Improved Bounds on Relaxations of a Parallel Machine Scheduling Problem*

Cynthia A. Phillips† Andreas S. Schulz‡ David B. Shmoys§ Cliff Stein¶ Joel Wein‖

Abstract

We consider the problem of scheduling n jobs with release dates on m identical parallel machines to minimize the average completion time of the jobs. We prove that the ratio of the average completion time of the optimal nonpreemptive schedule to that of the optimal preemptive schedule is at most $\frac{7}{3}$, improving a bound of $(3 - \frac{1}{m})$ due to Phillips, Stein and Wein. We then use our technique to give an improved bound on the quality of a linear programming relaxation of the problem considered by Hall, Schulz, Shmoys and Wein.

*A preliminary presentation of these results was given in the Proceedings of the 1996 International Colloquium on Automata, Languages and Programming [1].

† caphill@cs.sandia.gov. Sandia National Labs, Albuquerque, NM. This work was performed under U.S. Department of Energy contract number DE-AC04-76AL85000.

‡ schulz@math.tu-berlin.de. Department of Mathematics, Technical University of Berlin, 10623 Berlin, Germany. Research partially supported by the graduate school Algorithmische Diskrete Mathematik (DFG), grant We 1265/2-1.

§ shmoys@cs.cornell.edu. School of Operations Research & Industrial Engineering and Department of Computer Science, Cornell University, Ithaca, NY 14853. Research partially supported by NSF grants CCR-9307391 and DMS-9505155 and ONR grant N00014-96-1-0050O.

¶ cliff@cs.dartmouth.edu. Department of Computer Science, Sudikoff Laboratory, Dartmouth College, Hanover, NH. Research partially supported by NSF Award CCR-9308701, a Walter Burke Research Initiation Award and a Dartmouth College Research Initiation Award.

‖ wein@mem.poly.edu. Department of Computer Science, Polytechnic University, Brooklyn, NY, 11201. Research partially supported by NSF Research Initiation Award CCR-9211494, NSF Grant CCR-9626831, and a grant from the New York State Science and Technology Foundation, through its Center for Advanced Technology in Telecommunications.

1 Introduction

Recently there has been much activity in the development of approximation algorithms for a number of scheduling problems in which the goal is to minimize the average completion time of the jobs scheduled [2, 3, 6, 8, 9, 17, 18, 22, 23]. The key idea behind most of these results is that, given certain sorts of relaxations of scheduling problems, a valid schedule can be inferred from a simple ordering that can be easily constructed from these relaxations.

In this note we improve the analytical methods available for the analysis of the schedules constructed from these relaxations. Specifically, we consider the problem of scheduling n jobs $\mathcal{J} = \{1, \ldots, n\}$ on m identical parallel machines. Each job j has a *release date* r_j before which it is not available for processing. We will denote the completion time of job j in a schedule S as C_j^S, and the total completion time of schedule S as $C^S = \sum_{j \in \mathcal{J}} C_j^S$. We will often drop the S when the schedule in question is clear. The goal is to construct a schedule S that minimizes C^S, which is an objective function equivalent to the average completion time $\frac{1}{n}C^S$. We require a *nonpreemptive schedule* in which each job must be processed in an uninterrupted fashion. This problem is often denoted by $P|r_j|\sum C_j$ [12], and is $\mathcal{NP}$-hard [13]; thus, it is natural to be interested in approximately-optimal solutions that can be computed in polynomial time. We define a *ρ-approximation algorithm* as an algorithm that runs in polynomial time and always produces a solution of value within a factor of ρ of the optimum.

Until recently little was known about approximation algorithms for this problem, or in general for scheduling problems with a minsum optimality criterion subject to release date constraints (or other types of constraints, such as precedence constraints). The first progress in developing approximation algorithms with constant-factor performance guarantees for these problems was made by Phillips, Stein and Wein [18], who developed approximation algorithms by using a *preemptive* schedule as a relaxation of the nonpreemptive schedule. Specifically, they showed that, in the special case of $m = 1$, a nonpreemptive schedule of average completion time within a factor of 2 of optimal can be constructed simply by ordering the jobs nonpreemptively by their completion times in an optimal preemptive schedule. Since the preemptive version of the one-machine problem can be solved in polynomial time, this leads immediately to a 2-approximation algorithm. For general m they showed that list scheduling in order of the completion times in a preemptive schedule yields a nonpreemptive schedule of average completion time at most a factor of $(3 - \frac{1}{m})$ larger than that of the preemptive. Unfortunately, in this case the underlying preemptive problem is $\mathcal{NP}$-hard. By also providing an approximation algorithm for the preemptive problem, Phillips, Stein, and Wein were able to apply this conversion technique to yield a 6-approximation algorithm for $P|r_j|\sum C_j$.

The idea of scheduling in a natural order dictated by a relaxation of the problem has proved to be quite powerful. Inspired by this idea, Hall, Schulz, Shmoys, and Wein in [8], which is a joint journal version of [9] and [23], studied a number of linear programming relaxations of constrained minsum scheduling problems and gave methods to round their solutions to feasible schedules. Their techniques yield improved performance guarantees for many problems, and in some cases, the first constant performance guarantees. In particular, they give a $(4 - \frac{1}{m})$-approximation algorithm for $P|r_j|\sum C_j$; this result was discovered independently by Queyranne [20]. This result makes use of a linear programming relaxation of the problem in completion-time variables and as a result establishes a bound on the quality of the relaxation; Hall et al. showed that, for any instance, the fractional solution to the linear program is no more than a factor of $(4 - \frac{1}{m})$ from optimal.

Our results improve upon both the techniques of Phillips, Stein and Wein [18] and of Hall et al. [8]. Specifically,

- we show that list scheduling in order of the completion times in a preemptive identical-parallel-machine schedule yields a nonpreemptive schedule of average completion time at most a factor of $\frac{7}{3}$ larger than that of the preemptive schedule, improving the previous bound of $(3 - \frac{1}{m})$ [18];

- we improve the analysis of Hall et al. to show that the linear programming relaxation that they consider yields a lower bound within a factor of 3.75 of optimal, improving on the bound of $(4 - \frac{1}{m})$ [8].

Neither of these results gives the best current performance guarantee for an approximation algorithm for $P|r_j|\sum C_j$. The current best approximation algorithm for the underlying preemptive problem is a 2-approximation algorithm [18]; thus our first technique yields only a 4.666-approximation algorithm. The second technique does improve over the best previously known approximation algorithm, but simultaneously with our discovery of this result we also discovered a $(2.89 + \epsilon)$-approximation algorithm for the problem using rather different techniques [1]. We note, however, that our 3.75-approximation algorithm can be implemented in $O(n \log n)$ time (by applying results of Queyranne and Schulz [21] and Goemans [5]) whereas the $(2.89 + \epsilon)$-approximation algorithm is rather computationally intensive. Subsequent to the results in this paper, Chekuri et al. [2] gave a 2.85-approximation algorithm that runs in $O(n \log n)$ time, and quite recently, Schulz and Skutella have given a randomized $O(n \log n)$ 2-approximation algorithm [22]. (A consequence of this result is also an improved bound of 2 on the ratio of the optimal nonpreemptive to preemptive schedule.)

Our results, however, are important for three reasons. First, the idea of scheduling in a natural order based on a linear programming relaxation in completion-time variables, or related formulations, has found many applications to a number of scheduling problems, e.g., [1, 2, 3, 6, 8, 17, 22, 24]. This is currently a very active area of research, and our techniques offer new ideas that we believe may find other applications and lead to further improvements. The analyses of Phillips, Stein, and Wein [18] and Hall et al. [8] show, respectively, that the completion time of job j in the schedule found is within a constant factor of the preemptive completion time of j and the LP completion time of j. We give a technique that strengthens this "job-by-job" approach, by showing how to make use of the total completion time in our analysis. We hope that this will lead to further progress in obtaining stronger cumulative bounds in other settings.

Second, the linear programming formulations considered in these results have received a great deal of attention both from the perspective of their strength when used to give lower bounds in computing optimal solutions to small problem instances, and from the perspective of their use in giving polyhedral characterizations of various scheduling problems [21]. Until the recent work to which we have referred, these polyhedral formulations have had no worst-case analysis; therefore our results in Section 3.2 create more precise connections between the worst-case analysis of scheduling problems and the study of polyhedral formulations of these problems.

Finally, our results are of interest from a perspective different from that of approximation algorithms and polyhedral methods. Researchers have long been interested in characterizing the power of preemption in scheduling, with the goal of optimizing both $\sum C_j$ [14, 16, 18]

Algorithm LIST

Input: An ordered list L of jobs; without loss of generality, let this ordering be $1, \ldots, n$.

Output: A nonpreemptive schedule N.

List schedule nonpreemptively using L, in the following manner:

For each job j on the list, schedule j as early as possible subject to the following constraints:

- Job j does not start before r_j.
- Jobs $1, \ldots, j-1$ have already been scheduled in N, and their position is fixed.

Figure 1: The algorithm LIST

and the makespan ($\max_j C_j$) [4, 7, 10, 15] of a schedule. For both of these optimality criteria, the optimal preemptive schedule for an instance will often be significantly better than that of the optimal nonpreemptive schedule; however, in the real world preemption is viewed as being difficult to implement and as incurring a cost, and a nonpreemptive schedule, if of reasonable quality, is more desirable.

McNaughton [16] proved that no finite number of preemptions can improve the average completion time of a set of jobs on parallel machines; combined with a result from open-shop scheduling theory [12], this establishes that that for the scheduling of jobs *without release dates* on identical parallel machines to minimize average completion time, the optimal preemptive schedule is no better than the optimal nonpreemptive schedule. When release dates are introduced the situation changes significantly; our result in Section 3.1 gives an improved understanding of the power of preemption in this scheduling environment. Specifically, by disallowing preemption, the average completion time can increase by no more than a factor of $\frac{7}{3}$. In contrast, T.C. Lai [11] has proven the best known lower bound: there exists an instance for which the optimal value for a nonpreemptive schedule is at least $\frac{18}{13}$ times the optimal value for a preemptive schedule.

Both of our results arise from the application of a list-scheduling algorithm to a list determined by the solution to a relaxation of the scheduling problem. In Section 2 we present this list-scheduling algorithm LIST and establish several basic bounds on the performance of LIST on any ordered list of jobs. In Section 3 we apply this technique to prove our two results by specializing the analysis to ordered lists derived from a preemptive schedule and from a solution to a linear programming relaxation of the problem.

2 The Algorithm LIST

We consider the algorithm LIST (see also Figure 1), which accepts as input an ordered list of jobs and uses that list to produce a nonpreemptive schedule N for the jobs on m identical parallel machines that respects the release dates of the jobs. LIST is essentially a list-scheduling algorithm; for each job in turn, it schedules it as early as possible without violating its release date, and without disturbing the jobs that have already been scheduled.

In order to analyze LIST we will also apply an algorithm POST to N that yields a schedule N'. It will always be the case that $C^{N'} \geq C^N$, but in certain cases it will be easier to analyze N' in order to bound C^N.

The algorithm POST takes as input the nonpreemptive schedule N that is produced by LIST, and the set $\mathcal{L}$ of the last m jobs in the list. These jobs, however, need not be the set of jobs that are the last to be processed on some machine in N. POST creates a schedule N'

Algorithm Post

Input: A nonpreemptive schedule N derived from application of List to a list L.
Output: A nonpreemptive schedule N'.

1. Let $\mathcal{L}$ be the last m jobs in the list L.
2. Let t_i be the earliest time such that all work in schedule N on machine M_i after time t_i is given to jobs from $\mathcal{L}$, and let j_i be the job that starts at time t_i. If no job from $\mathcal{L}$ is scheduled last on machine M_i in schedule N, then $t_i = \infty$ and j_i does not exist. Let $\mathcal{H}$ be the set of jobs $j \in \mathcal{L}$ which are not j_i for some machine M_i. Let $\mathcal{M}'$ be the set of machines whose last job in schedule N is not in $\mathcal{L}$ ($t_i = \infty$).
3. Remove the jobs in $\mathcal{H}$ from their present machines, and instead assign them to run last on machines in $\mathcal{M}'$. The assignment can be any 1−to−1 assignment.

Figure 2: The algorithm Post

in which the jobs in $\mathcal{L}$ also are the jobs that are the last to be processed on some machine in N'. Consider each machine in N. On some machines, the last job to be processed is not in $\mathcal{L}$; on these machines Post removes from the schedule all jobs in $\mathcal{L}$ and places them in a set $\mathcal{H}$. On the remaining machines the last job to be processed is in $\mathcal{L}$; in fact, the last several to be processed might be in $\mathcal{L}$. On these machines, Post removes from the schedule every job in $\mathcal{L}$ except for the first in the "concluding block" of jobs in $\mathcal{L}$ on that machine and adds them to $\mathcal{H}$. Post then assigns the jobs in $\mathcal{H}$ in a 1−to−1 fashion to all machines on which the last job to be processed is not from $\mathcal{L}$. These newly-assigned jobs are scheduled last on the machine to which they are assigned. Figure 2 gives a more formalized version of Post. We note again that the schedule N' is constructed by Post solely for the sake of our analysis; $C^{N'}$ is never smaller than C^N, but at times N' will be easier to analyze.

We now establish several general bounds on the completion times of jobs in N and the average completion time of the schedules N and N'; in the next section we will specialize the bounds to each of our applications.

Lemma 2.1 *Let L be an ordered list of n jobs and without loss of generality let this ordering be $1, \ldots, n$. Let $r'(j) = \max_{i=1,\ldots,j} r_i$. Let N be the nonpreemptive schedule resulting from the application of* List *to L. Then*

$$C_j^N \leq r'(j) + (\sum_{i=1}^{j-1} p_i)/m + p_j \quad .$$

Proof. The proof is essentially identical to arguments given by both Phillips, Stein and Wein [18] and Hall et al. [8]. Consider the schedule of jobs $1, 2, \ldots, j-1$, and suppose that job j is then scheduled to start at time t. Trace back in this (partial) schedule to find the latest time t' that a machine is idle prior to time t. Since all machines are busy throughout the interval $(t', t]$, we see that $\sum_{i=1}^{j-1} p_i \geq m(t - t')$, or equivalently, $t \leq t' + (\sum_{i=1}^{j-1} p_i)/m$. Consider a machine that is idle immediately prior to time t', and let j' be the job that starts processing on this machine at t'. Since the algorithm List does not start j' earlier on this machine, we can conclude that $t' = r_{j'} \leq r'(j)$. Since job j completes at time $t + p_j$, we see that $C_j^N \leq r'(j) + (\sum_{i=1}^{j-1} p_i)/m + p_j$. ■

If we let $F(j) = r'(j) + (\sum_{i=1}^{j-1} p_i)/m + p_j$, then we can succinctly state this upper bound as $C_j^N \leq F(j)$, for each $j = 1, \ldots, n$. We next show that the schedule N' is dominated by the schedule N.

Lemma 2.2 *For each job $j = 1, \ldots, n$, $C_j^N \leq C_j^{N'}$.*

PROOF. Suppose that the lemma were false, and that there exists a job j for which $C_j^N > C_j^{N'}$. Assume that j starts on machine M' at some time $t + t_1$ in N and on machine M'' at time t in N' (where $t_1 > 0$). Since POST rescheduled job j to be the last job processed by M'' in N', it follows that in N, no job runs on M'' after t. Therefore, in the construction of N by LIST, at the point when job j is to be scheduled, machine M'' was free at time t; hence job j would have been scheduled at this earlier time, which is a contradiction. ■

We now divide $\mathcal{L}$, the last m jobs in the list L, into two sets $\mathcal{L}_1$ and $\mathcal{L}_2$. The set $\mathcal{L}_2$ contains those jobs $j \in \mathcal{L}$ for which $C_j^{N'} = r_j + p_j$ and the set $\mathcal{L}_1 = \mathcal{L} - \mathcal{L}_2$. Let $\mathcal{K} = \mathcal{J} - \mathcal{L}$. Lemma 2.2 has the following immediate corollary.

Lemma 2.3 *For each job $j \in \mathcal{L}_2$, we have that $C_j^N = r_j + p_j$.*

PROOF. Since N is a feasible schedule, we have that $r_j + p_j \leq C_j^N$ for each job $j = 1, \ldots, n$. However, by Lemma 2.2, $C_j^N \leq C_j^{N'} = r_j + p_j$ for each job $j \in \mathcal{L}_2$, and so we have proved the claim. ■

Lemma 2.4 *Let C^N be the total completion time of the schedule constructed by Algorithm* LIST. *Then,*

$$C^N \leq \sum_{j \in \mathcal{K} \cup \mathcal{L}_1} F(j) + \sum_{j \in \mathcal{L}_2} (r_j + p_j) \ .$$

PROOF. We apply the bound $C_j^N \leq F(j)$ to the jobs in $\mathcal{L}_1 \cup \mathcal{K}$, apply the bound $C_j^N = r_j + p_j$ to the jobs in $\mathcal{L}_2$, and sum over all jobs. ■

Lemma 2.5 *Let $C^{N'}$ be the total completion time of the schedule constructed by Algorithm* POST. *Then,*

$$C^{N'} \leq \sum_{j \in \mathcal{K}} F(j) + \sum_{j \in \mathcal{L}_2} (r_j + p_j) + \sum_{k \in \mathcal{K}} r_k + \sum_{j \in \mathcal{L}_1 \cup \mathcal{K}} p_j \ .$$

PROOF. Since the algorithm POST only changes the times at which jobs in $\mathcal{L}$ are processed, we have that for each job $j \in \mathcal{K}$, $C_j^{N'} = C_j^N \leq F(j)$. For each job $j \in \mathcal{L}_2$, we will apply the bound $C_j^{N'} = r_j + p_j$, which follows directly from the definition of $\mathcal{L}_2$.

We now bound the completion times of jobs in $\mathcal{L}_1$, which are each on a different machine in the schedule N'. Consider a particular job $j \in \mathcal{L}_1$ on some machine M. Let t be the last time that machine M was idle before running j. Let k be the job that starts processing at time t on machine M. We will show that $t = r_k$.

Assume, for a contradiction, that job k is not scheduled to start at its release date. If either of the algorithms LIST or POST schedules a job to start later than its release date, it must schedule that job to immediately follow the processing of another job on the same machine without any intervening idle time. Hence, when the algorithm schedules job k, there must be another job k' that is processed up until time t on machine M. This implies that job k was not scheduled in this position by POST. Consequently, job k is scheduled by the algorithm LIST, and then POST has rescheduled job k'. This implies that $k' \in \mathcal{L}$. Furthermore, for each machine, POST reschedules all jobs in $\mathcal{L}$ except the first of the final block of jobs in $\mathcal{L}$. This implies that $k \notin \mathcal{L}$: since k' is rescheduled by POST, then k is not the first of the final block of jobs in $\mathcal{L}$; hence if k were in $\mathcal{L}$, then POST would have also rescheduled k. But it is also impossible for k to be in $\mathcal{K}$, since then k precedes k' in the list L: when LIST schedules k, the job k' is not yet scheduled, and so job k would be scheduled to start earlier than time t. Hence, $t = r_k$.

Since job k begins processing at time $t = r_k$ and $j \notin \mathcal{L}_2$, we know that $k \neq j$, and hence $k \in \mathcal{K}$. Let $j_1, \ldots, j_\ell$ be the jobs that run on M between k and j in the schedule N'. Then

$$C_j^{N'} = r_k + p_k + p_{j_1} + \ldots + p_{j_\ell} + p_j \quad . \tag{1}$$

If we sum (1) over all jobs in $\mathcal{L}_1$, then each job in $\mathcal{K} \cup \mathcal{L}_1$ contributes to the right-hand side at most once, since each job in $\mathcal{L}_1$ is on a different machine. The jobs in $\mathcal{L}_2$ do not contribute at all, since they are run on machines distinct from the ones which run jobs in $\mathcal{L}_1$ in N'. Thus, summing over all jobs $j \in \mathcal{L}_1$, we obtain $\sum_{j\in\mathcal{L}_1} C_j^{N'} \leq \sum_{k\in\mathcal{K}} r_k + \sum_{j\in\mathcal{L}_1\cup\mathcal{K}} p_j$. Combining this with the above-mentioned bounds for the other jobs, we obtain our lemma. ■

3 Applications

3.1 Preemptive and Nonpreemptive Schedules

In this section we consider preemptive and nonpreemptive schedules for minimizing average completion time of jobs with release dates on identical parallel machines. We will show that the ratio between the optimal nonpreemptive and preemptive average completion times is at most $\frac{7}{3}$ by analyzing the application of LIST to the list that orders the jobs by their completion times in the preemptive schedule. We will show that this yields a nonpreemptive schedule with average completion time at most $\frac{7}{3}$ times greater. This improves on the bound of $(3 - \frac{1}{m})$ given by Phillips, Stein and Wein [18].

For any preemptive schedule P, let $\mathcal{L}(P)$ denote the set of m jobs with the largest completion times (where ties are broken, for example, by job index). In general, it need not be the case that the set of jobs that are the last to be processed on each machine is equal to $\mathcal{L}(P)$. However, the following lemma shows that this property can be assumed without loss of generality.

Lemma 3.1 *For each preemptive schedule P, there exists a preemptive schedule P' such that $C_j^P = C_j^{P'}$ for each $j = 1, \ldots, n$, with the property that $\mathcal{L}(P')$ is equal to the set of jobs that are the last to be processed on each machine in P'.*

PROOF. Consider some job $j \in \mathcal{L}(P)$, which completes on machine M, but is not the last job to be processed on M. Assume that the final piece of job j runs from time t to C_j^P.

By a pigeonhole argument, there must be some job $j' \notin \mathcal{L}(P)$ which is the last to complete on some machine M'. Observe that $C_{j'}^P \leq C_j^P$, and that M' is idle after $C_{j'}^P$.

We make a simple interchange: we schedule job j on M' from time t to C_j^P, and swap whatever is scheduled on M' in P during this interval to be scheduled on M instead. Job j now completes on M', and is the last job to be processed on M'. Furthermore, all job completion times are unaffected by this interchange.

This interchange reduces the number of jobs in $\mathcal{L}$ that are not the last to be processed on some machine, and so after at most m iterations of this procedure, we will obtain the desired schedule. ■

Thus, when we refer to $\mathcal{L} = \mathcal{L}(P)$ for some preemptive schedule, we will be make use of the fact that it is equivalent to think of $\mathcal{L}$ as the set of jobs that finish last on some machine. We define $C^{L_1} = \sum_{j\in\mathcal{L}_1} C_j^P$, $C^K = \sum_{j\in\mathcal{K}} C_j^P$, and $C^{L_2} = \sum_{j\in\mathcal{L}_2} C_j^P$. For our improved analysis it will be necessary to analyze $\sum_{j\in\mathcal{J}} p_j$ in terms of the preemptive schedule, so we begin by establishing a useful bound on this sum.

Lemma 3.2 *For any preemptive schedule P,*

$$\sum_{j\in\mathcal{J}} p_j \le C^{L_1} + C^{L_2} \ .$$

Proof. Let P' be the schedule obtained from P by Lemma 3.1; we shall prove the lemma by considering P' instead. For each machine M, there exists a distinct job in $j \in \mathcal{L}$ which is the last job processed on M in P', and hence this machine runs from time 0 until C_j^P. Thus, we can view the total processing capacity of this schedule as $\sum_{j\in\mathcal{L}} C_j^P = C^{L_1} + C^{L_2}$. Since this capacity is sufficient for all of the jobs in $\mathcal{J}$, the lemma follows. ■

We are now ready to establish the 7/3 bound; we do this by establishing two new upper bounds, one on C^N and one on $C^{N'}$. We will order the jobs by their completion times in the schedule P and call the resulting list L_P.

Lemma 3.3 *The algorithm* List, *when applied to the list L_P, produces a nonpreemptive schedule N in which $C^N \le 2C^K + 3C^{L_1} + 2C^{L_2}$.*

Proof. We will make use of the bound of Lemma 2.4:

$$C^N \le \sum_{j\in\mathcal{K}\cup\mathcal{L}_1} F(j) + \sum_{j\in\mathcal{L}_2} (r_j + p_j) \ .$$

We first relate $F(j)$ to the completion times of the jobs in P. Recall that

$$F(j) = r'(j) + (\sum_{i=1}^{j-1} p_i)/m + p_j \ .$$

We can bound $r'(j) \le C_j^P$, since jobs $1, \ldots, j$ all complete in P by time C_j^P, and thus must be released by then. We may also bound $(\sum_{i=1}^{j-1} p_i)/m \le C_j^P$ since jobs $1, \ldots, j$ all complete in P by time C_j^P; thus $F(j) \le 2C_j^P + p_j$, and we obtain

$$\begin{aligned} C^N &\le \sum_{j\in\mathcal{K}\cup\mathcal{L}_1} (2C_j^P + p_j) + \sum_{j\in\mathcal{L}_2} (r_j + p_j) \\ &\le 2(C^K + C^{L_1}) + \sum_{j\in\mathcal{J}} p_j + \sum_{j\in\mathcal{L}_2} r_j \\ &\le 2(C^K + C^{L_1}) + \sum_{j\in\mathcal{J}} p_j + C^{L_2} \ . \qquad (2) \end{aligned}$$

We now apply Lemma 3.2 to bound $\sum_{j \in \mathcal{J}} p_j$, and substituting into (2) we obtain the lemma. ■

Roughly, when C^{L_1} is small, this bound is good. However, when C^{L_1} is large we need to apply Post and use the resulting schedule to bound C^N.

Lemma 3.4 *The algorithm* List, *when applied to* L_P *and followed by* Post, *produces a nonpreemptive schedule* N' *in which* $C^{N'} \leq 3C^K + C^{L_1} + 2C^{L_2}$.

Proof. We make use of the bound of Lemma 2.5:

$$C^{N'} \leq \sum_{j \in \mathcal{K}} F(j) + \sum_{j \in \mathcal{L}_2} (r_j + p_j) + \sum_{k \in \mathcal{K}} r_k + \sum_{j \in \mathcal{L}_1 \cup \mathcal{K}} p_j \ ,$$

and substitute $F(j) \leq 2C_j^P + p_j$. Then,

$$\begin{aligned} C^{N'} &\leq \sum_{j \in \mathcal{K}} (2C_j^P + p_j) + \sum_{j \in \mathcal{L}_2} (r_j + p_j) + \sum_{k \in \mathcal{K}} r_k + \sum_{j \in \mathcal{L}_1 \cup \mathcal{K}} p_j \\ &\leq 2C^K + \sum_{k \in \mathcal{K}} p_k + C^{L_2} + \sum_{k \in \mathcal{K}} r_k + \sum_{j \in \mathcal{L}_1 \cup \mathcal{K}} p_j \\ &\leq 3C^K + C^{L_2} + \sum_{j \in \mathcal{L}_1 \cup \mathcal{K}} p_j \\ &\leq 3C^K + C^{L_2} + (C^{L_1} + C^{L_2}) \ , \end{aligned}$$

where the last inequality follows from Lemma 3.2. ■

We can now combine the previous two lemmas to prove that the average completion time of the nonpreemptive schedule constructed by algorithm List is at most $\frac{7}{3}$ times the average completion time of the preemptive schedule that served as the input. Recall that $C^P = C^K + C^{L_1} + C^{L_2}$. If $C^{L_1} \leq \frac{1}{3}C^P$, then the bound from Lemma 3.3 becomes $C^N \leq 2(C^K + C^{L_1} + C^{L_2}) + C^{L_1} \leq 2C^P + \frac{1}{3}C^P = \frac{7}{3}C^P$. If $C^{L_1} > \frac{1}{3}C^P$ then the bound from Lemma 3.4 becomes $C^{N'} \leq 3C^K + C^{L_1} + 2C^{L_2} = 2C^P + C^K - C^{L_1}$. But $C^{L_1} > \frac{1}{3}C^P$ and thus $C^K \leq \frac{2}{3}C^P$, so this simplifies to $C^{N'} \leq \frac{7}{3}C^P$. By Lemma 2.2, we have that $C^N \leq C^{N'}$, and so we see that in both cases, $C^N \leq \frac{7}{3}C^P$.

Theorem 3.5 *Given a preemptive schedule* P *for an instance of scheduling jobs with release dates on identical parallel machines, algorithm* List, *when applied to* L_P, *produces a nonpreemptive schedule for that instance whose total completion time is at most* $\frac{7}{3}C^P$.

3.2 Bounding a Linear Programming Relaxation

The $(4 - \frac{1}{m})$-approximation algorithm of Hall et al. [8] is based on the LP relaxation of a formulation in which there are *completion-time variables:* with each job j we associate a variable C_j. We show that by letting $\overline{C}_j$, $j = 1, \ldots, n$, a feasible solution to the LP relaxation, play the role of the preemptive completion times in the previous section, we obtain an improved approximation algorithm and an equivalent bound on the quality of the lower bounds delivered by that linear programming formulation.

The linear programming formulation that we consider is the following (lpc):

$$\begin{array}{lrcl} \text{minimize} & \sum_{j \in J} C_j & & \\ \text{subject to} & \sum_{j \in \mathcal{A}} p_j C_j & \geq & \frac{1}{2m}\Big(\sum_{j \in \mathcal{A}} p_j\Big)^2 + \frac{1}{2m}\sum_{j \in \mathcal{A}} p_j^2 \ \text{ for each } \mathcal{A} \subseteq J \\ & C_j & \geq & r_j + p_j \ \text{ for each } j \in J. \end{array}$$

The second class of constraints are quite simple, merely stating that the completion time of job j in a valid schedule can be no earlier than the sum of its release date and processing time. The first set of constraints is less intuitive, but can be readily derived by summing over load-based constraints on job completion times [8]. We note that the first set of constraints is exponential in size, but as a consequence of the results of Queyranne [19] we can separate over these constraints in polynomial time, and thus an optimal solution to this linear programming relaxation can be obtained in polynomial time. More surprisingly, as a consequence of the results of Queyranne and Schulz [21] and Goemans [5, 6], an optimal solution to this linear program can actually be obtained in $O(n \log n)$ time. For further discussion of this formulation, see [8] or [21].

Let $\overline{C}_j, j = 1, \ldots, n$, be an optimal solution to the linear program LPC, and renumber the jobs so that $\overline{C}_1 \leq \cdots \leq \overline{C}_n$. In this setting, we apply LIST to the list $1, \ldots, n$, or in other words, to the list of jobs ordered by their $\overline{C}_j$ values in the optimal LP solution. We call this list L_{LP}.

In this setting, $\mathcal{L}$ is the set of jobs with the m largest values $\overline{C}_j$; analogously, $\mathcal{K} = \mathcal{J} - \mathcal{L}$, $\mathcal{L}_2$ is the set of jobs $j \in \mathcal{L}$ for which $C_j^{N'} = r_j + p_j$ and $\mathcal{L}_1 = \mathcal{L} - \mathcal{L}_2$. We let $C^{L_1} = \sum_{j \in \mathcal{L}_1} \overline{C}_j$, $C^K = \sum_{j \in \mathcal{K}} \overline{C}_j$, and $C^{L_2} = \sum_{j \in \mathcal{L}_2} \overline{C}_j$.

The following lemma was the central step in the earlier analysis that led to a 4-approximation algorithm, and is equally important for our improved analysis; for completeness, we include its proof.

Lemma 3.6 *For any subset $A \subseteq \mathcal{J}$, $\frac{1}{2m} \sum_{j \in \mathcal{A}} p_j \leq \max_{j \in \mathcal{A}} \overline{C}_j$.*

PROOF. Since $\overline{C}_j$, $j = 1, \ldots, n$, is a feasible solution to LPC, we have that

$$\sum_{j \in \mathcal{A}} p_j \overline{C}_j \geq \frac{1}{2m}\Big(\sum_{j \in \mathcal{A}} p_j\Big)^2.$$

However,

$$(\max_{j \in \mathcal{A}} \overline{C}_j)\Big(\sum_{j \in \mathcal{A}} p_j\Big) \geq \sum_{j \in \mathcal{A}} p_j \overline{C}_j \ ,$$

and so

$$(\max_{j \in \mathcal{A}} \overline{C}_j)\Big(\sum_{j \in \mathcal{A}} p_j\Big) \geq \frac{1}{2m}\Big(\sum_{j \in \mathcal{A}} p_j\Big)^2 \ ,$$

which is equivalent to the inequality claimed in the lemma. ∎

We use this lemma to derive an upper bound on $\sum_{j \in \mathcal{K} \cup \mathcal{L}_1} p_j$.

Lemma 3.7
$$\sum_{j\in\mathcal{K}\cup\mathcal{L}_1} p_j \le 3C^{L_1} + 2C^{L_2} \quad .$$

PROOF. We apply Lemma 3.6 with $A = \mathcal{K}$ to obtain $\frac{1}{2m}\sum_{j\in\mathcal{K}} p_j \le \max_{j\in\mathcal{K}} \overline{C}_j$. Since for each $j \in \mathcal{L}$, $\max_{k\in\mathcal{K}} \overline{C}_k \le \overline{C}_j$, we also have that $\frac{1}{2m}\sum_{k\in\mathcal{K}} p_k \le \overline{C}_j$. Summing over each $j \in \mathcal{L}$, we see that
$$\sum_{j\in\mathcal{L}} \frac{1}{2m} \sum_{k\in\mathcal{K}} p_k = |\mathcal{L}| \frac{1}{2m} \sum_{k\in\mathcal{K}} p_k = \frac{1}{2} \sum_{k\in\mathcal{K}} p_k \le \sum_{j\in\mathcal{L}} \overline{C}_j \quad .$$
This enables us to bound
$$\sum_{j\in\mathcal{K}\cup\mathcal{L}_1} p_j = \sum_{k\in\mathcal{K}} p_k + \sum_{j\in\mathcal{L}_1} p_j \le 2C^{L_1} + 2C^{L_2} + C^{L_1} \quad .$$

■

Lemma 3.8 *The algorithm* LIST, *when applied to list L_{LP}, produces a nonpreemptive schedule N in which*
$$C^N \le 3C^K + 6C^{L_1} + 3C^{L_2} \quad .$$

PROOF. We again make use of the bound of Lemma 2.4:
$$C^N \le \sum_{j\in\mathcal{K}\cup\mathcal{L}_1} F(j) + \sum_{j\in\mathcal{L}_2} (r_j + p_j) \quad .$$
We wish to relate $F(j)$ to the optimal solution to the linear program $\overline{C}_j, j = 1, \ldots, n$. We can bound $\overline{C}_j \ge r'(j)$, since $\overline{C}_j \ge \overline{C}_i$ for $i = 1, \ldots, j$, and we know that $\overline{C}_i \ge r_i + p_i$ for each $i = 1, \ldots, n$ because of the constraints of the linear program. We are not guaranteed, however, that $\overline{C}_j \ge (\sum_{i=1}^{j-1} p_i)/m$, since the $\overline{C}_j$ do not describe a valid schedule but rather a feasible solution to this linear programming relaxation of the scheduling problem. By setting $A = \{1, \ldots, j\}$, we can apply Lemma 3.6 to obtain the following bound:
$$\frac{1}{2m} \sum_{i=1}^{j-1} p_i \le \overline{C}_j \quad .$$
As a result, we see that
$$F(j) = r'(j) + (\sum_{i=1}^{j-1} p_i)/m + p_j \le 3\overline{C}_j + p_j \quad .$$
We therefore obtain
$$\begin{aligned} C^N &\le \sum_{j\in\mathcal{K}\cup\mathcal{L}_1} (3\overline{C}_j + p_j) + \sum_{j\in\mathcal{L}_2} (r_j + p_j) \\ &\le 3(C^K + C^{L_1}) + \sum_{j\in\mathcal{K}\cup\mathcal{L}_1} p_j + \sum_{j\in\mathcal{L}_2} (p_j + r_j) \\ &\le 3(C^K + C^{L_1}) + \sum_{j\in\mathcal{K}\cup\mathcal{L}_1} p_j + C^{L_2} \quad . \end{aligned}$$

Now, we substitute the bound of Lemma 3.7, which yields

$$C^N \leq 3C^K + 6C^{L_1} + 3C^{L_2} \quad .$$

$\blacksquare$

Lemma 3.9 *The algorithm* LIST, *when applied to L_{LP} and followed by* POST, *produces a nonpreemptive schedule N' in which*

$$C^{N'} \leq 4C^K + 3C^{L_1} + 3C^{L_2} \quad .$$

PROOF. We make use of the bound of Lemma 2.5:

$$C^{N'} \leq \sum_{j \in \mathcal{K}} F(j) + \sum_{j \in \mathcal{L}_2} (r_j + p_j) + \sum_{k \in \mathcal{K}} r_k + \sum_{j \in \mathcal{L}_1 \cup \mathcal{K}} p_j,$$

and substitute $F(j) \leq 3\overline{C}_j + p_j$. Using the bounds $\sum_{j \in \mathcal{K}}(r_j + p_j) \leq C^K$ and $\sum_{j \in \mathcal{L}_2}(r_j + p_j) \leq C^{L_2}$, we obtain

$$C^{N'} \leq 3C^K + C^K + C^{L_2} + \sum_{j \in \mathcal{L}_1 \cup \mathcal{K}} p_j \quad .$$

We can now apply Lemma 3.7 to obtain a bound of $4C^K + 3C^{L_1} + 3C^{L_2}$. $\blacksquare$.

Again, by balancing the two cases we can prove the desired bound on the ratio of the total completion time C^N of the nonpreemptive schedule N to the LP value $\overline{C} = \sum_j \overline{C}_j$. If $C^K \leq \frac{3}{4}\overline{C}$, Lemma 3.9 implies $C^N \leq 3\overline{C} + C^K \leq \frac{15}{4}\overline{C}$. On the other hand, if $C^K > \frac{3}{4}\overline{C}$, then $C^{L_1} \leq \frac{1}{4}\overline{C}$. Consequently, Lemma 3.8 implies that $C^N \leq 3\overline{C} + 3C^{L_1} \leq \frac{15}{4}\overline{C}$. The following theorem summarizes our result.

Theorem 3.10 *There is a* 3.75*-approximation algorithm to minimize the average completion time of jobs with release dates on identical parallel machines.*

Note that we have created a nonpreemptive schedule of average completion time at most 3.75 times $\sum_{j \in \mathcal{J}} \overline{C}_j$, the solution to linear program LPC. This implies the following corollary on the quality of the solution delivered by this relaxation.

Corollary 3.11 *For any instance of $P|r_j|\sum C_j$, its optimal value is within a factor of* 3.75 *of the optimal value to its linear programming relaxation* LPC.

Acknowledgments We thank Leslie Hall and Soumen Chakrabarti for helpful discussions.

References

[1] 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. Springer, Berlin, 1996. Proceedings of the 23rd International Colloquium (ICALP'96).

[2] C. 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, January 1997.

[3] 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*, 1997. To appear.

[4] E.G. Coffman Jr. and M.R. Garey. Proof of the 4/3 conjecture for preemptive vs. nonpreemptive two-processor scheduling. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 241–248, 1991.

[5] M. Goemans. A supermodular relaxation for scheduling with release dates. In *Proceedings of the 5th Conference on Integer Programming and Combinatorial Optimization*, pages 288–300, June 1996. Published as Lecture Notes in Computer Science 1084, Springer-Verlag.

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

[7] D.K. Goyal. Nonpreemptive scheduling of unequal execution time tasks on two identical processors. Technical Report CS-77-039, Washington State University, Pullman, WA, 1977.

[8] 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, August 1997.

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

[10] K.S. Hong and J.Y. Leung. Some results on liu's conjecture. *SIAM Journal on Discrete Mathematics*.

[11] T. C. Lai, Personal communication. May, 1995.

[12] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy 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, *Handbooks in Operations Research and Management Science, Vol 4., Logistics of Production and Inventory*, pages 445–522. North-Holland, 1993.

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

[14] J.H. Lin and J.S. Vitter. ϵ-approximation with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[15] C. L. Liu. Optimal scheduling on multiprocessor computing systems. In *Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory*, pages 155–60, 1972.

[16] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6:1–12, 1959.

[17] R. H. Möhring, M. W. Schäffter, and A. S. Schulz. Scheduling jobs with communication delays: Using infeasible solutions for approximation. In J. Diaz and M. Serna, editors, *Algorithms – ESA'96*, volume 1136 of *Lecture Notes in Computer Science*, pages 76 – 90. Springer, Berlin, 1996. Proceedings of the 4th Annual European Symposium on Algorithms.

[18] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82:199–223, 1998.

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

[20] M. Queyranne, 1995. Private communication.

[21] M. Queyranne and A.S. Schulz. Polyhedral approaches to machine scheduling. Technical Report Technical Report 474/1995, Technical University of Berlin, 1994.

[22] Andreas S. Schulz and Martin Skutella. Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria. Technical Report 533/1996, Technische Universität Berlin, Fachbereich Mathematik, 1996.

[23] A.S. Schulz. Scheduling to minimize total weighted completion time: performance guarantees of lp based heuristics and lower bounds. Technical Report 474/1995, Technical University of Berlin, 1995.

[24] Yaoguang Wang. Improved one-machine scheduling algorithms. Manuscript, November 1996.