Operations Research Letters 44 (2016) 676–679

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

A 2.542-approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective

Martin Skutella

TU Berlin, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany

ARTICLE INFO

Article history:
Received 15 March 2016
Received in revised form
23 July 2016
Accepted 24 July 2016
Available online 29 July 2016

Keywords:
Approximation algorithm
Machine scheduling
Precedence constraints
Total weighted completion time
LP relaxation

ABSTRACT

We present a $\sqrt{e}/(\sqrt{e}-1)$-approximation algorithm for the nonpreemptive scheduling problem to minimize the total weighted completion time of jobs on a single machine subject to release dates and precedence constraints. The previously best known approximation algorithm dates back to 1997; its performance guarantee can be made arbitrarily close to the Euler constant e (Schulz and Skutella, 1997).

1. Introduction

We consider the following classical machine scheduling problem denoted by $1|r_j, prec|\sum w_jC_j$ in the standard classification scheme of Graham, Lawler, Lenstra, and Rinnooy Kan [12]. We are given a set of jobs $N = \{1, 2, \ldots, n\}$ and for every job $j \in N$ a processing time $p_j \geq 0$, a release date $r_j \geq 0$, and a weight $w_j \geq 0$. The jobs $j \in N$ need to be processed during non-overlapping time intervals of length p_j, and j's processing must not start before its release date r_j. Moreover, there are precedence constraints given by a partial order "$\prec$" on N where $j \prec k$ means that job j must be completed before job k may be started, that is, j's processing interval must precede k's. We may therefore without loss of generality assume throughout the paper that $j \prec k$ implies $r_j \leq r_k$. The objective is to minimize the total weighted completion time $\sum_{j\in N} w_jC_j$ where C_j denotes the first point in time at which j's processing is completed.

Complexity. Even for unit job weights, the special cases of the problem without non-trivial release dates $1|prec|\sum C_j$ (i.e., $r_j = 0$ for all $j \in N$) or without precedence constraints $1|r_j|\sum C_j$ are strongly NP-hard; see, e.g., [8, problem SS4]. In preemptive scheduling, the processing of a job may be repeatedly interrupted and resumed at a later point in time. In the absence of precedence constraints, the problem with unit job weights $1|r_j, pmtn|\sum C_j$ can be solved in polynomial time [3], but for arbitrary weights $1|r_j, pmtn|\sum w_jC_j$ is strongly NP-hard. Without non-trivial release dates preemptions are superfluous such that $1|prec, pmtn|\sum C_j$ is equivalent to $1|prec|\sum C_j$ and thus strongly NP-hard.

List scheduling. Before dipping into the rich history of approximation algorithms for these scheduling problems, we first discuss the most important algorithmic ingredient for both heuristic and exact solutions: *list scheduling*. Consider a list representing a total order on the set of jobs N, extending the given partial order "$\prec$". A straightforward way to construct a feasible schedule is to process the jobs in the given order as early as possible with respect to release dates. A schedule constructed in this way is a *list schedule*.

Depending on the given list and the release dates of jobs, the machine might remain idle when one job is completed but the next job in the list is not yet released. On the other hand, if job preemptions are allowed, it is certainly not advisable to leave the machine idle while another job at a later position in the list is already available (released) and waiting. Instead, we better start this job and preempt it from the machine as soon as the next job in the list is released. In *preemptive list scheduling* we process at any point in time the first available job in the list. The resulting preemptive schedule is feasible (as $j \prec k$ implies $r_j \leq r_k$) and is called *preemptive list schedule*.

Known techniques and results. There is a vast literature on approximation algorithms for the various scheduling problems mentioned

E-mail address: martin.skutella@tu-berlin.de.

http://dx.doi.org/10.1016/j.orl.2016.07.016
0167-6377/

above. Here we only mention those results that are particularly relevant in the context of this paper and refer to Chekuri and Khanna [5] for a more comprehensive overview. Various kinds of linear programming (LP) relaxations have proved to be useful in designing approximation algorithms. One of the simplest and most intuitive classes of LP relaxations is based on completion time variables only. These LP relaxations were introduced by Queyranne [16] and first used in the context of approximation algorithms by Schulz [17], who presents a 2-approximation algorithm for the problem $1|\,prec\,|\sum w_jC_j$ and a 3-approximation algorithm for $1|\,r_j,\ prec\,|\sum w_jC_j$; see also Hall, Schulz, Shmoys, and Wein [13]. These algorithms compute an optimal LP solution and then do list scheduling in order of increasing LP completion times. Moreover, Hall et al. [13] show that preemptive list scheduling in order of increasing LP completion times is a 2-approximation algorithm for $1|\,r_j,\ prec,\ pmtn\,|\sum w_jC_j$.

Phillips, Stein, and Wein [15] and Hall, Shmoys, and Wein [14] introduce the idea of list scheduling in order of so-called α-points to convert preemptive schedules to nonpreemptive ones. For $\alpha \in (0, 1]$, the α-point of a job with respect to a preemptive schedule is the first point in time when an α-fraction of the job has been completed. Goemans [10] and Chekuri, Motwani, Natarajan, and Stein [6] show that choosing α randomly leads to better results. In particular, Chekuri et al. [6] present an $e/(e-1)$-approximation algorithm for $1|\,r_j\,|\sum C_j$ by starting from an optimal preemptive schedule. Goemans [10] and Goemans, Queyranne, Schulz, Skutella, and Wang [11] give approximation results for the more general weighted problem $1|\,r_j\,|\sum w_jC_j$ based on a preemptive schedule that is an optimal solution to an LP relaxation in time-indexed variables. Similarly, Schulz and Skutella [18] give an $(e+\varepsilon)$-approximation algorithm for $1|\,r_j,\ prec\,|\sum w_jC_j$ for any $\varepsilon > 0$.

Bansal and Khot prove in a recent landmark paper [4] that there is no $(2-\varepsilon)$-approximation algorithm for $1|\,prec\,|\sum w_jC_j$, assuming a stronger version of the Unique Games Conjecture. Ambühl, Mastrolilli, Mutsanas, and Svensson [2], based on earlier work of Correa and Schulz [7] and Ambühl and Mastrolilli [1], prove an interesting relation between the approximability of $1|\,prec\,|\sum w_jC_j$ and the vertex cover problem

Our contribution. We present a $\sqrt{e}/(\sqrt{e}-1)$-approximation algorithm for the problem $1|\,r_j,\ prec\,|\sum w_jC_j$ based on the following two ingredients: (i) For the problem $1|\,r_j,\ prec,\ pmtn\,|\sum w_jC_j$ we slightly strengthen the 2-approximation result of Hall et al. [13] and show that preemptive list scheduling in order of increasing LP completion times on a machine running at double speed yields a schedule whose cost is at most the cost of an optimal schedule on a regular machine; see Section 2. (ii) Modifying the analysis of Chekuri et al. [6] we show how to turn the preemptive schedule on the double speed machine into a nonpreemptive schedule on a regular machine while increasing the objective function by at most a factor of $\sqrt{e}/(\sqrt{e}-1)$; see Section 3. We conclude with a conjecture in Section 4.

2. Optimal preemptive schedules under resource augmentation

In this section we consider the preemptive single machine scheduling problem with release dates, precedence constraints and total weighted completion time objective $1|\,r_j, prec, pmtn\,|\sum w_jC_j$. The best known approximation result for this problem is a 2-approximation algorithm due to Hall et al. [13] that is based on an LP relaxation in completion time variables originally introduced by Queyranne [16] and later refined by Goemans [9,10] for problems involving release dates. Let $S \subseteq N$ denote a set of jobs and define

$$p(S) := \sum_{j\in S} p_j \quad \text{and} \quad r_{\min}(S) := \min_{j\in S} r_j.$$

The LP relaxation in completion time variables C_j, $j \in N$, looks as follows:

$$\min \sum_{j\in N} w_jC_j$$

$$\text{s.t. } C_j \le C_k \quad \text{for all } j \prec k, \tag{1}$$

$$\frac{1}{p(S)}\sum_{j\in S} p_jC_j \ge r_{\min}(S) + \tfrac{1}{2}p(S) \quad \text{for all } S \subseteq N. \tag{2}$$

Notice that constraints (1) could be strengthened to $C_j + p_k \le C_k$, which is however not necessary for our purposes. Goemans [10] argues that constraints (2) hold for a feasible schedule, even if $(C_j)_{j\in N}$ denotes the vector of *mean busy times* of jobs instead of the larger completion times. Moreover, despite their exponential number, these constraints can be separated in polynomial time by efficient submodular function minimization [9]. Thus, an optimal solution C^* to the LP relaxation can be found in polynomial time and yields the LP lower bound $\sum_{j\in N} w_jC_j^*$ on the total weighted completion time of an optimal preemptive schedule. Reindex the set of jobs such that

$$C_1^* \le C_2^* \le \cdots \le C_n^* \quad \text{and} \quad (j \prec k \Rightarrow j < k). \tag{3}$$

The second condition in (3) is necessary to ensure that the total order of jobs by increasing indices extends the partial order given by the precedence constraints; notice, that in an optimal LP solution C_j^* might be equal to C_k^* for some pair of jobs with $j \prec k$.

Hall et al. [13] show that preemptive list scheduling according to list (3) yields a feasible preemptive schedule with completion times $C_j \le 2\cdot C_j^*$, $j \in N$, and thus a 2-approximate solution. Exactly the same analysis implies a slightly stronger result in terms of resource augmentation as we show in the next lemma. We imagine a machine running at double speed such that each job $j \in N$ needs to be processed for $p_j/2$ time units only.

Lemma 1. *Preemptive list scheduling according to list (3) on a machine running at double speed yields a feasible preemptive schedule with completion times $C_j' \le C_j^*$ for all $j \in N$.*

Proof. For a fixed $k \in N$, let S denote the subset of jobs $j \le k$ such that (i) $C_j' \le C_k'$, (ii) the preemptive list schedule does not leave the double speed machine idle between times C_j' and C_k', and (iii) only jobs $\ell \le k$ are being processed between times C_j' and C_k'. In particular, $k \in S$ by definition.

Claim. *The preemptive list schedule processes the set of jobs S without interruption during the time interval $I := [r_{\min}(S), C_k']$.*

To prove the claim, first notice that the preemptive list schedule never leaves the machine idle between the release and the completion of any job, as $j \prec h$ implies $r_j \le r_h$ such that no job will ever have to wait for another job that is not yet released. Consider a job $h \in S$ with $r_h = r_{\min}(S)$. As already mentioned, there is no idle time within the time interval $[r_h, C_h']$, and only jobs ℓ with $\ell \le h \le k$ are being processed there. Moreover, due to (ii) there is no idle time within the time interval $[C_h', C_k']$ and only jobs ℓ with $\ell \le k$ are being processed there due to (iii). As a consequence, there is no idle time in I and only jobs ℓ with $\ell \le k$ are being processed in I. Therefore, every job j with $C_j' \in I$ satisfies conditions (i), (ii), and (iii), and is thus contained in S. Finally, since any job ℓ that the preemptive list schedule processes in I satisfies $\ell \le k$ and is therefore completed before job k in I, the claim follows.

The claim implies that $C_k' = r_{\min}(S) + \frac{1}{2}p(S)$. Finally,

$$C_k^* \ge \frac{1}{p(S)}\sum_{j\in S} p_jC_j^* \ge r_{\min}(S) + \tfrac{1}{2}p(S) = C_k',$$

where the first inequality holds as $C_j^* \le C_k^*$ for $j \le k$ and the second inequality follows by the LP constraints (2). □