European Journal of Operational Research 219 (2012) 27–33

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Preemptive scheduling of two uniform parallel machines to minimize total tardiness

Irina N. Lushchakova *

Belarusian State University of Informatics and Radioelectronics 6, P. Brovka St., Minsk 220013, Republic of Belarus

ARTICLE INFO

Article history:
Received 16 July 2010
Accepted 9 November 2011
Available online 18 November 2011

Keywords:
Scheduling
Uniform machines
Total tardiness
Polynomial algorithm

ABSTRACT

We consider the problem of preemptive scheduling n jobs on two uniform parallel machines. All jobs have equal processing requirements. For each job we are given its due date. The objective is to find a schedule minimizing total tardiness $\sum T_i$. We suggest an $O(n \log n)$ algorithm to solve this problem.

1. Introduction

We consider the following scheduling problem.

There are $M = 2$ uniform parallel machines and a set $N = \{1,2,\ldots,n\}$ of jobs. All jobs are available simultaneously. For each job $i \in N$ we are given the processing requirement p_i and its due date d_i. We suppose that all jobs have equal processing requirements, i.e. all $p_i = p$. Each job i has to be processed on any machine. Machine L, $1 \leqslant L \leqslant 2$, processes any job with the same speed v_L. This means that the processing time of any job on machine L is equal to p/v_L. Preemptions are allowed. After interrupting, job i may be processed on the other machine. Each machine can process at most one job at a time and each job can be processed on at most one machine at a time. For a feasible schedule s, let $C_i(s)$ be the completion time of job i. Determine the tardiness $T_i(s)$ of job i over its due date d_i in the following way: $T_i(s) = \max\{0; C_i(s) - d_i\}$. The objective is to find a schedule s^* minimizing the total tardiness $\sum_{i \in N} T_i(s)$.

Following the notation system introduced by Graham et al. [8], we denote the described problem by $Q2|p_i = p, pmtn|\sum T_i$. This problem is indicated by Brucker and Knust [4] as minimal open one. We suggest an $O(n \log n)$ algorithm to solve it.

Baptiste et al. [1] showed that the analogous problem $P|p_i = p, pmtn|\sum T_i$ with an arbitrary number of parallel identical machines can be solved in polynomial time. On the other hand, the problem $P|pmtn|\sum T_i$ with arbitrary processing times of jobs and the problem $P|r_i, p_i = p, pmtn|\sum T_i$ with release dates for jobs are shown to be NP-hard (in the ordinary sense) [11]. Recently, Kravchenko and Werner [11] suggested a polynomial algorithm to solve the problem $Q|p_i = p, pmtn|\sum T_i$ with an arbitrary number of uniform machines. Their approach uses a linear program with $O(Mn^3)$ variables, where M is the number of uniform machines. Notice that for the case $M = 2$ our algorithm solves the problem much faster.

When preemptions are forbidden, the $Q|p_i = p|\sum w_i T_i$ problem with an arbitrary number of uniform machines and the weighted total tardiness criterion can be solved in $O(n^3)$ time using assignment problem [3,8,18]. We would like to emphasize that for the problem under consideration preemptions are advantageous.

For preemptive scheduling with uniform machines we can also note the following results. When the number M of machines is variable, the problem $Q|pmtn|\sum C_i$ of minimizing the total completion time was shown by Gonzalez [6] to be solved in $O(n \log n + Mn)$ time using the SPT-rule (see also [8,12]). The problem $Q2|r_i, p_i = p, pmtn|\sum C_i$ with release dates for jobs can be solved in $O(n^3)$ time [14]. When the number of machines is arbitrary, Kravchenko and Werner [10] showed that the problem $Q|r_i, p_i = p, pmtn|\sum C_i$ can be solved in polynomial time. Lawler and Martel solved the problem $Q2|pmtn|\sum U_i$ of minimizing the number $\sum U_i$ of late jobs in $O(n^3)$ time [13]. Besides, for the problem $Q2|pmtn|\sum w_i U_i$ they suggested an $O(Wn^2)$ algorithm, where W is the sum of the job weights, as well as presented a fully polynomial approximation scheme [13]. Baptiste et al. [2] showed that the problem $Q|pmtn, p_i = p|\sum U_i$ can be solved in polynomial time. The preemptive scheduling problems for uniform machines with some other objective functions and restrictions were investigated in [5,7,15–17].

The paper is organized as follows. In Section 2 we give Algorithm *Q2TT* for solving the $Q2|p_i = p, pmtn|\sum T_i$ problem. Algorithm *Q2TT* uses the transformation of partial schedules. This transformation process and the justification of its correctness are discussed in Section 3. In Section 4 we prove that Algorithm

* Fax: +375 172 932333.
E-mail address: IrinaLushchakova@yandex.ru

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.11.023

Q2TT constructs an optimal schedule for the problem under consideration.

2. The algorithm

Suppose that $v_1 \geqslant v_2$. Define $q = v_1/v_2$. Without loss of generality we assume that $v_2 = 1$. Therefore, $v_1 = q \geqslant 1$.

In this paper we use the variant of the Shortest Processing Time (SPT) rule for preemptive scheduling of uniform machines [6,8,12] which in the case of two uniform machines may be described in the following way.

The variant of the SPT rule:

Order the jobs according to nondecreasing order of their processing requirements.

Schedule job 1 on machine 1. Having scheduled jobs $1,2,\ldots,i$, schedule job $i+1$ on machine 2 until machine 1 becomes available, then interrupt the processing of job $i+1$ on machine 2 and resume its processing on machine 1, thereby completing job $i+1$ as soon as possible (see Fig. 1). □

The above variant of the SPT rule constructs the schedule minimizing total completion time [6,8,12]. Recall that in the problem under consideration all jobs have equal processing requirements and are available simultaneously.

Suppose that the set N of jobs is ordered according to nondecreasing order of their due dates d_i. Using the described variant of the SPT-rule, we assign jobs for processing one after another. Let job k be the first job, which has the following property: job k completes after its due date d_k while the previous job $k-1$ completes before its due date d_{k-1}. In this case it is natural to try to diminish the completion time of job k by means of increasing the completion time of job $k-1$.

Consider the partial schedule s_k for the first k jobs. Suppose that in the schedule s_k jobs $k-1$ and k complete the processing at the time moments $C_{k-1} = C_{k-1}(s_k)$ and $C_k = C_k(s_k)$, respectively. Set $\mu = d_{k-1} - C_{k-1}$ and $\lambda = C_k - d_k$ (see Fig. 2).

Notice, that for any part of a job with p' unit processing requirement its processing time on machine 1 is equal to p'/q time units, $q > 1$, while its processing time on machine 2 is equal to p' time units. We want to diminish the completion time of job k in such a way that job $k-1$ will not violate its due date d_{k-1}. Notice, that in the time interval $(d_k, C_k]$ job k is processed on machine 1. In the partial schedule s_k only machine 2 has an idle interval before the due date d_{k-1}, and the length of this idle interval is μ. As a result, job k can complete the processing on machine 1 earlier by no more than μ/q units. Thus, we decrease the completion time of job k by $\delta = \min\{\mu/q, \lambda\}$ time units.

Below in Section 3 we describe the procedure *TRANS*(s_k,δ). This procedure transforms the partial schedule s_k into the new partial schedule $\widetilde{s_k}$. In the schedule $\widetilde{s_k}$ the value of the partial objective function for the first $k-1$ jobs is the same as in the schedule s_k, while job k completes the processing at the time moment $C_k - \delta$.

Further, having constructed the schedule $\widetilde{s_k}$, we repeat the above process by assigning jobs one after another to machines and transforming, if necessary, the obtained partial schedules.

The following algorithm describes all these actions more formally. Hereafter, if a job violates its due date, we call it a plus-job, otherwise a job is called a minus-job.

Algorithm Q2TT

```
Stage 1
Schedule job 1 on machine 1.
For $k$ from 2 to $n$ do
Stage $k$
1. Having scheduled jobs $1,2,\ldots,k-1$, schedule job $k$ on
   machine 2 until machine 1 becomes available, then
   interrupt the processing of job $k$ on machine 2 and resume
   its processing on machine 1.
2. If job $k$ is a plus-job, while the previous job $k-1$ is a minus-
   job, go to step 3; otherwise go to step 6.
3. Find the value $\delta = \min\{\mu/q, \lambda\}$.
4. If $\delta = 0$, set $\widetilde{s_k} = s_k$ and go to step 6.
5. TRANS$(s_k,\delta)$
6. Complete Stage $k$.
                end;
After Stage $n$ has been completed, stop. □
```

Step 5 of Algorithm *Q2TT* requires the constant time (see Section 3). Therefore, Algorithm *Q2TT* can be implemented in $O(n)$ time, provided that the set N of jobs is ordered in advance. Ordering the jobs according to nondecreasing order of their due dates requires $O(n\log n)$ time. Thus, the problem under consideration can be solved in $O(n\log n)$ time.

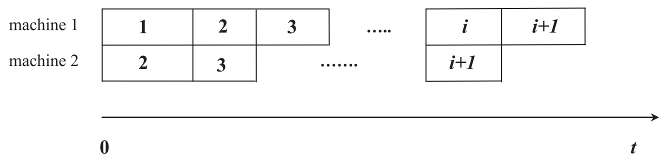

Fig. 1. The SPT-schedule for two uniform machines.

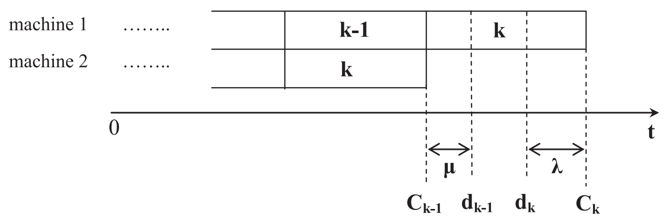

Fig. 2. The partial schedule s_k.

3. The procedure *TRANS*(s_k, δ) and its justification

For each job i, $1 \leqslant i \leqslant k$, let C_i denote its completion time in the partial schedule s_k, i.e. $C_i = C_i(s_k)$. Besides, we set $C_0 = 0$.

In this section we describe the procedure *TRANS*(s_k,δ) which transforms the schedule s_k into the new partial schedule $\widetilde{s_k}$ with the following completion times of jobs: the completion time of job k decreases by the value $\delta = \min\{\mu/q,\lambda\}$ and becomes equal to $C_k - \delta$, the completion time of job $k-1$ increases by the value $q\delta$, the completion times of jobs $1,2,\ldots,k-2$ do not change. For each job i, $1 \leqslant i \leqslant k$, we denote its completion time in the new schedule by $\widetilde{C}_i$, i.e. $\widetilde{C}_i = C_i(\widetilde{s_k})$. Besides, let y denote the processing time of job k on machine 2 in the schedule $\widetilde{s_k}$.

Two cases may occur, each of them being handled separately. The motivation to each case and the justification of the correctness are done after the procedure.

TRANS(s_k,δ)

(a) If

$$\widetilde{C}_k = C_k - \delta \geqslant \begin{cases} C_{k-2} + \frac{p}{q}, & k > 2, \\ \frac{p}{q}, & k = 2, \end{cases}$$