Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems

Alix Munier* Maurice Queyranne† Andreas S. Schulz‡

Abstract

A well studied and difficult class of scheduling problems concerns parallel machines and precedence constraints. In order to model more realistic situations, we consider precedence delays, associating with each precedence constraint a certain amount of time which must elapse between the completion and start times of the corresponding jobs. Release dates, among others, may be modeled in this fashion. We provide the first constant-factor approximation algorithms for the makespan and the total weighted completion time objectives in this general class of problems. These algorithms are rather simple and practical forms of list scheduling. Our analysis also unifies and simplifies that of a number of special cases heretofore separately studied, while actually improving some of the former approximation results.

1 Introduction

Scheduling problems involving precedence constraints are among the most difficult problems in the area of machine 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 [16] with performance ratio $2-1/m$ is not only more than thirty years old, but it is also still essentially the best one available for this problem. On the other hand, it is only known that no polynomial-time algorithm can have a better approximation ratio than $4/3$ unless P = NP [25]. (ii) The computational complexity of the problem $\mathrm{P}m|p_j = 1, \mathrm{prec}|C_{\max}$, open problem OPEN8 from the original list of Garey and Johnson [12] is still open. (iii) The situation is also unsatisfactory with machines running at different speed, for which no constant-factor approximation algorithms are known. For the makespan objective, Chudak and Shmoys [10] only recently improved to $\mathrm{O}(\log m)$ an almost twenty year old approximation ratio of $\mathrm{O}(\sqrt{m})$ due to Jaffe [22]. They obtained the same approximation ratio for the total weighted completion time objective. (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, a better

*Université Paris Val de Marne, Laboratoire LACL, 61 avenue du Général de Gaulle, 94010, Créteil, France, *Alix.Munier@lip6.fr*

†Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, B.C., Canada V6T 1Z2, *Maurice.Queyranne@commerce.ubc.ca*

‡MIT, Sloan School of Management and Operations Research Center, E53-361, Cambridge, MA 02139, USA, *schulz@mit.edu*

understanding of linear programming (LP) relaxations and their use to guide solution strategies led to a 2– and a 2.719–approximation algorithm for $1|\text{prec}|\sum w_jC_j$ and $1|r_j,\text{prec}|\sum w_jC_j$, respectively [18, 35, 36], and to a 5.328–approximation algorithm for $\text{P}|r_j,\text{prec}|\sum w_jC_j$ [8]. Few deep negative results are known for these problems (see [21] for the total completion time objective).

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. [18, Page 530]:

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

Indeed their algorithm, as well as its improvement by Chakrabarti et al. [8], is rather elaborate and its performance ratio does not match the quality of the lower bound it uses. We show that using the same LP relaxation 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.

In this paper, we consider a general class of precedence-constrained scheduling problems on identical parallel machines. We have a set N of n jobs, and m identical parallel machines. Each job j has a nonnegative processing requirement (size) p_j and must be processed for that amount of time on any one of the machines. A job must be processed in an uninterrupted fashion, and a machine can process only one job at a time. We are interested in constrained scheduling problems in which each job j may have a release date r_j before which it cannot be processed, and there may be a partial order A on the jobs. We associate with each precedence-constrained job pair $(i,j)\in A$ a nonnegative *precedence delay* d_{ij}, with the following meaning: in every feasible schedule, job j cannot start until d_{ij} time units after job i is completed. Special cases include ordinary precedence constraints ($d_{ij}=0$); and release dates $r_j\geq 0$ (which may be modeled by adding a dummy job 0 with zero processing time and precedence delays $d_{0j}=r_j$ for all other jobs). Delivery times (or lags), which must elapse between the end of a job's processing and its actual completion time, may also be modeled by adding one or several dummy jobs and the corresponding precedence delays.

We denote the completion time of a job j in a schedule S as C_j^S and will drop the superscript S when it is clear to which schedule we refer. We consider the usual objectives of minimizing the makespan $C_{\max}=\max_j C_j$ and, for given nonnegative weights $w_j\geq 0$, a weighted sum $\sum_j w_jC_j$ of completion times. In an extension of the common notation introduced in [17], we may denote these problems as $\text{P}|\text{prec. delays } d_{ij}|C_{\max}$ and $\text{P}|\text{prec. delays } d_{ij}|\sum w_jC_j$, respectively. These problems are NP-hard (see, e.g., Lawler et al. [24]), and we discuss here the quality of relaxations and approximation algorithms. An *α–approximation algorithm* is a polynomial-time algorithm that delivers a solution with objective value at most α times the optimal value. Sometimes α is called the (worst-case) *performance guarantee* or *performance ratio* of the algorithm.

Precedence delays were considered for resource-constrained project scheduling under the name of "finish-to-start lags", e.g., by Bartusch, Möhring, and Radermacher [4] and Herroelen and Demeulemeester [20], for one-machine scheduling by Wikum, Llewellyn, and Nemhauser [39] under the name of "generalized precedence constraints", and by Balas, Lenstra, and Vazacopoulos [3] under that of "delayed precedence constraints"; the latter authors use 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. Most of the theoretical studies concerning this class of precedence constraints consider the one machine problem $1|\text{prec. delays } d_{ij}=k, p_j=1|C_{\max}$ which corresponds to a basic pipeline scheduling problem (see [23] for a survey). Leung, Vornberger, and Witthoff [27] showed that this problem is strongly NP-complete. Several other authors (e.g., [7, 5, 29, 11, 6]) obtained 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 [19] present polynomial time approximation schemes for $1|r_j, q_j|C_{\max}$ (here, "q_j" denotes delivery times), and Schuurman [38] presents a fully polynomial approximation scheme for $1|\text{prec. delays } d_{ij}|C_{\max}$ when the partial order A has a special structure introduced by Wikum et al. [39]. Besides these approximation schemes, the main approximation result is that Graham's list scheduling algorithm [16] was extended to $\text{P}|\text{prec. delays } d_{ij} = k, p_j = 1|C_{\max}$ to give a worst-case performance ratio of $2 - 1/(m(k+1))$ [23, 29]. We extend this result, in Section 3, to nonidentical precedence delays and processing times, and we refer to Brucker and Knust [6] for a nice overview of complexity results for single-machine problems with precedence delays including polynomially solvable cases with total completion time objective.

List scheduling algorithms, first analyzed by Graham [16] are among the simplest and most commonly used approximate solution methods for parallel machine scheduling problems. These algorithms use priority rules, or job rankings, which are often derived from solutions to relaxed versions of the problems. For example, several algorithms of Hall, Schulz, Shmoys, and Wein [18] use the job completion times obtained from linear programming relaxations. In Section 2 we show that the *non-idling property*, a typical consequence of list scheduling, may lead to an arbitrarily poor performance ratio for a weighted sum of completion times objective $\sum_{j\in N} w_j C_j$. On the other hand, a modified, "job-driven" version of list scheduling based on job completion times can, in the presence of precedence constraints, lead to solutions that are about as bad as m times the optimum, for both the $C_{\max}$ and $\sum_j w_j C_j$ objectives; this behavior may also occur when using actual *optimum* completion times.

Graham's original list scheduling, however, works well for minimizing the makespan, as we show in Section 3. There, we extend Graham's list scheduling to the case of precedence delays, and provide a detailed analysis of it worst-case performance ratio as a function of two parameters, the number m of machines and a parameter $\rho = \max_{(j,k)\in A} d_{jk} / \min_{i\in N} p_i$ which measures the "granularity" of the given instance.

For minimizing a weighted sum $\sum_j w_j C_j$ of completion times, we present in Section 4 a new algorithm with approximation ratio bounded by 4 for the general problem with precedence delays. This algorithm is based on an LP relaxation of this problem, which is an immediate extension of earlier LP relaxations proposed by Hall et al. [18]. The decision variables in this relaxation are the completion time C_j of every job j, and we choose to ignore the machine assignments in these relaxations. There are two sets of linear constraints, one representing the precedence delays (and, through the use of a dummy job, the release dates) 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 bounded by 4 for the general problem described above. Recall that 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 (or may be considered as) nonpreemptive then the midpoint of job j is simply $C_j^R - p_j/2$ where C_j^R is its completion time in the relaxation R. The advantage of using midpoints in the analysis of approximation algorithms was first observed by Goemans [13] and has since then been used by several authors (e.g., [36, 37, 14, 15]). Our result seems to be the first, however, where midpoints are really needed within the algorithm itself. We also show how the analysis yields tighter bounds for some special cases, and then conclude with some additional remarks in Section 5. We believe that the approach of applying a list-scheduling rule in which the jobs are ordered based on their midpoints in an LP solution will have further consequences for the design of approximation algorithms.

In summary, the main contributions of this paper are as follows.

1. We clarify the relationship between two forms of List Scheduling Algorithms (LSAs): Graham's non-idling LSAs and job-based LSAs. In particular, it is shown that the former are 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 $\sum_j w_j C_j$. In contrast, we present job-based LSAs with bounded performance ratio for the latter objective.

2. We show that using *job completion times* as a basis for job-based list scheduling may yield very poor schedules for problems with parallel machines, precedence constraints and weighted sum of completion times objective. This may happen even if the completion times are those of an *optimal* schedule.

3. In contrast, we show that job-based list scheduling according to *job midpoints* from an appropriate LP relaxation leads to job-by-job error ratios of at most 4 for a broad class of problems.

4. We present a general model of scheduling with precedence delays. This also allows us to treat in a unified framework ordinary precedence constraints, release dates and delivery times. In particular, this simplifies and unifies the analysis and proof techniques.

5. Finally, 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 (including release dates and delivery times) and either a makespan or total weighted completion time objective. These bounds are obtained by using relatively simple LSAs which should be of practical as well as theoretical interest. We also present the best polynomially solvable relaxations known so far for such problems with the latter objective. The approximation results are summarized in Table 1. Recall that the parameter ρ is defined as $\rho = \max_{(j,k)\in A} d_{jk} / \min_{i\in N} p_i$, and m denotes the number of identical parallel machines.

Problem	**New bound**	**Best earlier bound**
$1\|\text{prec. delays } d_{ij}\|C_{\max}$	$\min(2-\frac{1}{1+\rho}, 1+\frac{\rho}{2})$	?
$P\|\text{prec. delays } d_{ij}\|C_{\max}$	$2-\frac{1}{m(1+\rho)}$	?
$P\|\text{prec. delays } d_{ij}\|\sum w_j C_j$	4	?
$P\|r_j, \text{prec}\|\sum w_j C_j$	4	5.328 Chakrabarti et al. [8]
$P\|\text{prec}\|\sum w_j C_j$	$4-\frac{2}{m}$	5.328 Chakrabarti et al. [8]
$1\|\text{prec. delays } d_{ij}\|\sum w_j C_j$	3	?

Table 1: Summary of results.

2 List Scheduling Algorithms

In a seminal paper, Graham (1966) showed that a simple list-scheduling rule is a $(2 - \frac{1}{m})$-approximation algorithm for P|prec|$C_{\max}$. In this algorithm, the jobs are ordered in some list, and whenever one of the m machines becomes idle, the next available job on the list is started on that machine, where a job is *available* if all of its predecessors have completed processing. By their non-idling property, Graham's List Scheduling Algorithms (GLSAs) are appropriate when machine utilization is an important consideration. Indeed, it is shown in Section 3 that, for the makespan minimization problem P|prec. delays $d_{ij}|C_{\max}$, any GLSA (i.e., no matter which list of jobs is used) produces a schedule with objective function value within a factor 2 of the optimum. In this case, a job is available if all its predecessors are completed and the corresponding precedence delays have elapsed.

In contrast, the elementary Example 2.1 below shows that the non-idling property may lead to an arbitrarily poor performance ratio for a weighted sum of completion times objective $\sum_{j\in N} w_j C_j$.

Example 2.1. Consider the following two-job instance of the single machine nonpreemptive scheduling problem $1|r_j|\sum w_j C_j$ (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 optimum schedule 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 non-idling heuristic starts processing job 1 at time 0, leading to an objective value at least $q^3 + q^2 + q$, and its performance ratio is unbounded as q may be arbitrarily large. □

A different example of the same type but using ordinary precedence constraints rather than release dates can be found in [34, Page 82, Ex. 2.20]. Thus to obtain a bounded performance for the weighted sum of completion times objective $\sum_j w_j C_j$, we must relax the non-idleness property. One strategy, leading to *job-based* nonpreemptive list scheduling algorithms, is to consider the jobs one by one, in the given list order, starting from an empty schedule. Each job is non-preemptively inserted in the current schedule without altering the jobs already scheduled. Specific list scheduling algorithms differ in how this principle is implemented, in particular, for parallel machines, regarding the assignment of the jobs to the machines. For definiteness, consider the following version, 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. Notice that the given list is assumed to be a linear extension of the poset defined by the precedence constraints.

Job-based List Scheduling Algorithm for P|prec. delays $d_{ij}|\cdot$

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 all $h = 1, \ldots, m$.
3. For $k = 1$ to n do:
   3.1 Let job $j = \ell(k)$, its start time $S_j = \max(\max\{C_i + d_{ij} : (i, j) \in A\}, \min\{\Gamma_h : h = 1, \ldots, m\})$ and its completion time $C_j = S_j + p_j$.
   3.2 Assign job j to a machine h such that $\Gamma_h \leq S_j$. Update $\Gamma_h = C_j$.

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). Note also that the above algorithm can be modified to allow insertion of a job in an idle period before Γ_h on a machine h. In effect, the observations below also apply to all these variants.

One method (e.g., Phillips et al. [30] and Hall et al. [18]) for defining the list L consists in 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 (Hall et al., ibid.; see also [35]), but Example 2.2 below 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; note that this optimal schedule defines the same completion time order as the relaxation in Hall et al. and its extension in Section 4 below.

Example 2.2. For a fixed number $m \geq 2$ of identical parallel machines and a positive number ϵ, let the job set be $N = \{1, \ldots, n\}$ with $n = m(m+1)+1$. The ordinary precedence constraints (j, k) (with $d_{jk} = 0$) are defined as follows: (i) $j = 1+h(m+1)$ and $k = j+g$, for all $h = 0, \ldots, m-1$ and all $g = 1, \ldots, m$; and (ii) for all $j < n$ and $k = n$. The processing times are $p_j = 1 + h(m+1)\epsilon$ for $j = 1+h(m+1)$ and $h = 0, \ldots, m-1$; and $p_j = \epsilon$ otherwise. The objective is either to minimize the makespan, or a weighted sum $\sum_j w_j C_j$ of job completion times with weights $w_j = 0$ for all $j < n$ and $w_n = 1$; note that, due to the precedence constraints (ii) above, these two objectives coincide for any feasible schedule.

An optimal solution has, for $h = 0, \ldots, m-1$, job $j = 1 + h(m+1)$ starting at time $S_j^* = 0$ on machine $h+1$, immediately followed by jobs $j+1, \ldots, j+m$ assigned as uniformly as possible to machines $1, \ldots, h+1$. Job n is then processed last on machine m, so that the optimal objective value is $C^*_{\max} = C^*_n = 1 + (m^2+1)\epsilon$. A corresponding list is $L = (1, 2, \ldots, n)$. Any version of the list scheduling algorithm described above produces the following schedule from this list: job 1 is scheduled with start time $S_1^L = 0$ and completion time C_1^L=1; the m jobs $k = 2, \ldots, m+1$ are then scheduled, each with $S_k^L = 1$ and $C_k^L = 1 + \epsilon$ on a different machine; this will force all subsequent jobs to be scheduled no earlier than time $1 + \epsilon$. As a result, for $h = 0, \ldots, m-1$, job $j = 1 + h(m+1)$ is scheduled with start time $S_j^L = h + (\frac{1}{2}(h-1)h(m+1) + h)\epsilon$, followed by jobs $k = j+1, \ldots, j+(m+1)$ each with $S_k^L = h + 1 + (\frac{1}{2}h(h+1)(m+1) + h)\epsilon$ on a different machine. Finally, job n is scheduled last with $S_n^L = m + (\frac{1}{2}(m-1)m(m+1) + m)\epsilon$ and thus the objective value is $C^L_{\max} = C^L_n = m + o(\epsilon)$, or arbitrarily close to m times the optimal value $C^*_{\max}$ for $\epsilon > 0$ small enough. □

The example shows that list scheduling according to completion times can lead to poor schedules on identical parallel machines with job precedence constraints. In contrast, we will present in Section 4 a linear programming relaxation of the general problem with precedence delays and show that job-based list scheduling according to job midpoints leads to a bounded performance ratio.

3 The Performance of Graham's List Scheduling for Makespan Minimization

In this section we show that Graham's (non-idling) List Scheduling Algorithms generate feasible schedules with makespan less than twice the optimum for instances of P|prec. delays $d_{ij}|C_{\max}$. As discussed in the introduction, this result and proof extend and unify earlier work. We provide a detailed analysis of its worst-case performance ratio as a function of the number m of machines and the "granularity" parameter $\rho = \max_{(j,k)\in A} d_{jk} / \min_{i\in N} p_i$. We may assume that $\min_{i\in N} p_i > 0$; otherwise, jobs with zero processing times may be eliminated using an obvious transformation of the graph.

Let $S^H = (S_1^H, \dots, S_n^H)$ denote the vector of start times of a schedule constructed by GLSA, as described in Section 2. Let $C_{\max}^H = \max_{i\in N}(S_i^H + p_i)$ denote the makespan of this schedule. For any pair (t, t') of dates such that $0 \le t \le t' \le C_{\max}^H$, let $\mathcal{I}[t, t')$ denote the total machine idle time during the interval $[t, t')$. (Thus, for example, if all m machines are idle during the whole interval, then $\mathcal{I}[t, t') = m(t' - t)$.) Let N^+ denote the set of jobs in N that have at least one predecessor.

Lemma 3.1. *Let j be a job in N^+ and let i be an immediate predecessor of j with largest value $S_i^H + p_i + d_{ij}$. Then $\mathcal{I}[S_i^H, S_j^H) \le m\, d_{ij} + (m-1)p_i$.*

Proof. Let i be an immediate predecessor of j with largest value of $S_i^H + p_i + d_{ij}$. Since H is a GLSA, job j may be processed by any available processor from date $S_i^H + p_i + d_{ij}$ on. Therefore $\mathcal{I}[S_i^H + p_i + d_{ij}, S_j^H) = 0$ and hence $\mathcal{I}[S_i^H, S_j^H) = \mathcal{I}[S_i^H, S_i^H + p_i + d_{ij})$. Since job i is processed during the time interval $[S_i^H, S_i^H + p_i)$, at most $m-1$ machines are idle during this interval, and we obtain the requisite inequality. □

The *precedence network* (N^0, A^0, ℓ) is defined by $N^0 = N \cup \{0\}$; $A^0 = A \cup \{(i,0) : i \in N\}$; and arc lengths $\ell(i,j) = p_i + d_{ij}$ for all $(i,j) \in A$, and $\ell(i,0) = p_i$ for all $i \in N$. Job 0 is a dummy job succeeding all other jobs with zero delays. It is added so that any maximal path in the precedence network is of the form $i \dots j0$ and its length is the minimum time required to process all the jobs $i, \dots, j$.

Lemma 3.2. *There exists a path $P = (i_k, i_{k-1}, \dots, i_0)$ in the precedence network such that*

$$\mathcal{I}[0, C_{\max}^H) \le m \sum_{q=0}^{k-1} d_{i_{q+1} i_q} + (m-1) \sum_{q=0}^{k} p_{i_q}.$$

Proof. Let $i_0 \in N$ such that $S_{i_0}^H + p_{i_0} = C_{\max}^H$. Starting with $q = 0$, we construct the path $P = (i_k, i_{k-1}, \dots, i_0)$ in the precedence network as follows:

1. While $i_q \in N^+$ do:
   choose as i_{q+1} an immediate predecessor of i_q with largest value $S_{i_{q+1}}^H + p_{i_{q+1}} + d_{i_{q+1} i_q}$;
   $q := q + 1$.
2. Set $k = q$ and stop.

Since i_k has no predecessor, $\mathcal{I}[0, S_{i_k}^H) = 0$. By repeated application of Lemma 3.1, we have $\mathcal{I}[S_{i_k}^H, S_{i_0}^H) \le m \sum_{q=0}^{k-1} d_{i_{q+1} i_q} + (m-1) \sum_{q=1}^{k} p_{i_q}$. In addition, $\mathcal{I}[S_{i_0}^H, C_{\max}^H) \le (m-1)p_{i_0}$, and therefore $\mathcal{I}[0, C_{\max}^H) \le m \sum_{q=0}^{k-1} d_{i_{q+1} i_q} + (m-1) \sum_{q=0}^{k} p_{i_q}$. □

Lemma 3.3. *Let L denote the length of a longest path in the precedence network. Then*

$$\mathcal{I}[0, C_{\max}^H) \le (m - 1/(1+\rho))L.$$

Proof. Let $P = (i_k, \dots, i_0)$ denote a path in the precedence network which verifies Lemma 3.2. By definition of L, we have $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} + \sum_{q=0}^{k} p_{i_q} \le L$. Moreover, $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} \le \rho \sum_{q=0}^{k-1} \min_{i\in N} p_i \le \rho \sum_{q=0}^{k-1} p_{i_q}$. Therefore, $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} \le \frac{\rho}{1+\rho} L$. Using that $\mathcal{I}[0, C_{\max}^H) \le m \sum_{q=0}^{k-1} d_{i_{q+1} i_q} + (m-1) \sum_{q=0}^{k} p_{i_q}$, we obtain the inequality. □

We are now in position to prove the first main result of this section:

Theorem 3.4. *For the scheduling problem* P|prec. delays $d_{ij}|C_{\max}$ *the performance ratio of Graham's List Scheduling Algorithm is no larger than* $2 - 1/(m(1+\rho))$ *for an arbitrary priority list. This bound is (asymptotically) tight for every* $m \geq 2$ *and* $\rho \geq 0$.

Proof. We have $m\, C_{\max}^H = \sum_{i \in N} p_i + \mathcal{I}[0, C_{\max}^H)$, and thus, by Lemma 3.3,

$$C_{\max}^H \leq \frac{\sum_{i \in N} p_i}{m} + \left(1 - \frac{1}{m(1+\rho)}\right) L$$

Since $\frac{1}{m} \sum_{i \in N} p_i$ and L are two lower bounds on the optimum makespan, the result follows. This bound is asymptotically best possible for $m \geq 2$, as shown by Example 3.5 below. □

Example 3.5. Fix $m \geq 2$ and $\rho \geq 0$, and let $k \geq 3$ be any positive integer such that $\rho(k-1)$ is integer. Let the set of jobs $N = \{1, \ldots, n\}$ with $n = 2k - 2 + (m-2)(k + \rho(k-1))$. The processing time of each of the first $n_1 = (m-2)(k + \rho(k-1))$ jobs is $p_j = 1$. For each of the next $n_2 = k-2$ jobs we set $p_j = 1 + 2\rho$ and for the last $n_3 = k$ jobs we set $p_j = 1$. The precedence constraints are $(j, j+1)$ for $j = n_1 + n_2 + 1, \ldots, n_1 + n_2 + n_3 - 1$ with $d_{j,j+1} = \rho$. Thus the length of the path $P = (n_1 + n_2 + 1, \ldots, n_1 + n_2 + n_3)$ is $L = k + \rho(k-1)$.

An optimal solution with makespan $C_{\max}^* = L$ can be built as follows: execute the n_3 jobs $n_1 + n_2 + 1, \ldots, n_1 + n_2 + n_3$ as soon as possible and according to the precedence constraints, alternatively on processors 1 and 2. Fill the n_2 idle periods of length $1 + 2\rho$ on these two processors with the jobs $n_1 + 1, \ldots, n_1 + n_2$. The $m-2$ other processors are fully used by the jobs $1, \ldots, n_1$.

Now, with the list $(1, \ldots, n)$, the $n_1 + n_2$ first jobs are executed by the m processors followed by the path P. So, $C_{\max}^H \geq \lfloor (n_1 + n_2(1+2\rho))/m \rfloor + L \geq (2 - 1/(m(1+\rho))L - b(m,\rho)$ where $b(m,\rho)$ is independent of k. As k goes to infinity, so does L and the ratio $C_{\max}^H / C_{\max}^*$ approaches $2 - 1/(m(1+\rho))$. □

For a single machine (i.e., $m = 1$), we have:

Theorem 3.6. *For the scheduling problem* 1|prec. delays $d_{ij}|C_{\max}$, *the performance ratio of Graham's List Scheduling Algorithm is* $\min(2 - 1/(1+\rho), 1 + \rho/2)$ *for an arbitrary priority list. This bound is (asymptotically) tight for every* $\rho \geq 0$.

Proof. By Theorem 3.4 we only need to prove that $C_{\max}^H \leq (1 + \rho/2) C_{\max}^*$, where $C_{\max}^*$ denotes the optimal value of the makespan. Let $P = (i_k, \ldots, i_0)$ be a path in the precedence network, constructed as in the proof of Lemma 3.2. Since $\mathcal{I}[0, S_{i_k}^H) = 0$ we have $C_{\max}^H \leq \sum_{i \in N} p_i + \sum_{q=0}^{k-1} d_{i_{q+1} i_q}$.

1. If $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} \leq \rho \sum_{i \in N \setminus P} p_i$ then, since $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} \leq \rho \sum_{i \in P} p_i$, we have $2 \sum_{q=0}^{k-1} d_{i_{q+1} i_q} \leq \rho \sum_{i \in N} p_i$ and therefore $C_{\max}^H \leq (1 + \rho/2) \sum_{i \in N} p_i \leq (1 + \rho/2) C_{\max}^*$, as needed.

2. We assume now that $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} > \rho \sum_{i \in N \setminus P} p_i$. Let Δ be the total idle time in a given optimal schedule, so $C_{\max}^* = \sum_{i \in N} p_i + \Delta$. Consider all the time intervals corresponding to the precedence delays $d_{i_{q+1} i_q}$ $(q = 0, \ldots, k-1)$ in the optimal schedule: during these intervals the machine is either idle or processes jobs from $N \setminus P$. Since at most $|N \setminus P|$ of these intervals can be covered by jobs from $N \setminus P$, the total idle time satisfies $\Delta \geq \sum_{q=0}^{k-1} d_{i_{q+1} i_q} - |N \setminus P| \max_{(j,k) \in A} d_{jk}$. Since $\sum_{i \in N \setminus P} p_i \geq |N \setminus P| \min_{i \in N} p_i$, we have $\rho \sum_{i \in N \setminus P} p_i \geq |N \setminus P| \max_{(j,k) \in A} d_{jk}$. So, $C_{\max}^* \geq \sum_{i \in N} p_i + (\sum_{q=0}^{k-1} d_{i_{q+1} i_q} - \rho \sum_{i \in N \setminus P} p_i)$ and then $C_{\max}^H \leq$

$C^*_{\max} + \rho \sum_{i \in N \setminus P} p_i$. From the assumption $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} > \rho \sum_{i \in N \setminus P} p_i$, it then follows $C^H_{\max} \leq C^*_{\max} + (\rho/2) \sum_{i \in N \setminus P} p_i + (1/2) \sum_{q=0}^{k-1} d_{i_{q+1} i_q}$. Since $\sum_{q=0}^{k-1} d_{i_{q+1} i_q} \leq \rho \sum_{i \in P} p_i$ we obtain $C^H_{\max} \leq (1 + \rho/2) \sum_{i \in N} p_i \leq (1 + \rho/2) C^*_{\max}$, as needed.

Example 3.7 below shows that this bound is asymptotically tight. □

Example 3.7. Let $m = 1$ and $n = 2k+1$. We assume that the precedence constraints are $(j, j+1)$ for $j = k+1, \ldots, n-1$ with $d_{j,j+1} = \rho$. We consider the two following cases:

1. If $\rho > 1$, then $1 + \rho/2 > 2 - 1/(1+\rho)$. We assume here that processing times are $p_j = \rho$ for $j = 1, \ldots, k$, and $p_j = 1$ otherwise. The optimum makespan is $C^*_{\max} = \sum_{j \in N} p_j = k(\rho+1)+1$, obtained by alternately processing jobs $k+1, 1, k+2, 2, \ldots, n$ starting at time 0 and without any machine idle time. With list $(1, 2, \ldots, n)$ we obtain a makespan $C^H_{\max} = k\rho + C^*_{\max}$. As k goes to infinity, the ratio $C^H_{\max}/C^*_{\max}$ approaches $2 - 1/(1+\rho)$.

2. Otherwise, $1 + \rho/2 \leq 2 - 1/(1+\rho)$. We assume that $p_j = 1$ for all $j = 1, \ldots, 2k+1$. The optimal makespan is $C^*_{\max} = 2k+1$ and is obtained as in the first case. With the list $(1, 2, \ldots, n)$, we get $C^H_{\max} = k + (k+1) + k\rho$. As k goes to infinity, the ratio $C^H_{\max}/C^*_{\max}$ approaches $1 + \rho/2$. □

4 An Approximation Algorithm for Minsum Objectives

In this section we present a linear programming relaxation of the problem of minimizing a weighted sum $\sum w_j C_j$ of job completion times subject to precedence delays, and use it to develop a 4–approximation algorithm for this problem. This formulation is a direct extension of a formulation given in [18], see also [34]. The decision variables are the job completion times C_j for all jobs $j \in N$. Note that this relaxation does not take into account the assignment of jobs to machines. The set of constraints is:

$$C_j \geq C_i + d_{ij} + p_j \qquad \text{all } (i,j) \in A, \tag{1}$$

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

Constraints (1) are the precedence delay constraints. Constraints (2) are a relatively weak way of expressing the requirement that each machine can process at most one job at a time[1]; for the single-machine case ($m = 1$) they were introduced by Wolsey [40] and Queyranne [31], and studied by Queyranne and Wang [32], von Arnim, Schrader, and Wang [1], von Arnim and Schulz [2], and Schulz [34] in the presence of ordinary precedence constraints; they were extended to $m \geq 2$ parallel machines by Hall et al. [18]. Note that these constraints, for $F = \{j\}$, imply $C_j \geq 0$; these and, as indicated above, the use of a dummy job 0 allow the formulation of release date constraints.

[1] We say that these constraints are "relatively weak" in the following sense: (i) in the absence of precedence constraints, all these constraints are facet inducing for the corresponding scheduling polyhedron in the completion time variables C_j (see [31]), whereas only few of them are for $m \geq 2$ (see [33]); (ii) with precedence constraints, the facet inducing inequalities among these constraints are easily characterized for $m = 1$ (see [32]), whereas deciding which ones are tight (not to mention facet inducing) is NP-hard for $m \geq 2$ (see [26]). The surprising fact is that using these "weak" constraints and ignoring the machine assignments is sufficient to obtain the best approximation bounds known so far for this class of problems.

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

$$\text{minimize} \sum_{j \in N} w_j C_j \quad \text{subject to } (1) - (2). \tag{3}$$

Let C^{LP} denote any feasible solution to the constraint set (1)–(2) of this LP; we will call C_j^{LP} the *LP completion time* of job j. We now 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* $M_j^{LP} = C_j^{LP} - p_j/2$. We now use the List Scheduling Algorithm of Section 2 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 4.1. *Let C^{LP} denote any feasible solution to the constraint set* (1)–(2) *and let M^{LP} denote the associated LP midpoints. Let S^H be the vector of start times of the feasible schedule constructed by the List Scheduling Algorithm using the LP midpoint list. Then*

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

Proof. Assume for simplicity that the jobs are indexed in the order of their LP midpoints, that is, $M_1^{LP} \leq M_2^{LP} \leq \cdots \leq M_n^{LP}$. 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 $[j] = \{1, \ldots, j\}$.

Let μ denote the total time between 0 and 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, we have $\mu \leq \frac{1}{m}\sum_{i=1}^{j-1} p_i$. Let $\lambda = S_j^H - \mu$. To prove (4), we only need to show that

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

Inequality (i) follows from a straightforward variant of Lemma 3.2 in [18]. For this, first observe that the inequalities (2) are equivalent to

$$\sum_{i \in F} p_i M_i \geq \frac{1}{2m}\left(\sum_{i \in F} p_i\right)^2 \qquad \text{all } F \subseteq N. \tag{5}$$

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

$$\left(\sum_{i \in [j-1]} p_i\right) M_j^{LP} \geq \sum_{i \in [j-1]} p_i M_i^{LP} \geq \frac{1}{2m}\left(\sum_{i \in [j-1]} p_i\right)^2$$

implying (i).

To show (ii), let q denote the number of time 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 $h = 1, \ldots, q$, so that $0 \leq a_1$; that $b_{h-1} < a_h < b_h$ for all $h = 2, \ldots, q$; and that $b_q \leq S_j^H$. We have $\lambda = \sum_{h=1}^{q}(b_h - a_h)$ and all machines are busy during the complementary intervals $[b_h, a_{h+1}]$, including intervals $[0, a_1]$ and $[b_q, S_j^H]$ if nonempty.

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

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

that is, $A^{[j]}$ is the set of precedence pairs in $[j]$ for which the precedence delay constraints (1) are tight for C^H. If $b_q > 0$, then a machine becomes busy at date b_q (or starts processing job j if $b_q = S_j^H$) and thus there exists a job $x(q) \in [j]$ with start time $S_{x(q)}^H = b_q$. Since $x(q) \in [j]$ we have $M_{x(q)}^{LP} \leq M_j^{LP}$. 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 $b_g < S_{v(1)}^H \leq a_{g+1}$ 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 $S_{v(1)}^H$ of job $v(1)$ and this job, not being constrained by any tight precedence delay constraint, should have started earlier than that date. This implies in particular that $s \geq 2$. We have

$$b_h - a_{g+1} \leq S_{v(s)}^H - S_{v(1)}^H = \sum_{i=1}^{s-1}\left(S_{v(i+1)}^H - S_{v(i)}^H\right) = \sum_{i=1}^{s-1}\left(p_{v(i)} + d_{v(i)v(i+1)}\right) . \tag{6}$$

On the other hand, the precedence delay constraints (1) imply

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

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

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

If $b_g > 0$, then let $x(g)$ denote a job with start time $S_{x(g)}^H$ satisfying $b_g \leq S_{x(g)}^H \leq a_{g+1}$ and with minimum value of $M_{x(g)}^{LP}$ under this condition. Therefore $M_{x(g)}^{LP} \leq M_{v(1)}^{LP}$ and

$$M_{x(h)}^{LP} - M_{x(g)}^{LP} \geq \frac{1}{2}(b_h - a_{g+1}) \ . \tag{7}$$

We also have $(k, x(g)) \in A^{[j]}$ for some $k \in [j]$ with $S_k^H < b_g \leq S_{x(g)}^H$, 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 (7) we obtain

$$\lambda \leq \sum_{i=1}^{q'}(b_{H(i)} - a_{H(i+1)+1}) \leq 2(M_{x(H(1))}^{LP} - M_{x(H(q'))}^{LP}) \leq 2(M_j^{LP} - 0) \ . \tag{8}$$

This establishes (ii). The proof of Theorem 4.1 is complete. $\square$

Example 4.2 below shows that the factor 4 in inequality (4) is (asymptotically) best possible for any number of machines.

Example 4.2. For a fixed number $m \geq 2$ of identical parallel machines, let the job set be $N = \{1, \ldots, n\}$ with $n = (m+1)^2$. The ordinary precedence constraints (j,k) (with $d_{ij} = 0$) are for $j = 1 + h(m+1)$ and $k = j + g$, for all $h = 0, \ldots, m-1$ and all $g = 1, \ldots, m$. The processing times are $p_j = 2^{h-m}$ for $j = 1 + h(m+1)$ and $h = 0, \ldots, m-1$; $p_j = 1$ for $m(m+1) < j < n$; and $p_j = 0$ for the remaining $m+1$ jobs. Finally, job n has a release date (possibly modeled using a dummy job and a precedence delay) $r_n = \frac{1}{2} + \frac{2}{m}$. The following solution C^{LP} is feasible for constraints (1)–(2): $C_j^{LP} = 2^{h-m}$ for $1 + h(m+1) \leq j \leq (h+1)(m+1)$ and $0 \leq h < m$; $C_j^{LP} = 1 + \frac{2}{m}$ for all other jobs $j < n$ and $C_n^{LP} = S_n^{LP} = r_n$. Therefore an LP-midpoint list is $L = (1, 2, \ldots, n)$, producing the following schedule: for $h = 0, \ldots, m-1$, schedule job $1 + h(m+1)$ on one machine, immediately followed by the m jobs $2 + h(m+1), \ldots, (h+1)(m+1)$ each on a different machine, and all with completion time $C_j^H = \sum_{i \leq h} 2^{i-m} = 2^{h+1-m} - 2^{-m}$; then schedule the m next jobs, each on a different machine and with completion time $C_j^H = 2 - 2^{-m}$; finally the last job with start time $S_n^H = 2 - 2^{-m}$. For m large enough, the latter expression is arbitrarily close to $4M_n^{LP} = 2 + \frac{8}{m}$. □

Using for C^{LP} an optimal LP solution, Theorem 4.1 implies performance ratios of 1/4 and 4 for the LP relaxation and the heuristic solution, respectively, for the $\sum w_j C_j$ objective.

Corollary 4.3. *Let C^{LP} denote an optimal solution to the LP defined in* (3) *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 List Scheduling Algorithm using the LP midpoint list, and let C^* denote an optimum schedule. Then*

$$\sum_{j \in N} w_j C_j^{LP} \geq \frac{1}{4} \sum_{j \in N} w_j C_j^* \quad \textit{and} \quad \sum_{j \in N} w_j C_j^H \leq 4 \sum_{j \in N} w_j C_j^* \ . \tag{9}$$

Example 4.4 below shows that the latter bound is (asymptotically) tight for an arbitrary number of machines.

Example 4.4. In Example 4.2, let the weights be $w_n = 1$ and all other $w_j = 0$, so the optimum solution has $wC^* = w_n(r_n + p_n) = \frac{1}{2} + \frac{2}{m}$. Then the solution C^{LP} described in Example 4.2 is optimal for the LP relaxation (3); its objective value is $wC^{LP} = \frac{1}{2} + \frac{2}{m} = wC^*$. Using the LP-midpoint list produces the same schedule as described in the above example, with $C_n^H = S_n^H + p_n = 2 - 2^{-m}$ and thus $wC^H = 2 - 2^{-m}$. For m large enough, the latter expression is arbitrarily close to $4wC^*$. □

We suspect that the first inequality in (9), bounding the performance ratio of the LP relaxation, is not tight. The worst instances we know have a performance ratio of 1/3 for the LP relaxation, see Example 4.5 below.

Example 4.5. For a fixed number $m \geq 2$ of identical parallel machines and an integer $q \geq 2$, let the job set be $N = \{1, \ldots, n\}$ with $n = q(2m+1) + 1$. The ordinary precedence constraints (j,k) (with $d_{jk} = 0$) are defined as follows: (i) $j = f + h(2m+1)$ and $k = g + (h+1)(2m+1)$, for all $h = 0, \ldots, q-1$ and all $f, g \in \{1, \ldots, 2m+1\}$, that is, for any j and k in two consecutive subsets of $2m+1$ jobs; and (ii) for all $j < n$ and $k = n$. The processing times are $p_j = 1$ for all $j < n$ and $p_n = 0$. As in Example 2.2, we let all $w_j = 0$ except that $w_n = 1$ so, with all precedences $(j,n) \in A$, the $C_{\max}$ and $\sum w_j C_j$ objectives coincide. The LP solution is as follows: for $0 \leq h < q$

we have $C_j^{LP} = h(1 + \frac{1}{2m}) + \frac{1}{2}$ for all $j = 1 + h(2m+1), \ldots, (h+1)(2m+1)$, as determined by constraint (2) with $F = \{1, \ldots, (h+1)(2m+1)\}$—as well as, for $h \geq 1$, by the precedence constraints; and $C_n^{LP} = q(1 + \frac{1}{2m}) + \frac{1}{2} = wC^{LP}$. An optimum schedule needs 3 time units to process each set $\{1 + h(2m+1), \ldots, (h+1)(2m+1)\}$ of $2m+1$ jobs, hence $wC^* = 3q$. For fixed m, wC^{LP} gets arbitrarily close to $\frac{1}{3} + \frac{1}{6m}$ times wC^* for q large enough; then for m large enough the ratio wC^{LP}/wC^* approaches the value $\frac{1}{3}$. □

The analysis in Theorem 4.1 may be refined for some special cases, yielding tighter performance ratios. For the problem P|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 4.1, 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 4.6. *List scheduling by LP midpoints is a $(4 - 2/m)$–approximation algorithm for the scheduling problem* P|prec|$\sum w_j C_j$.

Note that for $m = 1$ we recover the performance ratio of 2 for 1|prec|$\sum w_j C_j$ in [18, 35], which is known to be tight for that special case.

In the case of a single machine, the idle intervals that add up to λ time units cannot contain any processing. Therefore, in the proof of Theorem 4.1 replace inequality (6) with $b_h - a_{g+1} \leq S_{v(s)}^H - C_{v(1)}^H = \sum_{i=1}^{s-1} d_{v(i)v(i+1)}$. (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 for all $i = 1, \ldots, s-1$ and omitting some processing times yield $M_{x(h)}^{LP} - M_{v(1)}^{LP} \geq \sum_{i=1}^{s-1} d_{v(i)v(i+1)} \geq b_h - a_{g+1}$. Therefore we may replace (8) with $\lambda \leq \sum_{i=1}^{q'} (b_{H(i)} - a_{H(i+1)+1}) \leq M_{x(H(1))}^{LP} - M_{x(H(q'))}^{LP} \leq M_j^{LP}$ and thus inequality (ii) with $\lambda \leq M_j^{LP}$. This implies $S_j^H \leq 3M_j^{LP}$.

Corollary 4.7. *List scheduling by LP midpoints is a 3–approximation algorithm for the scheduling problem* 1|prec. delays d_{ij}|$\sum w_j C_j$.

Note that for the special case $1|r_j, \text{prec}|\sum w_j C_j$ we recover the performance ratio of 3 in [18, 35]. The best known approximation algorithm for this problem, however, has a performance guarantee of $e \approx 2.719$ [36]. Corollaries 4.6 and 4.7 also imply corresponding bounds on the quality of the LP relaxation (3) for these special cases.

5 Concluding Remarks

The appropriate introduction of idle time is an important element in the design of approximation algorithms to minimize the weighted sum of completion times subject to precedence delays. As Example 2.1 illustrates, idle time is needed to avoid that profitable jobs which become available soon are delayed by other, less important jobs. On the other hand, too much idle time is undesired as well. The necessity to balance these two effects contributes to the difficulty of this problem. Interestingly, 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. [18] partition jobs into groups that are individually scheduled according to a GLSA, and then these schedules are concatenated to obtain a solution of the original problem. To find a good partition, this scheme was enriched with randomness by Chakrabarti et al. [8]. Chekuri et al. [9] presented a different variant of a GLSA 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. Hence, the algorithm analyzed in Section 4 seems to be the first one within this context that does not take the machine-based point of view of GLSAs.

However, an advantage of the rather simple scheme of Chekuri et al. is its small running time (though the performance ratios obtained are worse). In fact, so far we have not even explained that the algorithms presented above are indeed polynomial-time algorithms. Whereas this is obvious for the GLSA variant for makespan minimization, we have to argue that in case of the total weighted completion time the linear programming relaxation (3) behaves well. In fact, it can be solved in polynomial time since the corresponding separation problem is polynomially solvable [34].

The job-by-job bounds obtained in Section 4 can be used to derive approximation bounds for more general objective functions than weighted sums of completion times. Since these bounds relate the heuristic start times to the LP midpoints, they may in particular be used to derive approximation bounds for weighted sums $\sum_j w_j C_j(\alpha)$ of α-points. Recall that, in a given schedule S and for a given value $0 < \alpha \leq 1$, the *α–point* $C_j^S(\alpha)$ of job j is the earliest time at which a fraction α of its processing has been performed; for $\alpha = 0$ we let $C_j^S(0) = C_j^S(0^+)$, i.e., the starting point in the schedule. If the schedule is nonpreemptive, the α–point of job j is thus simply $C_j^S - (1-\alpha)p_j$. Note that, for every $\alpha \in [0,1]$, $\sum_j w_j(C_j^{LP} - (1-\alpha)p_j)$ is a lower bound on the optimum value $\mathrm{OPT}(\alpha) = \min\{\sum_j w_j C_j^S(\alpha) : S \text{ feasible}\}$. As in Corollary 4.3 (resp., Corollary 4.7) it follows from Theorem 4.1 that $\sum_j w_j C_j^H(\alpha) \leq 4\,\mathrm{OPT}(\alpha)$ for all $\alpha \in [2/3, 1]$ (resp., $\leq 3\,\mathrm{OPT}(\alpha)$ for all $\alpha \in [3/4, 1]$ for the single machine problems). Incidentally, we also note that we may use other points than the LP midpoints in Section 4: we may use any *LP α-point* $C_j^{LP}(\alpha) = C_j^{LP} - (1-\alpha)p_j$. But our analysis requires that $\alpha \geq 1/2$, and it then turns out[2] that using the midpoint $M_j^{LP} = C_j^{LP}(1/2)$ leads to the best bound.

Acknowledgments

We are grateful to Maxim Sviridenko for identifying an error in the statement and proof of Theorem 4.1 in an earlier version of this paper, and for pointing out references [19] and [38]. We also thank the organizers of the workshop on "Parallel Scheduling" held at Schloß Dagstuhl, Germany, July 14–18, 1997, where this work was initiated. An extended abstract of this paper appeared as [28].

The research of the second author is supported in part by a research grant from NSERC (the Natural Sciences and Research Council of Canada), and by the UNI.TU.RIM. S.p.a. (Società per l'Università nel riminese), whose support is gratefully acknowledged.

References

[1] A. von Arnim, R. Schrader, and Y. Wang. The permutahedron of N-sparse posets. *Mathematical Programming*, 75:1 – 18, 1996.

[2] Indeed, inequality (i) in the proof of Theorem 4.1 may be replaced with $(1/m)\sum_{i=1}^{j-1} p_i \leq 2C_j^{LP}(\alpha)$, provided that $\alpha \geq 1/2$. On the other hand, inequality (ii) becomes $\lambda \leq (1-\alpha)^{-1} C_j^{LP}(\alpha)$.

[2] A. von Arnim and A. S. Schulz. Facets of the generalized permutahedron of a poset. *Discrete Applied Mathematics*, 72:179 – 192, 1997.

[3] E. Balas, J. K. Lenstra, and A. Vazacopoulos. The one machine problem with delayed precedence constraints and its use in job-shop scheduling. *Management Science*, 41:94 – 109, 1995.

[4] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16:201 – 240, 1988.

[5] D. Bernstein and I. Gertner. Scheduling expressions on a pipelined processor with a maximum delay of one cycle. *ACM Transactions on Programming Languages and Systems*, 11:57 – 66, 1989.

[6] P. Brucker and S. Knust. Complexity results for single–machine problems with positive finish–start time–lags. Osnabrücker Schriften zur Mathematik, Reihe P, No. 202, 1998.

[7] J. Bruno, J. W. Jones, and K. So. Deterministic scheduling with pipelined processors. *IEEE Transactions on Computers*, C-29:308 – 316, 1980.

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

[9] C. S. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. In *Proceedings of the 8th ACM–SIAM Symposium on Discrete Algorithms*, pages 609 – 618, 1997.

[10] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence–constrained scheduling problems on parallel machines that run at different speeds. In *Proceedings of the 8th ACM–SIAM Symposium on Discrete Algorithms*, pages 581 – 590, 1997.

[11] L. Finta and Z. Liu. Single machine scheduling subject to precedence delays. *Discrete Applied Mathematics*, 70:247 – 266, 1996.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness.* Freeman, San Francisco, 1979.

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

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

[15] M. X. Goemans and D. P. Williamson. Two-dimensional Gantt charts and a scheduling algorithm of Lawler. In *Proceedings of the 10th ACM–SIAM Symposium on Discrete Algorithms*, pages 366 – 375, 1999.

[16] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. J.*, 45:1563 – 1581, 1966.

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

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

[19] L. A. Hall and D. B. Shmoys. Jackson's rule for single-machine scheduling: Making a good heuristic better. *Mathematics of Operations Research*, 17:22 – 35, 1992.

[20] W. Herroelen and E. Demeulemeester. Recent advances in branch-and-bound procedures for resource-constrained project scheduling problems. In P. Chrétienne, E. G. Coffman Jr. , J. K. Lenstra, and Z. Liu, editors, *Scheduling Theory and its Applications*, chapter 12, pages 259 – 276. John Wiley & Sons, 1995.

[21] J. A. Hoogeveen, P. Schuurman, and G. J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. In R. Bixby, E. A. Boyd, and R. Z. Rios Mercado, editors, *Integer Programming and Combinatorial Optimization*, number 1412 in Lecture Notes in Computer Science, pages 353 – 366. Springer, Berlin, 1998. Proceedings of the 6th International IPCO Conference.

[22] J. M. Jaffe. Efficient scheduling of tasks without full use of processor resources. *Theoretical Computer Science*, 12:1 – 17, 1980.

[23] E. Lawler, J. K. Lenstra, C. Martel, B. Simons, and L. Stockmeyer. Pipeline scheduling: A survey. Technical Report RJ 5738 (57717), IBM Research Division, San Jose, California, 1987.

[24] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, editors, *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, chapter 9, pages 445 – 522. North–Holland, Amsterdam, The Netherlands, 1993.

[25] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research 26*, 26:22 – 35, 1978.

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

[27] J. Y.–T. Leung, 0. Vornberger, and J. Witthoff. On some variants of the bandwidth minimization problem. *SIAM J. Computing*, 13:650 – 667, 1984.

[28] A. Munier, M. Queyranne, and A. S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In R. Bixby, E. A. Boyd, and R. Z. Rios Mercado, editors, *Integer Programming and Combinatorial Optimization*, number 1412 in Lecture Notes in Computer Science, pages 367 – 382. Springer, Berlin, 1998. Proceedings of the 6th International IPCO Conference.

[29] K. W. Palem and B. Simons. Scheduling time critical instructions on risc machines. In *Proceedings of the 17th Annual Symposium on Principles of Programming Languages*, pages 270 – 280, 1990.

[30] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82:199 – 223, 1998. Conference version appeared previously as "Scheduling jobs that arrive over time" in Lecture Notes in Computer Science 955, Springer, Berlin, 86 – 97 (1995), Proceedings of the Fourth Workshop on Algorithms and Data Structures.

[31] M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263 – 285, 1993.

[32] M. Queyranne and Y. Wang. Single–machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research*, 16:1 – 20, 1991.

[33] A. S. Schulz. Polyedrische Charakterisierung von Scheduling Problemen. Diploma Thesis, Department of Mathematics, Technical University of Berlin, Berlin, Germany, 1993.

[34] A. S. Schulz. *Polytopes and Scheduling.* PhD thesis, Technical University of Berlin, Berlin, Germany, 1996.

[35] A. S. Schulz. Scheduling to minimize total weighted completion time: Performance guarantees of LP–based heuristics and lower bounds. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization*, number 1084 in Lecture Notes in Computer Science, pages 301 – 315. Springer, Berlin, 1996.

[36] A. S. Schulz and M. Skutella. Random–based scheduling: New approximations and LP lower bounds. In J. Rolim, editor, *Randomization and Approximation Techniques in Computer Science*, volume 1269 of *Lecture Notes in Computer Science*, pages 119 – 133. Springer, Berlin, 1997.

[37] A. S. Schulz and M. Skutella. Scheduling–LPs bear probabilities: Randomized approximations for min–sum criteria. In R. Burkard and G. Woeginger, editors, *Algorithms – ESA'97*, volume 1284 of *Lecture Notes in Computer Science*, pages 416 – 429. Springer, Berlin, 1997.

[38] P. Schuurman. A fully polynomial approximation scheme for a scheduling problem with intree-type precedence delays. *Operations Research Letters*, 23:9 – 11, 1998.

[39] E. D. Wikum, D. C. Llewellyn, and G. L. Nemhauser. One–machine generalized precedence constrained scheduling problems. *Operations Research Letters*, 16:87 – 99, 1994.

[40] L. A. Wolsey, August 1985. Invited Talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge.