SIAM J. COMPUT.
Vol. 35, No. 5, pp. 1241–1253

APPROXIMATION BOUNDS FOR A GENERAL CLASS OF PRECEDENCE CONSTRAINED PARALLEL MACHINE SCHEDULING PROBLEMS*

MAURICE QUEYRANNE† AND ANDREAS S. SCHULZ‡

Abstract. An important class of scheduling problems concerns parallel machines and precedence constraints. We consider precedence delays, which associate with each precedence constraint a certain amount of time that must elapse between the completion and start times of the corresponding jobs. Together with ordinary precedence constraints, release dates and delivery times can be modeled in this manner. We present a 4-approximation algorithm for the total weighted completion time objective for this general class of problems. The algorithm is a rather simple form of list scheduling. The list is in order of job midpoints derived from a linear programming relaxation. Our analysis unifies and simplifies that of a number of special cases heretofore separately studied, while actually improving many of the former approximation results.

Key words. approximation algorithm, linear programming relaxation, performance guarantee, precedence constraints, scheduling

AMS subject classifications. 68W25, 90B35, 90C59, 68W40, 68Q25, 68M20, 06A99, 90C27

DOI. 10.1137/S0097539799358094

1. Introduction.

Scheduling problems with precedence constraints are among the most difficult problems in the area of machine and processor scheduling, in particular for the design of good approximation algorithms. Our understanding of the structure of these problems and our ability to generate near-optimal solutions remain limited. The following examples illustrate this point:

(i) The first approximation algorithm for $\mathrm{P}|\mathrm{prec}|C_{\max}$ by Graham (1969) with performance ratio $2 - 1/m$ is still the algorithm of choice for this problem. On the other hand, it is known only that no polynomial-time algorithm can have a better approximation ratio than 4/3, unless P = NP (Lenstra and Rinnooy Kan (1978)).

(ii) The computational complexity of the problem $\mathrm{P}m|\mathrm{prec}, p_j = 1|C_{\max}$, open problem "OPEN8" from the original list of Garey and Johnson (1979), is still open.

(iii) No constant-factor approximation algorithms are known for machines running at different speeds. For the makespan and total weighted completion time objectives, Chudak and Shmoys (1999) only recently improved to $\mathrm{O}(\log m)$ performance ratios of $\mathrm{O}(\sqrt{m})$ due to Jaffe (1980) and Schulz (1996a), respectively.

(iv) Progress is also quite recent for the latter objective on a single machine or identical parallel machines. Until recently, no constant-factor approximation algorithms were known. Lately, the use of linear programming (LP) relaxations has led to 2- and 2.7183-approximation algorithms for $1|\mathrm{prec}|\sum w_j C_j$ and $1|r_j, \mathrm{prec}|\sum w_j C_j$, respectively (Schulz (1996b), Schulz and Skutella (1997)), and to a 5.3281-approximation algorithm for $\mathrm{P}|r_j, \mathrm{prec}|\sum w_j C_j$ (Chakrabarti et al. (1996)). (Since then, Chudak

*Received by the editors June 28, 1999; accepted for publication (in revised form) June 3, 2005; published electronically March 17, 2006. An extended abstract of a preliminary version of this paper appeared in *Proceedings of the Conference on Integer Programming and Combinatorial Optimization (IPCO* VI*)*, Houston, TX, 1998, pp. 367–382.

http://www.siam.org/journals/sicomp/35-5/35809.html

†Sauder School of Business, University of British Columbia, 2053 Main Mall, Henry Angus 457, Vancouver, BC V6T 1Z2, Canada (Maurice.Queyranne@sauder.ubc.ca).

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

and Hochbaum (1999), Chekuri and Motwani (1999), and Margot, Queyranne, and Wang (2003) have proposed various combinatorial 2-approximation algorithms for $1|\text{prec}|\sum w_jC_j$.) Few deep negative results are known for these problems (Hoogeveen, Schuurman, and Woeginger (2001)).

We consider (a generalization of) the scheduling problem $\text{P}|r_j, \text{prec}|\sum w_jC_j$ and answer a question of Hall et al. (1997, page 530), which they raised in the context of a 7-approximation algorithm for this problem:

> *Unfortunately, we do not know how to prove a good performance guarantee for this model by using a simple list-scheduling variant.*

We show that using a similar LP relaxation as Hall et al. (1997) and Chakrabarti et al. (1996) in a different way (reading the list order from the LP midpoints instead of LP completion times) yields a simple 4-approximation algorithm for $\text{P}|r_j, \text{prec}|\sum w_jC_j$. We actually obtain this result in the more general framework of precedence delays.

Let us describe the model in detail. We are given a set N of n jobs, and m identical parallel machines. Each job j has a nonnegative processing time p_j; it must be processed uninterruptedly for that amount of time on any one of the machines. Each machine can process only one job at a time. An acyclic, directed graph $D = (N, A)$ specifies precedence constraints between jobs. A nonnegative precedence delay d_{ij} is associated with each precedence-constrained job pair $(i, j) \in A$, with the following meaning: in every feasible schedule, job j cannot start until d_{ij} time units after job i is completed. Precedence delays can be used to model ordinary precedence constraints ($d_{ij} = 0$), release dates $r_j \geq 0$ (by adding a dummy job 0 with zero processing time and precedence delays $d_{0j} = r_j$ for all other jobs), or delivery times $q_j \geq 0$, which must elapse between the end of a job's processing and its actual completion time.

Precedence delays were considered for project scheduling under the name of "finish-to-start lags," e.g., by Bartusch, Möhring, and Radermacher (1988) and Herroelen and Demeulemeester (1995), for one-machine scheduling by Wikum, Llewellyn, and Nemhauser (1994) under the name of "generalized precedence constraints," and by Balas, Lenstra, and Vazacopoulos (1995) under that of "delayed precedence constraints"; the latter authors used the $L_{\max}$ minimization problem as a key relaxation in a modified version of the shifting bottleneck procedure for the classic job-shop scheduling problem. The one-machine problem $1|\text{prec. delays } d_{ij} = k,\ p_j = 1|C_{\max}$ corresponds to a basic pipeline scheduling problem; see Lawler et al. (1987) for a survey. Leung, Vornberger, and Witthoff (1984) showed that this problem is strongly NP-complete. Several other authors, including Bruno, Jones III, and So (1980), Bernstein and Gertner (1989), Palem and Simons (1993), Finta and Liu (1996), and Brucker and Knust (1999), derived polynomial-time algorithms for particular instances by utilizing well-known algorithms for special cases of the classical m-machine problem. In the context of approximation algorithms, Hall and Shmoys (1989, 1990, 1992) presented polynomial-time approximation schemes for the problems $1|r_j, q_j|C_{\max}$, $\text{P}|r_j, q_j|C_{\max}$, and $1|r_j, \text{prec}, q_j|C_{\max}$, respectively. Schuurman (1998) gave a fully polynomial-time approximation scheme for $1|\text{prec. delays } d_{ij}|C_{\max}$ when the partial order A has a special structure introduced by Wikum, Llewellyn, and Nemhauser (1994). Graham's list-scheduling algorithm (Graham (1969)) was extended to $\text{P}|\text{prec. delays } d_{ij} = k,\ p_j = 1|C_{\max}$ to yield a worst-case performance ratio of $2 - 1/(m(k + 1))$ (Lawler et al. (1987), Palem and Simons (1993)). This result was in turn extended by Munier to nonidentical precedence delays and processing times; see Munier, Queyranne, and Schulz (1998) for details. We refer to Brucker and Knust (1999) for an overview of complexity results for single-machine problems with precedence delays, including polynomially solvable cases with total completion time objective.

For given nonnegative job weights $w_j \geq 0$, we consider the objective of minimizing the weighted sum $\sum_{j \in N} w_j C_j$ of completion times. Here, C_j denotes the completion time of job j in the corresponding schedule. Even special cases of this problem P|prec. delays $d_{ij}| \sum w_j C_j$ are NP-hard;[1] we therefore discuss the quality of relaxations and approximation algorithms. An α-approximation algorithm is a polynomial-time algorithm that produces a solution with objective value at most α times the optimal value. Sometimes α is called the (worst-case) performance guarantee of the algorithm. Similarly, an α-relaxation is a relaxation with objective value at least $1/\alpha$ times the optimal value.

For P|prec. delays $d_{ij}| \sum w_j C_j$, we present in section 3 a new algorithm with a performance guarantee of 4. This algorithm is based on an LP relaxation of this problem, which is a direct extension of earlier LP relaxations proposed by Schulz (1996b) and Hall et al. (1997). The decision variables are the job completion times C_j; in particular, this relaxation ignores the machine assignments. There are two sets of linear constraints: one represents the precedence delays in a straightforward fashion; the other set of constraints is a relatively simple way of enforcing the total capacity of the m machines. Although the machine assignments are ignored and the machine capacities are modeled in a simplistic way, this is sufficient to obtain the best relaxation and approximation bounds known so far for these problems and several special cases thereof. We show that using job midpoints (instead of completion times) derived from the LP relaxation leads to a performance ratio of 4 for the general problem described above. In a given schedule, the midpoint of a job is the earliest point in time at which half of its processing has been performed; if the schedule is nonpreemptive, then the midpoint of job j is simply $C_j - p_j/2$. Midpoints and more general notions of α-points have previously been used in the design and analysis of approximation algorithms for a variety of scheduling problems with the weighted sum of completion times objective; see, e.g., Phillips, Stein, and Wein (1998), Hall, Shmoys, and Wein (1996), Goemans (1997), Chekuri et al. (2001), Goemans et al. (2002), and Schulz and Skutella (2002a, 2002b).

In summary, the contributions of this paper are as follows:

(i) We shed further light on the relationship between two forms of list-scheduling algorithms: Graham's nonidling, machine-based list scheduling and job-driven list scheduling.[2] It is well known that the former is appropriate for optimizing objectives, such as the makespan $C_{\max}$, that are related to maximizing machine utilization, whereas they are inappropriate (leading to unbounded performance ratio) for job-oriented objectives, such as the weighted sum of completion times $\sum_j w_j C_j$. This difficulty was recognized by, among others, Chekuri et al. (2001), who proposed a variant of machine-based list scheduling that allows for insertion of idle time, using a mechanism for "charging" such idle time to jobs. This technique leads to a 5.8285-approximation algorithm for $\mathrm{P}|r_j, \mathrm{prec}| \sum w_j C_j$. We show that job-driven list-scheduling algorithms have bounded performance ratio for the $\sum_j w_j C_j$ objective if the priority list is sensibly chosen.

(ii) Using job completion times as a basis for job-driven list scheduling may yield very poor schedules for problems with parallel machines, precedence constraints, and a weighted sum of completion times objective. This may happen even if the completion

[1] For example, $\mathrm{P2}|\,|\sum w_j C_j$, $1|r_j|\sum w_j C_j$, and $1|\mathrm{prec}|\sum w_j C_j$ are NP-hard (Bruno, Coffman, Jr., and Sethi (1974), Lenstra, Rinnooy Kan, and Brucker (1977), Lawler (1978), Lenstra and Rinnooy Kan (1978)).

[2] Sometimes also called the parallel and serial methods; see Kolisch (1996) for a recent review in the context of resource-constrained project scheduling.

times are those of an optimal schedule. In contrast, we show that job-driven list scheduling according to job midpoints from an appropriate LP relaxation leads to job-by-job error bounds of at most 4 for a broad class of problems.

(iii) The general model of scheduling with precedence delays allows us to treat in a unified framework ordinary precedence constraints and release dates. In particular, this simplifies and unifies the analysis and proof techniques.

(iv) We present the best polynomial-time approximation bounds known so far for a broad class of parallel machine scheduling problems with precedence constraints or delays and a total weighted completion time objective. These approximation results are summarized in Table 1.1.

TABLE 1.1
Summary of results.

Problem	Performance Guarantee	Reference
P|prec. delays d_{ij}|$\sum w_jC_j$	4	Corollary 3.3
P|prec. delays d_{ij}, $p_j = p$|$\sum w_jC_j$	3	Corollary 3.7
P|r_j, prec|$\sum w_jC_j$	4	Corollary 3.3
P|r_j, prec, $p_j = p$|$\sum w_jC_j$	3	Corollary 3.7
P|prec|$\sum w_jC_j$	$4 - 2/m$	Corollary 3.5
P|prec, $p_j = p$|$\sum w_jC_j$	$3 - 1/m$	Corollary 3.7
P|prec = stiff|$\sum w_jC_j$	$3 - 1/m$	Corollary 3.8
1|prec. delays d_{ij}|$\sum w_jC_j$	3	Corollary 3.6

2. List-scheduling algorithms. List-scheduling algorithms, first analyzed by Graham (1966, 1969), are among the simplest and most commonly used approximate solution methods for parallel machine scheduling problems. These algorithms use priority rules, or job rankings. Whenever one of the m machines becomes idle, the next available job in the list is started on that machine. In the presence of precedence constraints, a job is available if all of its predecessors have completed processing. By their nonidling property, Graham's list-scheduling algorithms are appropriate when machine utilization is an important consideration. Indeed, Graham (1969) showed that list scheduling is a $(2 - 1/m)$-approximation algorithm for P|prec|$C_{\max}$, no matter which priority order is used. In contrast, the two examples below show that the nonidling property may lead to poor performance ratios for a weighted sum of completion times objective $\sum_j w_jC_j$.

Example 2.1. Consider the following two-job instance of the single-machine nonpreemptive scheduling problem $1|r_j|\sum w_jC_j$ (a special case of a precedence delay problem, as discussed in the introduction). For a parameter $q \geq 2$, job 1 has $p_1 = q$, $r_1 = 0$, and $w_1 = 1$, whereas job 2 has $p_2 = 1$, $r_2 = 1$, and $w_2 = q^2$. The optimal strategy is to leave the machine idle during the time interval $[0, 1)$ so as to process job 2 first. The optimum objective value is $2q^2 + q + 2$. Any nonidling heuristic starts processing job 1 at time 0, leading to an objective value of $q^3 + q^2 + q$. The performance ratio is unbounded as q may be arbitrarily large. □

The following example is of the same type but uses ordinary precedence constraints instead of release dates.

Example 2.2. Consider an instance with $m \geq 3$ machines and three types of jobs. Unit-time job 1 precedes jobs $a(1), a(2), \ldots, a(m)$, each of which has processing time 1 as well. Jobs $b(1), b(2), \ldots, b(m-1)$ are independent and have processing time m each. Job 1 and jobs $b(1), b(2), \ldots, b(m-1)$ have zero weight, whereas $w_{a(h)} = 1$ for

$h = 1, 2, \ldots, m$. The optimal schedule starts job 1 at time 0 on some machine and leaves $m - 1$ machines idle during the time interval $[0, 1)$ so as to complete all jobs $a(1), a(2), \ldots, a(m)$ by time 2. Hence, its objective value is $2m$. Any form of Graham's machine-based list scheduling starts processing jobs $b(1), b(2), \ldots, b(m-1)$ at time 0. These jobs occupy their machines until time m, forcing jobs $a(1), a(2), \ldots, a(m)$ onto the same machine as job 1, which results in an objective value of $(m+1)(m+2)/2-1$. Therefore, the performance ratio increases with m. □

Evidently, the appropriate introduction of idle time is an important element in the design of approximation algorithms to minimize a weighted sum of completion times subject to precedence delays. As Examples 2.1 and 2.2 illustrate, idle time is needed to prevent large-weight jobs, which may soon become available, from being delayed by other, less important jobs.[3] On the other hand, too much idle time is undesirable as well. The necessity to balance these two effects contributes to the difficulty of this problem. All former approximation algorithms for $P|r_j, \text{prec}|\sum w_j C_j$ with constant-factor performance ratios are based on variants of Graham's original list scheduling, which actually tries to avoid machine idle time. In fact, Hall et al. (1997) partitioned jobs into groups that are individually scheduled according to Graham's list-scheduling rule, and then these schedules are concatenated to obtain a solution for the original problem. To find a good partition, this scheme was enriched with randomness by Chakrabarti et al. (1996). Chekuri et al. (2001) presented a different variant of Graham's list scheduling by artificially introducing idle time whenever it seems that a further delay of the next available job in the list (if it is not the first) can be afforded. It is worth mentioning that these techniques, analyses, and approximation results also generalize to precedence delays.

Another, arguably simpler, strategy is to consider the jobs one by one, in the given list order, starting from an empty schedule. Each job is nonpreemptively inserted into the current schedule without altering the jobs already scheduled. Specific job-driven list-scheduling algorithms differ in how this principle is implemented. For definiteness, consider the version in Figure 2.1, whereby every job is considered in the list order and is scheduled at the earliest feasible time at the end of the current schedule on a machine. We assume that the given list is a linear extension of the partial order defined by the precedence constraints.

1. The list $L = (\ell(1), \ell(2), \ldots, \ell(n))$ is given.
2. Initially, all machines are empty, with machine completion times $\Gamma_h := 0$ for $h = 1, 2, \ldots, m$.
3. For $k = 1$ to n do:
   3.1 Let job $j = \ell(k)$; set its start time $S_j := \max\big(\max\{C_i + d_{ij} : (i, j) \in A\}, \min\{\Gamma_h : h = 1, 2, \ldots, m\}\big)$ and its completion time $C_j := S_j + p_j$.
   3.2 Assign job j to a machine h such that $\Gamma_h \le S_j$. Update $\Gamma_h := C_j$.

FIG. 2.1. *Job-driven list-scheduling algorithm for* P|prec. delays $d_{ij}|\sum w_j C_j$.

Various rules may be used in step 3.2 for the choice of the assigned machine h, for example, one with largest completion time Γ_h (so as to reduce the idle time between Γ_h and S_j). Moreover, the above algorithm can be modified to allow the insertion of

[3]It is important to point out that this difficulty results from the nonpreemptive mode; Graham's list scheduling with jobs ordered according to LP completion times gives a 3-approximation for $P|r_j, \text{prec}, \text{pmtn}|\sum w_j C_j$ (Hall et al. (1997)).

a job in an idle period before time Γ_h. In effect, the observations below also apply to all these variants.

One method (e.g., Phillips, Stein, and Wein (1998) and Hall et al. (1997)) for defining the list L consists of sorting the jobs in nondecreasing order of their completion times in a relaxation of the scheduling problem under consideration. In the presence of ordinary precedence constraints, this works well for the case of a single machine (Schulz (1996b), Hall et al. (1997)), but Example 2.3 shows that this may produce very poor schedules for the case of identical parallel machines. This example uses the list which is produced by an optimal schedule, the tightest kind of relaxation that can be defined; moreover, this optimal schedule defines the same completion time order as the LP relaxation in section 3.1.

Example 2.3. For a fixed number $m \geq 2$ of identical parallel machines and a positive number ϵ, let the job set N consist of m sets J_h $(h = 1, 2, \ldots, m)$ of $m + 1$ jobs each, plus a "last job" n. (Thus $n = m(m + 1) + 1$.) Each set J_h consists of a "long job" $a(h)$, with processing time $p_{a(h)} = 1 + (h - 1)(m + 1)\epsilon$, and m "small jobs" $b(h, g)$ (for $g = 1, 2, \ldots, m$), each with processing time $p_{b(h,g)} = \epsilon$ and subject to the precedence constraint $(a(h), b(h, g))$. In addition, there is a precedence constraint $(b(h, g), n)$ from each small job $b(h, g)$ (for $h = 1, 2, \ldots, m$ and $g = 1, 2, \ldots, m$) to the last job n, which has processing time ϵ. The objective is to minimize either the makespan or a weighted sum $\sum_j w_j C_j$ of job completion times with weights $w_j = 0$ for all $j \neq n$ and $w_n = 1$; due to the precedence constraints, these two objectives coincide for any feasible schedule. An optimal schedule has, for $h = 1, 2, \ldots, m$, long job $a(h)$ starting at time $S^*_{a(h)} = 0$ on machine h, immediately followed by all small jobs $b(h, g)$ (for $g = 1, 2, \ldots, m$) in the same set J_h, assigned as uniformly as possible to machines $1, 2, \ldots, h$. Note that all the jobs in J_h are completed before the next long job $a(h + 1)$ completes. Job n is then processed last on any machine, so that the optimal objective value is $C^*_{\max} = C^*_n = 1 + (m^2 + 1)\epsilon$. On the other hand, any version of the job-driven list-scheduling algorithm with all jobs listed in order of their optimal completion times produces the following schedule: long job $a(1)$ is scheduled first, with start time $S^L_{a(1)} = 0$ and completion time $C^L_{a(1)} = 1$; the m small jobs $b(1, g)$ (for $g = 1, 2, \ldots, m$) in J_1 are then scheduled, each with start time $S^L_{b(1,g)} = 1$ and completion time $C^L_{b(1,g)} = 1 + \epsilon$ on a different machine; this forces all subsequent jobs to be scheduled no earlier than date $1 + \epsilon$. Consequently, for $h = 1, 2, \ldots, m$, long job $a(h)$ is scheduled with start time $S^L_{a(h)} = (h - 1) + (h - 1 + (h - 1)(h - 2)(m + 1)/2)\epsilon$, followed by all small jobs $b(h, g)$ (for $g = 1, 2, \ldots, m$) in the same set J_h, each with start time $S^L_{b(h,g)} = S^L_{a(h)} + p_{a(h)}$ on a different machine. Finally, job n is scheduled last with $S^L_n = m + (m + m(m - 1)(m + 1)/2)\epsilon$, and thus the objective value is $C^L_{\max} = C^L_n = m + o(\epsilon)$, arbitrarily close to m times the optimal value $C^*_{\max}$ when $\epsilon > 0$ is small enough. □

3. LP-based approximation algorithms. In this section we present an LP relaxation for the problem of minimizing a weighted sum $\sum_j w_j C_j$ of job completion times subject to precedence delays, and use it to develop a 4-approximation algorithm for this problem. Thereafter, we refine the analysis to give improved bounds for various relevant special cases.

3.1. The LP relaxation. The decision variables are the job completion times C_j. The set of constraints is

$$C_j \geq p_j \quad \text{for all } j \in N, \tag{3.1}$$

$$C_j \geq C_i + d_{ij} + p_j \quad \text{for all } (i,j) \in A, \tag{3.2}$$

$$\sum_{j \in F} p_j C_j \geq \frac{1}{2m}\left(\sum_{j \in F} p_j\right)^2 + \frac{1}{2}\sum_{j \in F} p_j^2 \quad \text{for all } F \subseteq N. \tag{3.3}$$

This relaxation is an extension of a relaxation introduced by Hall et al. (1997) for $P|r_j, \text{prec}|\sum w_j C_j$: on the one hand, ordinary precedence constraints are replaced by inequalities (3.2), which model the precedence delays; on the other hand, inequalities (3.3) are stronger than the analogous class of inequalities used by Hall et al. (1997). Our analysis requires this strengthened class of inequalities, which was proposed by Schulz (1996b). For $m = 1$ (the single-machine case), they are identical to inequalities introduced by Wolsey (1985) to model the constraint that the machine can process at most one job at a time; see Queyranne (1993) for further details. Constraints (3.1) impose the trivial lower bounds on job completion times.

For a weighted sum of completion times objective, the LP formulation is simply

$$\text{minimize } \sum_{j \in N} w_j C_j \quad \text{subject to (3.1)–(3.3).} \tag{3.4}$$

Although there is an exponential number of constraints in (3.3), the separation problem for these inequalities can be solved in polynomial time (Schulz 1996a). It follows that this LP relaxation can be solved in polynomial time, using the ellipsoid method.

3.2. The approximation algorithm. Let C^{LP} denote any feasible solution to the constraint set (3.1)–(3.3) of this LP. We use this LP solution to define a feasible schedule with completion time vector C^{H} and analyze the job-by-job relationship between C_j^{H} and C_j^{LP} for every job $j \in N$. We define the LP midpoint as $M_j^{\text{LP}} := C_j^{\text{LP}} - p_j/2$. We now use the list-scheduling algorithm of Figure 2.1 with the LP midpoint list L defined by sorting the jobs in nondecreasing order of their midpoints M_j^{LP}. The next theorem contains our main result.

THEOREM 3.1. *Let C^{LP} denote any feasible solution to the constraint set* (3.1)–(3.3)*, and let M^{LP} denote the corresponding vector of LP midpoints. Let S^{H} be the vector of start times of the feasible schedule constructed by the job-driven list-scheduling algorithm using the LP midpoint list. Then*

$$S_j^{\text{H}} \leq 4\, M_j^{\text{LP}} \quad \textit{for all jobs } j \in N. \tag{3.5}$$

Proof. Assume for simplicity that the jobs are indexed in the order of their LP midpoints; that is, $M_1^{\text{LP}} \leq M_2^{\text{LP}} \leq \cdots \leq M_n^{\text{LP}}$. We fix job $j \in N$ and consider the schedule constructed by the list-scheduling heuristic using the LP midpoint list $L = (1, 2, \ldots, n)$ up to and including the scheduling of job j, that is, up to the completion of step 3 with $k = j$. Let $[\,j\,] := \{1, 2, \ldots, j\}$.

Let μ denote the total time between 0 and the start time S_j^{H} of job j when all m machines are busy at this stage of the algorithm. Since only jobs in $[\,j-1\,]$ have been scheduled so far, $\mu \leq \sum_{i=1}^{j-1} p_i/m$. Let $\lambda := S_j^{\text{H}} - \mu$. To prove (3.5), we need only show that

$$\text{(i)} \quad \frac{1}{m}\sum_{i=1}^{j-1} p_i \leq 2M_j^{\text{LP}} \quad \text{and} \quad \text{(ii)} \quad \lambda \leq 2M_j^{\text{LP}}.$$

Inequality (i) follows from a straightforward variant of Lemma 1 in Schulz (1996b) or Lemma 3.2 in Hall et al. (1997). For this, first observe that the inequalities (3.3) are

equivalent to

$$\sum_{i\in F} p_i M_i \geq \frac{1}{2m}\left(\sum_{i\in F} p_i\right)^2 \qquad \text{for all } F \subseteq N.$$

Since M^{LP} satisfies all these inequalities, letting $F = [\,j-1]$ and using $M_1^{\mathrm{LP}} \leq M_2^{\mathrm{LP}} \leq \cdots \leq M_j^{\mathrm{LP}}$, we obtain

$$\left(\sum_{i=1}^{j-1} p_i\right) M_j^{\mathrm{LP}} \geq \sum_{i=1}^{j-1} p_i M_i^{\mathrm{LP}} \geq \frac{1}{2m}\left(\sum_{i=1}^{j-1} p_i\right)^2,$$

implying (i).

To show (ii), let q denote the number of maximal intervals between dates 0 and S_j^{H} when at least one machine is idle (i.e., not processing a job in $[\,j-1]$) in the schedule C^{H}. Denote these idle intervals as $[a_h, b_h)$ for $h = 1, 2, \ldots, q$, so that $0 \leq a_1$, $b_{h-1} < a_h < b_h$ for all $h = 2, \ldots, q$, and $b_q \leq S_j^{\mathrm{H}}$. Hence, $\lambda = \sum_{h=1}^{q}(b_h - a_h)$ and all machines are busy during the complementary intervals $[b_h, a_{h+1})$, including intervals $[0, a_1)$ and $[b_q, S_j^{\mathrm{H}})$, if nonempty.

Consider the digraph $G^{[\,j]} = ([\,j], A^{[\,j]})$, where

$$A^{[\,j]} := \{(k,\ell) \in A : k, \ell \in [\,j] \text{ and } C_\ell^{\mathrm{H}} = C_k^{\mathrm{H}} + d_{k\ell} + p_\ell\}.$$

That is, $A^{[\,j]}$ is the set of precedence pairs in $[\,j]$ for which the precedence delay constraints (3.2) are tight for C^{H}. If $b_q > 0$, then a machine becomes busy at date b_q (or starts processing job j if $b_q = S_j^{\mathrm{H}}$) and thus there exists a job $x(q) \in [\,j]$ with start time $S_{x(q)}^{\mathrm{H}} = b_q$. Since $x(q) \in [\,j]$ we have $M_{x(q)}^{\mathrm{LP}} \leq M_j^{\mathrm{LP}}$. We repeat the following process for decreasing values of the interval index h, starting with $h = q$, until we reach the date 0 or the busy interval $[0, a_1)$. Let $(v(1), \ldots, v(s))$ denote a maximal path in $G^{[\,j]}$ with last node (job) $v(s) = x(h)$. Note that we must have $b_g < S_{v(1)}^{\mathrm{H}} \leq a_{g+1}$ for some busy interval $[b_g, a_{g+1})$ with $a_{g+1} < b_h$, for otherwise some machine is idle immediately before the start time $S_{v(1)}^{\mathrm{H}}$ of job $v(1)$ and this job, not being constrained by any tight precedence delay constraint, should have started earlier than that date. (Unless $S_{v(1)}^{\mathrm{H}} = 0$, of course. In this case, a_{g+1} is the first point in time at which some machine falls idle.) We have

$$b_h - a_{g+1} \leq S_{v(s)}^{\mathrm{H}} - S_{v(1)}^{\mathrm{H}} = \sum_{i=1}^{s-1}\left(S_{v(i+1)}^{\mathrm{H}} - S_{v(i)}^{\mathrm{H}}\right) = \sum_{i=1}^{s-1}\left(p_{v(i)} + d_{v(i)v(i+1)}\right). \tag{3.6}$$

On the other hand, the precedence delay constraints (3.2) imply

$$M_{v(i+1)}^{\mathrm{LP}} \geq M_{v(i)}^{\mathrm{LP}} + \frac{1}{2}p_{v(i)} + d_{v(i)v(i+1)} + \frac{1}{2}p_{v(i+1)}$$

for all $i = 1, 2, \ldots, s-1$. Therefore,

$$M_{x(h)}^{\mathrm{LP}} - M_{v(1)}^{\mathrm{LP}} \geq \frac{1}{2}\sum_{i=1}^{s-1}\left(p_{v(i)} + d_{v(i)v(i+1)}\right) \geq \frac{1}{2}(b_h - a_{g+1}).$$

If $b_g > 0$, then let $x(g)$ be a job with $S_{x(g)}^{\mathrm{H}} = b_g$. Because of the order of jobs in the priority list L, $S_{x(g)}^{\mathrm{H}} < S_{v(1)}^{\mathrm{H}}$ implies $M_{x(g)}^{\mathrm{LP}} \leq M_{v(1)}^{\mathrm{LP}}$. Therefore,

$$M_{x(h)}^{\mathrm{LP}} - M_{x(g)}^{\mathrm{LP}} \geq \frac{1}{2}(b_h - a_{g+1}). \tag{3.7}$$

We also have $(k, x(g)) \in A^{[j]}$ for some $k \in [j]$ with $S_k^{\mathrm{H}} < b_g = S_{x(g)}^{\mathrm{H}}$, for otherwise job $x(g)$ should have started processing on some idle machine before date b_g. We may thus repeat the above process with $h = g$ and job $x(h) = x(g)$. Since $g < h$ at each step, this whole process must terminate, generating a decreasing sequence of indices $q = H(1) > \cdots > H(q') = 0$ such that every idle interval is contained in some interval $[a_{H(i+1)+1}, b_{H(i)})$. Adding the corresponding inequalities (3.7), we obtain

$$\lambda \leq \sum_{i=1}^{q'-1} (b_{H(i)} - a_{H(i+1)+1}) \leq 2(M_{x(H(1))}^{\mathrm{LP}} - M_{x(H(q'))}^{\mathrm{LP}}) \leq 2(M_j^{\mathrm{LP}} - 0). \tag{3.8}$$

This establishes (ii). The proof of Theorem 3.1 is complete. □

The following example shows that the factor 4 in inequality (3.5) is (asymptotically) best possible, even for ordinary precedence constraints only.

Example 3.2. For a given number $m \geq 2$ of identical parallel machines, the job set N includes m sets J_h (for $h = 1, 2, \ldots, m$) of $m + 1$ jobs each: a job $a(h)$ with processing time $p_{a(h)} = 2^{h-1-m}$, and m "small jobs" $b(h, g)$ (for $g = 1, 2, \ldots, m$), each with processing time $p_{b(h,g)} = 0$ and subject to the ordinary precedence constraint $(a(h), b(h, g))$ (with zero delay). In addition, there are m "unit jobs" $u(i)$ (for $i = 1, 2, \ldots, m$) with processing time $p_{u(i)} = 1$, and two jobs, $n-1$ and n, with processing times $p_{n-1} = 2/m$ and $p_n = 0$ (thus $n = (m + 1)^2 + 1$). There are two additional (ordinary) precedence constraints: $a(m)$ precedes $n - 1$ and $n - 1$ precedes n.

For sufficiently large m, the following solution C^{LP} is feasible for constraints (3.1)–(3.3): $C_j^{\mathrm{LP}} = p_{a(h)} = 2^{h-1-m}$ for all jobs $j \in J_h$, for $h = 1, 2, \ldots, m$; $C_{u(i)}^{\mathrm{LP}} = 1 + 2/m$ for all unit jobs $u(i)$; and $C_{n-1}^{\mathrm{LP}} = C_n^{\mathrm{LP}} = M_n^{\mathrm{LP}} = p_{a(m)} + p_{n-1} = 1/2 + 2/m$. Therefore, an LP midpoint list starts with sets $J_1, J_2, \ldots, J_m$ (each with its medium job before all its small jobs), followed by job $n - 1$, all unit jobs $u(i)$, and finally job n. This LP midpoint list produces the following schedule. First schedule the sets J_h in sequence $h = 1, 2, \ldots, m$, beginning with the medium job $a(h)$ on one machine (starting just after set $J(h - 1)$ is complete), immediately followed by the m small jobs in J_h, each on a different machine. All jobs j in J_h have completion time $C_j^{\mathrm{H}} = \sum_{i=1}^{h} p_{a(i)} = 2^{h-m} - 2^{-m}$; since all machines are occupied at that date, this forces all subsequent jobs to start no earlier than that date. After the last set J_m is complete, schedule job $n-1$ and $m-1$ unit jobs $u(i)$, each on a different machine. After this, start the remaining unit job on the same machine as job $n - 1$. The first $m - 1$ unit jobs have completion time $C_{u(i)}^{\mathrm{H}} = 2 - 2^{-m}$. Finally, start job n at date $S_n^{\mathrm{H}} = 2 - 2^{-m}$. For m large enough, the latter expression is arbitrarily close to $4M_n^{\mathrm{LP}} = 2 + 8/m$. □

If C^{LP} is an optimal LP solution, Theorem 3.1 implies performance ratios of 1/4 and 4 for the LP relaxation and the heuristic solution, respectively.

COROLLARY 3.3. *Let C^{LP} denote an optimal solution to the LP relaxation defined in* (3.4) *for the problem* P|prec. delays $d_{ij}|\sum w_j C_j$. *Let C^{H} denote the solution constructed from C^{LP} by the job-driven list-scheduling algorithm using the LP midpoint list, and let C^* denote an optimum schedule. Then,*

$$\sum_{j \in N} w_j C_j^{\mathrm{LP}} \geq \frac{1}{4} \sum_{j \in N} w_j C_j^* \quad \text{and} \quad \sum_{j \in N} w_j C_j^{\mathrm{H}} \leq 4 \sum_{j \in N} w_j C_j^*. \tag{3.9}$$

The following example shows that the latter bound is (asymptotically) tight, even for ordinary precedence constraints only, i.e., for problem P|prec|$\sum w_j C_j$.

Example 3.4. In Example 3.2, let the weights be $w_n = 1$ and all other $w_j = 0$, so that the optimum solution has $\sum_{j\in N} w_j C_j^* = w_n(p_{a(m)} + p_{n-1} + p_n) = 1/2 + 2/m$. Since the precedence constraints (3.2) and the lower bounds (3.1) imply $C_n^{\mathrm{LP}} \geq p_{a(m)} + p_{n-1} + p_n = 1/2 + 2/m$, the solution C^{LP} described in Example 3.2 is optimal for the LP relaxation (3.4); its objective value is $\sum_{j\in N} w_j C_j^{\mathrm{LP}} = 1/2 + 2/m = \sum_{j\in N} w_j C_j^*$. Using the LP midpoint list produces the schedule described in the above example, with $\sum_{j\in N} w_j C_j^{\mathrm{H}} = 2 - 2^{-m}$. For m large enough, the latter expression is arbitrarily close to $4\sum_{j\in N} w_j C_j^*$. □

We suspect that the first inequality in (3.9), bounding the performance ratio of the LP relaxation, is not tight. The worst instances we know arise actually for $m = 1$ and lead to a gap of 2; see Hall et al. (1997) for details.

3.3. Special cases. The analysis in Theorem 3.1 can be refined for some special cases, yielding tighter performance ratios. For the problem $\mathrm{P}|\mathrm{prec}|\sum w_j C_j$, observe that the list-scheduling algorithm will not allow all machines to be simultaneously idle at any date before the start time of any job $j \in N$. Therefore, in the proof of Theorem 3.1, all the idle intervals, with total length λ, contain some processing of some job(s) $i < j$; as a result the total work during the busy intervals is at most $\sum_{i=1}^{j-1} p_i - \lambda$. Hence, we obtain the following result.

COROLLARY 3.5. *Job-driven list scheduling by optimal LP midpoints is a* $(4 - 2/m)$*-approximation algorithm for the scheduling problem* $\mathrm{P}|\mathrm{prec}|\sum w_j C_j$.

Note that for $m = 1$ we recover the performance ratio of 2 for $1|\mathrm{prec}|\sum w_j C_j$ in Schulz (1996b), which is known to be tight for that special case (Hall et al. (1997)).

In the case of precedence delays and a single machine, the idle intervals that add up to λ time units cannot contain any processing. Therefore, in the proof of Theorem 3.1 replace inequality (3.6) with $b_h - a_{g+1} \leq S^{\mathrm{H}}_{v(s)} - C^{\mathrm{H}}_{v(1)} = \sum_{i=1}^{s-1} d_{v(i)v(i+1)}$. (Note that, if all processing times are positive, then $s = 2$ and the summation in the right-hand side consists of a single term $d_{v(1)v(2)}$.) Adding up the precedence delay constraints (3.2) for all $i = 1, 2, \ldots, s-1$ and omitting some processing times yields $M^{\mathrm{LP}}_{x(h)} - M^{\mathrm{LP}}_{v(1)} \geq \sum_{i=1}^{s-1} d_{v(i)v(i+1)} \geq b_h - a_{g+1}$. Therefore, we may replace (3.8) with $\lambda \leq \sum_{i=1}^{q'-1}(b_{H(i)} - a_{H(i+1)+1}) \leq M^{\mathrm{LP}}_{x(H(1))} - M^{\mathrm{LP}}_{x(H(q'))} \leq M_j^{\mathrm{LP}}$ and thus inequality (ii) with $\lambda \leq M_j^{\mathrm{LP}}$. This implies $S_j^{\mathrm{H}} \leq 3M_j^{\mathrm{LP}}$.

COROLLARY 3.6. *Job-driven list scheduling by optimal LP midpoints is a* 3*-approximation algorithm for the scheduling problem* $1|\mathrm{prec.\ delays}\ d_{ij}|\sum w_j C_j$.

Note that for the special case $1|r_j, \mathrm{prec}|\sum w_j C_j$ we recover the performance ratio of 3 in Schulz (1996b). The best approximation algorithm known for this problem, however, has a performance guarantee of $e \approx 2.7183$ (Schulz and Skutella (1997)).

In principle, we may use any LP α-point $C_j^{\mathrm{LP}}(\alpha) := C_j^{\mathrm{LP}} - (1-\alpha)p_j$ for some $0 \leq \alpha < 1$ in section 3.2. Indeed, inequality (i) in the proof of Theorem 3.1 can be replaced with $\sum_{i=1}^{j-1} p_i/m \leq 2C_j^{\mathrm{LP}}(\alpha)$, provided that $\alpha \geq 1/2$. On the other hand, inequality (ii) becomes $\lambda \leq (1-\alpha)^{-1} C_j^{\mathrm{LP}}(\alpha)$. While it turns out that using the midpoint $M_j^{\mathrm{LP}} = C_j^{\mathrm{LP}}(1/2)$ leads to the best bound for the general case, we can take advantage of this observation for some special cases.

For example, if the LP midpoint list coincides with the LP start-time list (which is the case, e.g., if $p_i = p_j$ for all $i, j \in N$), then we can apply inequality (i) in the proof of Theorem 3.1 with $\alpha = 1/2$ to bound the total busy time, whereas we can use inequality (ii) with $\alpha = 0$ to bound the total idle time λ by $S_j^{\mathrm{LP}} = C_j^{\mathrm{LP}}(0)$. We obtain the following result.

COROLLARY 3.7. *Let C^{LP} denote an optimal solution to the LP relaxation* (3.4). *If $M_1^{\mathrm{LP}} \le M_2^{\mathrm{LP}} \le \cdots \le M_n^{\mathrm{LP}}$ implies $S_1^{\mathrm{LP}} \le S_2^{\mathrm{LP}} \le \cdots \le S_n^{\mathrm{LP}}$, then job-driven list scheduling by LP midpoints is a 3-approximation algorithm for the problem* P|prec. delays $d_{ij}|\sum w_jC_j$.

Because the arguments leading to Corollary 3.5 apply here as well, Corollary 3.7 actually gives a $(3-1/m)$-approximation algorithm for this special case of P|prec|$\sum w_jC_j$. Hall et al. (1997) had earlier proved these performance ratios for the special cases P|r_j, prec, $p_j = 1|\sum w_jC_j$ and P|prec, $p_j = 1|\sum w_jC_j$, respectively.

Let us finally consider "stiff" instances of P|prec|$\sum w_jC_j$. Margot, Queyranne, and Wang (2003) called an instance (of the single-machine problem 1|prec|$\sum w_jC_j$) stiff if $w(I)/p(I) \le w(N)/p(N)$ for all initial sets $I \subseteq N$. A set I is initial if $j \in I$ and $(i,j) \in A$ imply $i \in I$. Chekuri and Motwani (1999) and Margot, Queyranne, and Wang (2003) showed that the total weighted completion time of any feasible one-machine schedule of a stiff instance is within a factor 2 of that of an optimum. Given an instance of P|prec|$\sum w_jC_j$, we define a corresponding single-machine instance with processing times p_j/m, and we keep the original job weights and precedence constraints. The objective value of an optimal solution to this single-machine instance is a lower bound on the cost of an optimal schedule C^* for the original instance on m identical parallel machines (Chekuri et al. (2001)). Let us argue that job-driven list scheduling according to optimal LP start times $S_j^{\mathrm{LP}} = C_j^{\mathrm{LP}} - p_j$ is a $(3-1/m)$-approximation for stiff instances of P|prec|$\sum w_jC_j$. In fact, it suffices when C^{LP} is an optimal solution to the LP with constraint set (3.1)–(3.2), which can be solved combinatorially using shortest-path computations. That is, inequalities (3.3) are not needed. So, let L be defined by $S_1^{\mathrm{LP}} \le S_2^{\mathrm{LP}} \le \cdots \le S_n^{\mathrm{LP}}$. From the preceding discussion, we already know that we can use inequality (ii) in the proof of Theorem 3.1 with $\alpha = 0$ to bound the total idle time λ by S_j^{LP}. It remains to bound $\sum_{i=1}^{j} p_i/m$. This time, we do not give a job-by-job bound, but bound the entire contribution of busy periods to the objective function value of the heuristic schedule. Let C^1 be the completion time vector of an optimal schedule to the corresponding single-machine instance. Note that $\sum_{i=1}^{j} p_i/m$ is the completion time of job j in the single-machine schedule where jobs are sequenced according to L. Hence, as the instance is stiff, $\sum_{j=1}^{n} w_j \sum_{i=1}^{j} p_i/m \le 2\sum_{j=1}^{n} w_jC_j^1$. Overall, we obtain

$$\begin{aligned}\sum_{j=1}^{n} w_jC_j^{\mathrm{H}} &\le \sum_{j=1}^{n} w_j \left(\sum_{i=1}^{j} p_i/m + \bigl(1-1/m\bigr)\bigl(S_j^{\mathrm{LP}} + p_j\bigr) \right) \\ &\le 2\sum_{j=1}^{n} w_jC_j^1 + \bigl(1-1/m\bigr)\sum_{j=1}^{n} w_jC_j^{\mathrm{LP}} \\ &\le \bigl(3-1/m\bigr)\sum_{j=1}^{n} w_jC_j^*.\end{aligned}$$

COROLLARY 3.8. *Let C^{LP} denote an optimal solution to the LP relaxation defined over* (3.1)–(3.2). *Job-driven list scheduling by LP start times is a combinatorial $(3-1/m)$-approximation algorithm for stiff instances of the scheduling problem* P|prec|$\sum w_jC_j$.

Finally, let us point out that Corollaries 3.5, 3.6, and 3.7 also imply corresponding bounds on the quality of the LP relaxation (3.4) for these special cases.

Acknowledgments. The authors are grateful to Alix Munier for helpful discussions and to Maxim Sviridenko for useful comments on an earlier version of this paper. They also thank the organizers of the workshop on "Parallel Scheduling" held July 14–18, 1997, at Schloß Dagstuhl, Germany, where this work was initiated.

REFERENCES

E. Balas, J. K. Lenstra, and A. Vazacopoulos (1995), *The one-machine problem with delayed precedence constraints and its use in job shop scheduling*, Manag. Sci., 41, pp. 94–109.

M. Bartusch, R. H. Möhring, and F. J. Radermacher (1988), *Scheduling project networks with resource constraints and time windows*, Ann. Oper. Res., 16, pp. 201–240.

D. Bernstein and I. Gertner (1989), *Scheduling expressions on a pipelined processor with a maximal delay of one cycle*, ACM Trans. Program. Lang. Syst., 11, pp. 57–66.

P. Brucker and S. Knust (1999), *Complexity results for single-machine problems with positive finish-start time-lags*, Computing, 63, pp. 299–316.

J. L. Bruno, J. W. Jones III, and K. So (1980), *Deterministic scheduling with pipelined processors*, IEEE Trans. Comput., 29, pp. 308–316.

L. Bruno, E. G. Coffman, Jr., and R. Sethi (1974), *Scheduling independent tasks to reduce mean finishing time*, Comm. ACM, 17, pp. 382–387.

S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein (1996), *Improved scheduling algorithms for minsum criteria*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 1099, F. Meyer auf der Heide and B. Monien, eds., Springer, Berlin, pp. 646–657.

C. Chekuri and R. Motwani (1999), *Precedence constrained scheduling to minimize sum of weighted completion time on a single machine*, Discrete Appl. Math., 98, pp. 29–38.

C. Chekuri, R. Motwani, B. Natarajan, and C. Stein (2001), *Approximation techniques for average completion time scheduling*, SIAM J. Comput., 31, pp. 146–166.

F. A. Chudak and D. S. Hochbaum (1999), *A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine*, Oper. Res. Lett., 25, pp. 199–204.

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

L. Finta and Z. Liu (1996), *Single machine scheduling subject to precedence delays*, Discrete Appl. Math., 70, pp. 247–266.

M. R. Garey and D. S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco.

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

M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang (2002), *Single machine scheduling with release dates*, SIAM J. Discrete Math., 15, pp. 165–192.

R. L. Graham (1966), *Bounds for certain multiprocessing anomalies*, Bell System Tech. J., 45, pp. 1563–1581.

R. L. Graham (1969), *Bounds on multiprocessing timing anomalies*, SIAM J. Appl. Math., 17, pp. 416–429.

L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein (1997), *Scheduling to minimize average completion time: Off-line and on-line approximation algorithms*, Math. Oper. Res., 22, pp. 513–544.

L. A. Hall and D. B. Shmoys (1989), *Approximation schemes for constrained scheduling problems*, in Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, Research Triangle Park, NC, pp. 134–139.

L. A. Hall and D. B. Shmoys (1990), *Near-optimal sequencing with precedence constraints*, in Proceedings of the 1st Integer Programming and Combinatorial Optimization Conference, R. Kannan and W. R. Pulleyblank, eds., University of Waterloo Press, Waterloo, ON, pp. 249–260.

L. A. Hall and D. B. Shmoys (1992), *Jackson's rule for single-machine scheduling: Making a good heuristic better*, Math. Oper. Res., 17, pp. 22–35.

L. A. Hall, D. B. Shmoys, and J. Wein (1996), *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, pp. 142–151.

W. S. Herroelen and E. L. Demeulemeester (1995), *Recent advances in branch-and-bound procedures for resource-constrained project scheduling problems*, in Scheduling Theory and Its Applications, P. Chrétienne, E. G. Coffman, Jr., J. K. Lenstra, and Z. Liu, eds., John Wiley and Sons, Chichester, UK, pp. 259–276.

J. A. Hoogeveen, P. Schuurman, and G. J. Woeginger (2001), *Non-approximability results for scheduling problems with minsum criteria*, INFORMS J. Comput., 13, pp. 157–168.

J. M. Jaffe (1980), *Efficient scheduling of tasks without full use of processor resources*, Theoret. Comput. Sci., 12, pp. 1–17.

R. Kolisch (1996), *Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation*, European J. Oper. Res., 90, pp. 320–333.

E. Lawler, J. K. Lenstra, C. Martel, B. Simons, and L. Stockmeyer (1987), *Pipeline Scheduling: A Survey*, Technical report RJ 5738 (57717), IBM Research Division, San Jose, CA.

E. L. Lawler (1978), *Sequencing jobs to minimize total weighted completion time subject to precedence constraints*, Ann. Discrete Math., 2, pp. 75–90.

J. K. Lenstra and A. H. G. Rinnooy Kan (1978), *Complexity of scheduling under precedence constraints*, Oper. Res., 26, pp. 22–35.

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

J. Y.-T. Leung, O. Vornberger, and J. D. Witthoff (1984), *On some variants of the bandwidth minimization problem*, SIAM J. Comput., 13, pp. 650–667.

F. Margot, M. Queyranne, and Y. Wang (2003), *Decompositions, network flows, and a precedence constrained single-machine scheduling problem*, Oper. Res., 51, pp. 981–992.

A. Munier, M. Queyranne, and A. S. Schulz (1998), *Approximation bounds for a general class of precedence constrained parallel machine scheduling problems*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1412, R. Bixby, E. A. Boyd, and R. Z. Rios Mercado, eds., Springer, Berlin, pp. 367–382.

K. V. Palem and B. B. Simons (1993), *Scheduling time-critical instructions on RISC machines*, ACM Trans. Program. Lang. Syst., 15, pp. 632–658.

C. Phillips, C. Stein, and J. Wein (1998), *Minimizing average completion time in the presence of release dates*, Math. Programming, 82, pp. 199–223.

M. Queyranne (1993), *Structure of a simple scheduling polyhedron*, Math. Programming, 58, pp. 263–285.

A. S. Schulz (1996a), *Polytopes and Scheduling*, Ph.D. thesis, Technische Universität Berlin, Berlin.

A. S. Schulz (1996b), *Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1084, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer, Berlin, pp. 301–315.

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

A. S. Schulz and M. Skutella (2002a), *The power of α-points in preemptive single machine scheduling*, J. Sched., 5, pp. 121–133.

A. S. Schulz and M. Skutella (2002b), *Scheduling unrelated machines by randomized rounding*, SIAM J. Discrete Math., 15, pp. 450–469.

P. Schuurman (1998), *A fully polynomial approximation scheme for a scheduling problem with intree-type precedence delays*, Oper. Res. Lett., 23, pp. 9–11.

E. D. Wikum, D. C. Llewellyn, and G. L. Nemhauser (1994), *One-machine generalized precedence constrained scheduling problems*, Oper. Res. Lett., 16, pp. 87–99.

L. A. Wolsey (1985), *Mixed Integer Programming Formulations for Production Planning and Scheduling Problems*, invited talk at the 12th International Symposium on Mathematical Programming, Massachusetts Institute of Technology, Cambridge, MA.