Computing 71, 375–386 (2003)
Digital Object Identifier (DOI) 10.1007/s00607-003-0029-z

Computing
Printed in Austria

Notes on Max Flow Time Minimization with Controllable Processing Times

Monaldo Mastrolilli*, Switzerland

Received February 11, 2003; revised June 16, 2003
Published online: October 30, 2003

Abstract

In a scheduling problem with *controllable processing times* the job processing time can be compressed through incurring an additional cost. We consider the identical parallel machines max flow time minimization problem with controllable processing times. We address the preemptive and non-preemptive version of the problem. For the preemptive case, a linear programming formulation is presented which solves the problem optimally in polynomial time. For the non-preemptive problem it is shown that the First In First Out (FIFO) heuristic has a tight worst-case performance of $3 - 2/m$, when jobs processing times and costs are set as in some optimal preemptive schedule.

Keywords: Scheduling, controllable processing times, parallel machines, approximation algorithms.

1. Introduction

Motivation. The m-machine scheduling problem is one of the most widely-studied problems in computer science, with an almost limitless number of variants (see [3] for a survey). The most common objective function is the *makespan*, which is the length of the schedule, or equivalently the time when the last job is completed. This objective function formalizes the viewpoint of the *owner* of the machines. If the makespan is small, the utilization of his machines is high; this captures the situation when the benefits of the owner are proportional to the work done. If we turn our attention to the viewpoint of a *user*, the time it takes to finish individual jobs may be more important; this is especially true in interactive environments. Thus, if many jobs that are released early are postponed at the end of the schedule, it is unacceptable to the user of the system even if the makespan is optimal.

For that reason other objective functions are studied. With this aim, a well-studied objective function is the *total flow time* [1, 12, 16]. The flow time of a job is the time the job is in the system, i.e., the completion time minus the time when it becomes first available. The above mentioned objective function is the sum of these values

*Supported by Swiss National Science Foundation project 20-63733.00/1, "Resource Allocation and Scheduling in Flexible Manufacturing Systems", and by the "Metaheuristics Network", grant HPRN-CT-1999-00106.

over all jobs. The *Shortest Remaining Processing Times* (SRPT) heuristic produces a schedule with optimum total flow time (see [11]) when there is a single processor. Unfortunately, this heuristic has the well-known drawback that it leads to *starvation*. That is, some jobs may be delayed to an unbounded extent. Inducing starvation is an inherent property of the total flow time metric. In particular, there exists inputs where any optimal schedule for total flow time forces the starvation of some job (see Lemma 2.1 in [2]). This property is undesirable.

From the discussion above, it is natural to conclude that in order to avoid starvation, one should bound the flow time of each job. This motivates the study of the minimization of the *maximum flow time*.

Most classical scheduling models assume *fixed* processing times for the jobs. However, in real-life applications the processing time of a job often depends on the amount of resources such as facilities, manpower, funds, etc. allocated to it, and so the processing time can be reduced when additional resources are given to the job. This speed up of the processing time of a job comes at a certain cost. A scheduling problem in which the processing times of the jobs can be reduced at some expense is called a scheduling problem with controllable processing times. Scheduling problems with controllable processing times have gained importance in scheduling research since the pioneering works of Vickson [24, 25]. For a survey of this area until 1990, the reader is referred to [20]. Recent results include [4, 5, 9, 10, 17, 19, 22, 23].

Problem Definition. We address the following scheduling problem. We have a set $\mathcal{J}$ of n jobs, $\mathcal{J} = \{J_1, \ldots, J_n\}$, and m identical machines $M = \{M_1, \ldots, M_m\}$. Each job J_j must be processed in an uninterrupted fashion on one of the machines, each of which can process at most one job at a time. We will also consider the *preemptive* case, in which a job may be interrupted on one machine and continued later (possibly on another machine) without penalty. Job J_j $(j = 1, \ldots, n)$ is released at time $r_j \geq 0$ and cannot start processing before that time.

The processing time of job J_j lies in an interval $[\ell_j, u_j]$ (with $0 \leq \ell_j \leq u_j$). For each job J_j we have to choose a machine $\mu_j \in M$, and $\delta_j \in [0, 1]$ and get then processing time and cost that depend linearly on δ_j:

$$p_j(\delta_j) = \delta_j \ell_j + (1 - \delta_j) u_j,$$
$$c_j(\delta_j) = \delta_j c_j.$$

We refer δ_j as the *compression level* of job J_j, since the processing time $p_j(\delta_j)$ of J_j decreases by increasing δ_j.

Solving a non-preemptive scheduling problem with controllable processing times amounts to specifying an *assignment*, $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$, of jobs to machines (where $\mu_j \in M$ is the machine job J_j is assigned to), and a *selection* of the compression levels, $\delta = (\delta_1, \delta_2, \ldots, \delta_n)$, that defines jobs processing times $p_j(\delta_j)$ and costs $c_j(\delta_j)$. The *total cost* $C(\delta)$ of δ is defined as $C(\delta) = \sum_{j=1}^{n} c_j(\delta_j)$. We denote the completion time of job J_j in a schedule S by E_j^S or E_j, if no confusion is

possible. The flow time of job J_j is defined as $F_j = E_j - r_j$, and the *maximum flow time* $F_{\max}$ is $\max_{j=1,\ldots,n} F_j$. We seek to minimize the maximum flow time when the total cost is constrained to be at most κ, i.e., $C(\delta) \leq \kappa$, for some given value $\kappa > 0$. According to Graham et al. [7], we denote the preemptive and non-preemptive version of the described problem by $P|pmtn; r_j; contr; C(\delta) \leq \kappa|F_{\max}$ and $P|r_j; contr; C(\delta) \leq \kappa|F_{\max}$, respectively.

Known Results. Problem $P|r_j; contr; C(\delta) \leq \kappa|F_{\max}$ is strongly NP-hard [6] since the special case with fixed processing times and identical release dates (i.e. the makespan minimization problem) is strongly NP-hard [15]. The practical importance of NP–hard problems necessitates tractable relaxations. A very fruitful approach has been to relax the notion of optimality and settle for *near–optimal* solutions. A near–optimal solution is one whose objective function value is within some small multiplicative "1" factor of the optimal value. *Approximation algorithms* are heuristics that in polynomial time provide provably good guarantees on the quality of the solutions they return. The book on approximation algorithms edited by Hochbaum [8] gives a good glimpse of the current knowledge on the subject.

When preemption is allowed and processing times are fixed, we observe that there are polynomial-time off-line algorithms for finding optimal preemptive solutions: these are obtained by adapting the approaches proposed in [13, 14] for the preemptive parallel machines problems with release times and deadlines. In [13, 14] the objective function is the minimization of the maximum lateness $L_{\max} = \max L_j$, where L_j is the lateness of job J_j, that is the completion time of J_j minus the its deadline (the time by which job J_j must be completed). We can use the algorithms in [13, 14] for the preemptive maximum flow time minimization by setting the deadline of each job equal to its release time.

When processing times are fixed and preemption is not allowed, to the best of our knowledge, the only known result about the non-preemptive max flow time scheduling problem is due to Bender et al. [2]. They address the on-line non-preemptive problem with identical parallel machines (in the notation of Graham et al. [7], this problem is noted $P|\text{on-line}; r_j|F_{\max}$). In [2] they claim that the *First In First Out* (FIFO) heuristic (that is, scheduling jobs in the order they arrive to the first available machine) is a $(3 - 2/m)$-competitive algorithm. They also claim a fully polynomial time approximation schemes when preemption is allowed (this works also for the max stretch minimization problem).

When processing times are controllable the only known results are about the makespan model. Nowicki and Zdrzalka [21] consider the preemptive scheduling of m identical parallel machines. They provide a $O(n^2)$ greedy algorithm which generates the set of Pareto-optimal points. When preemption is not allowed and the machines are not identical, Trick [23] gave a polynomial time 2.618-approximation algorithm (i.e., an algorithm that returns a solution whose value is within 2.618 times the optimal value) to minimize a weighted sum of the cost and the makespan. The latter result was improved by Shmoys and Tardos [22] by providing a polynomial time 2-approximation algorithm.

New Results. We consider the preemptive and non-preemptive max flow time minimization problems with controllable processing times, i.e. $P|pmtn; r_j; contr; C(\delta) \leq \kappa|F_{\max}$ and $P|r_j; contr; C(\delta) \leq \kappa|F_{\max}$. In Section 2 we observe that the preemptive problem $P|pmtn; r_j; contr; C(\delta) \leq \kappa|F_{\max}$ is polynomially solvable (and hence it is polynomially solvable also the special case with fixed processing times addressed in [2]). A polynomial time algorithm may work as follows. First consider the following problem: Given some threshold value ϕ, compute, if it exists, a schedule that minimizes the total cost when the max flow time is constrained to be at most ϕ. This problem may be reduced to a linear program. Then, apply binary search on different ϕ-values, and return the solution with minimum max flow time and cost at most κ.

In Section 3 we address the non-preemptive problem $P|r_j; contr; C(\delta) \leq \kappa|F_{\max}$. We show that a solution with cost at most κ and max flow within factor $(3 - 2/m)$ of the optimal value may be obtained by using FIFO, when jobs processing times and costs set as in some optimal preemptive schedule. We prove that this bound is asymptotically tight.

2 The Preemptive Problem

The goal of this section is to find a preemptive schedule on m identical machines with minimum max flow time and total cost at most κ. We adopt and extend some ideas presented in [14] for the preemptive parallel machines problem with release times, deadlines and fixed processing times.

The proposed algorithm is sketched below. We first consider the following problem: Given some threshold value ϕ, compute, if it exists, a schedule that minimizes the total cost when the max flow time is constrained to be at most ϕ, i.e., $F_{\max} \leq \phi$. This problem may be reduced to a linear program. Then, we apply binary search on different ϕ-values, and return the solution with minimum max flow time and cost at most κ.

2.1 Linear Programming

We start observing that the max flow time is at most ϕ if and only if

$$E_j \leq d_j^{\phi} := r_j + \phi \text{ for all } j = 1, \ldots, n.$$

Thus, all jobs J_j must finish before the deadline d_j^{ϕ} and cannot start before the release time r_j, i.e., each job J_j must be processed in an interval $[r_j, d_j^{\phi}]$ associated with J_j. We call these intervals *time windows*.

Next we address the problem of finding a preemptive schedule for jobs J_j $(j = 1, \ldots, n)$ on m identical machines such that all jobs J_j are processed within their time windows $[r_j, d_j^{\phi}]$ and the total cost is minimized. This problem may be reduced to a linear programming constructed as follows.

Let

$$z_1 < z_2 < \ldots < z_{w+1}$$

be the ordered sequence of all different r_j values and d_j^{ϕ} values. Consider the intervals

$$I_v := [z_v, z_{v+1}] \text{ of length } Z_v = z_{v+1} - z_v \text{ for } v = 1, \ldots, w.$$

Then we have to solve

$$\begin{aligned}
& \min \quad \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{v=1}^{w} c_j x_{ij}^{(v)} \\
(A) \quad & s.t. \quad \sum_{i=1}^{m} \sum_{v=1}^{w} (x_{ij}^{(v)} + y_{ij}^{(v)}) = 1 \qquad j = 1, \ldots, n; \\
(B) \quad & \sum_{j=1}^{n} (x_{ij}^{(v)} \ell_j + y_{ij}^{(v)} u_j) \le Z_v \quad i = 1, \ldots, m; \quad v = 1, \ldots, w; \\
(C) \quad & \sum_{i=1}^{m} (x_{ij}^{(v)} \ell_j + y_{ij}^{(v)} u_j) \le Z_v \quad j = 1, \ldots, n; \quad v = 1, \ldots, w; \\
(D) \quad & x_{ij}^{(v)},\ y_{ij}^{(v)} \ge 0 \qquad j = 1, \ldots, n; \quad i = 1, \ldots, m; \\
& \qquad\qquad\qquad\qquad\qquad\quad v = 1, \ldots, w; \\
(E) \quad & x_{ij}^{(v)} = y_{ij}^{(v)} = 0 \qquad j = 1, \ldots, n; \quad i = 1, \ldots, m; \\
& \qquad\qquad\qquad\qquad\qquad\quad v = 1, \ldots, w; \quad z_v < r_j;\ d_j^{\phi} \le z_v.
\end{aligned} \tag{1}$$

Note that $x_{ij}^{(v)} \ell_j + y_{ij}^{(v)} u_j$ denotes the total amount of time that machine M_i spends on job J_j in time period I_v (therefore $x_{ij}^{(v)}$ represents the compression level of job J_j in time interval I_v when processed by machine M_i). The first set (A) of constraints ensures that every job gets assigned to some machines and time windows. The second set (B) guarantees that the total load on each machine and for each time window is at most the interval length. The third set (C) says that at each time window the total time spent processing one job is at most the interval length (this is a necessary condition to avoid overlapping). Finally, set (E) ensures that no job starts processing before its release date and ends later its deadline d_j^{ϕ}.

If there exists a feasible solution of the linear program (1) then there exists a preemptive schedule respecting all time windows. In this case the processing time and cost of job J_j ($j = 1, \ldots, n$) are, respectively,

$$\sum_{v=1}^{w} \sum_{i=1}^{m} (x_{ij}^{(v)} \ell_j + y_{ij}^{(v)} u_j) = \delta_j \ell_j + (1 - \delta_j) u_j, \tag{2}$$

$$\sum_{v=1}^{w} \sum_{i=1}^{m} x_{ij}^{(v)} c_j = \delta_j c_j, \tag{3}$$

where the compression level δ_j of job J_j is $\delta_j = \sum_{v=1}^{w} \sum_{i=1}^{m} x_{ij}^{(v)}$ and $1 - \delta_j = \sum_{v=1}^{w} \sum_{i=1}^{m} y_{ij}^{(v)}$, by the first set (A) of constraints of (1). Moreover, the total amount of processing time in I_v is at most mZ_v, which is the capacity of m machines, i.e.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij}^{(v)} \ell_j + y_{ij}^{(v)} u_j) \leq mZ_v. \tag{4}$$

Furthermore, the total time job J_j is processed in interval I_v is at most the interval length, i.e.

$$\sum_{i=1}^{m} (x_{ij}^{(v)} \ell_j + y_{ij}^{(v)} u_j) \leq Z_v. \tag{5}$$

A feasible solution for the preemptive scheduling problem with time windows is constructed by scheduling partial jobs J_{jv} with processing times $\sum_{i=1}^{m} (x_{ij}^{(v)} \ell_j + y_{ij}^{(v)} u_j)$ in the intervals I_v on m identical machines. For each interval I_v, this is essentially the preemptive makespan minimization of the identical parallel machines scheduling problem, which has a solution with total length at most Z_v because of (4) and (5). (A schedule meeting this bound can be constructed in linear time by using McNaughton's rule [18]: fill the machines successively, scheduling the jobs in any order and splitting jobs into two parts whenever the above time bound Z_v is met. Schedule the second part of a preempted job on the next machine at the beginning of interval I_v. Because of (5) for all J_j, the two parts of preempted job do not overlap.)

2.2 Binary Search

A solution with minimum max flow and cost at most κ can be computed by embedding the algorithm described in the previous subsection within a binary search procedure. Assume, without loss of generality, that the input instance has integral data. Clearly, the value of the optimal flow time is in the interval $[0, nu_{\max}]$, where $u_{\max} = \max_j u_j$. Thus we can use $UB := 0$ and $LB := nu_{\max}$ as initial upper and lower bounds, respectively, for the binary search; in each iteration the algorithm performs the following steps:

a) it uses the linear program (1) to find a schedule of minimum cost and max flow time at most the value ϕ, where $\phi = (LB + UB)/2$ (if a solution of max flow time ϕ exists);

b) if there exists a feasible solution and the total cost is not greater than κ, then update the upper bound UB to ϕ, otherwise update the lower bound LB to $\phi + 1$.

The algorithm terminates when $LB = UB$, and outputs the best schedule found. A straightforward proof by induction shows that, throughout the execution of the

binary search, the *LB* value is never greater than the optimal max flow value when the cost is constrained to be at most κ. After $O(\log(nu_{\max}))$ iterations the search converges. The resulting algorithm is polynomial in the binary encoding of the input size.

We conclude that problem $P|pmtn; r_j; contr; C(\delta) \leq \kappa|F_{\max}$ is polynomially solvable.

3 The Non-Preemptive Problem

In this section we present an approximation algorithm for minimizing the max flow time when preemption is not allowed. The algorithm is as follows. First, jobs processing times and costs are set as in some optimal preemptive schedule SOL^p (see (2) and (3) of Section 2.1). Then, jobs are scheduled according to FIFO heuristic (that is, scheduling jobs in the order they are released to the first available machine). The analysis shows that the proposed algorithm is fairly good, and it comes within the same bound as when processing times are fixed [2].

Theorem 1 *When jobs processing times and costs are set as in some optimal preemptive schedule, FIFO returns a* $(3 - 2/m)$*-approximate solution for problem* $P|r_j; contr; C(\delta) \leq \kappa|F_{\max}$.

Proof. Assuming, without loss of generality, that jobs are renumbered such that $r_1 \leq r_2 \leq \ldots \leq r_n$ holds, then jobs $J_1, J_2, \ldots, J_n$ are assigned in this order to the first available (i.e. idle) machine. Consider the schedule *SOL* returned by FIFO and let $F_{\max}$ denote the maximum flow time of *SOL*. Denote by $p_1, \ldots, p_n$ and $c_1, \ldots, c_n$ the jobs processing times and costs, respectively, when they are set as in some optimal preemptive schedule SOL^p (see (2) and (3) of Section 2.1). Note that the total cost is bounded by κ by construction, i.e., $\sum_j c_j \leq \kappa$. Let $F^p_{\max}$ and $F^*_{\max}$ denote the optimal preemptive and the optimal non-preemptive solution values, respectively.

Let J_f be a job that attains the maximum flow time value, i.e. $F_{\max} = s_f + p_f - r_f$, where s_f is the starting time of job J_f according to *SOL*. Without loss of generality, we can assume that job J_f was the last job released, i.e. $J_f = J_n$ and $r_1 \leq r_2 \leq \ldots \leq r_n$, since otherwise we can truncate the instance at this point and obtain a new instance on which FIFO performs at least as badly relative to the optimal schedule.

Let E_i $(i = 1, \ldots, m)$ denote the ending time of machine M_i. Without loss of generality, let us assume that J_f is processed on machine M_1. Since jobs are assigned to the first available machine, we have

$$E_i \geq s_f, \text{ for } i = 2, \ldots, m.$$

Summing this inequality over all machines M_i for $i > 1$, and adding s_f yields

$$\sum_{i>1} E_i + s_f \geq m s_f,$$

and hence

$$s_f \leq \frac{1}{m}\left(\sum_{i=1}^{m} E_i - p_f\right). \tag{6}$$

According to *SOL*, let J_h be the last job that starts processing as soon as it is released, i.e. its starting time s_h is equal to r_h. It is not difficult to recognize that such a job J_h always exists.

If $J_h = J_f$ then the returned solution is optimal, since $F_{\max} = s_f + p_f - r_f = p_f \leq F^p_{\max} \leq F^*_{\max}$, and the claim is proved.

Otherwise, observe that after time r_h each machine M_i $(i = 1, \ldots, m)$ has no idle time up to its ending time E_i. This means that the sum of machines ending times $\sum_{i=1}^{m} E_i$ is equal to $m \cdot r_h$ plus the total time T_h spent by machines after time r_h.

To bound T_h, we first note that at time r_h there are at most $m - 1$ jobs that are released before time r_h and that have not yet been completed. Indeed, if there had been more jobs J_j with $r_j < r_h$ at time r_h, than they would have taken priority over J_h, and J_h would not have been scheduled at time r_h. Their contribution to the value of T_h is bounded by $(m-1)p_{\max}$, where $p_{\max} = \max_j p_j$.

The second contribution to the value of T_h is given by those jobs J_j with $r_h \leq r_j \leq r_f$. Observe that these jobs in the preemptive optimal solution start not before time r_h and end not later than $r_f + F^p_{\max}$. It follows that the sum or their processing times is at most $m(r_f - r_h + F^p_{\max})$.

We then obtain

$$\sum_{i=1}^{m} E_i \leq (m-1)p_{\max} + m r_h + m(r_f - r_h + F^p_{\max}),$$

and by inequality (6) we have

$$\begin{aligned} s_f &\leq \frac{1}{m}\left(\sum_{i=1}^{m} E_i - p_f\right) \\ &= \frac{1}{m}\left((m-1)p_{\max} + m(r_f + F^p_{\max}) - p_f\right) \\ &= p_{\max} + r_f + F^p_{\max} - \frac{p_{\max} + p_f}{m}. \end{aligned}$$

Therefore, the proposed algorithm returns a solution with max flow time

$$\begin{aligned} F_{\max} &= s_f + p_f - r_f \\ &\leq p_{\max} + r_f + F^p_{\max} - \frac{p_{\max} + p_f}{m} + p_f - r_f \\ &\leq 2p_{\max}(1 - \frac{1}{m}) + F^p_{\max} \\ &\leq (3 - \frac{2}{m})F^p_{\max} \\ &\leq (3 - \frac{2}{m})F^*_{\max}. \end{aligned}$$

□

Finally, we observe that the approximation ratio of the proposed algorithm is asymptotically tight. Indeed, this follows by observing that the approximation ratio of FIFO is asymptotically tight when processing times are fixed[1].

Theorem 2 *The approximation ratio of FIFO is asymptotically tight.*

Proof. We consider the following family of instances (see Fig. 1 for an example with three machines). For any number of machines m, jobs are released at time $r(k) = km$, for $k = 0, 1, \ldots, l$ where l is a non-negative integer.

At time $r(k)$, for $k = 0, 1, 2, \ldots, l-1$, a set $J_{1(k)}, \ldots, J_{2m(k)}$ of $2m$ jobs are released. Let $p_{j(k)}$ denote the processing time of the j-th job $J_{j(k)}$ released at time $r(k)$. The processing times are as follows

- $p_{1(k)} = \begin{cases} 0, & \text{if } k = 0; \\ p_{1(k-1)} + p_{2(k-1)}, & \text{otherwise;} \end{cases}$
- $p_{2(k)} = p_{3(k)} = \ldots = p_{m+1(k)} = 1 - p_{1(k)}/m;$
- $p_{m+2(k)} = p_{m+3(k)} = \ldots = p_{2m(k)} = m.$

At time $r(l)$ we submit a set $J_{1(l)}, \ldots, J_{2(m+1)(l)}$ of $2(m+1)$ jobs with processing times as follows

- $p_{1(l)} = \begin{cases} 0, & \text{if } l = 0; \\ p_{1(l-1)} + p_{2(l-1)}, & \text{otherwise;} \end{cases}$
- $p_{2(l)} = p_{3(l)} = \ldots = p_{m+1(l)} = 1 - p_{1(l)}/m;$
- $p_{m+2(l)} = p_{m+3(l)} = \ldots = p_{2m+1(l)} = 1;$
- $p_{2(m+1)(l)} = m.$

It is easy to check that jobs released at time $r(k)$, for $k = 0, 1, 2, \ldots, l$, can be completed by time $r(k) + m$, therefore always before new jobs are released (if any). It follows that the optimal maximum flow time is $F^*_{\max} = m$.

[1] In [2] it is claimed that FIFO is a $(3 - 2/m)$ approximation algorithm and this bound is tight. However no proof is available in literature and we provide our own proof for completeness.

l=3	$p_{1(k)}$	$p_{2(k)}$	$p_{3(k)}$	$p_{4(k)}$	$p_{5(k)}$	$p_{6(k)}$	$p_{7(k)}$	$p_{8(k)}$
k=0	0	1	1	1	3	3	-	-
k=1	1	2/3	2/3	2/3	3	3	-	-
k=2	5/3	4/9	4/9	4/9	3	3	-	-
k=3	19/9	8/27	8/27	8/27	1	1	1	3

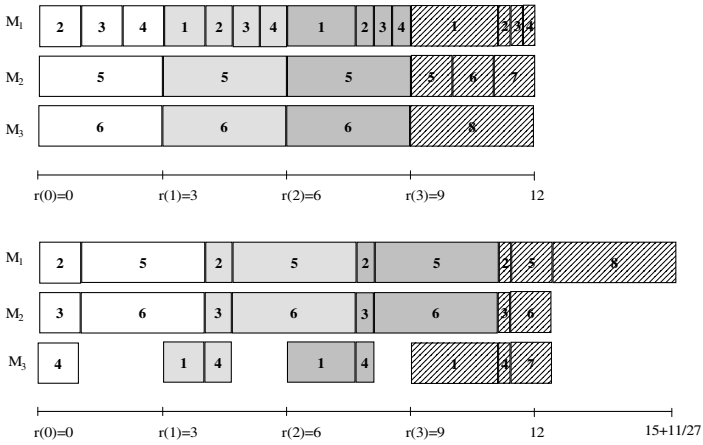

Fig. 1. An optimal schedule versus a possible FIFO schedule, for an instance with three machines and $l = 3$

Starting from $k = 0, 1, 2, \ldots, l-1$, assume, without loss of generality, that FIFO schedules jobs according to the order $J_{1(k)}, \ldots, J_{2m(k)}$. According to FIFO, let $C_i(k)$ be the time the last job released before time $r(k)$ completes on machine M_i. Since our instances are such that all jobs released before $r(k)$ can be completed by time $r(k)$ (according to some optimal off-line schedule), we define $H_i(k) = \max\{C_i(k) - r(k); 0\}$ as the *overflow* of machine M_i at time $r(k)$.

By induction, the reader should have no difficulty to check that for $k = 0, 1, 2, \ldots, l-1$, there are $m-1$ machines each with overflow

$$\begin{cases} H_i(0) = 0 \\ H_i(k+1) = 1 + H_i(k)(1 - \frac{1}{m}) \quad \text{for } k = 0, 1, 2, \ldots \end{cases}$$

Solving the previous recurrences we have that at time $r(k)$ there are $m-1$ machines with overflow

$$H_i(k) = m - m(1 - \frac{1}{m})^k.$$

At time $r(l)$ there are $m-1$ machines with overflow $H(l) = m - m(1 - \frac{1}{m})^l$. Moreover, observe that $p_{1(l)} = H(l)$ and all jobs but $J_{2(m+1)(l)}$ are completed at the same time, i.e. at time

$$r(l) + \frac{(m-1)(H(l)+m)}{m} = r(l) + (m-1)(2 - (1 - \frac{1}{m})^l).$$

It follows that $J_{2(m+1)(l)}$ completes at time $r(l) + (m-1)(2 - (1 - \frac{1}{m})^l) + m$ and its flow time, that is also the maximum flow time, is equal to $3m - 2 - (m-1)(1 - \frac{1}{m})^l$. Therefore the competitive ratio is

$$3 - 2/m - (1 - \frac{1}{m})^{l+1},$$

which converges to $3 - 2/m$ by increasing l. □

References

[1] Awerbuch, B., Azar, Y., Leonardi, S., Regev, O.: Minimizing the flow time without migration. In: Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC'99), pages 198–205, 1999.

[2] Bender, M. A., Chakrabarti, S., Muthukrishnan, S.: Flow and Stretch Metrics for Scheduling Continuous Job Streams. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98), pp. 270–279, 1998.

[3] Chen, B., Potts, C., Woeginger, G.: A review of machine scheduling: complexity, algorithms and approximability. Handbook of combinatorial optimization, vol. *3*, pp. 21–169, 1998.

[4] Chen, Z., Lu, Q., Tang, G.: Single machine scheduling with discretely controllable processing times. Operation Research Letters *21*, 69–76 (1997).

[5] Cheng, T., Shakhlevich, N.: Proportionate flow shop with controllable processing times. Journal of Scheduling *2*, 253–265 (1999).

[6] Garey, M. R., Johnson, D. S.: Computers and intractability; a guide to the theory of NP-completeness. W.H. Freeman 1979.

[7] Graham, R., Lawler, E., Lenstra, J., Kan, A. R.: Optimization and approximation in deterministic sequencing and scheduling: A survey. vol. 5, pp. 287–326. Amsterdam: North-Holland 1979.

[8] Hochbaum, D. (ed.): Approximation algorithms for NP-hard problems. Boston: PWS Publishing Company 1995.

[9] Hoogeveen, H., Woeginger, G. J.: Some comments on sequencing with controllable processing times. Computing *68*, 181–192 (2002).

[10] Jansen, K., Mastrolilli, M., Solis-Oba, R.: Job shop scheduling problems with controllable processing times. In: Proceedings of the 7th Italian Conference on Theoretical Computer Science, vol. LNCS 2202, pp. 107–122, 2001.

[11] Karger, D., Stein, C., Wein, J.: Scheduling algorithms. In: Atallah, M. J. (ed.): Handbook of algorithms and theory of computation. CRC Press 1997.

[12] Kellerer, H., Tautenhahn, T., Woeginger G. J.: Approximability and nonapproximability results for minimizing total flow time on a single machine. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC'96), pp. 418–426, 1996.

[13] Labetoulle, J., Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R.: Preemptive scheduling of uniform machines subject to release dates. In: Pulleyblank, W. R. (ed.): Progress in combinatorial optimization, pp. 245–261. Academic Press 1984.

[14] Lawler, E., Labetoulle, J.: On preemptive scheduling of unrelated parallel processors by linear programming. Journal of the ACM *25*, 612–619 (1978).

[15] Lenstra, J. K., Kan, A. H. G. R., Brucker, P.: Complexity of machine scheduling problems. Annals of Operations Research *1*, 343–362 (1977).

[16] Leonardi, S., Raz, D.: Approximating total flow time on parallel machines. In: Proc. 28th Annual ACM Symposium on the Theory of Computing (STOC'96), pp. 110–119, 1997.

[17] Mastrolilli, M.: A PTAS for the single machine scheduling problem with controllable processing times. In: Algorithm Theory – SWAT 2002, 8th Scandinavian Workshop on Algorithm Theory, vol. LNCS 2368, pp. 51–59, 2002.

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

[19] Nowicki, E.: An approximation algorithm for the m-machine permutation flow shop scheduling problem with controllable processing time. European Journal of Operational Research *70*, 342–349 (1993).

[20] Nowicki, E., Zdrzalka, S.: A survey of results for sequencing problems with controllable processing times. Discrete Applied Mathematics *26*, 271–287 (1990).
[21] Nowicki, E., Zdrzalka, S.: A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times. Discrete Applied Mathematics, *63* 237–256 (1995).
[22] Shmoys, D., Tardos, E.: An approximation algorithm for the generalized assignment problem. Mathematical Programming *62*, 461–474 (1993).
[23] Trick, M.: Scheduling multiple variable-speed machines. Operations Research *42*, 234–248 (1994).
[24] Vickson, R.: Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine. Operations Research *28*, 1155–1167 (1980).
[25] Vickson, R.: Two single machine sequencing problems involving controllable job processing times. AIIE Trans., *12* 258–262 (1980).

Monaldo Mastrolilli
IDSIA
Galleria 2
6928 Manno
Switzerland
e-mail: monaldo@idsia.ch

Verleger: Springer-Verlag KG, Sachsenplatz 4–6, A-1201 Wien. – Herausgeber: Prof. Dr. Wolfgang Hackbusch, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstraße 22–26, D-04103 Leipzig – Satz und Umbruch: Scientific Publishing Services (P) Ltd., Madras; Offsetdruck: Manz Crossmedia. 1051 Wien. – Verlagsort: Wien. – Herstellungsort: Wien. – Printed in Austria.