A 1.47-approximation algorithm for a preemptive single-machine scheduling problem

*Michel X. Goemans** M.I.T.

Joel M. Wein† Polytechnic University

David P. Williamson‡ IBM Watson

Abstract

In this note, we give a 1.47-approximation algorithm for the preemptive scheduling of jobs with release dates on a single machine so as to minimize the weighted sum of job completion times; this problem is denoted by $1|r_j, pmtn|\sum_j w_j C_j$ in the notation of Lawler et al. [6]. Our result improves on a 2-approximation algorithm due to Hall, Schulz, Shmoys and Wein [4], and also yields an improved bound on the quality of a well-known linear programming relaxation of the problem.

Keywords: Analysis of algorithms: suboptimal algorithms. Production/scheduling: approximations/heuristic. Production/scheduling: single machine deterministic scheduling.

1 Introduction

In this note we consider the problem of scheduling n jobs on a single machine. Each job j must be scheduled for p_j units of time, and only one job can be scheduled at any point in time. We will consider *preemptive* scheduling: that is, we may suspend the scheduling of some job before processing it completely and resume it at a later point in time without penalty. In a *non-preemptive* schedule, we would have to schedule all p_j units of job j consecutively. We will impose the restriction that job j is not available to be scheduled before time r_j, called the *release date* of job j. A positive weight w_j is associated with job j, and our goal is to find a schedule which minimizes the sum

*Address: MIT, Room 2-351, 77 Mass. Ave, Cambridge, MA 02139. Email: `goemans@math.mit.edu`. Research supported in part by NSF contract 9623859-CCR, ARPA contract N00014-95-1-1246, a Sloan fellowship, and IBM. This research was conducted in part while the author was visiting IBM.

†Address: Department of Computer Science, Polytechnic University, Five MetroTech Center, Brooklyn, NY 11201. Email: `wein@mem.poly.edu`. Research partially supported by NSF Grant CCR-9626831,a grant from the New York State Science and Technology Foundation, through its Center for Advanced Technology in Telecommunications, and IBM. This research was conducted in part while the author was visiting IBM.

‡Address: IBM T.J. Watson Research Center, Room 33-219, P.O. Box 218, Yorktown Heights, NY, 10598. Email: `dpw@watson.ibm.com`.

$\sum_j w_j C_j$, where C_j is the time at which job j completes in the given schedule (the *completion time* of j). This scheduling problem is denoted $1|r_j, pmtn|\sum_j w_j C_j$ in the notation of Lawler et al. [6].

The problem $1|r_j, pmtn|\sum_j w_j C_j$ was shown to be strongly NP-hard by Labetoulle, Lawler, Lenstra, and Rinnooy Kan [5]. One approach to providing solutions for NP-hard optimization problems is to give an *approximation algorithm* for the problem. A ρ-approximation algorithm for $1|r_j, pmtn|\sum_j w_j C_j$ runs in polynomial time, and for any instance constructs a schedule whose sum of weighted completion times is no more than a factor of ρ times the value of an optimal schedule for the instance. The value ρ is sometimes called the *performance guarantee* of the algorithm. One can also consider *randomized* ρ-approximation algorithms for $1|r_j, pmtn|\sum_j w_j C_j$. Such an algorithm runs in randomized polynomial time (which means that its running time is guaranteed to be polynomial in the problem size, but that it is allowed to make random choices in the course of the algorithm) and for any instance constructs a schedule whose *expected* sum of weighted completion times is no more than a factor of ρ times the value of an optimal schedule for the instance.

There has been a good deal of research recently on the design and analysis of approximation algorithms for scheduling problems in which the objective is to minimize the sum of weighted completion times [7, 4, 1, 9]. These papers draw on interesting theoretical and experimental work on polyhedral approaches to these problems. As corollaries of the performance guarantees of these approximation algorithms, these papers also give bounds on the quality of linear-programming relaxations of the problems that had had no prior worst-case analysis.

The first approximation algorithm for $1|r_j, pmtn|\sum_j w_j C_j$ was given by Phillips, Stein and Wein [7], who gave an $(8+\epsilon)$-approximation algorithm based on a certain linear-programming relaxation of the problem. This was improved by Hall, Schulz, Shmoys and Wein [4] who made use of a linear-programming relaxation in C_j variables that was a valid relaxation of both $1|r_j, pmtn|\sum_j w_j C_j$ and the non-preemptive variant $1|r_j|\sum_j w_j C_j$. Based on this relaxation, they gave a 3-approximation algorithm for the non-preemptive problem and a 2-approximation algorithm for the preemptive problem. Subsequent to these results, Goemans [2] developed a new combinatorial understanding of this C_j-based formulation and then, based on it, developed a 2-approximation algorithm for $1|r_j|\sum_j w_j C_j$ [3]. All of these results also yielded bounds on the quality of the relevant linear-programming relaxation: for example, Goemans' result establishes that the optimal solution to the C_j-based relaxation of $1|r_j|\sum_j w_j C_j$ is no more than a factor of 2 smaller than the optimal schedule.

In this note we show that some of the ideas of Goemans [2, 3] can be used to give an improved preemptive algorithm as well. Specifically, we give a randomized 1.47-approximation algorithm for $1|r_j, pmtn|\sum_j w_j C_j$. We then note how it can be derandomized to give a (deterministic) 1.47-approximation algorithm as well. In work subsequent to an announcement of our result, Schulz and Skutella [8] gave a 4/3-approximation algorithm for this problem.

This note is structured as follows. In Section 2, we review relevant ideas and theorems from

Goemans [3], and in Section 3 we present our algorithm and its analysis.

2 Some preliminaries

The algorithm of Goemans [3] for the non-preemptive problem works as follows. First, the algorithm solves a linear programming relaxation of the scheduling problem. As shown in [2] there is an $O(n \log n)$ time algorithm to solve the relaxation. The algorithm then uses randomization and the structure of an optimal solution to the linear programming relaxation to create a non-preemptive schedule. Goemans shows that the expected value of the sum of the weighted completion times of his schedule is no more than twice the value of the linear programming relaxation, proving that the algorithm is a randomized 2-approximation algorithm. He also shows that it is possible to obtain a 2-approximation algorithm which does not need randomization.

The LP relaxation used is

$$
(LP) \qquad
\begin{aligned}
Z^*_{LP} \quad = \quad \text{Min} \quad & \sum_j \frac{w_j p_j}{2} + \sum_j w_j M_j \\
\text{subject to:} & \\
& \sum_{j \in S} p_j M_j \geq p(S)\left(r(S) + \frac{p(S)}{2}\right) \qquad \forall S,
\end{aligned}
$$

where $r(S) = \min_{j \in S} r_j$, $p(S) = \sum_{j \in S} p_j$, and variable M_j represents the *mean busy time* of job j in a schedule: if job j is processed during intervals in the set I in the schedule, then the mean busy time of job j is $(\int_I t\, dt)/(\int_I dt) = (\int_I t\, dt)/p_j$. For any schedule, preemptive or non-preemptive, setting the variables M_j to be the mean busy time of job j produces a feasible solution for (LP) [4, 3]. Note that in a non-preemptive schedule, M_j is simply the "midpoint" of job j, $M_j = C_j - p_j/2$, and the objective function of (LP) is thus the sum of weighted completion times. In contrast, note that in a preemptive schedule the objective function of (LP) is merely a lower bound on the sum of weighted completion times: the mean busy time of a job in a preemptive schedule is at most, but is not necessarily equal to, $C_j - p_j/2$.

Assume throughout this note that jobs are indexed so that $w_1/p_1 \geq w_2/p_2 \geq \cdots \geq w_n/p_n$, unless stated otherwise. Goemans proves that an optimal solution M for (LP) corresponds to the mean busy times of jobs in a preemptive schedule constructed by always scheduling the available job with the minimum index. Call this schedule $\mathcal{S}$. Because of release dates, this schedule will sometimes be a preemptive one. Observe that although $\mathcal{S}$ is an optimal solution to an LP relaxation of the preemptive scheduling problem, it is not an optimal preemptive schedule because the objective function of (LP) can not be interpreted as the sum of weighted completion times in a preemptive schedule. However, any preemptive schedule is not only feasible for (LP), but also has an objective function value no greater than the sum of the weighted completion times for that preemptive schedule, and thus Z^*_{LP} gives a lower bound on the optimal value of the preemptive scheduling problem.

We will principally need the following result from the analysis in [3]. $\mathcal{S}$ can be decomposed into sets of jobs S with certain properties. This decomposition is called the *canonical decomposition*, and sets S in the decomposition *canonical sets*. These sets S have the property that in schedule $\mathcal{S}$, jobs in S are processed continuously between $r(S)$ and $r(S)+p(S)$. Furthermore, the following theorem is proved in [3].

Theorem 1 [Goemans [3]] Let $\hat{C}_j$ be the completion time of job j in a preemptive schedule constructed by a randomized algorithm. If for all canonical sets S,

$$E\left[\sum_{j\in S} p_j(\hat{C}_j - p_j/2)\right] \le \rho \cdot p(S)(r(S)+p(S)/2),$$

then $E\left[\sum_j w_j \hat{C}_j\right]$ is no greater than $\rho \cdot Z^*_{LP}$.

An implication of this theorem is that if we can construct a preemptive (resp. non-preemptive) schedule in randomized polynomial time that obeys the condition of the theorem, then we have a randomized ρ-approximation algorithm for the preemptive (resp. non-preemptive) problem. Furthermore, this would prove that the worst-case ratio between the optimal preemptive (resp. non-preemptive) schedule and the optimal value of (LP) is at most ρ. Goemans constructs a non-preemptive schedule obeying the condition for $\rho = 2$. This yields a randomized 2-approximation algorithm for both the non-preemptive and the preemptive problems, and a proof that the worst-case ratio is at most two in both cases.

We will also be using the idea of an α-point in a preemptive schedule, first introduced in [7]. Given some parameter α, $0 \le \alpha \le 1$, and a preemptive schedule, the α-point of job j is the time at which αp_j units of job j have been processed.

3 The algorithm and its analysis

Our scheduling algorithm is as follows. We find the schedule $\mathcal{S}$ that is the optimal solution to (LP) in $O(n \log n)$ time. Given a parameter α $(0 \le \alpha \le 1)$, we construct a new preemptive schedule $\tilde{\mathcal{S}}$ from $\mathcal{S}$ as follows. We order the jobs by their α-points in $\mathcal{S}$ (e.g. the job with the earliest α-point is ordered first). We construct $\tilde{\mathcal{S}}$ by always scheduling the available job that appears earliest in the ordering of jobs by α-points. Observe that the machine is busy at exactly the same times in $\mathcal{S}$ and in $\tilde{\mathcal{S}}$. Denote the completion time of job j in $\tilde{\mathcal{S}}$ by $\hat{C}_j$.

We note that this algorithm can be derandomized to give a deterministic approximation algorithm with the same performance guarantee. Goemans proves that for a fixed preemptive schedule there are at most n possible different orderings of the jobs by α-points, where $0 \leq \alpha \leq 1$ [3, Proposition 3.1]. Since the schedule constructed by our algorithm is entirely determined by the ordering of jobs by their α-points, there are thus at most n possible schedules that can be constructed by our algorithm. We inspect every one of these schedules and choose the one with minimum sum of weighted completion times, which is guaranteed to be at most the expected performance of the algorithm.

We analyze the performance of our algorithm by examining what happens to a canonical set S of jobs. Recall that such a set S is processed continuously from time $r(S)$ to time $r(S)+p(S)$ in schedule $\mathcal{S}$, and observe that this implies that no job $j \in S$ is processed for any amount of time outside this interval.

We begin with the following lemma.

Lemma 2 Given a canonical set S, let α_j denote the fraction of job j processed in $\mathcal{S}$ before time $r(S)$, for all $j \notin S$. If jobs in S are indexed so that job 1 finishes first in $\tilde{\mathcal{S}}$, job 2 second, etc., then for $i \in S$,

$$\hat{C}_i \leq r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) + (1-\alpha)[p_1 + \cdots + p_i].$$

Proof: Note that in $\tilde{\mathcal{S}}$ from time $r(S)$ on, there may be a period of time during which $\tilde{\mathcal{S}}$ schedules some jobs with α points that occur before $r(S)$ in $\mathcal{S}$. This period of time, however, completes no later than $r(S) + \sum_{j:\alpha_j \geq \alpha}(1-\alpha_j)p_j$ in $\tilde{\mathcal{S}}$.

Subsequently, for $p(S)$ units of time, $\tilde{\mathcal{S}}$ only processes jobs in S. Consider the first job 1, and note that this job must also be the first of the jobs in S in the ordering of jobs by α-points. Thus its α-point occurs in $\mathcal{S}$ between times $r(S)$ and $r(S)+\alpha p(S)$. It follows that the job must be released by time $r(S)+\alpha p(S)-\alpha p_1$, and thus it completes in $\tilde{\mathcal{S}}$ by time $r(S)+\sum_{j:\alpha_j \geq \alpha}(1-\alpha_j)p_j + \alpha p(S) + (1-\alpha)p_1$. Therefore we have that

$$\hat{C}_1 \leq r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) + (1-\alpha)p_1.$$

Similarly, job 2 is released no later than $r(S)+\alpha p(S)+(1-\alpha)p_1 - \alpha p_2$ and thus

$$\hat{C}_2 \leq r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) + (1-\alpha)[p_1 + p_2].$$

The lemma follows by similar reasoning. ■

Lemma 3 Given a canonical set S, let α_j denote the fraction of job j processed in $\mathcal{S}$ before time $r(S)$ for all $j \notin S$. Then

$$\sum_{j \in S} p_j(\hat{C}_j - p_j/2) \leq p(S)\left[r(S) + \sum_{j:\alpha_j \geq \alpha}(1-\alpha_j)p_j + \frac{1}{2}(\alpha+1)p(S)\right].$$

Proof: By Lemma 2, for the ith job to complete in S,

$$\hat{C}_i \leq r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) + (1-\alpha)[p_1 + \cdots + p_i],$$

so that

$$p_i(\hat{C}_i - p_i/2) \leq p_i \left[r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) \right] + (1-\alpha)p_i[p_1 + \cdots + p_i] - p_i^2/2.$$

Let $p^2(S)$ denote $\sum_{i \in S} p_i^2$, and $p(S)^2 = \left(\sum_{i \in S} p_i\right)^2$. Then by summing this inequality over all jobs $j \in S$ we obtain

$$\begin{aligned}
\sum_{j \in S} p_j(\hat{C}_j - p_j/2) &\leq p(S) \left[r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) \right] + (1-\alpha) \sum_{i=1}^{|S|} p_i(p_1 + \cdots + p_i) - p^2(S)/2 \\
&= p(S) \left[r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) \right] + \frac{1}{2}(1-\alpha)(p(S)^2 + p^2(S)) - p^2(S)/2 \\
&\leq p(S) \left[r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \alpha p(S) + \frac{1}{2}(1-\alpha)p(S) \right] \\
&= p(S) \left[r(S) + \sum_{j:\alpha_j \geq \alpha} (1-\alpha_j)p_j + \frac{1}{2}(\alpha+1)p(S) \right].
\end{aligned}$$

■

We can now prove the following theorem.

Theorem 4 Suppose we choose α from the interval $[0, \beta]$ according to the probability distribution $f(\alpha) = \frac{1-\beta}{\beta(1-\alpha)^2}$. Then for any canonical set S,

$$E\left[\sum_{j \in S} p_j(\hat{C}_j - p_j/2) \right] \leq p(S) \left(\frac{1}{\beta} r(S) + \left(1 + \frac{1-\beta}{2\beta} \ln(1-\beta) \right) p(S) \right).$$

By setting $\beta \approx .682$, we have that

$$E\left[\sum_{j \in S} p_j(\hat{C}_j - p_j/2) \right] \leq 1.466 \cdot p(S)(r(S) + p(S)/2).$$

By Theorem 1, our algorithm is a randomized 1.466-approximation algorithm. It can be made deterministic by our observation earlier. Recall also that this also shows that the worst-case ratio between the value of the optimal preemptive schedule and the optimal value of (LP) is at most 1.466.

Proof: We wish to compute the expected value of $\sum_{j \in S} p_j(\hat{C}_j - p_j/2)$. We observe that, for α_j as defined in Lemma 3,

$$\Pr[\alpha_j \geq \alpha] \leq \int_0^{\alpha_j} f(\alpha) d\alpha$$

$$
\begin{aligned}
&= \frac{1-\beta}{\beta}\left(\frac{1}{1-\alpha_j}-1\right) \\
&= \frac{(1-\beta)\alpha_j}{\beta(1-\alpha_j)},
\end{aligned}
$$

so that

$$E\left[\sum_{j:\alpha_j\geq\alpha}(1-\alpha_j)p_j\right] = \sum_j (1-\alpha_j)p_j \Pr[\alpha_j \geq \alpha] = \frac{1-\beta}{\beta}\sum_j \alpha_j p_j \leq \frac{1-\beta}{\beta}r(S),$$

since $\sum_j \alpha_j p_j$, the total amount of jobs processed in $\mathcal{S}$ before time $r(S)$, can be no more than $r(S)$. Furthermore,

$$
\begin{aligned}
E[\alpha] &= \int_0^\beta \alpha f(\alpha) d\alpha \\
&= \frac{1-\beta}{\beta}\left[\frac{\alpha}{1-\alpha}+\ln(1-\alpha)\right]_0^\beta \\
&= \frac{1-\beta}{\beta}\left[\frac{\beta}{1-\beta}+\ln(1-\beta)\right] \\
&= 1+\frac{1-\beta}{\beta}\ln(1-\beta).
\end{aligned}
$$

Then by Lemma 3

$$
\begin{aligned}
E\left[\sum_{j\in S} p_j(\hat{C}_j - p_j/2)\right] &\leq p(S)\left(r(S)+\frac{1-\beta}{\beta}r(S)+\left(\frac{1}{2}+\frac{1-\beta}{2\beta}\ln(1-\beta)+\frac{1}{2}\right)p(S)\right) \\
&= p(S)\left(\frac{1}{\beta}r(S)+\left(1+\frac{1-\beta}{2\beta}\ln(1-\beta)\right)p(S)\right).
\end{aligned}
$$

■

Acknowledgements: We thank the anonymous referee for the suggestion to include a derandomization of the algorithm.

References

[1] S. Chakrabarti, C. Phillips, A. S. Schulz, D. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for minsum criteria. In *Automata, Languages, and Programming*, number 1099 in Lecture Notes in Computer Science, pages 646–657, Berlin, 1996. Springer-Verlag.

[2] M. X. Goemans. A supermodular relaxation for scheduling with release dates. 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 288–300, Berlin, 1996. Springer-Verlag.

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

[4] 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, Aug. 1997.

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

[6] 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, *Handbooks in Operations Research and Management Science, Vol. 4, Logistics of Production and Inventory*, pages 445–522. North-Holland, 1993.

[7] C. Phillips, C. Stein, and J. Wein. Scheduling jobs that arrive over time. In S. G. Akl, editor, *Algorithms and Data Structures*, number 955 in Lecture Notes in Computer Science, pages 86–97, Berlin, 1995. Springer-Verlag. Journal version to appear in Mathematical Programming B.

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

[9] 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 *LNCS*, pages 416–429. Springer, Berlin, 1997.