SCHEDULING UNRELATED MACHINES BY RANDOMIZED ROUNDING†

ANDREAS S. SCHULZ* AND MARTIN SKUTELLA**

Abstract. We present a new class of randomized approximation algorithms for unrelated parallel machine scheduling problems with the average weighted completion time objective. The key idea is to assign jobs randomly to machines with probabilities derived from an optimal solution to an LP relaxation in time-indexed variables. Our main results are a $(2+\varepsilon)$–approximation algorithm for the model with individual job release dates, and a $(3/2+\varepsilon)$–approximation algorithm if all jobs are released simultaneously. We obtain corresponding bounds on the quality of the LP relaxation.

It is an interesting implication for identical parallel machine scheduling that jobs are randomly assigned to machines, in which each machine is equally likely. In addition, in this case the algorithm has running time $O(n \log n)$ and performance guarantee 2. Moreover, the approximation result for identical parallel machine scheduling applies to the on-line setting in which jobs arrive over time as well, with no difference in performance guarantee.

Key words. approximation algorithm, randomized rounding, LP relaxation, scheduling, on-line algorithm

AMS subject classifications. 90C27, 68Q25, 90B35, 68M20

1. Introduction. It is well-known that randomization can help in the design of (approximation) algorithms, cf., e. g., [28, 29]. The use of linear programs (LPs) is one way of guiding randomness. In this paper, we give LP-based, randomized approximation algorithms for parallel machine scheduling problems with the average weighted completion time objective. A randomized ρ–approximation algorithm for a minimization problem is a polynomial-time algorithm that produces for every instance a feasible solution whose expected objective function value is within a factor of ρ of the optimum; ρ is also called the (expected) performance guarantee of the algorithm. Most often, we actually compare the output of an algorithm to a lower bound given by an optimal solution to a certain LP relaxation. Hence, at the same time we obtain a result on the quality of the respective LP. All of our off-line algorithms can be derandomized with no difference in performance guarantees, but at the expense of increased, but still polynomial running times.

We consider the following scheduling model. We are given a set J of n jobs, and m unrelated parallel machines. The processing time of job j depends on the machine on which j will be processed; it is a positive integer p_{ij} on machine i. Each job j must be processed for the respective amount of time on one of the m machines, and may be assigned to any of them. Every machine can process at most one job at a time. Each job j also has an integral release date $r_j \geqslant 0$ before which it cannot be started. We denote the completion time of job j in a schedule S by C_j^S; we also use C_j if no confusion is possible as to which schedule we refer. The objective is to minimize the total weighted completion time: a weight $w_j \geqslant 0$ is associated with each job j and the goal is to find a schedule S that minimizes $\sum_{j \in J} w_j C_j^S$. The classification scheme introduced by Graham, Lawler, Lenstra, and Rinnooy Kan [19] offers a convenient way to refer to individual scheduling problems. The problem that we just described is de-

†Different parts of this paper appeared in preliminary form in [35, 36].

*Massachusetts Institute of Technology, Sloan School of Management, E53-361, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA (schulz@mit.edu)

**Technische Universität Berlin, Fakultät II, Institut für Mathematik, MA 6-1, Straße des 17. Juni 136, D-10623 Berlin, Germany (skutella@math.tu-berlin.de)

noted by $\mathrm{R}\,|\,r_j\,|\,\sum w_jC_j$, and, if all jobs share the same release date, by $\mathrm{R}\,|\,|\,\sum w_jC_j$. We will also consider the special case of identical parallel machines, which arises by assuming that, for each job j, $p_{ij} = p_j$ for all machines i. In general, we are only interested in non-preemptive schedules, in which each job must be processed without interruption. Yet, for the identical parallel machine case we also discuss preemptive schedules in which jobs may repeatedly be interrupted and continued later, possibly on a different machine. Hence, the class of problems for which we will present approximation algorithms includes $\mathrm{P}\,|\,|\,\sum w_jC_j$, $\mathrm{P}\,|\,r_j, \mathrm{pmtn}\,|\,\sum w_jC_j$, and $\mathrm{P}\,|\,r_j\,|\,\sum w_jC_j$. These problems are strongly NP-hard [24, 25].

Scheduling to minimize the total weighted completion time (or, equivalently, the average weighted completion time) has recently received a great deal of attention, partly because of its importance as a classic objective function in scheduling, but also because of new applications, for instance, in compiler optimization [6] or in parallel computing [4]. There has been significant progress in the design of approximation algorithms for this class of problems. This progress essentially results from the use of preemptive schedules to construct non-preemptive ones, and from solving an LP relaxation and then constructing a schedule by list scheduling in an order dictated by the LP solution [5, 7, 9, 16, 17, 18, 20, 26, 27, 30, 33, 37, 42].

In this paper, we propose a new technique: random assignments of jobs to machines. In fact, we first introduce an LP relaxation in time-indexed variables for the problem $\mathrm{R}\,|\,r_j\,|\,\sum w_jC_j$, and we then show that a certain variant of randomized rounding leads to an algorithm with performance guarantee 2. If all jobs are released at the same time, $\mathrm{R}\,|\,|\,\sum w_jC_j$, the performance guarantee of this algorithm is $3/2$. The latter observation was independently made by Chudak [8]. The corresponding LP is a 2–relaxation and a $3/2$–relaxation, respectively, i. e., the true optimum is always within this factor of the optimal value of the LP relaxation. Our algorithm improves upon a $16/3$–approximation algorithm of Hall, Schulz, Shmoys, and Wein [20], which is based on a related interval-indexed LP relaxation. In contrast to their approach, our algorithm does not rely on Shmoys and Tardos' rounding technique for the generalized assignment problem [39]. Rather, we exploit the LP by interpreting the values of the LP variables in an optimal solution as probabilities with which jobs are assigned to machines. For an introduction to randomized rounding and its application to other combinatorial optimization problems, the reader is referred to [28, 34].

Since the time-indexed LP relaxation is of exponential size, we have to resort to an interval-indexed formulation in order to obtain polynomial running times. The resulting algorithm is a $(2 + \varepsilon)$–approximation algorithm for $\mathrm{R}\,|\,r_j\,|\,\sum w_jC_j$, and a $(3/2 + \varepsilon)$–approximation algorithm for $\mathrm{R}\,|\,|\,\sum w_jC_j$, for any $\varepsilon > 0$. The second author subsequently developed a different approach to overcome this difficulty. Based on compact convex quadratic programming relaxations in assignment variables, the same rounding technique yields a 2–approximation algorithm for $\mathrm{R}\,|\,r_j\,|\,\sum w_jC_j$, and a $3/2$–approximation algorithm for $\mathrm{R}\,|\,|\,\sum w_jC_j$ directly, see [41].

Actually, for $\mathrm{P}\,|\,r_j\,|\,\sum w_jC_j$, our algorithm produces in time $\mathrm{O}(n \log n)$ a solution that is expected to be within a factor of 2 of the optimum. Since the underlying LP relaxation also is a relaxation of the corresponding preemptive problem, this algorithm is a 2–approximation algorithm for $\mathrm{P}\,|\,r_j, \mathrm{pmtn}\,|\,\sum w_jC_j$ as well. The best previously known approximation algorithms had performance guarantees of $(2.89+\varepsilon)$ and 3, respectively [5, 20]. In addition, our result implies that the value of an optimal non-preemptive schedule is at most twice the value of an optimal preemptive schedule. Moreover, an optimal solution to the LP used in the case of identical parallel

Model	Off-line			On-line		
	Known		New	Known		New
$\mathrm{P}\,\|\,r_j\,\|\,\sum C_j$	2.85	[7]	2	$2.89+\varepsilon$	[5]	2
$\mathrm{P}\,\|\,r_j\,\|\,\sum w_j C_j$	$2.89+\varepsilon$	[5]	2	$2.89+\varepsilon$	[5]	2
$\mathrm{P}\,\|\,r_j, \mathrm{pmtn}\,\|\,\sum C_j$	2	[33]	2	2	[33]	2
$\mathrm{P}\,\|\,r_j, \mathrm{pmtn}\,\|\,\sum w_j C_j$	3	[20]	2	—		2
$\mathrm{R}\,\|\,\|\,\sum w_j C_j$	16/3	[20]	$3/2+\varepsilon$	—		—
$\mathrm{R}\,\|\,r_j\,\|\,\sum w_j C_j$	16/3	[20]	$2+\varepsilon$	—		—

TABLE 1.1

Summary of performance guarantees for the minimization of the total weighted completion time. The "Known" columns list the best previously known performance guarantees, whereas the "New" columns list new results from this paper; "—" indicates the absence of a relevant result, and ε is an arbitrarily small, positive constant. While the off-line results are achieved by deterministic algorithms, all on-line results refer to randomized algorithms.

machines is attained by the following preemptive schedule, which can be obtained in a greedy manner. Consider a virtual single machine, which is m times as fast as any of the original machines. At any point in time, schedule from the jobs that are already released, but not yet completed, one with largest ratio of weight to processing time. We call this schedule the LP schedule. The idea of using a preemptive relaxation on a virtual single machine was employed before by Chekuri, Motwani, Natarajan, and Stein [7], among others. They showed that any preemptive schedule on such a machine can be converted into a non-preemptive schedule on m identical parallel machines such that the completion time of each job j in the non-preemptive schedule is at most $(3-1/m)$ times its preemptive completion time. For the problem of minimizing the average completion time, $\mathrm{P}\,|\,r_j\,|\,\sum C_j$, they refined this to a 2.85–approximation algorithm. In the single-machine context, the LP schedule is the key ingredient of the 1.6853–approximation algorithm for $1\,|\,r_j\,|\,\sum w_j C_j$ [17] and the 4/3–approximation algorithm for $1\,|\,r_j, \mathrm{pmtn}\,|\,\sum w_j C_j$ [37].

Since an optimal solution to the LP relaxation can be obtained greedily, these single-machine algorithms as well as our algorithm for identical parallel machine scheduling also work in the corresponding on-line setting where jobs continually arrive to be processed and, for each time t, one must construct the schedule until time t without any knowledge of the jobs that will arrive afterwards; our algorithm maintains an (expected) competitive ratio of 2 for both the non-preemptive and the preemptive variant of this problem. A randomized on-line algorithm for a minimization problem is ρ–competitive if it outputs for any instance a solution such that its expected value is within a factor of ρ of the value of an optimal off-line solution. A summary of our results, along with a comparison to previously known performance guarantees, is given in Table 1.1.

Recently, Skutella and Woeginger [43] developed a polynomial-time approximation scheme for the problem $\mathrm{P}\,|\,|\,\sum w_j C_j$ that improves upon the previously best known $(1+\sqrt{2})/2$–approximation algorithm due to Kawaguchi and Kyan [23]. Subsequently, Afrati et al. [1] gave polynomial-time approximation schemes for the problem $\mathrm{P}\,|\,r_j\,|\,\sum w_j C_j$, its preemptive variant $\mathrm{P}\,|\,r_j, \mathrm{pmtn}\,|\,\sum w_j C_j$, and also for the corresponding problems on a constant number of unrelated machines, $\mathrm{R}m\,|\,r_j\,|\,\sum w_j C_j$ and $\mathrm{R}m\,|\,r_j, \mathrm{pmtn}\,|\,\sum w_j C_j$. On the other hand, Hoogeveen, Schuurman and Woeginger [22] showed that the problems $\mathrm{R}\,|\,r_j\,|\,\sum C_j$ and $\mathrm{R}\,|\,|\,\sum w_j C_j$ are APX-hard; hence, they do not possess a polynomial-time approximation scheme, unless $\mathrm{P} = \mathrm{NP}$.

The remainder of this paper is organized as follows. In Section 2, we present our main result: a pseudo-polynomial time algorithm with performance guarantee 2 in the general context of unrelated parallel machine scheduling. We give a combinatorial 2–approximation algorithm for identical parallel machine scheduling in Section 3, and show how to use it in an on-line setting. Then, in Section 4, we discuss the derandomization of the previously given randomized algorithms. Finally, in Section 5, we elaborate on the details of turning the pseudo-polynomial time algorithm of Section 2 into a polynomial-time algorithm with performance guarantee $2+\varepsilon$. We conclude by pointing out some open problems in Section 6.

2. Scheduling Unrelated Parallel Machines with Release Dates. In this section, we consider the problem $\mathrm{R}\,|\,r_j\,|\,\sum w_jC_j$. As in [20, 32, 41], we will actually discuss a slightly more general problem, in which the release date of job j may also depend on the machine i to which j is assigned, and is thus denoted by r_{ij}. Machine-dependent release dates are relevant to model situations in which parallel machines are connected by a network, each job is located at a given machine at date 0, and cannot be started on another machine until sufficient time elapses to allow the job to be transmitted to its new machine. This model, called network scheduling, was introduced in [2, 10].

The problem $\mathrm{R}\,|\,r_{ij}\,|\,\sum w_jC_j$ is strongly NP-hard; in fact, $\mathrm{P}\,2\,|\,|\,\sum w_jC_j$ is NP-hard, and $1\,|\,r_j\,|\,\sum C_j$ as well as $\mathrm{P}\,|\,|\,\sum w_jC_j$ are strongly NP-hard [3, 25]. Phillips, Stein and Wein presented the first non-trivial approximation algorithm for this problem [32]. It has performance guarantee $\mathrm{O}(\log^2 n)$. Subsequently, Hall et al. [20] gave a 16/3–approximation algorithm that relies on an interval-indexed LP relaxation whose optimal value serves as a surrogate for the true optimum in their estimations. We use a related LP relaxation and construct feasible schedules from LP solutions by randomized rounding, whereas Hall et al. invoke the deterministic rounding technique of Shmoys and Tardos [39].

Let $T+1=\max_{i,j} r_{ij}+\sum_{j\in J}\max_i p_{ij}$ be the time horizon. We introduce for every job $j\in J$, every machine $i=1,\dots,m$, and every point $t=r_{ij},\dots,T$ in time a variable y_{ijt} that represents the amount of time job j is processed on machine i within the time interval $(t,t+1]$. Equivalently, one can say that a y_{ijt}/p_{ij}–fraction of job j is being processed on machine i within the time interval $(t,t+1]$. The LP relaxation, which is an extension of a single-machine LP relaxation considered by Dyer and Wolsey [11], is as follows:

$$
(\mathrm{LP_R})\qquad
\begin{aligned}
\min\quad & \sum_{j\in J} w_j C_j^{LP} && \\
\text{s. t.}\quad & \sum_{i=1}^{m}\sum_{t=r_{ij}}^{T}\frac{y_{ijt}}{p_{ij}}=1 && \text{for all } j, && (2.1)\\
& \sum_{j\in J} y_{ijt}\leqslant 1 && \text{for all } i \text{ and } t, && (2.2)\\
& C_j^{LP}\geqslant \sum_{i=1}^{m}\sum_{t=r_{ij}}^{T}\Big(\frac{y_{ijt}}{p_{ij}}\big(t+\tfrac12\big)+\tfrac12\,y_{ijt}\Big) && \text{for all } j, && (2.3)\\
& C_j^{LP}\geqslant \sum_{i=1}^{m}\sum_{t=r_{ij}}^{T} y_{ijt} && \text{for all } j, && (2.4)\\
& y_{ijt}\geqslant 0 && \text{for all } i,\ j, \text{ and } t. &&
\end{aligned}
$$

Equations (2.1) ensure that the processing requirement of every job is satisfied. The Machine Capacity Constraints (2.2) express that each machine can process at most one job at a time. For (2.3), consider an arbitrary feasible schedule S in which job j is being continuously processed between date $C_j^S - p_{hj}$ and C_j^S on machine h. Then, the right-hand side of (2.3) corresponds to the real completion time C_j^S of j if we assign the values to the LP variables y_{ijt} as defined above, i.e., $y_{ijt} = 1$ if $i = h$ and $t \in \{C_j^S - p_{hj}, \dots, C_j^S - 1\}$, and $y_{ijt} = 0$ otherwise. The right-hand side of (2.4) equals the processing time p_{hj} of job j in the schedule S, and is therefore a lower bound on its completion time C_j^S. Hence, ($\mathrm{LP_R}$) is a relaxation of the scheduling problem $\mathrm{R}\,|\,r_{ij}\,|\sum w_j C_j$. In fact, even the corresponding mixed-integer program, where the y–variables are forced to be binary, is only a relaxation — it allows preemptions of jobs and a job may use the capacity of more than one machine at a time.

Due to the exponentially large number of variables, the Linear Programming Relaxation ($\mathrm{LP_R}$) cannot be solved in time polynomial in the input size of an instance of the problem $\mathrm{R}\,|\,r_{ij}\,|\sum w_j C_j$. Therefore, the running time of the following algorithm, which turns an optimal LP solution into a feasible schedule, is only pseudo-polynomial. In particular, the results on the quality of the computed schedule that we prove in the remainder of this section do not directly lead to an approximation algorithm for the considered scheduling problem. However, we can overcome this drawback by introducing new variables that are not associated with exponentially many time intervals of length 1, but rather with a polynomial number of intervals of geometrically increasing size. We discuss the technical details of this remedy in Section 5.

The following algorithm takes an optimal LP solution, and then constructs a feasible schedule by using a variant of randomized rounding.

Algorithm LP Rounding

(1) Compute an optimal solution y to ($\mathrm{LP_R}$).
(2) For all $j \in J$, assign job j to a machine-time pair (i,t), where the machine-time pair is chosen from the probability distribution that assigns job j to (i,t) with probability $\frac{y_{ijt}}{p_{ij}}$; set $t_j := t$.
(3) Schedule on each machine i the jobs that were assigned to it non-preemptively as early as possible in order of non-decreasing t_j; ties are broken independently at random.

In the analysis of the algorithm, we assume that the random decisions for different jobs in the second step are pairwise independent.

Lemma 2.1. *The expected completion time $E[C_j]$ of job j in the schedule constructed by Algorithm* LP Rounding *can be bounded from above by*

$$\mathrm{E}[C_j] \;\leqslant\; 2\sum_{i=1}^{m}\sum_{t=r_{ij}}^{T}\frac{y_{ijt}}{p_{ij}}(t+\tfrac{1}{2}) + \sum_{i=1}^{m}\sum_{t=r_{ij}}^{T} y_{ijt} \;.$$

If all jobs are released at date 0 on all machines, the following stronger bound holds:

$$\mathrm{E}[C_j] \;\leqslant\; \sum_{i=1}^{m}\sum_{t=0}^{T}\frac{y_{ijt}}{p_{ij}}(t+\tfrac{1}{2}) + \sum_{i=1}^{m}\sum_{t=0}^{T} y_{ijt} \;. \tag{2.5}$$

Proof. We start by analyzing the structure of a schedule produced by Algorithm LP Rounding. We consider an arbitrary, but fixed job $j \in J$ and denote the machine-time pair to which job j has been assigned by (i,t). Let $\tau \geqslant 0$ be the earliest point in time such that there is no idle time in the constructed schedule during $(\tau, C_j]$

on machine i. Let K be the set of jobs processed in this time interval on machine i; hence

$$\sum_{k \in K} p_{ik} \;=\; C_j - \tau \;. \tag{2.6}$$

Since all jobs $k \in K$ are started no later than j, they must have been assigned to machine-time pairs (i, t_k) with $t_k \leqslant t$. In particular, $r_{ik} \leqslant t_k \leqslant t$ for all $k \in K$, and therefore $\tau \leqslant t$. Together with (2.6) we obtain

$$C_j \;\leqslant\; t + \sum_{k \in K} p_{ik} \;.$$

To analyze the expected completion time $\mathrm{E}[C_j]$ of job j, we first keep the assignment of j to machine-time pair (i, t) fixed and prove a bound on the conditional expectation $\mathrm{E}_{i,t}[C_j]$:

$$\begin{aligned}
\mathrm{E}_{i,t}[C_j] \;\leqslant\; t + \mathrm{E}_{i,t}\Big[\sum_{k \in K} p_{ik}\Big] \;&\leqslant\; t + p_{ij} + \sum_{k \neq j} p_{ik} \cdot \mathrm{Pr}_{i,t}[k \text{ on } i \text{ before } j] \\
&=\; t + p_{ij} + \sum_{k \neq j} p_{ik} \Big(\sum_{\ell = r_{ik}}^{t-1} \frac{y_{ik\ell}}{p_{ik}} + \frac{1}{2}\, \frac{y_{ikt}}{p_{ik}} \Big)
\end{aligned}$$

(Note that the factor $\frac{1}{2}$ before the term $\frac{y_{ikt}}{p_{ik}}$ results from breaking ties randomly.)

$$\leqslant\; t + p_{ij} + (t + \tfrac{1}{2}) \;\leqslant\; 2\,(t + \tfrac{1}{2}) + p_{ij} \;.$$

The second but last inequality follows from the Machine Capacity Constraints (2.2). Finally, unconditioning the expectation by the formula of total expectation leads to

$$\mathrm{E}[C_j] \;=\; \sum_{i=1}^{m} \sum_{t=r_{ij}}^{T} \frac{y_{ijt}}{p_{ij}}\, \mathrm{E}_{i,t}[C_j] \;\leqslant\; 2 \sum_{i=1}^{m} \sum_{t=r_{ij}}^{T} \frac{y_{ijt}}{p_{ij}} (t + \tfrac{1}{2}) + \sum_{i=1}^{m} \sum_{t=r_{ij}}^{T} y_{ijt} \;.$$

In the absence of non-trivial release dates, we can prove a stronger bound. Observe that $\tau = 0$ and $C_j = \sum_{k \in K} p_{ik}$ in this case. This yields $\mathrm{E}_{i,t}[C_j] \leqslant p_{ij} + (t + \frac{1}{2})$ and eventually the claimed result. ∎

THEOREM 2.2. *For instances of* $\mathrm{R}\,|\,r_{ij}\,|\,\sum w_j C_j$*, the expected objective function value of the schedule constructed by Algorithm* LP ROUNDING *is at most twice the value of an optimal solution.*

Proof. Lemma 2.1 together with Constraints (2.3) implies that the expected completion time of every job j is bounded from above by twice its LP completion time C_j^{LP}. Since the optimal LP value is a lower bound on the value of an optimal schedule and the weights are non-negative, the result follows from linearity of expectations. ∎

Note that Theorem 2.2 still holds if we use the weaker LP relaxation where Constraints (2.4) are missing. However, this is not true for the following result.

THEOREM 2.3. *For instances of* $\mathrm{R}\,|\,|\,\sum w_j C_j$*, Algorithm* LP ROUNDING *constructs a schedule of expected objective function value at most* $3/2$ *times the value of an optimal schedule.*

Proof. The claimed result follows from Lemma 2.1 and LP Constraints (2.3) and (2.4). ∎

In the absence of non-trivial release dates, Algorithm LP ROUNDING can be improved and simplified:

ALGORITHM LP SIMPLE ROUNDING

(1) Compute an optimal solution y to ($\mathrm{LP_R}$).
(2) For all $j \in J$, assign job j to a machine i, where machine i is chosen from the probability distribution that assigns job j to i with probability $\sum_{t=0}^{T} \frac{y_{ijt}}{p_{ij}}$.
(3) Sequence on each machine i the assigned jobs in order of non-increasing ratios w_j/p_{ij}.

Again, the random decisions in Step (2) are performed pairwise independently.

COROLLARY 2.4. *For instances of* $\mathrm{R}\,|\;|\sum w_jC_j$*, the approximation result of Theorem 2.3 also holds for Algorithm* LP SIMPLE ROUNDING.

Proof. Notice that the random assignment of jobs to machines is identical in Algorithms LP ROUNDING and LP SIMPLE ROUNDING. Moreover, for a fixed assignment of jobs to machines, sequencing the jobs according to Smith's ratio rule [44] on each machine is optimal. In particular, it improves upon the random sequence used in the final step of Algorithm LP ROUNDING. ◻

In the analysis, we have compared the value of the solution computed by Algorithm LP ROUNDING to the optimal LP value, which is a lower bound on the value of an optimal solution. Hence, we obtain the following result on the quality of the LP relaxation:

COROLLARY 2.5. *The linear program* (LP_R) *is a* 2*–relaxation for* $\mathrm{R}\,|\,r_{ij}\,|\sum w_jC_j$ *(even without Constraints* (2.4)*), and a* $\frac{3}{2}$*–relaxation for* $\mathrm{R}\,|\;|\sum w_jC_j$.

We show in the following section that ($\mathrm{LP_R}$) without Constraints (2.4) is not better than a 2–relaxation, even for instances of $\mathrm{P}\,|\;|\sum w_jC_j$. On the other hand, the relaxation can be strengthened by adding the constraints

$$\sum_{i=1}^{m} y_{ijt} \leqslant 1 \qquad \text{for } j \in J,\ t = 0,\dots,T. \tag{2.7}$$

These constraints ensure that no job can use the capacity of more than one machine in each time period. We do not know whether these constraints can be used to get provably stronger results on the quality of the LP relaxation and better performance guarantees for Algorithm LP ROUNDING.

In Section 5, we eventually derive from Theorems 2.2 and 2.3 a $(2+\varepsilon)$–approximation algorithm for the problem $\mathrm{R}\,|\,r_{ij}\,|\sum w_jC_j$, and a $(3/2+\varepsilon)$–approximation algorithm for $\mathrm{R}\,|\;|\sum w_jC_j$.

The techniques presented in this section (and in Section 5) can be modified to design approximation algorithms for the corresponding preemptive scheduling problems as well. Notice that, although the LP Relaxation ($\mathrm{LP_R}$) allows preemptions of jobs, it is not a relaxation of $\mathrm{R}\,|\,r_{ij},\mathrm{pmtn}\,|\sum w_jC_j$; one can easily show (see, e.g., [40, Example 2.10.8.]) that the right-hand side of (2.3) can in fact overestimate the actual completion time of a job in the preemptive schedule corresponding to a solution of ($\mathrm{LP_R}$). However, replacing (2.3) with the following slightly weaker constraint yields an LP relaxation for the preemptive scheduling problem:

$$C_j^{LP} \geqslant \sum_{i=1}^{m} \sum_{t=r_{ij}}^{T} \frac{y_{ijt}}{p_{ij}} \left(t + \tfrac{1}{2}\right) \qquad \text{for all } j \in J.$$

This leads to a $(3+\varepsilon)$–approximation algorithm for $\mathrm{R}\,|\,r_{ij},\mathrm{pmtn}\,|\sum w_jC_j$ and a $(2+\varepsilon)$–approximation algorithm for $\mathrm{R}\,|\,\mathrm{pmtn}\,|\sum w_jC_j$. These results can again be slightly improved by using convex quadratic programming relaxations, see [41].

In the next section, we consider the special case of identical parallel machines and give a different interpretation of Algorithm LP ROUNDING in terms of so-called α–points. The following variant of Algorithm LP ROUNDING will turn out to be useful in this context.

REMARK 2.6. *The following is an equivalent way of breaking ties randomly in the last step of Algorithm* LP ROUNDING*: At the end of the second step, draw t_j from the interval $(t, t+1]$ independently at random with uniform distribution; then, in the last step, ties occur with probability zero and can therefore be neglected.*

3. Identical Parallel Machine Scheduling with Release Dates. We now consider the special case of m *identical* parallel machines. The processing time and the release date of job j do not anymore depend on the machine job j is assigned to; consequently, they are denoted by p_j and r_j, respectively. As mentioned before, even $\mathrm{P}2 \,|\,|\, \sum w_j C_j$ is NP-hard. We consider $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$.

In this context, we can turn Algorithm LP ROUNDING into a purely combinatorial algorithm. Following earlier work (e.g., Eastman, Even, and Isaacs [12]), we first reduce an identical parallel machine instance to a single machine instance. Here, the single machine is assumed to be m times as fast as each of the original m machines, i.e., the processing time of job j on this virtual single machine is $p'_j := p_j/m$. (We assume without loss of generality that p_j is a multiple of m.) Its weight and its release date remain the same. The crucial part of our algorithm is to assign jobs to machines uniformly at random. Then, on each machine, we schedule the assigned jobs in order of random α–points with respect to the LP schedule on the fast single machine.

For $0 < \alpha \leqslant 1$, the *α–point* $C_j^S(\alpha)$ of job j with respect to a given preemptive schedule S on the fast single machine is the first point in time when an α–fraction of job j has been completed, i.e., when j has been processed for $\alpha \cdot p'_j$ time units. In particular, $C_j^S(1) = C_j^S$, and for $\alpha = 0$ we define $C_j^S(0)$ to be the starting time of job j. Slightly varying notions of α-points were considered in [21, 33], but their full potential was revealed when Chekuri et al. [7] as well as Goemans [16] chose the parameter α at random. The following procedure may be seen as an extension of their single-machine algorithms to identical parallel machines.

ALGORITHM RANDOM ASSIGNMENT

(1) Construct the preemptive LP schedule S on the virtual single machine by scheduling, at any point in time, among the already released, but not yet completed jobs the one with largest w_j/p'_j ratio.
(2) Independently, for all $j \in J$, assign job j to a machine $i \in \{1, \dots, m\}$, where machine i is chosen from the probability distribution that assigns job j to i with probability $\frac{1}{m}$.
(3) For all $j \in J$, let α_j be a realization of an independent, uniformly distributed random variable in $[0, 1]$.
(4) Schedule on each machine i the assigned jobs non-preemptively as early as possible in order of non-decreasing $C_j^S(\alpha_j)$.

Notice that in the first step, whenever a job is released, the job being processed (if any) will be preempted if the released job has a larger w_j/p'_j ratio; here, the p'_j values are the original processing times and not the remaining processing times. The Appendix provides an illustration of Algorithm RANDOM ASSIGNMENT. Its running time is dominated by the computation of the preemptive LP schedule in the first step, which can be done in $\mathrm{O}(n \log n)$ time using a priority queue [16].

We will show in the following that Algorithm RANDOM ASSIGNMENT can be interpreted as the reformulation of Algorithm LP ROUNDING discussed in Remark 2.6. We notice first that the preemptive LP schedule on the virtual single machine corresponds to an optimal solution to an LP relaxation which is equivalent to ($\mathrm{LP_R}$). We introduce a variable y_{jt} for every job j and every time period $(t, t+1]$, $t = r_j, \dots, T$; the variable y_{jt} is set to $1/m$ if job j is being processed on one of the m machines in this period, and to 0 otherwise. In contrast to the unrelated parallel machine case, we do not need machine-dependent variables since it is not necessary to distinguish between the identical parallel machines. We can express the new variables y_{jt} with the help of the old variables y_{ijt} by setting $y_{jt} = \frac{1}{m}(y_{1jt} + \cdots + y_{mjt})$ for $j \in J$, $t = r_j, \dots, T$. This leads to the following simplified LP (ignoring Constraints (2.4) of ($\mathrm{LP_R}$)):

$$
(\mathrm{LP_P}) \qquad
\begin{array}{lll}
\min & \displaystyle\sum_{j \in J} w_j C_j^{LP} & \\
\text{s. t.} & \displaystyle\sum_{t=r_j}^{T} y_{jt} = p'_j & \text{for all } j \in J, \\
& \displaystyle\sum_{j \in J} y_{jt} \leqslant 1 & \text{for } t = 0, \dots, T, \\
& \displaystyle C_j^{LP} = \frac{p_j}{2} + \frac{1}{p'_j} \sum_{t=r_j}^{T} y_{jt}\big(t + \tfrac{1}{2}\big) & \text{for all } j \in J, \\
& y_{jt} \geqslant 0 & \text{for all } j \in J \text{ and } t = r_j, \dots, T.
\end{array}
$$

Dyer and Wolsey noticed that this linear program can be solved in $\mathrm{O}(n \log n)$ time for the special case $m = 1$ [11]. Goemans showed (also for the case $m = 1$) that the preemptive schedule that is constructed in the first step of Algorithm RANDOM ASSIGNMENT defines an optimal solution to ($\mathrm{LP_P}$) [15]. This result as well as its proof easily generalize to an arbitrary number of identical parallel machines:

LEMMA 3.1. *For instances of* $\mathrm{P} \mid r_j \mid \sum w_j C_j$, *the preemptive LP schedule on the fast single machine is an optimal solution to the Relaxation* (LP_P). *Moreover, it can be computed in* $O(n \log n)$ *time.*

THEOREM 3.2. RANDOM ASSIGNMENT *is a randomized* 2*–approximation algorithm for* $\mathrm{P} \mid r_j \mid \sum w_j C_j$.

Proof. We show that Algorithm RANDOM ASSIGNMENT can be interpreted as a special case of the variant of Algorithm LP ROUNDING discussed in Remark 2.6. The result then follows from its polynomial running time and Theorem 2.2.

Lemma 3.1 implies that we compute in the first step of Algorithm RANDOM ASSIGNMENT an optimal solution to the LP Relaxation ($\mathrm{LP_P}$), which is equivalent to ($\mathrm{LP_R}$) without Constraints (2.4). In particular, the corresponding solution to ($\mathrm{LP_R}$) is symmetric with regard to the m machines. Therefore, Algorithm LP ROUNDING assigns each job uniformly at random to one of the machines. The symmetry also yields that for each job j the choice of t_j is not correlated with the choice of i in Algorithm LP ROUNDING.

We observe next that the probability distributions of the random variable t_j in Algorithm LP ROUNDING and of $C_j^S(\alpha_j)$ in Algorithm RANDOM ASSIGNMENT are the same. In fact, the probability that $C_j^S(\alpha_j) \in (t, t+1]$ for some t equals the fraction y_{jt}/p'_j of job j that is being processed in this time interval. Moreover, each point in $(t, t+1]$ is equally likely to be obtained by $C_j^S(\alpha_j)$. Therefore, the random choice of $C_j^S(\alpha_j)$ in Algorithm RANDOM ASSIGNMENT is an alternate way of choosing t_j as described in Algorithm LP ROUNDING. Consequently, the two algorithms coincide for the identical parallel machine case. The result eventually follows from Theorem 2.2. ∎

At this point, let us briefly compare the approximation results of this section for the single machine case ($m = 1$) with related results. If we only work with one α for all jobs instead of individual and independent α_j's and if we draw α uniformly from $[0, 1]$, then RANDOM ASSIGNMENT coincides with Goemans' randomized 2–approximation algorithm RANDOM$_\alpha$ for $1 \,|\, r_j \,|\, \sum w_j C_j$ [16]. Goemans, Queyranne, Schulz, Skutella, and Wang have improved this result to performance guarantee 1.6853 by using job-dependent α_j's (as in Algorithm RANDOM ASSIGNMENT) together with a non-uniform choice of the α_j's [17]. The latter idea can also be applied in the parallel machine setting to obtain a performance guarantee better than 2 for Algorithm RANDOM ASSIGNMENT. However, this improvement depends on m. A comprehensive overview of the use of α–points for machine scheduling problems can be found in [40, Chapter 2].

We have already argued in the previous section that ($\mathrm{LP_R}$), and thus ($\mathrm{LP_P}$) is a 2–relaxation of the scheduling problem under consideration:

COROLLARY 3.3. *The Relaxation* (LP_{P}) *is a 2–relaxation of the scheduling problem* $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$. *This bound is tight, even for* $\mathrm{P} \,|\,|\, \sum w_j C_j$.

Proof. The positive result follows from Corollary 2.5. For the tightness of this bound, consider an instance with m machines and one job of length m and unit weight. The optimal LP completion time is $(m + 1)/2$, whereas the optimal completion time is m. When m goes to infinity, the ratio of the two values converges to 2. ∎

The following lemma helps to extend Theorem 3.2 and Corollary 3.3 to the corresponding preemptive scheduling problem.

LEMMA 3.4. *Linear program* (LP_{P}) *is a relaxation of the preemptive scheduling problem* $\mathrm{P} \,|\, r_j, \, pmtn \,|\, \sum w_j C_j$.

Proof. Since all release dates and processing times are integral, there exists an optimal preemptive schedule where preemptions only occur at integral points in time. Take such an optimal schedule S and construct the corresponding feasible solution to ($\mathrm{LP_P}$) by setting $y_{jt} = 1/m$ if job j is being processed on one of the m machines within the interval $(t, t + 1]$, and $y_{jt} = 0$ otherwise. We observe that $C_j^{LP} \leqslant C_j^S$, and equality holds if and only if job j is continuously processed in the interval $(C_j^S - p_j, C_j^S]$. Thus, the value of the constructed solution to ($\mathrm{LP_P}$) is a lower bound on the value of an optimal schedule. ∎

COROLLARY 3.5. *For instances of* $\mathrm{P} \,|\, r_j, \, pmtn \,|\, \sum w_j C_j$, *the value of the (non-preemptive) schedule produced by Algorithm* RANDOM ASSIGNMENT *is at most twice the value of an optimal preemptive schedule. Moreover,* (LP_{P}) *is a 2–relaxation of this scheduling problem. This bound is tight.*

Another consequence is the following result on the power of preemption:

COROLLARY 3.6. *For identical parallel machine scheduling with release dates so as to minimize the weighted sum of completion times, the value of an optimal non-preemptive schedule is at most twice the value of an optimal preemptive schedule.*

Moreover, for identical parallel machines, Steps (2)–(3) of Algorithm LP ROUNDING can be used to convert an arbitrary preemptive schedule into a non-preemptive one such that the objective function value increases at most by a factor of 2: for a given preemptive schedule, construct the corresponding solution to ($\mathrm{LP_R}$). The value of this feasible solution to the LP relaxation is a lower bound on the value of the given preemptive schedule. (As discussed at the end of Section 2, this is in general not true for unrelated parallel machines.) Using Algorithm LP ROUNDING, the solution to ($\mathrm{LP_R}$) can be turned into a non-preemptive schedule whose expected value is bounded by twice the value of the LP solution, and thus by twice the value of the preemptive schedule we started with. This improves upon a bound of 7/3 due to Phillips et al. [31].

Several different on-line paradigms have been studied in the area of scheduling, see [38] for a survey. We consider the setting where the only on-line feature is the lack of knowledge of jobs arriving in the future. In particular, the processing time and the weight of a job become known at its arrival. Let us show that Algorithm RANDOM ASSIGNMENT can easily be turned into an on-line algorithm. First, note that we can immediately determine α_j when job j is released; this drawing does not depend on any other decision of the algorithm. The same argument holds for the random machine assignment. Moreover, we can construct the LP schedule until time t without any knowledge of jobs that are released afterwards. Finally, it follows from the analysis in the proof of Lemma 2.1 that we obtain the same performance guarantee if job j is not started before time $t_j = C_j^S(\alpha_j)$. Thus, we modify Step (4) in the on-line variant of Algorithm RANDOM ASSIGNMENT: on each machine we schedule the assigned jobs as early as possible in order of non-decreasing $C_j^S(\alpha_j)$, with the additional constraint that no job j may start before time $C_j^S(\alpha_j)$. This technique was introduced in a similar context in [33].

COROLLARY 3.7. *The on-line variant of Algorithm* RANDOM ASSIGNMENT *has competitive ratio* 2.

One appealing aspect of Algorithm RANDOM ASSIGNMENT is that the assignment of jobs to machines does not depend on job characteristics; a job is put with probability $1/m$ to any of the machines. This technique also proves useful for the problem without release dates:

THEOREM 3.8. *Assigning jobs independently and uniformly at random to the machines and then applying Smith's ratio rule on each machine is a $3/2$–approximation algorithm for* $\mathrm{P} \mid \mid \sum w_j C_j$. *There exist instances for which this bound is asymptotically tight.*

Proof. First, notice that the described algorithm is identical to Algorithm RANDOM ASSIGNMENT, and therefore to the variant of LP ROUNDING discussed in Remark 2.6. Because of the negative result in Corollary 3.3, we cannot derive the bound of 3/2 by comparing the expected value of the computed solution to the optimal value of ($\mathrm{LP_P}$). Remember that we used a stronger relaxation including Constraints (2.4) in order to derive this bound in the unrelated parallel machine setting. However, Lemma 2.1 implies that

$$\mathrm{E}[C_j] \leqslant C_j^{LP} + \tfrac{1}{2} p_j$$

because the second term on the right-hand side of (2.5) is equal to p_j for the case of identical parallel machines. Since both $\sum_j w_j C_j^{LP}$ and $\sum_j w_j p_j$ are lower bounds on the value of an optimal solution, the result follows.

In order to show that this performance guarantee is tight, we consider instances with m identical parallel machines and m jobs of unit length and weight. We obtain an optimal schedule with value m by assigning one job to each machine. On the other hand, we can show that the expected completion time of a job in the schedule constructed by random machine assignments is $\frac{3}{2} - \frac{1}{2m}$, which converges to $\frac{3}{2}$ for increasing m. Since the ratio w_j/p_j equals 1 for all jobs, we can without loss of generality schedule on each machine the jobs that were assigned to it in random order.

Consider a fixed job j and the machine i it has been assigned to. The probability that a job $k \neq j$ was assigned to the same machine is $1/m$. In this case, job k is processed before j with probability $1/2$. We therefore get $\mathrm{E}[C_j] = 1 + \sum_{k \neq j} \frac{1}{2m} = \frac{3}{2} - \frac{1}{2m}$. □

Interestingly, the derandomized variant of the algorithm considered in Theorem 3.8 coincides with the WSPT-rule: sort the jobs according to non-increasing ratios w_j/p_j, and schedule the next job from this list whenever a machine becomes available. Kawaguchi and Kyan proved that this algorithm has performance guarantee $(1+\sqrt{2})/2 \approx 1.21$ [23]. While their proof is somewhat intricate, our simpler, probabilistic analysis yields a performance guarantee of $3/2$. However, this weaker result also follows from the work of Eastman et al. [12]. They gave a combinatorial lower bound for $\mathrm{P}\,|\,|\sum w_j C_j$ that coincides with the lower bound obtained from $(\mathrm{LP_P})$. The latter observation is due to Uma and Wein [46] and Williamson [49]. Details on the derandomization are given in the next section.

4. Derandomization. The hitherto presented algorithms are randomized and compute feasible schedules whose expected value can be bounded from above. While this shows that our algorithms perform well on average, we cannot give firm guarantees for the performance of a single execution. In certain situations, it may be more desirable to have deterministic algorithms with bounded worst-case ratio. Fortunately, there exists a deterministic version of every algorithm proposed in this paper that maintains the performance guarantee of its randomized companion, as we are about to illustrate now. We can derandomize the randomized algorithms in this paper by using the method of conditional probabilities. The method of conditional probabilities is one of the most important techniques for derandomization. This method is implicitly contained in a paper of Erdös and Selfridge [14], and was extended to a more general context by Spencer [45]. It considers the random decisions one after another and chooses the most promising alternative at every decision point. Here, it is assumed that all remaining decisions are random. Thus, an alternative is said to be most promising if the corresponding conditional expected objective function value is smallest. We shall demonstrate this technique for the most general problem, $\mathrm{R}\,|\,r_{ij}\,|\sum w_j C_j$, and Algorithm LP ROUNDING.

Our analysis of Algorithm LP ROUNDING in the proof of Lemma 2.1 does not give a precise expression for the expected value of the computed solution, but only an upper bound. Hence, we modify Algorithm LP ROUNDING by replacing its last step with the following variant:

(3') Schedule on each machine i the assigned jobs non-preemptively in order of non-decreasing t_j; ties are broken by preferring jobs with smaller indices. At the starting time of job j, the amount of idle time on its machine has to be exactly t_j.

Since $r_{ij} \leqslant t_j$ for each job j that has been assigned to machine i and $t_j \leqslant t_k$ if job k is scheduled after job j, Step (3') defines a feasible schedule. In the proof of Lemma 2.1, we have bounded the idle time before the start of job j from above by t_j. Thus, the analysis still works for the modified version of Algorithm LP ROUNDING. The advantage of the modification is that we are now in a position to give precise expressions for the expectations and conditional expectations of completion times.

Let y be an optimal solution to $(\mathrm{LP_R})$. Using the same arguments as in the proof of Lemma 2.1, we obtain the following expression for the expected completion time of job j in the schedule output by the modified Algorithm LP Rounding:

$$\mathrm{E}[C_j] \;=\; \sum_{i=1}^{m}\sum_{t=r_{ij}}^{T}\frac{y_{ijt}}{p_{ij}}\Big(p_{ij}+t+\sum_{k\neq j}\sum_{\ell=r_{ik}}^{t-1}y_{ik\ell}+\sum_{k<j}y_{ikt}\Big)\;.$$

Moreover, we are also interested in the conditional expectation of j's completion time if some of the jobs have already been assigned to a machine-time pair. Let $K \subseteq J$ be such a subset of jobs. For each job $k \in K$, let the 0/1–variable x_{ikt} for $t \geqslant r_{ik}$ indicate whether k has been assigned to the machine-time pair (i,t) (i.e., $x_{ikt}=1$) or not ($x_{ikt}=0$). This enables us to give the following expressions for the conditional expectation of j's completion time. If $j \notin K$, we have

$$\begin{aligned}\mathrm{E}_{K,x}[C_j] \;=\; \sum_{i=1}^{m}\sum_{t=r_{ij}}^{T}\frac{y_{ijt}}{p_{ij}}\Big(p_{ij}+t+\sum_{k\in K}\sum_{\ell=r_{ik}}^{t-1}x_{ik\ell}p_{ik}+\sum_{k\in K,\,k<j}x_{ikt}p_{ik}\\ +\sum_{k\in J\setminus(K\cup\{j\})}\sum_{\ell=r_{ik}}^{t-1}y_{ik\ell}+\sum_{k\in J\setminus K,\,k<j}y_{ikt}\Big)\;,\end{aligned} \tag{4.1}$$

and, if $j \in K$, we obtain

$$\begin{aligned}\mathrm{E}_{K,x}[C_j] \;&=p_{ij}+t+\sum_{k\in K}\sum_{\ell=r_{ik}}^{t-1}x_{ik\ell}p_{ik}+\sum_{k\in K,\,k<j}x_{ikt}p_{ik}\\ &\quad+\sum_{k\in J\setminus K}\sum_{\ell=r_{ik}}^{t-1}y_{ik\ell}+\sum_{k\in J\setminus K,\,k<j}y_{ikt}\;,\end{aligned} \tag{4.2}$$

where (i,t) is the machine-time pair job j has been assigned to, i. e., $x_{ijt}=1$. The following lemma is the most important part of derandomizing Algorithm LP Rounding.

Lemma 4.1. *Let y be an optimal solution to (LP_R), $K \subseteq J$, and x be a fixed assignment of the jobs in K to machine-time pairs. Furthermore, let $j \in J \setminus K$. Then, there exists an assignment of j to a machine-time pair (i,t) (i. e., $x_{ijt}=1$) with $r_{ij} \leqslant t$ such that*

$$\mathrm{E}_{K\cup\{j\},x}\Big[\sum_{\ell}w_\ell C_\ell\Big] \;\leqslant\; \mathrm{E}_{K,x}\Big[\sum_{\ell}w_\ell C_\ell\Big]\;. \tag{4.3}$$

Proof. Using the formula of total expectation, the conditional expectation on the right-hand side of (4.3) can be written as a convex combination of conditional expectations $\mathrm{E}_{K\cup\{j\},x}\big[\sum_\ell w_\ell C_\ell\big]$ over all possible assignments of job j to machine-time pairs (i,t) with coefficients $\frac{y_{ijt}}{p_{ij}}$. The best one satisfies (4.3). ∎

Lemma 4.1 leads to a derandomized version of Algorithm LP Rounding if we replace the second step by

(2') Set $K := \emptyset$; $x := 0$; for all $j \in J$ do
  (i) for all possible assignments of j to machine-time pairs (i,t) (i. e., $x_{ijt}=1$) compute $\mathrm{E}_{K\cup\{j\},x}\big[\sum_\ell w_\ell C_\ell\big]$;
  (ii) determine the machine-time pair (i,t) that minimizes the conditional expectation in (i);
  (iii) set $K := K \cup \{j\}$; $x_{ijt}=1$.

Notice that we have replaced Step (3) of Algorithm LP ROUNDING by (3') only to give a more accessible analysis of its derandomization. Since the value of the schedule constructed in Step (3) is always at least as good as the one constructed in Step (3'), the following theorem can be formulated for Algorithm LP ROUNDING with the original Step (3).

THEOREM 4.2. *If we replace Step* (2) *in Algorithm* LP ROUNDING *with* (2'), *we obtain a deterministic algorithm with performance guarantee* 2 *for* $\mathrm{R}\,|\,r_{ij}\,|\sum w_jC_j$, *and with performance guarantee* $3/2$ *for* $\mathrm{R}\,|\,|\sum w_jC_j$. *Moreover, the running time of this algorithm is polynomial in the number of variables of* (LP_R).

Proof. The result follows by an inductive use of Lemma 4.1, and from Theorems 2.2 and 2.3. The computation of (4.1) and (4.2) is polynomially bounded by the number of variables. Therefore, the running time of each of the n iterations in Step (2') is polynomially bounded by this number as well. □

The same derandomization process also works for the polynomial-time approximation algorithms in Section 5 that are based on interval-indexed LP relaxations. Since these LP relaxations only contain a polynomial number of variables, the running time of the derandomized algorithms is polynomially bounded in the input size of the scheduling problem.

The derandomization of Algorithm RANDOM ASSIGNMENT for $\mathrm{P}\,|\,|\sum w_jC_j$ by the method of conditional probabilities leads to an interesting result, as indicated at the end of Section 3. It essentially follows from the considerations above that the derandomized version of this algorithm assigns each job to the machine with the smallest load. If we consider the jobs in order of non-increasing ratios w_j/p_j, the resulting algorithm coincides with the WSPT-rule.

5. Interval-Indexed LP Relaxations. We pointed out earlier that the LP-based Algorithms LP ROUNDING and LP SIMPLE ROUNDING for unrelated parallel machine scheduling suffer from the exponential number of variables in the corresponding LP Relaxation ($\mathrm{LP_R}$). However, we can overcome this drawback by using new variables that are not associated with exponentially many time intervals of length 1, but with a polynomial number of intervals of geometrically increasing size. This idea was earlier introduced by Hall et al. [21]. We show that Algorithm LP ROUNDING can be turned into a polynomial-time algorithm for $\mathrm{R}\,|\,r_{ij}\,|\sum w_jC_j$ at the cost of an increased performance guarantee of $2+\varepsilon$. The same technique can be used to modify Algorithm LP SIMPLE ROUNDING to a $(3/2+\varepsilon)$–approximation algorithm for $\mathrm{R}\,|\,|\sum w_jC_j$.

For a given $\eta > 0$, the number L is chosen to be the smallest integer such that $(1+\eta)^L \geqslant T+1$. Consequently, L is polynomially bounded in the input size of the considered scheduling problem. Let $I_0 = \big[0,1\big]$ and for $1 \leqslant \ell \leqslant L$ let $I_\ell = \big((1+\eta)^{\ell-1},(1+\eta)^\ell\big]$. We denote with $|I_\ell|$ the length of the ℓ–th interval, i. e., $|I_\ell| = \eta(1+\eta)^{\ell-1}$ for $1 \leqslant \ell \leqslant L$. To simplify notation we define $(1+\eta)^{\ell-1}$ to be $\frac{1}{2}$ for $\ell = 0$. We introduce variables $y_{ij\ell}$ for $i = 1,\dots,m$, $j \in J$, and $(1+\eta)^{\ell-1} \geqslant r_{ij}$ with the following interpretation: $y_{ij\ell}\cdot|I_\ell|$ is the time job j is processed on machine i within time interval I_ℓ or, equivalently, $(y_{ij\ell}\cdot|I_\ell|)/p_{ij}$ is the fraction of job j that is being processed on machine i within I_ℓ. Consider the following linear program in these interval-indexed variables:

$$\begin{aligned}
&\min && \sum_{j\in J} w_j C_j \\
&\text{s. t.} && \sum_{i=1}^{m} \sum_{\substack{\ell=0\\(1+\eta)^{\ell-1}\geqslant r_{ij}}}^{L} \frac{y_{ij\ell}\cdot|I_\ell|}{p_{ij}} = 1 && \text{for all } j, && (5.1)\\
&(\mathrm{LP}_\mathrm{R}^\eta) && \sum_{j\in J} y_{ij\ell} \leqslant 1 && \text{for all } i \text{ and } \ell, && (5.2)\\
& && C_j = \sum_{i=1}^{m} \sum_{\substack{\ell=0\\(1+\eta)^{\ell-1}\geqslant r_{ij}}}^{L} \Big(\frac{y_{ij\ell}\cdot|I_\ell|}{p_{ij}}(1+\eta)^{\ell-1} + \tfrac{1}{2}\cdot y_{ij\ell}\cdot|I_\ell|\Big) && \text{for all } j, && (5.3)\\
& && y_{ij\ell} \geqslant 0 && \text{for all } i,\, j, \text{ and } \ell.
\end{aligned}$$

Consider a feasible schedule and assign the values to the variables $y_{ij\ell}$ as defined above. This solution is clearly feasible: Constraints (5.1) are satisfied since a job j consumes p_{ij} time units if it is processed on machine i; Constraints (5.2) are satisfied since the total processing time of jobs that are processed within the interval I_ℓ on machine i cannot exceed its length. Finally, if job j is continuously being processed between $C_j - p_{hj}$ and C_j on machine h, then the right-hand side of Equation (5.3) is a lower bound on the real completion time. Thus, $(\mathrm{LP}_\mathrm{R}^\eta)$ is a relaxation of the scheduling problem $\mathrm{R}\,|\,r_{ij}\,|\,\sum w_j C_j$. Since $(\mathrm{LP}_\mathrm{R}^\eta)$ is of polynomial size, an optimal solution can be computed in polynomial time. We rewrite Algorithm LP ROUNDING for the new LP:

ALGORITHM LP ROUNDING

(1) Compute an optimal solution y to $(\mathrm{LP}_\mathrm{R}^\eta)$.
(2) Independently, for all $j \in J$, assign job j to a machine-interval pair (i, I_ℓ), where the machine-interval pair is chosen from the probability distribution that assigns job j to (i, I_ℓ) with probability $\frac{y_{ij\ell}\cdot|I_\ell|}{p_{ij}}$; set t_j to the left end-point $(1+\eta)^{\ell-1}$ of the time interval I_ℓ.
(3) Schedule on each machine i the assigned jobs in order of non-decreasing t_j; ties are randomly broken.

THEOREM 5.1. *The expected completion time of each job j in the schedule constructed by Algorithm* LP ROUNDING *is at most* $2\cdot(1+\eta)\cdot C_j^{LP}$.

Proof. We argue almost exactly as in the proof of Lemma 2.1. We consider an arbitrary, but fixed job $j \in J$. We also consider a fixed assignment of j to machine i and time interval I_ℓ. Again, the conditional expectation of j's starting time equals the expected idle time plus the expected processing time on machine i before j is started.

With similar arguments as in the proof of Lemma 2.1, we can bound the sum of the idle time plus the processing time by $2\cdot(1+\eta)\cdot(1+\eta)^{\ell-1}$. This, together with the expected processing time of job j itself, which is $\sum_{i=1}^m \sum_{\ell=0}^L y_{ij\ell}\cdot|I_\ell|$, and (5.3) yields the theorem. ∎

For any given $\varepsilon > 0$, we can choose $\eta = \varepsilon/2$. Then, Algorithm LP ROUNDING is a $(2+\varepsilon)$–approximation algorithm for the problem $\mathrm{R}\,|\,r_{ij}\,|\,\sum w_j C_j$, and $(\mathrm{LP}_\mathrm{R}^\eta)$ is a $(2+\varepsilon)$–relaxation.

6. Concluding Remarks and Open Problems. In this paper, we have developed LP-based approximation algorithms for a variety of parallel machine scheduling problems with the average weighted completion time objective. A by-product of our analysis are results on the quality of the underlying LP relaxations.

Our central off-line result is the $(2+\varepsilon)$–approximation algorithm for the problem $\mathrm{R} \,|\, r_{ij} \,|\, \sum w_j C_j$, and there exist instances which show that the underlying LP Relaxation ((LP$_\mathrm{R}$) without Inequalities (2.4)) is indeed not better than a 2–relaxation. However, it is open whether the quality of (LP$_\mathrm{R}$) (with (2.4) and/or (2.7)) is better than 2 and also whether it can be used to derive an approximation algorithm with performance guarantee strictly less than 2. On the negative side, $\mathrm{R} \,|\, r_j \,|\, \sum C_j$ is APX-hard [22]. In other words, the best known approximation algorithm for $\mathrm{R} \,|\, r_{ij} \,|\, \sum w_j C_j$ has performance guarantee 2 (we proved $2+\varepsilon$ here and [41] gets rid of the ε using a convex quadratic relaxation), but the only known limit to its approximation is the non-existence of a polynomial-time approximation scheme, unless P = NP. The situation for $\mathrm{R} \,|\, \,|\, \sum w_j C_j$ is similar. (LP$_\mathrm{R}$) is a 3/2–relaxation, the quality of (LP$_\mathrm{R}$) together with (2.7) is unknown, the 3/2–approximation given in [41] (improving upon the $(3/2+\varepsilon)$–approximation in Section 2) is best known, and again there cannot be a PTAS, unless P = NP [22]. For identical parallel machines, one important property of our 2-approximation algorithm for $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$ is that it runs in time $\mathrm{O}(n \log n)$. The running time of the recent PTAS is $\mathrm{O}\big((m+1)^{poly(1/\varepsilon)} n + n \log n\big)$ [1]. The other important feature of the $\mathrm{O}(n \log n)$ algorithm is that it is capable of working in an on-line context as well, which brings us to the second set of open problems.

If jobs arrive over time and if the performance of algorithms is measured in terms of their competitiveness to optimal off-line algorithms, it is important to distinguish between deterministic and randomized algorithms. For identical parallel machine scheduling, there is a significant gap between the best-known lower bound and competitive ratio of a deterministic algorithm. A universal lower bound of 1.309 is proved in [47, Chapter 3], while a $(4+\varepsilon)$–competitive algorithm emerges from a more general framework given in [20]. For randomized algorithms, the situation is only slightly better. No relevant lower bound on the competitive ratio of any randomized on-line algorithm is known, and the modified version of Algorithm RANDOM ASSIGNMENT discussed in Section 3 is a randomized 2–competitive algorithm.

An interesting application of the approximation results for $\mathrm{R} \,|\, r_{ij} \,|\, \sum w_j C_j$ and $\mathrm{R} \,|\, \,|\, \sum w_j C_j$ was subsequently proposed in [13]. In a generalization of standard scheduling models, Engels et al. introduce the possibility to outsource a job j, at a cost e_j. Since there is an approximation-preserving reduction from an instance of the resulting scheduling with rejection problem $\mathrm{R} \,|\, (r_j) \,|\, \sum_S w_j C_j + \sum_{\overline{S}} e_j$ to an instance of $\mathrm{R} \,|\, (r_{ij}) \,|\, \sum w_j C_j$, the approximation results are inherited.

Finally, in a computational study of $\mathrm{R} \,|\, \,|\, \sum w_j C_j$, Vredeveld and Hurkens [48] compared the algorithms based on time-indexed and interval-indexed formulations proposed herein to the algorithm based on a convex quadratic programming relaxation in [41], and to a variant based on a time-indexed LP that uses variables x_{ijt}. Here, $x_{ijt} = 1$ if and only if job j is started at time t on machine i. The latter relaxation is tighter than (LP$_\mathrm{R}$) and the convex program in [41] (see, e.g., [48]), and this is confirmed by their studies. In addition, the algorithm based on the relaxation in x–variables also leads to the best upper bounds on their test instances. It follows from Theorem 2.3 that randomized rounding based on this relaxation has performance guarantee 3/2 as well.

Acknowledgements. The authors are grateful to Chandra S. Chekuri, Michel X. Goemans, and David B. Shmoys for helpful comments on an earlier version of this paper [36]. They would also like to thank two anonymous referees for their careful reading of the manuscript and numerous helpful comments that helped to improve the presentation of this paper.

REFERENCES

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

[2] B. Awerbuch, S. Kutten, and D. Peleg, *Competitive distributed job scheduling*, in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, Victoria, BC, 1992, pp. 571–581.

[3] J. L. Bruno, E. G. Coffman Jr., and R. Sethi, *Scheduling independent tasks to reduce mean finishing time*, Communications of the Association for Computing Machinery, 17 (1974), pp. 382–387.

[4] S. Chakrabarti and S. Muthukrishnan, *Resource scheduling for parallel database and scientific applications*, in Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures, Padua, Italy, 1996, pp. 329–335.

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

[6] C. S. Chekuri, R. Johnson, R. Motwani, B. Natarajan, B. R. Rau, and M. Schlansker, *Profile–driven instruction level parallel scheduling with applications to super blocks*, in Proceedings of the 29th Annual International Symposium on Microarchitecture, Paris, France, 1996, pp. 58–67.

[7] C. S. Chekuri, R. Motwani, B. Natarajan, and C. Stein, *Approximation techniques for average completion time scheduling*, SIAM Journal on Computing, 31 (2001), pp. 146–166.

[8] F. A. Chudak, *A min-sum 3/2-approximation algorithm for scheduling unrelated parallel machines*, Journal of Scheduling, 2 (1999), pp. 73–77.

[9] F. A. Chudak and D. B. Shmoys, *Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds*, Journal of Algorithms, 30 (1999), pp. 323–343.

[10] X. Deng, H. Liu, J. Long, and B. Xiao, *Deterministic load balancing in computer networks*, in Proceedings of the 2nd Annual IEEE Symposium on Parallel and Distributed Processing, Dallas, TX, 1990, pp. 50–57.

[11] 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 (1990), pp. 255–270.

[12] W. L. Eastman, S. Even, and I. M. Isaacs, *Bounds for the optimal scheduling of n jobs on m processors*, Management Science, 11 (1964), pp. 268–279.

[13] D. W. Engels, D. R. Karger, S. G. Kolliopoulos, S. Sengupta, R. N. Uma, and J. Wein, *Techniques for scheduling with rejection*, in Algorithms — ESA '98, G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci, eds., vol. 1461 of Lecture Notes in Computer Science, Springer, Berlin, 1998, pp. 490–501.

[14] P. Erdös and J. L. Selfridge, *On a combinatorial game*, Journal of Combinatorial Theory A, 14 (1973), pp. 298–301.

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

[16] ———, *Improved approximation algorithms for scheduling with release dates*, in Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms, New Orleans, LA, 1997, pp. 591–598.

[17] M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang, *Single machine scheduling with release dates*, SIAM Journal on Discrete Mathematics. To appear.

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

[19] 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 (1979), pp. 287–326.

[20] 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 (1997), pp. 513–544.

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

[22] H. Hoogeveen, P. Schuurman, and G. J. Woeginger, *Non-approximability results for scheduling problems with minsum criteria*, INFORMS Journal on Computing, 13 (2001), pp. 157–168.

[23] T. Kawaguchi and S. Kyan, *Worst case bound of an LRF schedule for the mean weighted flow–time problem*, SIAM Journal on Computing, 15 (1986), pp. 1119–1129.

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

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

[26] R. H. Möhring, M. W. Schäffter, and A. S. Schulz, *Scheduling jobs with communication delays: Using infeasible solutions for approximation*, in Algorithms — ESA '96, J. Diaz and M. Serna, eds., vol. 1136 of Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 76–90.

[27] R. H. Möhring, A. S. Schulz, and M. Uetz, *Approximation in stochastic scheduling: The power of LP-based priority policies*, Journal of the ACM, 46 (1999), pp. 924–942.

[28] R. Motwani, J. Naor, and P. Raghavan, *Randomized approximation algorithms in combinatorial optimization*, in Approximation Algorithms for NP-Hard Problems, D. S. Hochbaum, ed., Thomson, 1996, ch. 11, pp. 447–481.

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

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

[31] C. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein, *Improved bounds on relaxations of a parallel machine scheduling problem*, Journal of Combinatorial Optimization, 1 (1998), pp. 413–426.

[32] C. Phillips, C. Stein, and J. Wein, *Task scheduling in networks*, SIAM Journal on Discrete Mathematics, 10 (1997), pp. 573–598.

[33] ———, *Minimizing average completion time in the presence of release dates*, Mathematical Programming, 82 (1998), pp. 199–223.

[34] P. Raghavan and C. D. Thompson, *Randomized rounding: A technique for provably good algorithms and algorithmic proofs*, Combinatorica, 7 (1987), pp. 365–374.

[35] A. S. Schulz and M. Skutella, *Random-based scheduling: New approximations and LP lower bounds*, in Randomization and Approximation Techniques in Computer Science, J. Rolim, ed., vol. 1269 of Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 119–133.

[36] ———, *Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria*, in Algorithms — ESA '97, R. Burkard and G. J. Woeginger, eds., vol. 1284 of Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 416–429.

[37] ———, *The power of α-points in preemptive single machine scheduling*, Journal of Scheduling, 5 (2002). To appear.

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

[39] D. B. Shmoys and E. Tardos, *An approximation algorithm for the generalized assignment problem*, Mathematical Programming, 62 (1993), pp. 461–474.

[40] M. Skutella, *Approximation and randomization in scheduling*, PhD thesis, Technische Universität Berlin, Germany, 1998.

[41] ———, *Convex quadratic and semidefinite programming relaxations in scheduling*, Journal of

the ACM, 48 (2001), pp. 206–242.

[42] M. Skutella and M. Uetz, *Scheduling precedence-constrained jobs with stochastic processing times on parallel machines*, in Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms, Washington, DC, 2001, pp. 589–590.

[43] M. Skutella and G. J. Woeginger, *A PTAS for minimizing the total weighted completion time on identical parallel machines*, Mathematics of Operations Research, 25 (2000), pp. 63–75.

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

[45] J. Spencer, *Ten Lectures on the Probabilistic Method*, no. 52 in CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, 1987.

[46] R. N. Uma and J. M. Wein, *On the relationship between combinatorial and LP-based approaches to NP-hard scheduling problems*, in Integer Programming and Combinatorial Optimization, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., vol. 1412 of Lecture Notes in Computer Science, Springer, Berlin, 1998, pp. 394–408.

[47] A. P. A. Vestjens, *On-line machine scheduling*, PhD thesis, Eindhoven University of Technology, The Netherlands, 1997.

[48] T. Vredeveld and C. Hurkens, *Experimental comparison of approximation algorithms for scheduling unrelated parallel machines*, SPOR report, Eindhoven University of Technology, The Netherlands, 2001.

[49] D. P. Williamson, 1997. Cited as personal communication in [46].

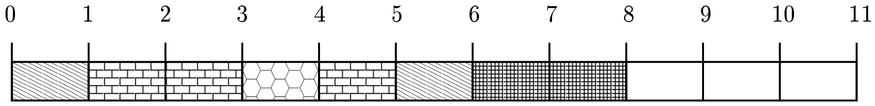
Appendix A. An Illustrating Example.

Consider the following instance of $\mathrm{P}\,2\,|\,r_j\,|\sum w_j C_j$, consisting of the job set $\{1, 2, 3, 4\}$ together with fixed values α_j:

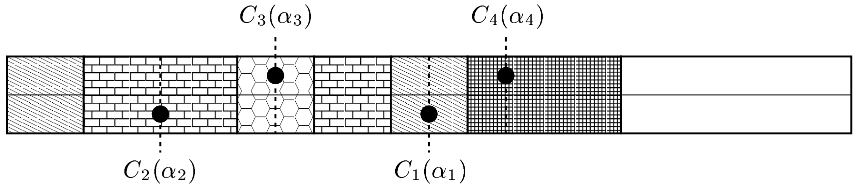
job j		r_j	p_j	w_j/p_j	α_j
1		0	4	1	3/4
2		1	6	2	1/3
3		3	2	3	1/2
4		6	4	2	1/4

The following figure illustrates the action of Algorithm Random Assignment for the given instance. In the first step, it computes the preemptive LP schedule S on a virtual single machine, which is twice as fast as each of the original two machines. Then, each job is randomly assigned to one of the two machines and the α–points are chosen. Finally, the jobs are scheduled on their machines non-preemptively in non-decreasing order of $C_j^S(\alpha_j)$.

(1) preemptive schedule:



(2) & (3) random choices:



(4) feasible schedule: