \leq C_j^P(0) + \sum_{\substack{k \in J_1 \\ \alpha_k \leq \eta_k}} (1 - \eta_k) p_k + \sum_{\substack{k \in J_2 \\ \alpha_j > \mu_k}} p_k + p_j . \tag{14}$$

The expressions (11), (12), (13), and (14) reflect the structural insights that we need for proving stronger bounds for α -schedules and preemptive α -schedules in the sequel.

As mentioned above, the second ingredient for an improvement on the bound of 2 in Theorem 11 is a more sophisticated probability distribution of the random variables α_j . In view of the bound on C_j^α given in (13), we have to cope with two contrary phenomena: On the one hand, small values of α_k keep the terms of the form $(1 + \alpha_k - \eta_k)$ and $(1 + \alpha_k)$ on the right-hand side of (13) small; on the other hand, choosing larger values decreases the number of terms in the first sum on the right-hand side of (13). The balancing of these two effects contributes to reducing the bound on the expected value of C_j^α . Similar considerations can be made for the preemptive case and the bound given in (14).

7.1 A 1.6853-Approximation Algorithm for $1 \mid r_j \mid \sum w_j C_j$

In this subsection, we will prove the following theorem that was achieved by Goemans et al. [17].

Theorem 12. *Let $\gamma \approx 0.4835$ be the unique solution to the equation*

$$\gamma + \ln(2 - \gamma) = e^{-\gamma}((2 - \gamma)e^\gamma - 1)$$

satisfying $0 < \gamma < 1$. Define $\delta := \gamma + \ln(2 - \gamma) \approx 0.8999$ and $c := 1 + e^{-\gamma}/\delta < 1.6853$. Let the α_j 's be chosen pairwise independently from a probability distribution over $[0, 1]$ with the density function

$$f(\alpha) = \begin{cases} (c - 1)e^\alpha & \text{if } \alpha \leq \delta, \\ 0 & \text{otherwise,} \end{cases}$$

see Figure 4. Then, the expected completion time of every job j in the non-preemptive schedule constructed by Algorithm RANDOM- α is at most cC_j^{LP} and the expected performance guarantee of Algorithm RANDOM- α is $c < 1.6853$.

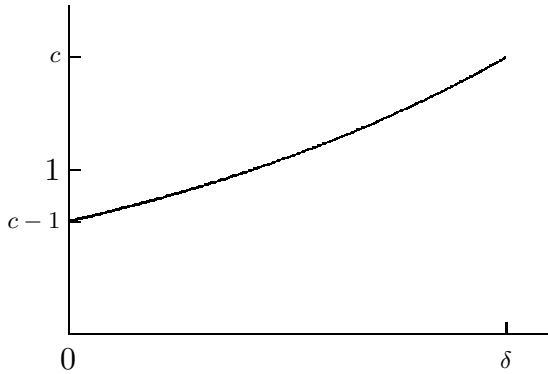


Fig. 4. The density function used for $1|r_j|\sum w_jC_j$ in Theorem 12

The bound in Theorem 12 yields also a bound on the quality of the relaxations (LP):

Corollary 5. *The relaxation (LP) is a 1.6853-relaxation of $1|r_j|\sum w_jC_j$.*

Before we prove Theorem 12 we state two properties of the density function f that are crucial for the analysis of the corresponding random α -schedule.

Lemma 13. *The function f given in Theorem 12 is a density function with the following properties:*

$$(i) \int_0^\eta f(\alpha)(1 + \alpha - \eta) d\alpha \leq (c - 1)\eta \quad \text{for all } \eta \in [0, 1],$$

$$(ii) (1 + E_f) \int_\mu^1 f(\alpha) d\alpha \leq c(1 - \mu) \quad \text{for all } \mu \in [0, 1],$$

where E_f denotes the expected value of a random variable α that is distributed according to f .

Property (i) is used to bound the delay to job j caused by jobs in J_1 which corresponds to the first summation on the right-hand side of (13). The second summation reflects the delay to job j caused by jobs in J_2 and will be bounded by property (ii).

Proof (of Lemma 13). A short computation shows that $\delta = \ln \frac{c}{c-1}$. The function f is a density function since

$$\int_0^1 f(\alpha) d\alpha = (c - 1) \int_0^\delta e^\alpha d\alpha = (c - 1) \left(\frac{c}{c-1} - 1 \right) = 1 .$$

In order to prove property (i), observe that for $\eta \in [0, \delta]$

$$\int_0^\eta f(\alpha)(1 + \alpha - \eta) d\alpha = (c - 1) \int_0^\eta e^\alpha(1 + \alpha - \eta) d\alpha = (c - 1)\eta .$$

For $\eta \in (\delta, 1]$ we therefore get

$$\int_0^\eta f(\alpha)(1 + \alpha - \eta) d\alpha < \int_0^\delta f(\alpha)(1 + \alpha - \delta) d\alpha = (c - 1)\delta < (c - 1)\eta .$$

In order to prove property (ii), we first compute

$$E_f = \int_0^1 f(\alpha)\alpha d\alpha = (c - 1) \int_0^\delta e^\alpha \alpha d\alpha = c\delta - 1 .$$

Property (ii) certainly holds for $\mu \in (\delta, 1]$. For $\mu \in [0, \delta]$ we get

$$\begin{aligned} (1 + E_f) \int_\mu^1 f(\alpha) d\alpha &= c\delta(c - 1) \int_\mu^\delta e^\alpha d\alpha \\ &= ce^{-\gamma}((2 - \gamma)e^\gamma - e^\mu) \\ &= c(2 - \gamma - e^{\mu - \gamma}) \\ &\leq c(2 - \gamma - (1 + \mu - \gamma)) \\ &= c(1 - \mu) . \end{aligned}$$

This completes the proof of the lemma. □

Proof (of Theorem 12). The analysis of the expected completion time of job j in the random α -schedule follows the line of argument developed in the proof of Theorem 11. First we consider a fixed choice of α_j and bound the corresponding conditional expectation $E_{\alpha_j}[C_j^\alpha]$. In a second step we bound the unconditional expectation $E[C_j^\alpha]$ by integrating the product $f(\alpha_j)E_{\alpha_j}[C_j^\alpha]$ over the interval $[0, 1]$.

For a fixed job j and a fixed value α_j , the bound in (13) and Lemma 13 (i) yield

$$\begin{aligned} E_{\alpha_j}[C_j^\alpha] &\leq C_j^P(0) + (c - 1) \sum_{k \in J_1} \eta_k p_k + \sum_{\substack{k \in J_2 \\ \alpha_j > \mu_k}} (1 + E_f) p_k + (1 + \alpha_j) p_j \\ &\leq cC_j^P(0) + (1 + E_f) \sum_{\substack{k \in J_2 \\ \alpha_j > \mu_k}} p_k + (1 + \alpha_j) p_j . \end{aligned}$$

The last inequality follows from (11). Using property (ii) and equation (12) yields

$$\begin{aligned} E[C_j^\alpha] &\leq cC_j^P(0) + (1 + E_f) \sum_{k \in J_2} p_k \int_{\mu_k}^1 f(\alpha_j) d\alpha_j + (1 + E_f) p_j \\ &\leq cC_j^P(0) + c \sum_{k \in J_2} (1 - \mu_k) p_k + cp_j = cC_j^{LP} . \end{aligned}$$

The result follows from linearity of expectations. □

One can further say that Algorithm RANDOM- α actually produces a schedule that is simultaneously expected to be near-optimal with respect to both the total weighted

completion time objective and the maximum completion time objective. Bicriteria results of similar spirit and also other results in this direction have been presented in [7, 11, 47].

Corollary 6. *Under the assumptions of Theorem 12, the expected makespan of the schedule constructed by Algorithm RANDOM- α is at most 1.5166 times the optimal makespan.*

Proof. The makespan of the preemptive list schedule P exactly equals the optimal makespan. Note that the expected makespan of the schedule constructed by Algorithm RANDOM- α can be bounded by the sum of the idle time that already existed in the preemptive schedule plus the idle time caused by jobs $k \in J$ plus their processing times. The corollary now immediately follows from the fact that the expected idle time caused by any job k is bounded by $E_f p_k \leq 0.5166 p_k$. \square

The performance of the 2-approximation algorithm with only one α given in Theorem 10 b) can also be improved through a more intricate density function; details can be found in [17].

Theorem 13. *If α is randomly chosen from $[0, 1]$ according to an appropriate truncated exponential density function, then Algorithm RANDOM- α achieves expected performance guarantee 1.7451. In particular, Algorithm BEST- α has performance ratio 1.7451.*

7.2 A 4/3-Approximation Algorithm for $1 \mid r_j, pmtn \mid \sum w_j C_j$

In this subsection we prove the following theorem of Schulz and Skutella [36].

Theorem 14. *Let the α_j 's be chosen from a probability distribution over $[0, 1]$ with the density function*

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

see Figure 5. Then, the expected completion time of every job j in the preemptive schedule constructed by the preemptive variant of Algorithm RANDOM- α is at most $4/3 C_j^{LP}$ and the expected performance guarantee of the preemptive variant of Algorithm RANDOM- α is $4/3$.

Notice that, in contrast to the non-preemptive case discussed in Theorem 12, we do not require the random variables $\alpha_j, j \in J$, to be independent but allow any correlation between them. In particular, the performance ratio of $4/3$ is achieved by Algorithm BEST- α and by Algorithm RANDOM- α when α is drawn from $[0, 1]$ with the density function given in Theorem 14. This result of Schulz and Skutella [36] improves upon a 1.466-approximation by Goemans, Wein, and Williamson [18]. They also analyzed Algorithm RANDOM- α with a density function similar to the one given in Theorem 14.

Again, the bound in Theorem 14 yields also a bound on the quality of the relaxations (LP):

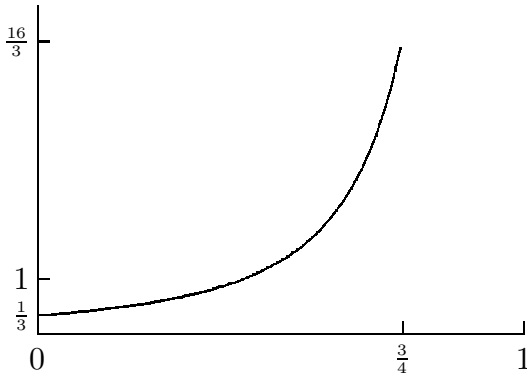


Fig. 5. The density function used for $1|r_j, pmtn | \sum w_j C_j$ in Theorem 14

Corollary 7. *The relaxation (LP) is a 4/3-relaxation of $1|r_j, pmtn | \sum w_j C_j$.*

Following the lines of the last subsection, we state two properties of the density function f that are crucial for the analysis of the corresponding preemptive random α -schedule.

Lemma 14. *The function f given in Theorem 14 is a density function with the following properties:*

- (i) $(1 - \eta) \int_0^\eta f(\alpha) d\alpha \leq \frac{1}{3} \eta$ for all $\eta \in [0, 1]$,
- (ii) $\int_\mu^1 f(\alpha) d\alpha \leq \frac{4}{3} (1 - \mu)$ for all $\mu \in [0, 1]$.

Similar to the situation in the last subsection, property (i) is used to bound the delay to job j caused by jobs in J_1 which corresponds to the first summation on the right-hand side of (14). The second summation reflects the delay to job j caused by jobs in J_2 and will be bounded by property (ii).

Proof (of Lemma 14). The function f is a density function since

$$\int_0^1 f(\alpha) d\alpha = \frac{1}{3} \int_0^{3/4} \frac{1}{(1 - \alpha)^2} d\alpha = 1 .$$

In order to prove property (i), observe that for $\eta \in [0, 3/4]$

$$(1 - \eta) \int_0^\eta f(\alpha) d\alpha = \frac{1}{3} (1 - \eta) \int_0^\eta \frac{1}{(1 - \alpha)^2} d\alpha = \frac{1}{3} \eta .$$

Since $f(\alpha) = 0$ if $\alpha > 3/4$, the bound also holds for $\eta > 3/4$. For the same reason, (ii) holds if $\mu > 3/4$. For $\mu \leq 3/4$ we get

$$\int_\mu^1 f(\alpha) d\alpha = \frac{1}{3} \int_\mu^{3/4} \frac{1}{(1 - \alpha)^2} d\alpha = \frac{4}{3} - \frac{1}{3} \frac{1}{1 - \mu} .$$

A short computation shows that the latter expression is bounded by $\frac{4}{3}(1 - \mu)$ which concludes the proof. \square

Proof (of Theorem 14). For a fixed job j , the bound in (14) and Lemma 14 yield

$$\begin{aligned} E[C_j^{\alpha-pmtn}] &\leq C_j^P(0) + \frac{1}{3} \sum_{k \in J_1} \eta_k p_k + \frac{4}{3} \sum_{k \in J_2} (1 - \mu_k) p_k + p_j \\ &\leq \frac{4}{3} C_j^P(0) + \frac{4}{3} \sum_{k \in J_2} (1 - \mu_k) p_k + \frac{4}{3} p_j = \frac{4}{3} C_j^{LP} . \end{aligned}$$

The second inequality follows from (11), the last equation follows from (12). This concludes the proof by linearity of expectations. \square

8 Approximations for Single Machine Scheduling with Precedence Constraints

In this section we present randomized approximation algorithms developed by Schulz and Skutella [35] for the scheduling problems $1|r_j, prec, pmtn|\sum w_j C_j$. The first constant-factor approximation algorithms for these problems have been given by Hall et al. [22]. They presented a $(4 + \epsilon)$ -approximation algorithm for $1|prec|\sum w_j C_j$ and a 5.83-approximation algorithm for $1|r_j, prec|\sum w_j C_j$. Their algorithms are based on the time-indexed LP relaxation (LP') and scheduling in order of α -points for a fixed value of α . However, their definition of α -points is slightly different from ours. As already mentioned in Section 6, Schulz [34] and Hall et al. [21] improved upon these results. Building upon the work of Sidney [41], combinatorial 2-approximation algorithms for $1|prec|\sum w_j C_j$ were given by Chekuri and Motwani [10] and Margot, Queyranne, and Wang [25] (see also [19]). Correa and Schulz [12] look at the problem from a polyhedral perspective and uncover a relation between the work of Sidney and different linear programming relaxations. Woeginger [51] discusses the approximability of $1|prec|\sum w_j C_j$ and presents approximation preserving reductions to special cases.

A straightforward combination of the 2-approximation algorithm for the preemptive scheduling problem $1|r_j, prec, pmtn|\sum w_j C_j$ [21] with the conversion technique of Chekuri et al. given in Theorem 4 achieves a performance guarantee of $\frac{2e}{e-1}$. In particular, it does not improve upon the 3-approximation for the non-preemptive variant $1|r_j, prec|\sum w_j C_j$ given by Schulz [34].

For an arbitrary α and a feasible fractional preemptive schedule, the order of α -points does in general not respect the precedence relations. Therefore, we only use one α for all jobs instead of individual α_j 's. Then, the corresponding α -schedule is feasible if the fractional preemptive schedule is feasible.

The first result discussed in this section is a $(2 + \epsilon)$ -approximation for the problems $1|prec|\sum w_j C_j$ and $1|r_j, prec, pmtn|\sum w_j C_j$. If we combine this algorithm with the conversion technique of Chekuri et al. in a somewhat more intricate way, we get a considerably improved approximation result for $1|r_j, prec|\sum w_j C_j$.

Again, we consider both the non-preemptive and the preemptive version of the algorithm:

Algorithm: RANDOM- α

- 1) Take a feasible solution to (LP) and the corresponding feasible fractional preemptive schedule P .
- 2) Draw α randomly from $[0, 1]$.
- 3) Output the resulting (preemptive) α -schedule.

The next theorem follows directly from Lemma 9.

Theorem 15. *Suppose that α is chosen from a probability distribution over $[0, 1]$ with density function $f(x) = 2x$. Then, for instances of $1|r_j, prec, pmtn | \sum w_j C_j$ and $1|prec | \sum w_j C_j$, the expected completion time of every job $j \in J$ in the schedule constructed by Algorithm RANDOM- α is bounded from above by twice its LP completion time C_j^{LP} .*

Proof. Lemma 9 and Lemma 10 a) yield for the preemptive and the non-preemptive case

$$E[C_j^\alpha] = \int_0^1 f(\alpha) \frac{1}{\alpha} C_j^P(\alpha) d\alpha = 2 \int_0^1 C_j^P(\alpha) d\alpha \leq 2 C_j^{LP} .$$

This concludes the proof. □

Goemans (personal communication, June 1996) applied the very same technique as in Theorem 15 to improve the performance guarantee of 4 due to Hall et al. [22] for $1|prec | \sum w_j C_j$ to a performance ratio of 2.

As a result of Theorem 15, (LP) is a 2-relaxation for the precedence constrained scheduling problems $1|r_j, prec, pmtn | \sum w_j C_j$ and $1|prec | \sum w_j C_j$, see Table 2.

Combining the idea of Chekuri et al. from Theorem 4 with the technique demonstrated in the proof of Theorem 15, we can prove the following theorem.

Theorem 16. *Suppose that α is chosen from a probability distribution over $[0, 1]$ with density function $f(x) = e - e^x$, see Figure 6. Then, for instances of the problem $1|r_j, prec | \sum w_j C_j$ the expected completion time of every job $j \in J$ in the schedule constructed by Algorithm RANDOM- α is bounded from above by $e C_j^{LP}$.*

Proof. Just for the analysis of the randomized algorithm we emulate the random choice of α in the following way: First draw a new random variable β from $[0, 1]$ with density function $h(x) = e \frac{e^x - 1}{e^x}$. Then, for fixed β , choose α from a probability distribution over the interval $[0, 1]$ with density function

$$g_\beta(x) = \begin{cases} \frac{e^x}{e^\beta - 1} & \text{if } x \in [0, \beta], \\ 0 & \text{otherwise.} \end{cases}$$

The resulting probability distribution of the random variable α is described by the density function f since

$$f(\alpha) = e - e^\alpha = \int_0^1 h(\beta) g_\beta(\alpha) d\beta .$$

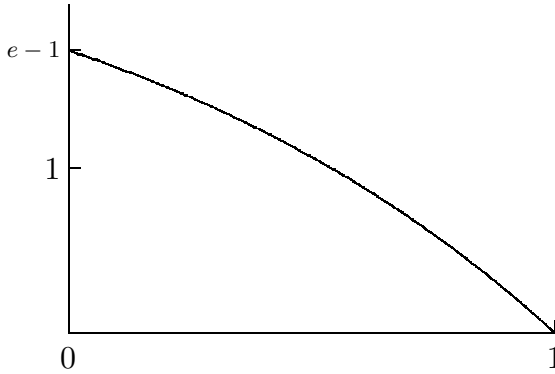


Fig. 6. The density function used for $1 | r_j, prec | \sum w_j C_j$ in Theorem 16

For fixed $\beta \in [0, 1]$ and fixed $\alpha \in [0, \beta]$ Corollary 1 and Lemma 6 yield

$$\begin{aligned} C_j^\alpha &\leq C_j^P(\alpha) + \sum_{\substack{k \\ \eta_k(\beta) \geq \alpha}} (1 + \alpha - \eta_k(\alpha)) p_k \\ &= C_j^P(\alpha) + \sum_{\substack{k \\ \eta_k(\beta) \geq \alpha}} (\eta_k(\beta) - \eta_k(\alpha)) p_k + \sum_{\substack{k \\ \eta_k(\beta) \geq \alpha}} (1 + \alpha - \eta_k(\beta)) p_k \\ &\leq C_j^P(\beta) + \sum_{\substack{k \\ \eta_k(\beta) \geq \alpha}} (1 + \alpha - \eta_k(\beta)) p_k . \end{aligned}$$

Since

$$\int_0^\eta g_\beta(x) (1 + \alpha - \eta) d\alpha \leq \frac{\eta}{e^\beta - 1}$$

for each $\eta \in [0, 1]$, the conditional expectation of j 's completion time for fixed β can be bounded by

$$\begin{aligned} E_\beta[C_j^\alpha] &\leq C_j^P(\beta) + \sum_k p_k \int_0^{\eta_k(\beta)} g_\beta(\alpha) (1 + \alpha - \eta_k(\beta)) d\alpha \\ &\leq C_j^P(\beta) + \frac{1}{e^\beta - 1} \sum_k \eta_k(\beta) p_k \leq \frac{e^\beta}{e^\beta - 1} C_j^P(\beta) . \end{aligned}$$

(Notice that for $\beta = 1$ this is precisely the result of Chekuri et al., see Theorem 4.) This yields

$$E[C_j^\alpha] = \int_0^1 h(\beta) E_\beta[C_j^\alpha] d\beta \leq e \int_0^1 C_j^P(\beta) d\beta \leq e C_j^{LP}$$

by Lemma 10 a).

□

As a result of Theorem 16, (LP) is an e -relaxation for the scheduling problem $1|r_j, prec|\sum w_j C_j$, see Table 2. Unfortunately, the results in Theorem 15 and Theorem 16 do not directly lead to approximation algorithms for the considered scheduling problems since we cannot solve (LP) in polynomial time. However, we can overcome this drawback by introducing new variables which are not associated with exponentially many time intervals of length 1, but rather with a polynomial number of intervals of geometrically increasing size (see [37] for details). In order to get polynomial-time approximation algorithms in this way, we have to pay for with a slightly worse performance guarantee. For any constant $\varepsilon > 0$ we get randomized approximation algorithms with performance guarantee $2 + \varepsilon$ respectively $e + \varepsilon$ for the scheduling problems under consideration. Again, those algorithms can be derandomized. For details we refer to [42, Chapter 2].

9 On-Line Results

In this section we consider a certain class of on-line scheduling problems. We show that the Algorithm $\text{RANDOM-}\alpha$, or a slightly modified version, also works in this on-line setting and the competitive ratios match the performance guarantees proved for the off-line variants. These observations are always due to the authors which also obtained the respective off-line approximation results discussed above.

In contrast to the off-line setting, an on-line algorithm does not get its entire input at one time, but receives it in partial amounts. This notion is intended to formalize the realistic scenario where we do not have access to the whole information about what will happen in the future. There are several different on-line paradigms that have been studied in the area of scheduling, see [39] for a survey. We consider the on-line setting, where jobs continually arrive over time and, for each time t , we must construct the schedule until time t without any knowledge of the jobs that will arrive afterwards. In particular, the characteristics of a job, i. e., processing time and weight are only known at its release date.

To measure the performance of a (randomized) on-line algorithm we compare the (expected) value of the schedule computed by the on-line algorithm to the value of an optimal schedule, i. e., we measure the (expected) performance of the on-line algorithms by an *oblivious adversary*, see [28, Chapter 13] for details. The worst case ratio of the two values is the *competitive ratio* of the on-line algorithm.

In order to apply $\text{RANDOM-}\alpha$ in the on-line setting we should first mention that for each job j its random variable α_j can be drawn immediately when the job is released since there is no interdependency with any other decisions of the randomized algorithms. Moreover, $\text{PREEMPTIVE LIST SCHEDULING}$ also works on-line if a job can be inserted at the correct position in the list with regard to the jobs that are already known as soon as it becomes available. In particular, the preemptive list schedule in order of non-decreasing ratios p_j/w_j can be constructed on-line in the first step of $\text{RANDOM-}\alpha$ 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 since the future development of the preemptive list schedule is not known.

However, the analysis of the non-preemptive variant of $\text{RANDOM-}\alpha$ still works if we schedule the jobs as early as possible in order of non-decreasing α -points, with the additional constraints that no job may start before its α -point. Notice that the non-preemptive schedule constructed by α -CONVERSION fulfills these constraints. The presented analysis of the non-preemptive variant of $\text{RANDOM-}\alpha$ in the proof of Theorem 12 relies on the bound given in Corollary 1 which also holds for the schedule constructed with α -CONVERSION by Lemma 7. Thus, we get an on-line variant of $\text{RANDOM-}\alpha$ with competitive ratio 1.6853 for $1|r_j|\sum w_j C_j$ if we replace the last step of the algorithm with the list scheduling routine described above.

This competitive ratio beats the deterministic on-line lower bound 2 for the weaker scheduling problem $1|r_j|\sum C_j$ [23]. Of course, against an oblivious adversary, a randomized algorithm can attain a substantially better competitiveness than any deterministic algorithm. The oblivious adversary cannot guess the random decisions and is therefore not able to adapt its own strategy completely to the behavior of the randomized on-line algorithm.

The deterministic 2-approximation algorithm of Phillips et al. for the scheduling problem $1|r_j|\sum C_j$ also works on-line and is therefore optimal. For the same problem Stougie and Vestjens proved the lower bound $\frac{e}{e-1}$ for randomized on-line algorithms [48, 50]. In particular, the on-line version of the algorithm of Chekuri et al. discussed in Section 4 is optimal. Recently, Anderson and Potts [3] gave a deterministic on-line algorithm for the problem $1|r_j|\sum w_j C_j$ with optimal competitive ratio 2.

The preemptive variant of $\text{RANDOM-}\alpha$ works without modifications in the on-line model. Notice that at any point in time and for an arbitrary pair of already available jobs we can predict whether $C_j(\alpha_j)$ will be smaller than $C_k(\alpha_k)$ 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 the first step, say j , will win since job k cannot be processed in the list schedule P before j is finished. This yields a randomized on-line approximation algorithm with competitive ratio $\frac{4}{3}$ for the problem $1|r_j, pmtn|\sum w_j C_j$. Notice that the (deterministic) list scheduling algorithm discussed in Lemma 5 also works on-line and has competitive ratio 2. On the other hand, Epstein and van Stee [14] recently obtained lower bounds of 1.073 and 1.038 for deterministic and randomized on-line algorithms, respectively.

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*, pages 32–43, New York City, NY, 1999.
2. J. M. van den Akker. *LP-Based Solution Methods for Single-Machine Scheduling Problems*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1994.
3. E. J. Anderson and C. N. Potts. On-line scheduling of a single machine to minimize total weighted completion time. *Mathematics of Operations Research*, 29:686–697, 2004.
4. K. R. Baker. *Introduction to Sequencing and Scheduling*. John Wiley & Sons, New York, 1974.

5. C. C. B. Cavalcante, C. C. de Souza, M. W. P. Savelsbergh, Y. Wang, and L. A. Wolsey. Scheduling projects with labor constraints. *Discrete Applied Mathematics*, 112:27–52, 2001.
6. S. Chakrabarti and S. Muthukrishnan. Resource scheduling for parallel database and scientific applications. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 329–335, Padua, Italy, 1996.
7. 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*, pages 646–657. Springer, Berlin, 1996.
8. L. M. A. Chan, A. Muriel, and D. Simchi-Levi. Parallel machine scheduling, linear programming and list scheduling heuristics. *Operations Research*, 46:729–741, 1998.
9. C. S. Chekuri, R. Johnson, R. Motwani, B. Natarajan, B. R. Rau, and M. Schlansker. Profile-driven instruction level parallel scheduling with applications to super blocks. In *Proceedings of the 29th Annual International Symposium on Microarchitecture*, pages 58–67, Paris, France, 1996.
10. C. S. Chekuri and R. Motwani. Precedence constrained scheduling to minimize weighted completion time on a single machine. *Discrete Applied Mathematics*, 98:29–38, 1999.
11. C. S. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. *SIAM Journal on Computing*, 31:146–166, 2001.
12. J. R. Correa and A. S. Schulz. Single machine scheduling with precedence constraints. In D. Bienstock and G. Nemhauser, editors, *Integer Programming and Combinatorial Optimization*, volume 3064 of *Lecture Notes in Computer Science*, pages 283–297. Springer, Berlin, 2004.
13. 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.
14. L. Epstein and R. van Stee. Lower bounds for on-line single machine scheduling. *Theoretical Computer Science*, 299:439–450, 2003.
15. 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*, pages 288–300. Springer, Berlin, 1996.
16. M. X. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 591–598, New Orleans, LA, 1997.
17. M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics*, 15:165–192, 2002.
18. M. X. Goemans, J. Wein, and D. P. Williamson. A 1.47-approximation algorithm for a preemptive single-machine scheduling problem. *Operations Research Letters*, 26:149–154, 2000.
19. M. X. Goemans and D. P. Williamson. Two-dimensional Gantt charts and a scheduling algorithm of Lawler. *SIAM Journal on Discrete Mathematics*, 13:281–294, 2000.
20. 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.
21. 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.
22. 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*, pages 142–151, Atlanta, GA, 1996.

23. 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*, pages 404–414. Springer, Berlin, 1996.
24. 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*, pages 245–261. Academic Press, New York, 1984.
25. F. Margot, M. Queyranne, and Y. Wang. Decomposition, network flows and a precedence constrained single machine scheduling problem. *Operations Research*, 51:981–992, 2003.
26. R. H. Möhring, A. S. Schulz, F. Stork, and M. Uetz. Solving project scheduling problems by minimum cut computations. *Management Science*, 49:330–350, 2003.
27. R. H. Möhring, A. S. Schulz, and M. Uetz. Approximation in stochastic scheduling: The power of LP-based priority policies. *Journal of the ACM*, 46:924–942, 1999.
28. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
29. C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82:199–223, 1998.
30. C. N. Potts. An algorithm for the single machine sequencing problem with precedence constraints. *Mathematical Programming Studies*, 13:78–87, 1980.
31. M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993.
32. M. W. P. Savelsbergh, R. N. Uma, and J. M. Wein. An experimental study of LP-based approximation algorithms for scheduling problems. In *Proceedings of the 9th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 453–462, San Francisco, CA, 1998.
33. A. S. Schulz. *Polytopes and scheduling*. PhD thesis, TU Berlin, Germany, 1996.
34. A. S. Schulz. Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*, pages 301–315. Springer, Berlin, 1996.
35. A. S. Schulz and M. Skutella. Random-based scheduling: New approximations and LP lower bounds. In J. Rolim, editor, *Randomization and Approximation Techniques in Computer Science*, volume 1269 of *Lecture Notes in Computer Science*, pages 119–133. Springer, Berlin, 1997.
36. A. S. Schulz and M. Skutella. The power of α -points in preemptive single machine scheduling. *Journal of Scheduling*, 5:121–133, 2002.
37. A. S. Schulz and M. Skutella. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15:450–469, 2002.
38. P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling*, 2:203–213, 1999.
39. J. Sgall. On-line scheduling — a survey. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 196–231. Springer, Berlin, 1998.
40. D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.
41. J. B. Sidney. Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research*, 23:283–298, 1975.
42. M. Skutella. *Approximation and randomization in scheduling*. PhD thesis, Technische Universität Berlin, Germany, 1998.
43. M. Skutella and M. Uetz. Scheduling precedence-constrained jobs with stochastic processing times on parallel machines. In *Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 589–590, Washington, DC, 2001.

44. M. Skutella and M. Uetz. Stochastic machine scheduling with precedence constraints. *SIAM Journal on Computing*, 2005. To appear.
45. W. E. Smith. Various optimizers for single-stage production. *Naval Research and Logistics Quarterly*, 3:59–66, 1956.
46. J. P. Sousa. *Time indexed formulations of non-preemptive single-machine scheduling problems*. PhD thesis, Université Catholique de Louvain, 1989.
47. C. Stein and J. Wein. On the existence of schedules that are near-optimal for both makespan and total weighted completion time. *Operations Research Letters*, 21:115–122, 1997.
48. L. Stougie and A. P. A. Vestjens. Randomized algorithms for on-line scheduling problems: How low can't you go? *Operations Research Letters*, 30:89–96, 2002.
49. R. N. Uma and J. M. Wein. On the relationship between combinatorial and LP-based approaches to NP-hard scheduling problems. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization*, volume 1412 of *Lecture Notes in Computer Science*, pages 394–408. Springer, Berlin, 1998.
50. A. P. A. Vestjens. *On-line machine scheduling*. PhD thesis, Eindhoven University of Technology, The Netherlands, 1997.
51. G. J. Woeginger. On the approximability of average completion time scheduling under precedence constraints. *Discrete Applied Mathematics*, 131:237–252, 2003.
52. L. A. Wolsey. Mixed integer programming formulations for production planning and scheduling problems, 1985. Invited talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge.