Single Machine Scheduling with Release Dates

Michel X. Goemans[*] Maurice Queyranne[†] Andreas S. Schulz[‡]
Martin Skutella[§] Yaoguang Wang[¶]

October 1999; revised August 2001

Abstract

We consider the scheduling problem of minimizing the average weighted completion time of n jobs with release dates on a single machine. We first study two linear programming relaxations of the problem, one based on a time-indexed formulation, the other on a completion-time formulation. We show their equivalence by proving that a $\mathrm{O}(n \log n)$ greedy algorithm leads to optimal solutions to both relaxations. The proof relies on the notion of mean busy times of jobs, a concept which enhances our understanding of these LP relaxations. Based on the greedy solution, we describe two simple randomized approximation algorithms, which are guaranteed to deliver feasible schedules with expected objective function value within factors of 1.7451 and 1.6853, respectively, of the optimum. They are based on the concept of common and independent α-points, respectively. The analysis implies in particular that the worst-case relative error of the LP relaxations is at most 1.6853, and we provide instances showing that it is at least $e/(e-1) \approx 1.5819$. Both algorithms may be derandomized; their deterministic versions run in $\mathrm{O}(n^2)$ time. The randomized algorithms also apply to the on-line setting, in which jobs arrive dynamically over time and one must decide which job to process without knowledge of jobs that will be released afterwards.

Keywords: approximation algorithm, LP relaxation, scheduling, on-line algorithm

AMS subject classification: 90C27, 68Q25, 90B35, 68M20

1 Introduction

We study the single-machine scheduling problem with release dates in which the objective is to minimize a weighted sum of completion times. It is defined as follows. A set $N = \{1, 2, \ldots, n\}$ of n jobs has to be scheduled on a single disjunctive machine. Job j has a processing time $p_j > 0$ and is released at time $r_j \geq 0$. We assume that release dates and processing times are integral. The completion time of job j in a schedule is denoted by C_j. The goal is to find a non-preemptive schedule that minimizes $\sum_{j \in N} w_j C_j$, where the w_j's are given positive weights.

[*]M.I.T., Department of Mathematics, Room 2-351, 77 Massachusetts Avenue, Cambridge, MA 02139, USA. Email: goemans@math.mit.edu

[†]University of British Columbia, Faculty of Commerce and Business Administration, Vancouver, B.C., Canada V6T 1Z2. Email: maurice.queyranne@commerce.ubc.ca

[‡]M.I.T., Sloan School of Management, Room E53-361, 77 Massachusetts Avenue, Cambridge, MA 02139, USA. Email: schulz@mit.edu

[§]Technische Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften, Institut für Mathematik, MA 6-1, Straße des 17. Juni 136, 10623 Berlin, Germany. Email: skutella@math.tu-berlin.de

[¶]PeopleSoft Inc., San Mateo, CA, USA 94404. Email: Yaoguang_Wang@peoplesoft.com

In the classical scheduling notation [12], this problem is denoted by $1|\, r_j \,|\sum w_j C_j$. It is strongly NP-hard, even if $w_j = 1$ for all jobs j [17].

One of the key ingredients in the design and analysis of approximation algorithms as well as in the design of implicit enumeration methods is the choice of a bound on the optimal value. Several linear programming based as well as combinatorial lower bounds have been proposed for this well studied scheduling problem, see for example, Dyer and Wolsey [9], Queyranne [22], and Queyranne and Schulz [23], as well as Belouadah, Posner and Potts [4]. The LP relaxations involve a variety of different types of variables which, e. g., either express whether job j is completed at time t (non-preemptive time-indexed relaxation), or whether it is being processed at time t (preemptive time-indexed relaxation), or when job j is completed (completion time relaxation). Dyer and Wolsey show that the non-preemptive time-indexed relaxation is stronger than the preemptive time-indexed relaxation. We will show that the latter relaxation is equivalent to the completion time relaxation that makes use of the so-called shifted parallel inequalities. In fact, it turns out that the polyhedron defined by these inequalities is supermodular and hence one can optimize over it by using the greedy algorithm. A very similar situation arises in [24]. The greedy solution may actually be interpreted in terms of the following preemptive schedule, which we call the LP schedule: at any point in time it schedules among the available jobs one with the largest ratio of weight to processing time. Uma and Wein [38] point out that the value of this LP solution coincides with one of the combinatorial bounds of Belouadah, Posner and Potts based on the idea of allowing jobs to be split into smaller pieces that can be scheduled individually.

We show that the optimal value of $1|\, r_j \,|\sum w_j C_j$ is at most 1.6853 times the lower bound given by any of these three equivalent relaxations — the preemptive time-indexed relaxation, the completion time relaxation or the combinatorial relaxation in [4]. We prove this result on the quality of these relaxations by converting the (preemptive) LP schedule into a non-preemptive schedule. This technique leads to approximation algorithms for $1|\, r_j \,|\sum w_j C_j$. 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. 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.

The technique of converting preemptive schedules to non-preemptive schedules in the design of approximation algorithms was introduced by Phillips, Stein and Wein [21]. More specifically, for $1|\, r_j \,|\sum w_j C_j$ they showed that list scheduling in order of the completion times of a given preemptive schedule produces a non-preemptive schedule while increasing the total weighted completion time by at most a factor of 2. In the same paper they also introduced a concept of α-points. This notion was also used by Hall, Shmoys and Wein [13], in connection with the non-preemptive time-indexed relaxation of Dyer and Wolsey to design approximation algorithms in various scheduling environments. For our purposes, the *α-point* of job j in a given preemptive schedule is the first point in time at which an α-fraction of j has been completed. When one chooses different values of α, sequencing in order of non-decreasing α-points in the same preemptive schedule may lead to different non-preemptive schedules. This increased flexibility led to improved approximation algorithms: Chekuri, Motwani, Natarajan and Stein [6] for $1|\, r_j \,|\sum C_j$ and Goemans [11] for $1|\, r_j \,|\sum w_j C_j$ chose α at random and analyzed the expected performance of the resulting randomized algorithms. We will show that, using a common value of α for all jobs and an appropriate probability distribution, sequencing in order of α-points of the LP schedule has expected performance no worse than 1.7451 times the optimal preemptive time-indexed LP value. We also prove that by selecting a separate value α_j for each job j, one can improve this bound to a factor of 1.6853. Our algorithms are inspired by and partly resemble the algorithms of Hall et al. [13] and Chekuri et al. [6]. In contrast to Hall et al.

Reference and/or type of schedule	Off-line deterministic	On-line randomized	On-line deterministic
Phillips et al. [21]	$16+\epsilon$		
Hall et al. [13]	4		$4+\epsilon$
Schulz [26]	3		
Hall et al. [14]	3		$3+\epsilon$
Chakrabarti et al. [5]	$2.8854+\epsilon$	$2.8854+\epsilon$	
Combining [5] and [14]	$2.4427+\epsilon$	$2.4427+\epsilon$	
α-schedule for $\alpha = 1/\sqrt{2}$ [11]	2.4143		2.4143
BEST α-schedule [11]	2 1.7451		
(random) (α_j)-schedule	1.6853	1.6853	

Table 1: Summary of approximation bounds for $1|\,r_j\,|\sum_j w_j C_j$. An *α-schedule* is obtained by sequencing the jobs in order of non-decreasing α-points of the LP schedule. The use of job-dependent α_j's yields an *(α_j)-schedule*. The results discussed in this paper are below the second double line. Subsequently, Anderson and Potts [2] gave a deterministic 2–competitive algorithm. For the unit-weight problem $1|\,r_j\,|\sum_j C_j$, the first constant-factor approximation algorithm is due to Phillips, Stein and Wein [21]. It has performance ratio 2, and it also works on-line. Further deterministic 2–competitive algorithms were given by Stougie [36] and Hoogeveen and Vestjens [15]. All these algorithms are optimal for deterministic on-line algorithms [15]. Chekuri, Motwani, Natarajan and Stein [6] gave a randomized $e/(e-1)$–approximation algorithm, which is optimal for randomized on-line algorithms [37, 39].

we exploit the preemptive time-indexed LP relaxation, which, on the one hand, provides us with highly structured optimal solutions and, on the other hand, enables us to work with mean busy times. We also use random α-points. The algorithm of Chekuri et al. starts from an arbitrary preemptive schedule and makes use of random α-points. They relate the value of the resulting α-schedule to that of the given preemptive schedule, and not to that of an underlying LP relaxation. While their approach gives better approximations for $1|\,r_j\,|\sum C_j$ and structural insights on limits of the power of preemption, the link of the LP schedule to the preemptive time-indexed LP relaxation helps us to obtain good approximations for the total weighted completion time.

Variants of our algorithms also work on-line when jobs arrive dynamically over time and, at each point in time, one has to decide which job to process without knowledge of jobs that will be released afterwards. Even in this on-line setting, we compare the value of the computed schedule to the optimal (off-line) schedule and derive the same bounds (competitive ratios) as in the off-line setting. See Table 1 for an account of the evolution of off-line and on-line approximation results for the single machine problem under consideration.

The main ingredient to obtain the results presented in this paper is the exploitation of the structure of the LP schedule. Not surprisingly, the LP schedule does not solve the strongly NP-hard [16] preemptive version of the problem, $1|\,r_j,\,pmtn\,|\sum w_j C_j$. However, we show that the LP schedule solves optimally the preemptive problem with the related objective function $\sum_j w_j M_j$, where M_j is the mean busy time of job j, i.e., the average point in time at which the machine is busy processing j. Observe that, although $1|\,r_j,\,pmtn\,|\sum w_j C_j$ and $1|\,r_j,\,pmtn\,|\sum w_j M_j$ are equivalent optimization problems in the non-preemptive case (since $C_j = M_j + \frac{p_j}{2}$), they are not when considering preemptive schedules.

The approximation techniques presented in this paper have also proved useful for more general scheduling problems. For the problem with precedence constraints $1|\, r_j,\, prec \,|\sum w_j C_j$, sequencing jobs in order of random α-points based on an optimal solution to a time-indexed LP relaxation leads to a 2.7183–approximation algorithm [27]. A 2–approximation algorithm for identical parallel machine scheduling $\mathrm{P}\,|\, r_j \,|\sum w_j C_j$ is given in [28]; the result is based on a time-indexed LP relaxation an optimal solution of which can be interpreted as a preemptive schedule on a fast single machine; jobs are then assigned randomly to the machines and sequenced in order of random α_j-points of this preemptive schedule. For the corresponding scheduling problem on unrelated parallel machines $\mathrm{R}\,|\, r_j \,|\sum w_j C_j$, a performance guarantee of 2 can be obtained by randomized rounding based on a convex quadratic programming relaxation [33], which is inspired by time-indexed LP relaxations like the one discussed herein [28]. We refer to [32] for a detailed discussion of the use of α-points for machine scheduling problems.

Significant progress has recently been made in understanding the approximability of scheduling problems with the average weighted completion time objective. Skutella and Woeginger [34] developed a polynomial-time approximation scheme for scheduling identical parallel machines in the absence of release dates, $\mathrm{P}\,|\,|\sum w_j C_j$. Subsequently, several research groups have found polynomial-time approximation schemes for problems with release dates such as $\mathrm{P}\,|\, r_j \,|\sum w_j C_j$ and $\mathrm{R}m\,|\, r_j \,|\sum w_j C_j$; see the resulting joint conference proceedings publication [1] for details.

We now briefly discuss some practical consequences of our work. Savelsbergh, Uma and Wein [25] and Uma and Wein [38] performed experimental studies to evaluate, in part, the quality of the LP relaxation and approximation algorithms studied herein, for $1|\, r_j \,|\sum w_j C_j$ and related scheduling problems. The first authors report that, except for instances that were deliberately constructed to be hard for this approach, the present formulation and algorithms "deliver surprisingly strong experimental performance." They also note that "the ideas that led to improved approximation algorithms also lead to heuristics that are quite effective in empirical experiments; furthermore [...] they can be extended to give improved heuristics for more complex problems that arise in practice." While the authors of the follow-up study [38] report that when coupled with local improvement the LP-based heuristics generally produce the best solutions, they also find that a simple heuristic often outperforms the LP-based heuristics. Whenever the machine becomes idle, this heuristic starts non-preemptively processing an available job of largest w_j/p_j ratio. By analyzing the differences between the LP schedule and this heuristic schedule, Chou, Queyranne and Simchi-Levi [7] have subsequently shown the asymptotic optimality of this on-line heuristic for classes of instances with bounded job weights and bounded processing times.

The contents of this paper are as follows. Section 2 is concerned with the LP relaxations and their relationship. We begin with a presentation and discussion of the LP schedule. In Section 2.1 we then review a time-indexed formulation introduced by Dyer and Wolsey [9] and show that it is solved to optimality by the LP schedule. In Section 2.2 we present the mean busy time relaxation (or completion time relaxation) and prove, among other properties, its equivalence to the time-indexed formulation. Section 2.3 explores some polyhedral consequences, in particular the fact that the mean busy time relaxation is (up to scaling by the job processing times) a supermodular linear program and that the "job-based" method for constructing the LP schedule is equivalent to the corresponding greedy algorithm. Section 3 then deals with approximation algorithms derived from these LP relaxations. In Section 3.1 we present a method for constructing (α_j)-schedules, which allows us to analyze and bound the job completion times in the resulting schedules. In Section 3.2 we derive simple bounds for α-schedules and (α_j)-schedules, using a deterministic common α or uniformly distributed random α_j's. Using appropriate probability distributions, we improve the approximation bound to the value of 1.7451 for α-schedules in Section 3.3 and to the value of 1.6853 for (α_j)-schedules in Section 3.4. We also indicate how these algorithms can be derandomized in $O(n^2)$ time for constructing deterministic schedules with these performance guarantees. In Section 3.5 we show that our randomized approximations also apply in an on-line setting and, in Section 3.6 we present a class of "bad" instances for which the ratio of the optimal objective function value and our LP bound is arbitrarily close to $\frac{e}{e-1} \approx 1.5819$. This constant defines a lower bound on the approximation results that can be obtained by the present approach. We conclude in Section 4 by discussing some related problems and open questions.

2 Relaxations

In this section, we present two linear programming relaxations for $1|\,r_j\,|\sum w_j C_j$. We show their equivalence and discuss some polyhedral consequences.

For both relaxations, the following preemptive schedule plays a crucial role: at any point in time, schedule (preemptively) the available job with highest w_j/p_j ratio. We assume (throughout the paper) that the jobs are indexed in order of non-increasing ratios $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \cdots \geq \frac{w_n}{p_n}$ and ties are broken according to this order. Therefore, whenever a job is released, the job being processed (if any) is preempted if the released job has a smaller index. We refer to this preemptive schedule as *the LP schedule*. See Figure 1 for an example of an LP schedule.

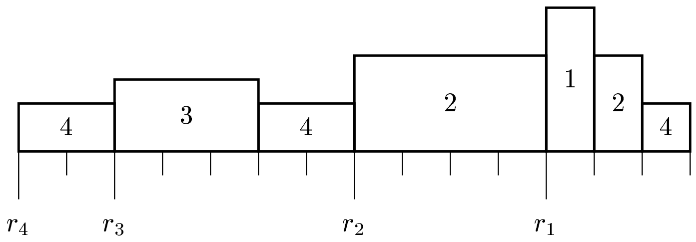

Figure 1: An LP schedule for a 4-job instance given by $r_1 = 11$, $p_1 = 1$, $r_2 = 7$, $p_2 = 5$, $r_3 = 2$, $p_3 = 3$, $r_4 = 0$, $p_4 = 5$. Higher rectangles represent jobs with larger weight to processing time ratio. Time is shown on the horizontal axis.

Notice that this LP schedule does not in general minimize $\sum_j w_j C_j$ over all preemptive schedules. This should not be surprising since the preemptive problem $1|\,r_j,\, pmtn\,|\sum w_j C_j$ is (strongly) NP-hard [16]. It can be shown, however, that the total weighted completion time of the LP schedule is always within a factor of 2 of the optimal value for $1|\,r_j,\, pmtn\,|\sum w_j C_j$ and this bound is tight; see [29].

The LP schedule can be constructed in $O(n \log n)$ time. To see this, we now describe an implementation, which may be seen as "dynamic" (event-oriented) or, using the terminology of [19], "machine-based" and can even be executed on-line while the jobs dynamically arrive over time. The algorithm keeps a priority queue [8] of the currently available jobs that have not yet been completely processed, with the ratio w_j/p_j as the key and with another field indicating the remaining processing time. A scheduling decision is made at only two types of events: when a job is released, and when a job completes its processing. In the former case, the released job is added to the priority queue. In the latter case, the completed job is removed from the priority queue. Then, in either case, the top element of the priority queue (the one with highest w_j/p_j ratio) is processed; if the queue is empty, then move on to the next job release; if there is none, then all jobs have been processed and the LP schedule is complete. This implementation results in a total of $O(n)$ priority queue operations. Since each such operation can be implemented in $O(\log n)$ time [8], the algorithm runs in $O(n \log n)$ time.

The LP schedule can also be defined in a somewhat different manner, which may be seen as "static" or "job-based" [19]. Consider the jobs one at a time in order of non-increasing w_j/p_j. Schedule each job j as early as possible starting at r_j and preempting it whenever the machine is busy processing another job (that thus came earlier in the w_j/p_j ordering). This point-of-view leads to an alternate $O(n \log n)$ construction of the LP schedule, see [10].

2.1 Time-Indexed Relaxation

Dyer and Wolsey [9] investigate several types of relaxations of $1|\, r_j \,|\sum w_j C_j$, the strongest ones being time-indexed. We consider the weaker of their two time-indexed formulations, which they call formulation (D). It uses two types of variables: $y_{j\tau} = 1$ if job j is being processed during time interval $[\tau, \tau+1)$, and zero otherwise; and t_j represents the start time of job j. For simplicity, we add p_j to t_j and replace the resulting expression by C_j; this gives an equivalent relaxation.

$$
\begin{array}{llll}
 & Z_D = \min & \displaystyle\sum_{j \in N} w_j C_j & \\
 & \text{subject to} & & \\
(D) & & \displaystyle\sum_{j: r_j \le \tau} y_{j\tau} \le 1 & \tau = 0, 1, \ldots, T-1 \\
 & & \displaystyle\sum_{\tau = r_j}^{T-1} y_{j\tau} = p_j & j \in N \\
 & & C_j = \dfrac{1}{2} p_j + \dfrac{1}{p_j} \displaystyle\sum_{\tau = r_j}^{T-1} \left(\tau + \dfrac{1}{2}\right) y_{j\tau} & j \in N \qquad (1) \\
 & & 0 \le y_{j\tau} & j \in N, \tau = r_j, r_j+1, \ldots, T-1
\end{array}
$$

where T is an upper bound on the makespan of an optimal schedule (for example, $T = \max_{j \in N} r_j + \sum_{j \in N} p_j$). We refer to this relaxation as the *preemptive time-indexed relaxation*. The expression for C_j given in (1) corresponds to the correct value of the completion time if job j is not preempted; an interpretation in terms of mean busy times is given in the next section for the case of preemptions. Observe that the number of variables of this formulation is pseudo-polynomial. If we eliminate C_j from the relaxation by using (1), we obtain a transportation problem [9] and, as a result, $y_{j\tau}$ can be assumed to be integral:

Lemma 2.1. *There exists an optimal solution to (D) for which $y_{j\tau} \in \{0, 1\}$ for all j and τ.*

As indicated in [9], (D) can be solved in $O(n \log n)$ time. Actually, one can derive a feasible solution to (D) from the LP schedule by letting $y_{j\tau}^{LP}$ be equal to 1 if job j is being processed in $[\tau, \tau+1)$, and 0 otherwise.

Theorem 2.2. *The solution y^{LP} derived from the LP schedule is an optimal solution to (D).*

Proof. The proof is based on an interchange argument. Consider any optimal 0/1-solution y^* to (D). If there exist $j < k$ and $\sigma > \tau \ge r_j$ such that $y_{j\sigma}^* = y_{k\tau}^* = 1$, then by replacing $y_{j\sigma}^*$ and $y_{k\tau}^*$ by 0, and $y_{j\tau}^*$ and $y_{k\sigma}^*$ by 1, we obtain another feasible solution with an increase in the objective function value of $(\sigma - \tau)\left(\frac{w_k}{p_k} - \frac{w_j}{p_j}\right) \le 0$. The resulting solution must therefore also be optimal. By repeating this interchange argument, we derive that there exists an optimal solution y^* such that there do not exist $j < k$ and $\sigma > \tau \ge r_j$ such that $y_{j\sigma}^* = y_{k\tau}^* = 1$. This implies that the solution y^* must correspond to the LP schedule. $\square$

In particular, despite the pseudo-polynomial number of variables in the LP relaxation (D) an optimal solution can be obtained efficiently. We will make use of this fact as well as of the special structure of the LP schedule in the design and analysis of the approximation algorithms, see Section 3. We note again that in spite of its nice properties the preemptive time-indexed LP relaxation (D) solves neither $1|\,r_j\,|\sum w_j C_j$ nor $1|\,r_j,\, pmtn\,|\sum w_j C_j$. In the former case, the processing of a job in the LP solution may fail to be consecutive; in the latter case equation (1) does not necessarily define the completion time of a job in the preemptive LP schedule, as will be shown in the next lemma.

2.2 Mean Busy Time Relaxation

Given any preemptive schedule, let I_j be the indicator function of the processing of job j at time t, i.e., $I_j(t)$ is 1 if the machine is processing j at time t and 0 otherwise. To avoid pathological situations, we require that, in any preemptive schedule, when the machine starts processing a job, it does so for a positive amount of time. Given any preemptive schedule, we define the *mean busy time* M_j of job j to be the average time at which the machine is processing j, i.e.,

$$M_j := \frac{1}{p_j} \int_{r_j}^{T} I_j(t)\, t\, dt \quad .$$

For instance, in the example given in Figure 1, which will be used throughout this paper, the mean busy time of job 4 is 5.5.

We first establish some important properties of M_j in the next two lemmas.

Lemma 2.3. *For any preemptive schedule, let C_j and M_j denote the completion and mean busy time, respectively, of job j. Then for any job j, we have $M_j + \frac{1}{2}p_j \le C_j$, with equality if and only if job j is not preempted.*

Proof. If job j is processed without preemption, then $I_j(t) = 1$ if $C_j - p_j \le t < C_j$ and 0 otherwise; therefore $M_j + \frac{1}{2}p_j = C_j$. Otherwise job j is not processed during some interval(s) of total length $L > 0$ between times $C_j - p_j$ and C_j. Since $\int_{r_j}^{T} I_j(t)\, dt = p_j$, job j must be processed during some time interval(s) of the same total length L *before* $C_j - p_j$. Therefore,

$$M_j = \frac{1}{p_j} \int_{r_j}^{C_j} I_j(t)\, t\, dt < \frac{1}{p_j} \int_{C_j - p_j}^{C_j} t\, dt = C_j - \frac{1}{2}p_j$$

and the proof is complete. □

Let $S \subseteq N$ denote a set of jobs and define

$$p(S) := \sum_{j \in S} p_j \qquad \text{and} \qquad r_{\min}(S) := \min_{j \in S} r_j \quad .$$

Let $I_S(t) := \sum_{j \in S} I_j(t)$. Since, by the machine capacity constraint, $I_S(t) \in \{0, 1\}$ for all t, we may view I_S as the indicator function for job set S. We can thus define the *mean busy time of set S* as $M_S := \frac{1}{p(S)} \int_0^T I_S(t)\, t\, dt$. Note that we have

$$p(S) M_S = \int_0^T \left(\sum_{j \in S} I_j(t) \right) t\, dt = \sum_{j \in S} \int_0^T I_j(t)\, t\, dt = \sum_{j \in S} p_j M_j \quad . \tag{2}$$

So, unlike its start and completion time, the mean busy time of a job set is a simple weighted average of the mean busy times of its elements. One consequence of this observation is the validity of the *shifted parallel inequalities* (3) (see, e.g., [10, 23, 24]) below.

Lemma 2.4. *For any set S of jobs and any preemptive schedule with mean busy time vector M, we have*

$$\sum_{j \in S} p_j M_j \geq p(S)\left(r_{\min}(S) + \frac{1}{2}p(S)\right) \tag{3}$$

and equality holds if and only if all the jobs in S are scheduled without interruption from $r_{\min}(S)$ to $r_{\min}(S) + p(S)$.

Proof. Note that $\sum_{j \in S} p_j M_j = p(S) M_S = \int_{r_{\min}(S)}^{T} I_S(t)\, t\, dt$; that $I_S(t) = 0$ for $t < r_{\min}(S)$ and $I_S(t) \leq 1$ for $t \geq r_{\min}(S)$; and that $\int_{r_{\min}(S)}^{T} I_S(t)\, dt = p(S)$. Under these constraints, M_S is minimized when $I_S(t) = 1$ for $r_{\min}(S) \leq t < r_{\min}(S) + p(S)$ and 0 otherwise. Therefore M_S is uniquely minimized among all feasible preemptive schedules when all the jobs in S are continuously processed from $r_{\min}(S)$ to $r_{\min}(S) + p(S)$. This minimum value is $p(S)(r_{\min}(S) + \frac{1}{2}p(S))$ and is a lower bound for $\sum_{j \in S} p_j M_j$ in any feasible preemptive schedule. □

As a result of Lemma 2.4, the following linear program provides a lower bound on the optimal value of $1|\, r_j,\, pmtn \,|\sum w_j C_j$, and hence on that of $1|\, r_j \,|\sum w_j C_j$.

$$Z_R \;=\; \min \sum_{j \in N} w_j \left(M_j + \frac{1}{2}p_j\right)$$

subject to

$$(R) \qquad \sum_{j \in S} p_j M_j \geq p(S)\left(r_{\min}(S) + \frac{1}{2}p(S)\right) \qquad S \subseteq N.$$

The proof of the following theorem and later developments use the notion of canonical decompositions [10]. For a set S of jobs, consider the schedule which processes jobs in S as early as possible, say, in order of their release dates. This schedule induces a partition of S into $\{S_1, \ldots, S_k\}$ such that the machine is busy processing jobs in S exactly in the disjoint intervals $[r_{\min}(S_\ell), r_{\min}(S_\ell) + p(S_\ell)]$ for $\ell = 1, \ldots, k$. We refer to this partition as the *canonical decomposition* of S. A set is *canonical* if it is identical to its canonical decomposition, i. e., if its canonical decomposition is $\{S\}$. Thus a set S is canonical if and only if it is feasible to schedule all its jobs in the time interval $[r_{\min}(S), r_{\min}(S) + p(S))$. Note that the set $N = \{1, 2, 3, 4\}$ in our example is canonical whereas the subset $\{1, 2, 3\}$ is not; it has the decomposition $\{\{3\}, \{1, 2\}\}$. Let

$$h(S) := \sum_{\ell=1}^{k} p(S_\ell)\left(r_{\min}(S_\ell) + \frac{1}{2}p(S_\ell)\right) \tag{4}$$

where $\{S_1, \ldots, S_k\}$ is the canonical decomposition of $S \subseteq N$. Then Lemma 2.4 implies that $\sum_{j \in S} p_j M_j = \sum_{\ell=1}^{k} \sum_{j \in S_\ell} p_j M_j \geq h(S)$ is a valid inequality for the mean busy time vector of any preemptive schedule. In other words, relaxation (R) may be written as:

$$\min \left\{ \sum_{j \in N} w_j \left(M_j + \frac{1}{2}p_j\right) \;:\; \sum_{j \in S} p_j M_j \geq h(S) \text{ for all } S \subseteq N \right\}.$$

Theorem 2.5. *Let M_j^{LP} be the mean busy time of job j in the LP schedule. Then M^{LP} is an optimal solution to (R).*

Proof. By Lemma 2.4, M^{LP} is a feasible solution for (R).

To prove optimality of M^{LP}, we construct a lower bound on the optimal value of (R) and show that it is equal to the objective function value of M^{LP}. Recall that the jobs are indexed in non-increasing order of the w_j/p_j ratios; let $[i] := \{1, 2, \ldots, i\}$, and let $S^i_1, \ldots, S^i_{k(i)}$ denote the canonical decomposition of $[i]$. Observe that for any vector $M = (M_j)_{j\in N}$ we have

$$\sum_{j\in N} w_j M_j = \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}}\right) \sum_{j\in[i]} p_j M_j = \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}}\right) \sum_{\ell=1}^{k(i)} \sum_{j\in S^i_\ell} p_j M_j \tag{5}$$

where we let $w_{n+1}/p_{n+1} := 0$. We have therefore expressed $\sum_{j\in N} w_j M_j$ as a nonnegative combination of expressions $\sum_{j\in S^i_\ell} p_j M_j$ over canonical sets. By construction of the LP schedule, the jobs in any of these canonical sets S^i_ℓ are continuously processed from $r_{\min}(S^i_\ell)$ to $r_{\min}(S^i_\ell) + p(S^i_\ell)$ in the LP schedule. Thus, for any feasible solution M to (R) and any such canonical set S^i_ℓ we have

$$\sum_{j\in S^i_\ell} p_j M_j \geq h(S^i_\ell) = p(S^i_\ell)\left(r_{\min}(S^i_\ell) + \frac{1}{2}p(S^i_\ell)\right) = \sum_{j\in S^i_\ell} p_j M^{LP}_j$$

where the last equation follows from Lemma 2.4. Combining this with (5), we derive a lower bound on $\sum_j w_j M_j$ for any feasible solution M to (R), and this lower bound is attained by the LP schedule. □

¿From Theorems 2.2 and 2.5, we derive that the values of the two relaxations (D) and (R) are equal.

Corollary 2.6. *The LP relaxations (D) and (R) of $1|\,r_j\,|\sum w_j C_j$ yield the same optimal objective function value, i. e., $Z_D = Z_R$, for any weights $w \geq 0$. This value can be computed in $O(n \log n)$ time.*

Proof. For the equivalence of the lower bounds, note that the mean busy time M^{LP}_j of any job j in the LP schedule can be expressed as

$$M^{LP}_j = \frac{1}{p_j} \sum_{\tau=r_j}^{T-1} y^{LP}_{j\tau}\left(\tau + \frac{1}{2}\right) , \tag{6}$$

where y^{LP} is the solution to (D) derived from the LP schedule. The result then follows directly from Theorems 2.2 and 2.5. We have shown earlier that the LP schedule can be constructed in $O(n \log n)$ time. □

Although the LP schedule does not necessarily minimize $\sum_j w_j C_j$ over the preemptive schedules, Theorem 2.5 implies that it minimizes $\sum_j w_j M_j$ over the preemptive schedules. In addition, by Lemma 2.3, the LP schedule is also optimal for both preemptive and non-preemptive problems $1|\,r_j, pmtn\,|\sum w_j C_j$ and $1|\,r_j\,|\sum w_j C_j$ whenever it does not preempt any job. For example, this is the case if all processing times are equal to 1 or if all jobs are released at the same date. Thus, the LP schedule provides an optimal solution to problems $1|\,r_j, p_j = 1\,|\sum w_j C_j$ and to $1|\,|\sum w_j C_j$. This was already known. In the latter case it coincides with Smith's ratio rule [35]; see Queyranne and Schulz [24] for the former case.

2.3 Polyhedral Consequences

We now consider some polyhedral consequences of the preceding results. Let P_D^∞ be the feasible region defined by the constraints of relaxation (D) when $T = \infty$, i.e.,

$$P_D^\infty := \Big\{ y \geq 0 : \sum_{j : r_j \leq \tau} y_{j\tau} \leq 1 \text{ for } \tau \in \mathbb{N}; \ \sum_{\tau \geq r_j} y_{j\tau} = p_j \text{ for all } j \in N \Big\} \ .$$

In addition, we denote by $P_R := \{M \in \mathbb{R}^N : \sum_{j \in S} p_j M_j \geq h(S) \text{ for all } S \subseteq N\}$ the polyhedron defined by the constraints of relaxation (R).

Theorem 2.7.

(i) *Polyhedron P_R is the convex hull of the mean busy time vectors M of all preemptive schedules. Moreover, every vertex of P_R is the mean busy time vector of an LP schedule.*

(ii) *Polyhedron P_R is also the image of P_D^∞ in the space of the M-variables under the linear mapping $\mathcal{M} : y \mapsto \mathcal{M}(y) \in \mathbb{R}^N$ defined by*

$$\mathcal{M}(y)_j = \frac{1}{p_j} \sum_{\tau \geq r_j} y_{j\tau} \left(\tau + \frac{1}{2} \right) \qquad \text{for all } j \in N.$$

Proof. (i) Lemma 2.4 implies that the convex hull of the mean busy time vectors M of all feasible preemptive schedules is contained in P_R. To show the reverse inclusion, it suffices to show that (a) every extreme point of P_R corresponds to a preemptive schedule; and (b) every extreme ray of P_R is a direction of recession for the convex hull of mean busy time vectors. Property (a) and the second part of statement (i) follow from Theorem 2.5 and the fact that every extreme point of P_R is the unique minimizer of $\sum_{j \in N} w_j M_j$ for some $w \geq 0$. For (b), note that the extreme rays of P_R are the n unit vectors of $\mathbb{R}^N$. An immediate extension to preemptive schedules and mean busy times of results in Balas [3] implies that these unit vectors of $\mathbb{R}^N$ are directions of recession for the convex hull of mean busy time vectors. This completes the proof of (i).

(ii) We first show that the image $\mathcal{M}(P_D^\infty)$ of P_D^∞ is contained in P_R. For this, let y be a vector in P_D^∞ and $S \subseteq N$ with canonical decomposition $\{S_1, \ldots, S_k\}$. By definition of $\mathcal{M}(y)_j$, we have

$$\begin{aligned}
\sum_{j \in S} p_j \mathcal{M}(y)_j &= \sum_{j \in S} \sum_{\tau \geq r_j} y_{j\tau} \left(\tau + \frac{1}{2} \right) \\
&\geq \sum_{\ell=1}^{k} \sum_{\tau = r_{\min}(S_\ell)}^{r_{\min}(S_\ell) + p(S_\ell) - 1} \left(\tau + \frac{1}{2} \right) \\
&= \sum_{\ell=1}^{k} p(S_\ell) \left(r_{\min}(S_\ell) + \frac{1}{2} p(S_\ell) \right) = h(S) \ .
\end{aligned}$$

The inequality follows from the constraints defining P_D^∞ and the interchange argument which we already used in the proof of Theorem 2.2. This shows $\mathcal{M}(y) \in P_R$ and thus $\mathcal{M}(P_D^\infty) \subseteq P_R$.

To show the reverse inclusion, we use the observation from the proof of part (i) that P_R can be represented as the sum of the convex hull of the mean busy time vectors of all LP schedules and the nonnegative orthant. Since, by equation (6), the mean busy time vector M^{LP} of any LP schedule is the projection of the corresponding 0/1-vector y^{LP}, it remains to show that every unit vector e_j is a direction of recession for $\mathcal{M}(P_D^\infty)$. For this, fix an LP schedule and let y^{LP} and

$M^{LP} = \mathcal{M}(y^{LP})$ denote the associated $0/1$ y-vector and mean busy time vector, respectively. For any job $j \in N$ and any real $\lambda > 0$, we need to show that $M^{LP} + \lambda\, e_j \in \mathcal{M}(P_D^\infty)$.

Let $\tau_{\max} := \operatorname{argmax}\{y_{k\tau}^{LP} = 1 : k \in N\}$. Choose θ such that $y_{j\theta}^{LP} = 1$, choose an integer $\mu > \max\{\lambda\, p_j, \tau_{\max} - \theta\}$ and define y' by $y'_{j\theta} = 0$, $y'_{j,\theta+\mu} = 1$, and $y'_{k\tau} = y_{k\tau}^{LP}$ otherwise. In the associated preemptive schedule, the processing of job j that was done in interval $[\theta, \theta+1)$ is now postponed, by μ time units, until interval $[\theta+\mu, \theta+\mu+1)$. Therefore, its mean busy time vector $M' = \mathcal{M}(y')$ satisfies $M'_j = M_j^{LP} + \mu/p_j$ and $M'_k = M_k^{LP}$ for all $k \neq j$. Let $\lambda' := \mu/p_j \geq \lambda$, so $M' = M^{LP} + \lambda' e_j$. Then the vector $M^{LP} + \lambda\, e_j$ is a convex combination of $M^{LP} = \mathcal{M}(y^{LP})$ and $M' = \mathcal{M}(y')$. Let y be the corresponding convex combination of y^{LP} and y'. Since P_D^∞ is convex then $y \in P_D^\infty$ and, since the mapping $\mathcal{M}$ is linear, $M^{LP} + \lambda\, e_j = \mathcal{M}(y) \in \mathcal{M}(P_D^\infty)$. □

In view of earlier results for single machine scheduling with identical release dates [22], as well as for parallel machine scheduling with unit processing times and integer release dates [24], it is interesting to note that the feasible set P_R of the mean busy time relaxation is, up to scaling by the job processing times, a supermodular polyhedron:

Proposition 2.8. *The set function h defined in* (4) *is supermodular.*

Proof. Consider any two elements j, $k \in N$ and any subset $S \subseteq N \setminus \{j, k\}$. We may construct an LP schedule minimizing $\sum_{i \in S \cup \{k\}} p_i M_i$ using the job-based method by considering first the jobs in S and then job k. (Note that considering the jobs in any sequence leads to a schedule minimizing $\sum_i p_i M_i$ because jobs are weighted by their processing times in this objective function). By definition (4) the resulting mean busy times M^{LP} satisfy $\sum_{i\in S} p_i M_i^{LP} = h(S)$ and $\sum_{i \in S\cup\{k\}} p_i M_i^{LP} = h(S \cup \{k\})$. Note that job k is scheduled, no earlier than its release date, in the first p_k units of idle time left after the insertion of all jobs in S. Thus M_k^{LP} is the mean of all these p_k time units. Similarly, we may construct an LP schedule, whose mean busy time vector will be denoted by $\widetilde{M}^{LP}$, minimizing $\sum_{i \in S\cup\{j,k\}} p_i M_i$ by considering first the jobs in S, so $\widetilde{M}_i^{LP} = M_i^{LP}$ for all $i \in S$; then job j, so $\sum_{i\in S\cup\{j\}} p_i \widetilde{M}_i^{LP} = h(S \cup \{j\})$; and then job k, so $\sum_{i \in S \cup\{j,k\}} p_i \widetilde{M}_i^{LP} = h(S \cup \{j, k\})$. Since job j has been inserted after subset S was scheduled, job k cannot use any idle time interval that is earlier than those it used in the former LP schedule M^{LP}—and some of the previously available idle time may now be occupied by job j, causing a delay in the mean busy time of job k; thus we have $\widetilde{M}_k^{LP} \geq M_k^{LP}$ and therefore

$$h(S \cup \{j, k\}) - h(S \cup \{j\}) = p_k \widetilde{M}_k^{LP} \geq p_k M_k^{LP} = h(S \cup \{k\}) - h(S).$$

This suffices to establish that h is supermodular. □

An alternate proof of the supermodularity of h can be derived, as in [10], from the fact, observed by Dyer and Wolsey and already mentioned above, that relaxation (D) becomes a transportation problem after elimination of the C_j's. Indeed, from an interpretation of Nemhauser, Wolsey and Fisher [20] of a result of Shapley [31], it then follows that the value of this transportation problem as a function of S is supermodular. One of the consequences of Proposition 2.8 is that the job-based method to construct an LP schedule is just a manifestation of the greedy algorithm for minimizing $\sum_{j\in N} w_j M_j$ over the supermodular polyhedron P_R.

We finally note that the separation problem for the polyhedron P_R can be solved combinatorially. One can separate over the family of inequalities $\sum_{j\in S} p_j M_j \geq p(S)(r_{\min}(S) + p(S))$ by trying all possible values for $r_{\min}(S)$ (of which there are at most n) and then applying a $\mathrm{O}(n \log n)$ separation routine of Queyranne [22] for the problem without release dates. The overall separation routine can be implemented in $\mathrm{O}(n^2)$ time by observing that the bottleneck step in Queyranne's algorithm — sorting the mean busy times of the jobs — needs to be done only once for the whole job set.

3 Provably Good Schedules and LP Relaxations

In this section, we derive approximation algorithms for $1|\,r_j\,|\sum w_j C_j$ that are based on converting the preemptive LP schedule into a feasible non-preemptive schedule whose value can be bounded in terms of the optimal LP value $Z_D = Z_R$. This yields results on the quality of both the computed schedule and the LP relaxations under consideration since the value of the computed schedule is an upper bound and the optimal LP value is a lower bound on the value of an optimal schedule.

In Section 3.6 below, we describe a family of instances for which the ratio between the optimal value of the $1|\,r_j\,|\sum w_j C_j$ problem and the lower bounds Z_R and Z_D is arbitrarily close to $\frac{e}{e-1} > 1.5819$. This lower bound of $\frac{e}{e-1}$ sets a target for the design of approximation algorithms based on these LP relaxations.

In order to convert the preemptive LP schedule into a non-preemptive schedule we make use of so-called α-points of jobs. For $0 < \alpha \le 1$ the *α-point* $t_j(\alpha)$ of job j is the first point in time when an α-fraction of job j has been completed in the LP schedule, i. e., when j has been processed for $\alpha\, p_j$ time units. In particular, $t_j(1)$ is equal to the completion time and we define $t_j(0^+)$ to be the start time of job j. Notice that, by definition, the mean busy time M_j^{LP} of job j in the LP schedule is the average over all its α-points

$$M_j^{LP} \;=\; \int_0^1 t_j(\alpha)\; d\alpha \;. \tag{7}$$

We will also use the following notation: For a fixed job j and $0 < \alpha \le 1$ we denote the fraction of job k that is completed in the LP schedule by time $t_j(\alpha)$ by $\eta_k(\alpha)$; in particular, $\eta_j(\alpha) = \alpha$. The amount of idle time that occurs between time 0 and the start of job j in the LP schedule is denoted by τ_{idle}. Note that η_k and τ_{idle} implicitly depend on the fixed job j. By construction, there is no idle time between the start and completion of job j in the LP schedule; therefore we can express j's α-point as

$$t_j(\alpha) \;=\; \tau_{idle} \;+\; \sum_{k\in N} \eta_k(\alpha) p_k \;. \tag{8}$$

For a given $0 < \alpha \le 1$, we define the α-schedule as the schedule in which jobs are processed non-preemptively as early as possible and in the order of non-decreasing α-points. We denote the completion time of job j in this schedule by C_j^α. The idea of scheduling non-preemptively in the order of α-points in a preemptive schedule was introduced by Phillips, Stein and Wein [21], and used in many of the subsequent results in the area.

This idea can be further extended to individual, i. e., job-dependent *α_j-points* $t_j(\alpha_j)$, for $j \in N$ and $0 < \alpha_j \le 1$. We denote the vector consisting of all α_j's by $\boldsymbol{\alpha} := (\alpha_j) := (\alpha_1, \dots, \alpha_n)$. Then, the *$(\alpha_j)$-schedule* is constructed by processing the jobs as early as possible and in non-decreasing order of their α_j-points; the completion time of job j in the (α_j)-schedule is denoted by $C_j^{\boldsymbol{\alpha}}$. Figure 2 compares an α-schedule to an (α_j)-schedule both derived from the LP schedule in Figure 1.

In the sequel we present several results on the quality of α-schedules and (α_j)-schedules. These results also imply bounds on the quality of the LP relaxations of the previous section. The main result is the construction of a random (α_j)-schedule whose expected value is at most a factor 1.6853 of the optimal LP value $Z_D = Z_R$. Therefore the LP relaxations (D) and (R) deliver a lower bound which is at least 0.5933 ($\approx 1.6853^{-1}$) times the optimal value. The corresponding randomized algorithm can be implemented on-line; it has competitive ratio 1.6853 and running time $O(n \log n)$; it can also be derandomized to run off-line in $O(n^2)$ time. We also investigate the case of a single common α and show that the best α-schedule is always within a factor of 1.7451 of the optimum.

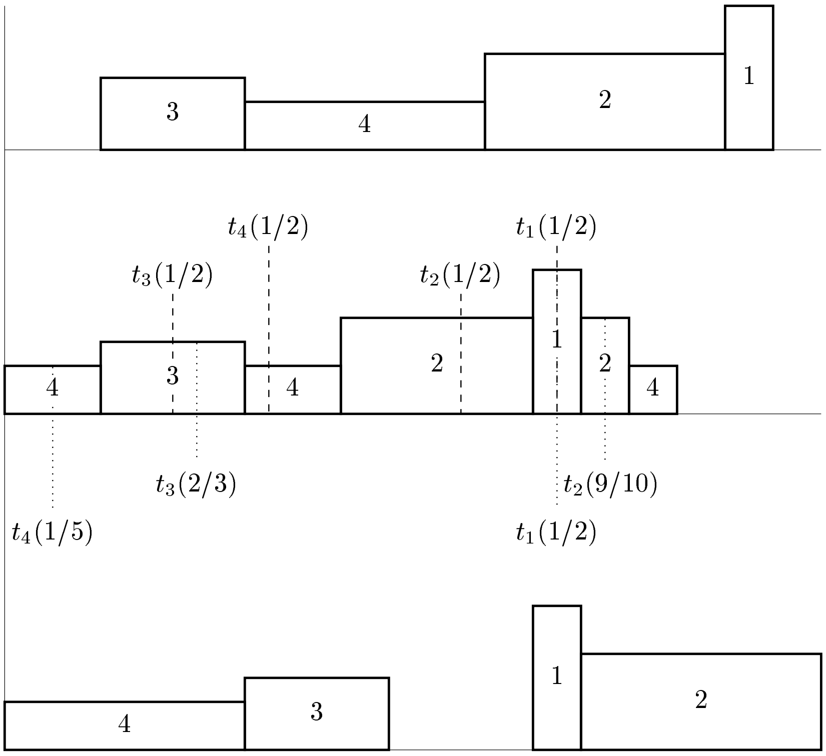

Figure 2: A non-preemptive α-schedule (for $\alpha = 1/2$) and an (α_j)-schedule shown above and below the LP schedule, respectively. Notice that there is no common α value that would lead to the latter schedule.

3.1 Bounding the completion times in (α_j)-schedules

To analyze the completion times of jobs in (α_j)-schedules, we consider non-preemptive schedules of similar structure that are, however, constructed by a slightly different conversion routine which we call (α_j)-Conversion:

> Consider the jobs $j \in N$ in order of non-increasing α_j-points $t_j(\alpha_j)$ and iteratively change the preemptive LP schedule 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 $t_j(\alpha_j)$ 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 $t_j(\alpha_j)$ 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 $[t_j(\alpha_j), t_j(\alpha_j) + p_j)$.

Figure 3 contains an example illustrating the action of (α_j)-Conversion starting from the LP schedule of Figure 1. Observe that in the resulting schedule jobs are processed in non-decreasing order of α_j-points and no job j is started before time $t_j(\alpha_j) \geq r_j$. The latter property will be useful in the analysis of on-line (α_j)-schedules.

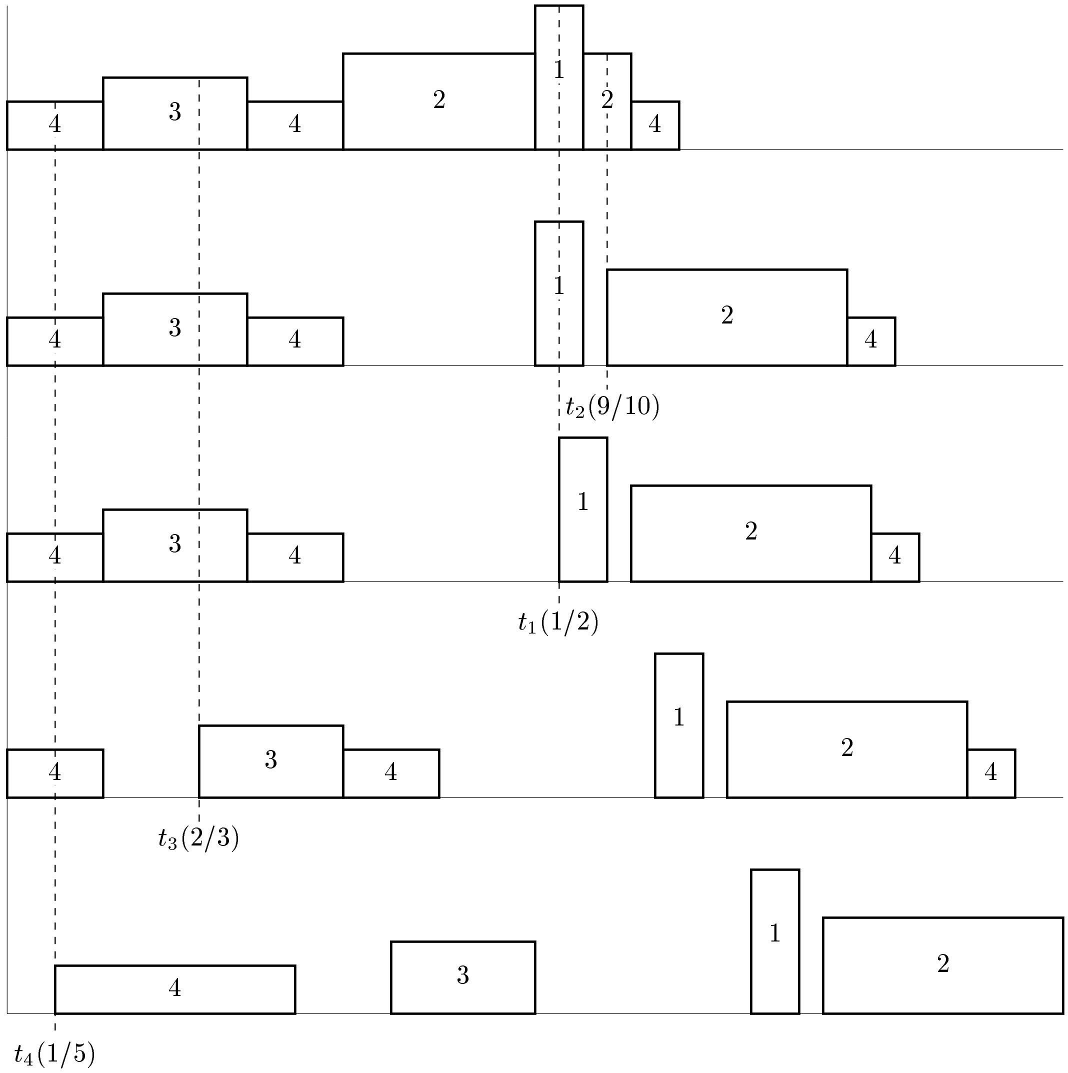

Figure 3: Illustration of the individual iterations of (α_j)-Conversion.

Lemma 3.1. *The completion time of job j in the schedule constructed by (α_j)-Conversion is equal to*

$$t_j(\alpha_j) \;+ \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} \bigl(1 + \alpha_k - \eta_k(\alpha_j)\bigr) p_k \;\; .$$

Proof. Consider the schedule constructed by (α_j)-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_j-points, the amount of processing before the completion of job j is

$$\sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} p_k \;\; . \tag{9}$$

The idle time before the start of job j can be written as the sum of the idle time τ_{idle} that already existed in the LP schedule before j's start plus the idle time before the start of job j

that is caused in steps i) of (α_j)-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

$$\tau_{idle} \;+ \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} \alpha_k\, p_k \;+ \sum_{\substack{k \\ \alpha_k > \eta_k(\alpha_j)}} \eta_k(\alpha_j)\, p_k \quad . \tag{10}$$

The completion time of job j in the schedule constructed by (α_j)-CONVERSION is equal to the sum of the expressions in (9) and (10); the result then follows from equation (8). □

It follows from Lemma 3.1 that the completion time C_j of each job j in the non-preemptive schedule constructed by (α_j)-CONVERSION satisfies $C_j \geq t_j(\alpha_j) + p_j \geq r_j + p_j$, hence is a feasible schedule. Since the (α_j)-schedule processes the jobs as early as possible and in the same order as the (α_j)-CONVERSION schedule, we obtain the following corollary.

Corollary 3.2. *The completion time of job j in an (α_j)-schedule can be bounded by*

$$C_j^{\boldsymbol{\alpha}} \;\leq\; t_j(\alpha_j) \;+ \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} \bigl(1 + \alpha_k - \eta_k(\alpha_j)\bigr) p_k \quad .$$

3.2 Bounds for α-schedules and (α_j)-schedules

We start with a result on the quality of the α-schedule for a fixed common value of α.

Theorem 3.3. *For fixed α,* (i) *the value of the α-schedule is within a factor* $\max\{1+\frac{1}{\alpha}, 1+2\alpha\}$ *of the optimal LP value; in particular, for $\alpha = 1/\sqrt{2}$ the bound is $1+\sqrt{2}$. Simultaneously,* (ii) *the length of the α-schedule is within a factor of $1+\alpha$ of the optimal makespan.*

Proof. While the proof of (ii) is an immediate consequence of (8) and Corollary 3.2, it follows from the proof of Theorem 2.5 that for (i) it is sufficient to prove that, for any canonical set S, we have

$$\sum_{j \in S} p_j C_j^{\alpha} \leq \max\left(1 + \frac{1}{\alpha}, 1 + 2\alpha\right) \left\{ p(S) \left(r_{\min}(S) + \frac{1}{2} p(S) \right) + \frac{1}{2} \sum_{j \in S} p_j^2 \right\} . \tag{11}$$

Indeed, using (5) and Lemma 2.4 it would then follow that

$$\begin{aligned}
\sum_{j \in N} w_j C_j^{\alpha} &= \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{\ell=1}^{k(i)} \sum_{j \in S_\ell^i} p_j C_j^{\alpha} \\
&\leq \max\left(1 + \frac{1}{\alpha}, 1 + 2\alpha\right) \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{\ell=1}^{k(i)} \sum_{j \in S_\ell^i} p_j \left(M_j^{LP} + \frac{p_j}{2} \right) \\
&= \max\left(1 + \frac{1}{\alpha}, 1 + 2\alpha\right) \sum_{j \in N} w_j \left(M_j^{LP} + \frac{p_j}{2} \right) \\
&= \max\left(1 + \frac{1}{\alpha}, 1 + 2\alpha\right) Z_R \ ,
\end{aligned}$$

proving the result.

Consider now any canonical set S and let us assume that, after renumbering the jobs, $S = \{1, 2, \dots, \ell\}$ and $t_1(\alpha) < t_2(\alpha) < \cdots < t_\ell(\alpha)$ (so the ordering is not necessarily anymore in non-increasing order of w_j/p_j). Fix now any job $j \in S$. From Corollary 3.2, we derive that

$$C_j^\alpha \le t_j(\alpha) + \sum_{k:\alpha\le\eta_k} (1+\alpha-\eta_k)\, p_k \ , \tag{12}$$

where $\eta_k := \eta_k(\alpha)$ represents the fraction of job k processed in the LP schedule before $t_j(\alpha)$. Let R denote the set of jobs k such that $t_k(\alpha) < r_{\min}(S)$ (and thus $\alpha \le \eta_k$). Since S is a canonical set, the jobs in S are processed continuously in the LP schedule between $r_{\min}(S)$ and $r_{\min}(S) + p(S)$, and therefore every job k with $\alpha \le \eta_k$ is either in S or in R. Observe that $\sum_{k\in R} \alpha\, p_k \le r_{\min}(S)$ implies that $p(R) \le \frac{1}{\alpha} r_{\min}(S)$. We can thus simplify (12) with:

$$C_j^\alpha \le t_j(\alpha) + \frac{1}{\alpha} r_{\min}(S) + \sum_{k=1}^{j} (1+\alpha-\eta_k)\, p_k \ . \tag{13}$$

Since the jobs in S are scheduled with no gaps in $[r_{\min}(S), r_{\min}(S) + p(S)]$, we have that

$$t_j(\alpha) = r_{\min}(S) + \sum_{k\in S} \eta_k p_k \le r_{\min}(S) + \sum_{k=1}^{j} \eta_k p_k + \sum_{k=j+1}^{\ell} \alpha\, p_k \ . \tag{14}$$

Combining (13) and (14), we derive that

$$C_j^\alpha \le \left(1 + \frac{1}{\alpha}\right) r_{\min}(S) + \alpha\, p(S) + \sum_{k=1}^{j} p_k \ .$$

Multiplying by p_j and summing over S, we get:

$$\begin{aligned}
\sum_{j\in S} p_j C_j^\alpha \ &\le \ \left(1+\frac{1}{\alpha}\right) r_{\min}(S) p(S) + \alpha\, p(S)^2 + \sum_{j\in S}\sum_{k=1}^{j} p_j p_k \\
&= \ \left(1+\frac{1}{\alpha}\right) r_{\min}(S) p(S) + \left(\frac{1}{2} + \alpha\right) p(S)^2 + \frac{1}{2}\sum_{j\in S} p_j^2 \ ,
\end{aligned}$$

which implies (11). □

In the sequel we will compare the completion time $C_j^{\boldsymbol{\alpha}}$ of every job j with its "completion time" $M_j^{LP} + \frac{1}{2} p_j$ in the LP schedule. However, for any fixed common value of α, there exist instances which show that this type of job-by-job analysis can give a bound no better than $1+\sqrt{2} > 2.4142$. One can also show that, for any given value of α, there exist instances for which the objective function value of the α-schedule can be as bad as twice the LP lower bound.

In view of these results, it is advantageous to use several values of α as it appears that no instance can be simultaneously bad for all choices of α. In fact, the α-points develop their full power in combination with randomization, i. e., when a common α or even job-dependent α_j's are chosen randomly from $(0, 1]$ according to an appropriate density function. This is also motivated by equation (7) which relates the expected α-point of a job under a uniform distribution of α to the LP variable M_j^{LP}. For random values α_j, we analyze the expected value of the resulting (α_j)-schedule and compare it to the optimal LP value. Notice that a bound on the expected value proves the existence of a vector $(\bar{\alpha}_j)$ such that the corresponding $(\bar{\alpha}_j)$-schedule meets this bound. Moreover, for our results we can always compute such an $(\bar{\alpha}_j)$ in polynomial time by derandomizing our algorithms with standard methods; see Propositions 3.8 and 3.13.

Although the currently best known bounds can only be achieved for (α_j)-schedules with job-dependent α_j's, we investigate α-schedules with a single common α as well. On the one hand, this helps to better understand the potential advantages of (α_j)-schedules; on the other hand, the randomized algorithm that relies on a single α admits a natural derandomization. In fact, we can easily 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. In Proposition 3.8 below, we will show that there are at most n different α-schedules. The BEST-α-schedule can be constructed in $O(n^2)$ time by evaluating all these different schedules.

As a warm-up exercise for the kind of analysis we use, we start by proving a bound of 2 on the expected worst-case performance ratio of uniformly generated (α_j)-schedules in the following theorem. This result will then be improved by using more intricate probability distributions and by taking advantage of additional insights into the structure of the LP schedule.

Theorem 3.4. *Let the random variables α_j be pairwise independently and uniformly drawn from $(0,1]$. Then, the expected value of the resulting (α_j)-schedule is within a factor 2 of the optimal LP value $Z_D = Z_R$.*

Proof. Remember that the optimal LP value is given by $\sum_j w_j(M_j^{LP} + \frac{1}{2}p_j)$. To get the claimed result, we prove that $\mathrm{E}_U[C_j^{\boldsymbol{\alpha}}] \le 2(M_j^{LP} + \frac{1}{2}p_j)$ for all jobs $j \in N$, where $\mathrm{E}_U[F(\boldsymbol{\alpha})]$ denotes the expectation of a function F of the random variable $\boldsymbol{\alpha}$ when the latter is uniformly distributed. The overall performance follows from this job-by-job bound by linearity of expectations.

Consider an arbitrary, but fixed job $j \in N$. To analyze the expected completion time of j, we first keep α_j fixed, and consider the conditional expectation $\mathrm{E}_U[C_j^{\boldsymbol{\alpha}} \mid \alpha_j]$. Since the random variables α_j and α_k are independent for each $k \neq j$, Corollary 3.2 and equation (8) yield

$$\begin{aligned}
\mathrm{E}_U[C_j^{\boldsymbol{\alpha}} \mid \alpha_j] \;&\le\; t_j(\alpha_j) \;+ \sum_{k \neq j} p_k \int_0^{\eta_k(\alpha_j)} \big(1 + \alpha_k - \eta_k(\alpha_j)\big)\, d\alpha_k \;+\; p_j \\
&=\; t_j(\alpha_j) \;+ \sum_{k \neq j} \Big(\eta_k(\alpha_j) - \frac{\eta_k(\alpha_j)^2}{2}\Big) p_k \;+\; p_j \\
&\le\; t_j(\alpha_j) \;+ \sum_{k \neq j} \eta_k(\alpha_j) p_k \;+\; p_j \;\le\; 2\Big(t_j(\alpha_j) + \frac{1}{2}p_j\Big) \;.
\end{aligned}$$

To obtain the unconditional expectation $\mathrm{E}_U[C_j^{\boldsymbol{\alpha}}]$ we integrate over all possible choices of α_j:

$$\mathrm{E}_U[C_j^{\boldsymbol{\alpha}}] \;=\; \int_0^1 \mathrm{E}_U[C_j^{\boldsymbol{\alpha}} \mid \alpha_j]\, d\alpha_j \;\le\; 2\Big(\int_0^1 t_j(\alpha_j)\, d\alpha_j + \frac{1}{2}p_j\Big) \;=\; 2\Big(M_j^{LP} + \frac{1}{2}p_j\Big) \;;$$

the last equation follows from (7). $\square$

We turn now to deriving improved results. We start with an analysis of the structure of the LP schedule. Consider any job j, and assume that, in the LP schedule, j is preempted at time s and its processing resumes at time $t > s$. Then all the 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 LP schedule, 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 $t_j(0^+)$ 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 $N \setminus \{j\}$ into two subsets N_1 and N_2: Let N_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 N_1. Notice that the function η_k is constant for jobs $k \in N_1$; to simplify notation we write $\eta_k := \eta_k(\alpha_j)$ for those jobs. For $k \in N_2$, 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

$$\eta_k(\alpha_j) \;=\; \begin{cases} 0 & \text{if } \alpha_j \le \mu_k, \\ 1 & \text{if } \alpha_j > \mu_k, \end{cases} \qquad \text{for } k \in N_2.$$

We can now rewrite equation (8) as

$$t_j(\alpha_j) \;=\; \tau_{\text{idle}} + \sum_{k \in N_1} \eta_k p_k + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} p_k + \alpha_j\, p_j \;=\; t_j(0^+) + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} p_k + \alpha_j\, p_j \;. \tag{15}$$

Plugging (15) into equation (7) yields

$$M_j^{LP} \;=\; t_j(0^+) + \sum_{k \in N_2} (1 - \mu_k) p_k + \frac{1}{2} p_j \;, \tag{16}$$

and Corollary 3.2 can be rewritten as

$$C_j^{\boldsymbol{\alpha}} \;\le\; t_j(0^+) + \sum_{\substack{k \in N_1 \\ \alpha_k \le \eta_k}} (1 + \alpha_k - \eta_k)\, p_k + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} (1 + \alpha_k)\, p_k + (1 + \alpha_j)\, p_j \;, \tag{17}$$

where, for $k \in N_2$, we have used the fact that $\alpha_k \le \eta_k(\alpha_j)$ is equivalent to $\alpha_j > \mu_k$. The expressions (15), (16), and (17) reflect the structural insights that we need for proving stronger bounds for (α_j)-schedules and α-schedules in the sequel.

As mentioned above, the second ingredient for an improvement on the bound of 2 is a more sophisticated probability distribution of the random variables α_j. In view of the bound on $C_j^{\boldsymbol{\alpha}}$ given in (17), 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 (17) small; on the other hand, choosing larger values decreases the number of terms in the first sum on the right-hand side of (17). The balancing of these two effects contributes to reducing the bound on the expected value of $C_j^{\boldsymbol{\alpha}}$.

3.3 Improved bounds for α-schedules

In this subsection we prove the following theorem.

Theorem 3.5. *Let $\gamma \approx 0.4675$ be the unique solution to the equation*

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

satisfying $0 < \gamma < 1$. Define $c := \frac{1+\gamma}{1+\gamma-e^{-\gamma}} < 1.7451$ and $\delta := 1 - \frac{\gamma^2}{1+\gamma} \approx 0.8511$. If α is chosen according to the density function

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

then the expected value of the resulting random α-schedule is bounded by c times the optimal LP value $Z_D = Z_R$.

Before we prove Theorem 3.5 we state two properties of the density function f that are crucial for the analysis of the corresponding random α-schedule.

Lemma 3.6. *The function f given in Theorem 3.5 is a density function with the following properties:*

(i) $\displaystyle\int_0^\eta f(\alpha)(1+\alpha-\eta)\; d\alpha \;\leq\; (c-1)\,\eta$ *for all* $\eta \in [0,1]$,

(ii) $\displaystyle\int_\mu^1 f(\alpha)(1+\alpha)\; d\alpha \;\leq\; c\,(1-\mu)$ *for all* $\mu \in [0,1]$.

Property (i) is used to bound the delay to job j caused by jobs in N_1 which corresponds to the first summation on the right-hand side of (17). The second summation reflects the delay to job j caused by jobs in N_2 and will be bounded by property (ii).

Proof of Lemma 3.6. A short computation shows that $\delta = \ln\frac{c}{c-1}$. The function f is a density function since

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

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

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

Property (ii) holds for $\mu \in (\delta,1]$ since the left-hand side is 0 in this case. For $\mu \in [0,\delta]$ we have

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

This completes the proof of the lemma. ☐

Proof of Theorem 3.5. In Lemma 3.6, both (i) for $\eta = 1$ and (ii) for $\mu = 0$ yield $\mathrm{E}_f[\alpha] \leq c-1$, where $\mathrm{E}_f[\alpha]$ denotes the expected value of a random variable α that is distributed according to the density f given in Theorem 3.5. Thus, using inequality (17) and Lemma 3.6 we derive that

$$\begin{aligned}
\mathrm{E}_f\left[C_j^\alpha\right] \;&\leq\; t_j(0^+) \;+\; (c-1)\sum_{k\in N_1}\eta_k p_k \;+\; c\sum_{k\in N_2}(1-\mu_k)p_k \;+\; c\,p_j\\
&\leq\; c\,t_j(0^+) \;+\; c\sum_{k\in N_2}(1-\mu_k)p_k \;+\; c\,p_j \;=\; c\Big(M_j^{LP}+\frac{1}{2}p_j\Big)\;;
\end{aligned}$$

the last inequality follows from the definition of N_1 and η_k and the last equality follows from (16). ☐

Notice that any density function satisfying properties (i) and (ii) of Lemma 3.6 for some value c' directly leads to the job-by-job bound $\mathrm{E}_f[C_j^\alpha] \leq c'\big(M_j^{LP} + \frac{1}{2}p_j\big)$ for the corresponding random α-schedule. It is easy to see that the unit function satisfies Lemma 3.6 with $c' = 2$, which establishes the following variant of Theorem 3.4.

Corollary 3.7. *Let the random variable α be uniformly drawn from $(0,1]$. Then, the expected value of the resulting α-schedule is within a factor 2 of the optimal LP value $Z_D = Z_R$.*

The use of an exponential density function is motivated by the first property in Lemma 3.6; notice that the function $\alpha \mapsto (c-1)e^\alpha$ satisfies it with equality. On the other hand, the exponential function is truncated in order to reduce the term $\int_\mu^1 f(\alpha)(1+\alpha)\,d\alpha$ in the second property. In fact, the truncated exponential function f in Theorem 3.5 can be shown to minimize c'; it is therefore optimal for our analysis. In addition, there exists a class of instances for which the ratio of the expected cost of an α-schedule, determined using this density function, to the cost of the optimal LP value is arbitrarily close to 1.745; this shows that the preceding analysis is essentially tight in conjunction with truncated exponential functions.

Theorem 3.5 implies that the BEST-α-schedule has a value of at most $1.7451\,Z_R$. The following proposition shows that the randomized algorithm that yields the α-schedule can be easily derandomized because the sample space is small.

Proposition 3.8. *There are at most n different α-schedules; they can be computed in $\mathrm{O}(n^2)$ 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 α-schedule is bounded from above by the total number of preemptions. Since a preemption can occur in the LP schedule only whenever a job is released, the total number of preemptions is at most $n-1$, and the number of α-schedules is at most n. Since each of these α-schedules can be computed in $\mathrm{O}(n)$ time, the result on the running time follows. □

3.4 Improved bounds for (α_j)-schedules

In this subsection, we prove the following theorem.

Theorem 3.9. *Let $\gamma \approx 0.4835$ be the unique solution to the equation*

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

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

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

Then, the expected value of the resulting random (α_j)-schedule is bounded by c times the optimal LP value $Z_D = Z_R$.

The bound in Theorem 3.9 yields also a bound on the quality of the LP relaxations:

Corollary 3.10. *The LP relaxations (D) and (R) deliver in $\mathrm{O}(n\log n)$ time a lower bound which is at least 0.5933 ($\approx 1.6853^{-1}$) times the objective function value of an optimal schedule.*

Following the lines of the last subsection, we state two properties of the density function g that are crucial for the analysis of the corresponding random (α_j)-schedule.

Lemma 3.11. *The function g given in Theorem 3.9 is a density function with the following properties:*

(i) $\displaystyle\int_0^{\eta} g(\alpha)(1+\alpha-\eta)\;d\alpha \;\le\; (c-1)\,\eta \quad$ *for all $\eta \in [0,1]$,*

(ii) $\displaystyle(1+\mathrm{E}_g[\alpha])\int_{\mu}^{1} g(\alpha)\;d\alpha \;\le\; c\,(1-\mu) \quad$ *for all $\mu \in [0,1]$,*

where $\mathrm{E}_g[\alpha]$ denotes the expected value of a random variable α that is distributed according to g.

Notice the similarity of Lemma 3.11 and Lemma 3.6 of the last subsection. Again, properties (i) and (ii) are used to bound the delay to job j caused by jobs in N_1 and N_2, respectively, in the right-hand side of inequality (17). Property (i) for $\eta = 1$ or Property (ii) for $\mu = 0$ again yield $\mathrm{E}_g[\alpha] \le c-1$.

Proof of Lemma 3.11. A short computation shows that $\delta = \ln\frac{c}{c-1}$. It thus follows from the same arguments as in the proof of Lemma 3.6 that g is a density function and that property (i) holds. In order to prove property (ii), we first compute

$$\mathrm{E}_g[\alpha] \;=\; \int_0^1 g(\alpha)\alpha\;d\alpha \;=\; (c-1)\int_0^{\delta} e^{\alpha}\alpha\;d\alpha \;=\; c\,\delta - 1\ .$$

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

$$\begin{aligned}(1+\mathrm{E}_g[\alpha])\int_{\mu}^{1} g(\alpha)\;d\alpha \;&=\; c\,\delta\,(c-1)\int_{\mu}^{\delta} e^{\alpha}\;d\alpha \;=\; c\,e^{-\gamma}\big((2-\gamma)e^{\gamma}-e^{\mu}\big)\\ &=\; c\,(2-\gamma-e^{\mu-\gamma}) \;\le\; c\,\big(2-\gamma-(1+\mu-\gamma)\big) \;=\; c\,(1-\mu)\ .\end{aligned}$$

This completes the proof of the lemma. □

Proof of Theorem 3.9. Our analysis of the expected completion time of job j in the random (α_j)-schedule follows the line of argument developed in the proof of Theorem 3.4. First we consider a fixed choice of α_j and bound the corresponding conditional expectation $\mathrm{E}_g[C_j^{\boldsymbol{\alpha}} \mid \alpha_j]$. In a second step we bound the unconditional expectation $\mathrm{E}_g[C_j^{\boldsymbol{\alpha}}]$ by integrating the product $g(\alpha_j)\mathrm{E}_g[C_j^{\boldsymbol{\alpha}} \mid \alpha_j]$ over the interval $(0,1]$.

For a fixed job j and a fixed value α_j, the bound in (17) and Lemma 3.11 (i) yield

$$\begin{aligned}\mathrm{E}_g[C_j^{\boldsymbol{\alpha}} \mid \alpha_j] \;&\le\; t_j(0^+) \;+\; (c-1)\sum_{k\in N_1}\eta_k\,p_k \;+\; \sum_{\substack{k\in N_2\\ \alpha_j>\mu_k}}(1+\mathrm{E}_g[\alpha_k])p_k \;+\; (1+\alpha_j)p_j\\ &\le\; c\,t_j(0^+) \;+\; (1+\mathrm{E}_g[\alpha_1])\sum_{\substack{k\in N_2\\ \alpha_j>\mu_k}} p_k \;+\; (1+\alpha_j)p_j.\end{aligned}$$

The last inequality follows from (15) and $\mathrm{E}_g[\alpha_k] = \mathrm{E}_g[\alpha_1]$ for all $k \in N$. Using property (ii) and equation (16) yields

$$\begin{aligned}\mathrm{E}_g[C_j^{\boldsymbol{\alpha}}] \;&\le\; c\,t_j(0^+) \;+\; (1+\mathrm{E}_g[\alpha_1])\sum_{k\in N_2} p_k\int_{\mu_k}^{1} g(\alpha_j)\;d\alpha_j \;+\; (1+\mathrm{E}_g[\alpha_j])p_j\\ &\le\; c\,t_j(0^+) \;+\; c\sum_{k\in N_2}(1-\mu_k)p_k \;+\; c\,p_j \;=\; c\Big(M_j^{LP}+\frac{1}{2}p_j\Big)\ .\end{aligned}$$

The result follows from linearity of expectations. □

While the total number of possible orderings of jobs is $n! = 2^{O(n \log n)}$, we show in the following lemma that the maximum number of (α_j)-schedules is at most 2^{n-1}. We will use the following observation. Let q_j denote the number of different pieces of job j in the LP schedule; thus q_j represents the number of times job j is preempted plus 1. Since there are at most $n-1$ preemptions, we have that $\sum_{j=1}^n q_j \le 2n-1$.

Lemma 3.12. *The maximum number of (α_j)-schedules is at most 2^{n-1} and this bound can be attained.*

Proof. The number of (α_j)-schedules is given by $s = \prod_{j=1}^n q_j$. Note that $q_1 = 1$ since this job is not preempted in the LP schedule. Thus, $s = \prod_{j=2}^n q_j$, while $\sum_{j=2}^n q_j \le 2(n-1)$. By the arithmetic-geometric mean inequality, we have that

$$s = \prod_{j=2}^{n} q_j \le \left(\frac{\sum_{j=2}^n q_j}{n-1} \right)^{n-1} \le 2^{n-1}.$$

Furthermore, this bound is attained if $q_j = 2$ for $j = 2, \ldots, n$ and this is achieved for example for the instance with $p_j = 2$, $w_j = n - j + 1$ and $r_j = n - j$ for all j. □

Therefore, and in contrast to the case of random α-schedules, we cannot afford to derandomize the randomized 1.6853–approximation algorithm by enumerating all (α_j)-schedules. We instead use the method of conditional probabilities [18].

¿From inequality (17) we obtain for every vector $\boldsymbol{\alpha} = (\alpha_j)$ an upper bound on the objective function value of the corresponding (α_j)-schedule, $\sum_j w_j C_j^{\boldsymbol{\alpha}} \le UB(\boldsymbol{\alpha})$, where $UB(\boldsymbol{\alpha}) = \sum_j w_j \, RHS_j(\boldsymbol{\alpha})$ and $RHS_j(\boldsymbol{\alpha})$ denotes the right-hand side of inequality (17). Taking expectations and using Theorem 3.9, we have already shown that

$$\mathrm{E}_g\left[\textstyle\sum_{j\in N} w_j C_j^{\boldsymbol{\alpha}}\right] \le \mathrm{E}_g[UB(\boldsymbol{\alpha})] \le c\, Z_D \quad ,$$

where $c < 1.6853$. For each job $j \in N$ let $\mathcal{Q}_j = \{Q_{j1}, \ldots, Q_{jq_j}\}$ denote the set of the intervals for α_j corresponding to the q_j pieces of job j in the LP schedule. We consider the jobs one by one in arbitrary order, say, $j = 1, \ldots, n$. Assume that, at step j of the derandomized algorithm, we have identified intervals $Q_1^d \in \mathcal{Q}_1, \ldots, Q_{j-1}^d \in \mathcal{Q}_{j-1}$ such that

$$\mathrm{E}_g[UB(\boldsymbol{\alpha}) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \ldots, j-1] \le c\, Z_D \, .$$

Using conditional expectations, the left-hand side of this inequality is

$$\begin{aligned} &\mathrm{E}_g[UB(\boldsymbol{\alpha}) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \ldots, j-1] \\ &\quad = \sum_{\ell=1}^{q_j} \mathrm{Prob}\{\alpha_j \in Q_{j\ell}\} \mathrm{E}_g \left[UB(\boldsymbol{\alpha}) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \ldots, j-1 \text{ and } \alpha_j \in Q_{j\ell} \right] . \end{aligned}$$

Since $\sum_{\ell=1}^{q_j} \mathrm{Prob}\{\alpha_j \in Q_{j\ell}\} = 1$, there exists at least one interval $Q_{j\ell} \in \mathcal{Q}_j$ such that

$$\begin{aligned} &\mathrm{E}_g[UB(\boldsymbol{\alpha}) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \ldots, j-1 \text{ and } \alpha_j \in Q_{j\ell}] \\ &\quad \le \mathrm{E}_g[UB(\boldsymbol{\alpha}) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \ldots, j-1] \ . \end{aligned} \tag{18}$$

Therefore, it suffices to identify such an interval $Q_j^d = Q_{j\ell}$ satisfying (18) and we may conclude that

$$\mathrm{E}_g\left[\sum_{h\in N} w_h C_h^{\boldsymbol{\alpha}} \mid \alpha_i \in Q_i^d \text{ for } i = 1, \ldots, j \right] \le \mathrm{E}_g[UB(\boldsymbol{\alpha}) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \ldots, j] \le c\, Z_D \, .$$

Having determined in this way an interval Q_j^d for every job $j = 1, \dots, n$, we then note that the (α_j)-schedule is the same for all $\boldsymbol{\alpha} \in Q_1^d \times Q_2^d \times \cdots \times Q_n^d$. The (now deterministic) objective function value of this (α_j)-schedule is

$$\sum_{j \in N} w_j C_j^{\boldsymbol{\alpha}} \leq \mathrm{E}_g[UB(\boldsymbol{\alpha}) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, n] \leq \mathrm{E}_g[UB(\boldsymbol{\alpha})] \leq c\, Z_D < 1.6853\, Z_D\,,$$

as desired. For every $j = 1, \dots, n$, checking whether an interval Q_j^d satisfies inequality (18) amounts to evaluating $\mathrm{O}(n)$ terms, each of which may be computed in constant time. Since, as observed just before Lemma 3.12, we have a total of $\sum_{j=1}^{n} q_j \leq 2\,n - 1$ candidate intervals, it follows that the derandomized algorithm runs in $\mathrm{O}(n^2)$ time.

Proposition 3.13. *The randomized* 1.6853*–approximation algorithm can be derandomized; the resulting deterministic algorithm runs in* $\mathrm{O}(n^2)$ *time and has performance guarantee* 1.6853 *as well.*

3.5 Constructing provably good schedules on-line

In this subsection we show that our randomized approximation results also apply in an on-line setting. There are several different on-line paradigms that have been studied in the area of scheduling; we refer to [30] for a survey. We consider the 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., its processing time and its weight become only known at its release date.

It has already been shown in Section 2 that the LP schedule can be constructed on-line. Unfortunately, for a given vector (α_j), the corresponding (α_j)-schedule cannot be constructed on-line. We only learn about the position of a job k in the sequence defined by non-decreasing α_j-points at time $t_k(\alpha_k)$; therefore we cannot start job k at an earlier point in time in the on-line setting. On the other hand, however, the start time of k in the (α_j)-schedule can be earlier than its α_k-point $t_k(\alpha_k)$.

Although an (α_j)-schedule cannot be constructed on-line, the above discussion reveals that the following variant, which we call *on-line-(α_j)-schedule*, can be constructed on-line: For a given vector (α_j), process the jobs as early as possible in the order of their α_j-points, with the additional constraint that no job k may start before time $t_k(\alpha_k)$. See Figure 4 for an example. We note that this idea of delaying the start of jobs until sufficient information for a good decision is available was in this setting introduced by Phillips, Stein and Wein [21].

Notice that the non-preemptive schedule constructed by (α_j)-CONVERSION fulfills these constraints; its value is therefore an upper bound on the value of the on-line-(α_j)-schedule. Our analysis in the last subsections relies on the bound given in Corollary 3.2 which also holds for the schedule constructed by (α_j)-CONVERSION by Lemma 3.1. This yields the following results.

Theorem 3.14. *For any instance of the scheduling problem* $1|\, r_j \,|\sum w_j C_j$*,*

a) *choosing* $\alpha = 1/\sqrt{2}$ *and constructing the on-line-α-schedule yields a deterministic on-line algorithm with competitive ratio* $1 + \sqrt{2} \leq 2.4143$ *and running time* $\mathrm{O}(n \log n)$*;*

b) *choosing the α_j's randomly and pairwise independently from* $(0, 1]$ *according to the density function g of Theorem* 3.9 *and constructing the on-line-(α_j)-schedule yields a randomized on-line algorithm with competitive ratio* 1.6853 *and running time* $\mathrm{O}(n \log n)$*.*

The competitive ratio 1.6853 in Theorem 3.14 beats the deterministic on-line lower bound 2 for the unit-weight problem $1|\, r_j \,|\sum C_j$ [15, 36]. For the same problem, Stougie and Vestjens proved the lower bound $\frac{e}{e-1} > 1.5819$ for randomized on-line algorithms [37, 39].

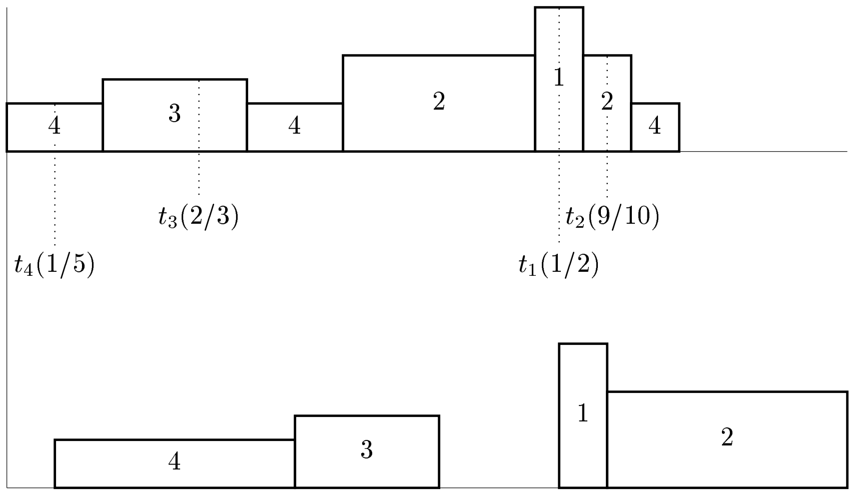

Figure 4: The on-line schedule for the previously considered instance and α_j-points. The LP schedule is shown above for comparison.

3.6 Bad instances for the LP relaxations

In this subsection, we describe a family of instances for which the ratio between the optimal value of the $1|\,r_j\,|\sum w_jC_j$ problem and the lower bounds Z_R and Z_D is arbitrarily close to $\frac{e}{e-1} > 1.5819$.

These instances I_n have $n \geq 2$ jobs as follows: 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 $w_n = \frac{1}{n}$ 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 $r_j = j$.

Throughout the paper, we have assumed that processing times are non-zero. In order to satisfy this assumption, we could impose a processing time of $1/k$ for all small jobs, multiply all processing times and release dates by k to make the data integral, and then let k tend to infinity. For simplicity, however, we just let the processing time of all small jobs be 0.

The LP solution has job n start at time 0, preempted by each of the small jobs; hence its mean busy times are: $M_j^{LP} = r_j$ for $j = 1, \dots, n-1$ and $M_n^{LP} = \frac{n}{2}$. Its objective function value is $Z_R = (1 + \frac{1}{n-1})^n - (1 + \frac{1}{n-1})$. Notice that the completion time of each job j is in fact equal to $M_j^{LP} + \frac{1}{2}p_j$ such that the actual value of the preemptive schedule is equal to Z_R.

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 $(1 + \frac{1}{n-1})^n - \frac{1}{n-1} - \frac{1}{n}$. As n grows large, the LP objective function value approaches $e - 1$ while the optimal non-preemptive cost approaches e.

4 Conclusion

Even though polynomial-time approximation schemes have now been discovered for the problem $1|\,r_j\,|\sum w_jC_j$ [1], the algorithms we have developed, or variants of them, are likely to be superior in practice. The experimental studies of Savelsbergh et al. [25] and Uma and Wein [38] indicate that LP-based relaxations and scheduling in order of α_j-points are powerful tools for a variety of scheduling problems.

Several intriguing questions remain open. Regarding the quality of linear programming relaxations, it would be interesting to close the gap between the upper (1.6853) and lower (1.5819) bound on the quality of the relaxations considered in this paper. We should point out that the situation for the strongly NP-hard [16] problem $1|\,r_j,\, pmtn\,|\sum w_j C_j$ is similar. It is shown in [29] that the completion time relaxation is in the worst case at least a factor of 8/7 and at most a factor of 4/3 off the optimum; the latter bound is achieved by scheduling preemptively by LP-based random α-points. Chekuri et al. [6] prove that the optimal non-preemptive value is at most $e/(e-1)$ times the optimal preemptive value; our example in Section 3.6 shows that this bound is tight.

Dyer and Wolsey [9] propose also a (non-preemptive) time-indexed relaxation which is stronger than the preemptive version studied here. This relaxation involves variables for each job and each time representing whether this job is being *completed* (rather than simply processed) at that time. This relaxation is at least as strong as the preemptive version, but its worst-case ratio is not known to be strictly better.

For randomized on-line algorithms, there is also a gap between the known upper and lower bound on the competitive ratios that are given at the end of Section 3.5. For deterministic on-line algorithms, the 2–competitve algorithm of Anderson and Potts [2] is optimal.

Acknowledgements

The research of the first author was performed partly when the author was at C.O.R.E., Louvain-la-Neuve, Belgium and supported in part by NSF contract 9623859-CCR. The research of the second author was supported in part by a research grant from NSERC (the Natural Sciences and Research Council of Canada) and by the UNI.TU.RIM. S.p.a. (Società per l'Università nel riminese), whose support is gratefully acknowledged. The research of the third author was performed partly when he was with the Department of Mathematics, Technische Universität Berlin, Germany. The fourth author was supported in part by DONET within the frame of the TMR Programme (contract number ERB FMRX-CT98-0202) while staying at C.O.R.E., Louvain-la-Neuve, Belgium, for the academic year 1998/99. The fifth author was supported by a research fellowship from Max-Planck Institute for Computer Science, Saarbrücken, Germany.

We are also grateful to an anonymous referee whose comments helped to improve the presentation of this paper.

References

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

[2] E.J. Anderson and C.N. Potts, "On-line scheduling of a single machine to minimize total weighted completion time", Manuscript, 2001.

[3] E. Balas, "On the facial structure of scheduling polyhedra", *Mathematical Programming Study*, **24**, 179–218 (1985).

[4] H. Belouadah, M.E. Posner and C.N. Potts, "Scheduling with release dates on a single machine to minimize total weighted completion time", *Discrete Applied Mathematics*, **36**, 213–231 (1992).

[5] S. Chakrabarti, C. Phillips, A.S. Schulz, D.B. Shmoys, C. Stein and J. Wein, "Improved algorithms for minsum criteria", in F. Meyer auf der Heide and B. Monien (eds.), *Automata, Languages and Programming* (Proceedings of the 23rd ICALP), Lecture Notes in Computer Science, vol. 1099, Springer, Berlin, 646–657 (1996).

[6] C. Chekuri, R. Motwani, B. Natarajan and C. Stein, "Approximation techniques for average completion time scheduling", *SIAM Journal on Computing*, **31**, 146–166 (2001). An extended abstract appeared in the *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, 609–618 (1997).

[7] C.-F. Chou, M. Queyranne and D. Simchi-Levi, "The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates", in K. Aardal and B. Gerards (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 8th International IPCO Conference), Lecture Notes in Computer Science, vol. 2081, Springer, Berlin, 45–59 (2001).

[8] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, MIT Press (1990).

[9] 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).

[10] M.X. Goemans, "A supermodular relaxation for scheduling with release dates", in W.H. Cunningham, S.T. McCormick and M. Queyranne (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 5th International IPCO Conference), Lecture Notes in Computer Science, vol. 1084, Springer, Berlin, 288–300 (1996).

[11] M.X. Goemans, "Improved approximation algorithms for scheduling with release dates", *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, 591–598 (1997).

[12] 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).

[13] L.A. Hall, D.B. Shmoys and J. Wein, "Scheduling to minimize average completion time: Off-line and on-line algorithms", *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, Atlanta, GA, 142–151 (1996).

[14] 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).

[15] 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 (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 5th International IPCO Conference), Lecture Notes in Computer Science, vol. 1084, Springer, Berlin, 404–414 (1996).

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

[17] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, **1**, 343–362 (1977).

[18] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press (1995).

[19] A. Munier, M. Queyranne and A.S. Schulz, "Approximation bounds for a general class of precedence constrained parallel machine scheduling problems", in R.E. Bixby, E.A. Boyd and R.Z. Ríos-Mercado (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 6th International IPCO Conference), Lecture Notes in Computer Science, vol. 1412, Springer, Berlin, 367-382 (1998). Journal version will appear in *SIAM Journal on Computing*.

[20] G.L. Nemhauser, L.A. Wolsey and M.L. Fisher, "An analysis of approximations for maximizing submodular set functions — I", *Mathematical Programming*, **14**, 265–294 (1978).

[21] C. Phillips, C. Stein and J. Wein, "Minimizing average completion time in the presence of release dates", *Mathematical Programming*, **82**, 199–223 (1998). An extended abstract appeared under the title "Scheduling jobs that arrive over time" in S.G. Akl, F. Dehne, J.-R. Sack, N. Santoro (eds.), *Algorithms and Data Structures* (Proceedings of the 4th International WADS), Lecture Notes in Computer Science, vol. 955, Springer, Berlin, 86–97 (1995).

[22] M. Queyranne, "Structure of a simple scheduling polyhedron", *Mathematical Programming*, **58**, 263–285 (1993).

[23] M. Queyranne and A.S. Schulz, "Polyhedral approaches to machine scheduling problems", Preprint 408/1994, Department of Mathematics, Technische Universität Berlin, Germany, 1994 (revised 1996).

[24] M. Queyranne and A.S. Schulz, "Scheduling unit jobs with compatible release dates on parallel machines with nonstationary speeds", in E. Balas and J. Clausen (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 4th International IPCO Conference), Lecture Notes in Computer Science, vol. 920, Springer, Berlin, 307–320 (1995).

[25] M.W.P. Savelsbergh, R.N. Uma and J. Wein, "An experimental study of LP-based approximation algorithms for scheduling problems", *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 453–462 (1998).

[26] 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 (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 5th International IPCO Conference), Lecture Notes in Computer Science, vol. 1084, Springer, Berlin, 301–315 (1996).

[27] A.S. Schulz and M. Skutella, "Random-based scheduling: New approximations and LP lower bounds", in J. Rolim (ed.), *Randomization and Approximation Techniques in Computer Science* (Proceedings of the 1st International Workshop RANDOM), Lecture Notes in Computer Science, vol. 1269, Springer, Berlin, 119–133 (1997).

[28] A.S. Schulz and M. Skutella, "Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria", in R. Burkard and G.J. Woeginger (eds.), *Algorithms — ESA '97*, Lecture Notes in Computer Science, vol. 1284, Springer, Berlin, 416–429 (1997). Journal version will appear under the title "Scheduling unrelated machines by randomized rounding" in *SIAM Journal on Discrete Mathematics*.

[29] A.S. Schulz and M. Skutella, "The power of α-points in preemptive single machine scheduling", *Journal of Scheduling*, to appear.

[30] J. Sgall, "On-line scheduling — a survey", in A. Fiat and G.J. Woeginger (eds.), *Online Algorithms: The State of the Art*, Lecture Notes in Computer Science, vol. 1442, Springer, Berlin, 196–231 (1998).

[31] L.S. Shapley, "Cores of Convex Games", *International Journal of Game Theory*, **1**, 11–26 (1971).

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

[33] M. Skutella, "Convex quadratic and semidefinite programming relaxations in scheduling", *Journal of the ACM*, **48**, 206–242 (2001). Extended abstracts of different parts appeared in the *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, Palo Alto, CA, 472–481 (1998), and in J. Nešetřil (ed.), *Algorithms — ESA '99*, Lecture Notes in Computer Science, vol. 1643, Springer, Berlin, 127–138 (1999).

[34] M. Skutella and G.J. Woeginger, "A PTAS for minimizing the total weighted completion time on identical parallel machines", *Mathematics of Operations Research*, **25**, 63–75 (2000). An extended abstract appeared in the *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, Atlanta, GA, 400–407 (1999).

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

[36] L. Stougie, 1995, cited as personal communication in [15].

[37] L. Stougie and A.P.A. Vestjens, "Randomized on-line scheduling: How low can't you go?". Manuscript, 1997.

[38] R.N. Uma and J. 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 (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 6th International IPCO Conference), Lecture Notes in Computer Science, vol. 1412, Springer, Berlin, 394–408 (1998).

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