Discrete Applied Mathematics 159 (2011) 1367–1376

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

# A closed-form solution for the optimal release times for the $F2|$ deteriorating jobs $|\sum w_jC_j$ problem

Edouard Wagneur [a,*], Edwin Cheng [b], Daniel Ng [b]

[a] GERAD, and MAGI, École Polytechnique de Montréal, Montreal, Canada

[b] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

*Article history:*
Received 6 October 2008
Received in revised form 17 March 2011
Accepted 3 April 2011
Available online 24 May 2011

*Keywords:*
Deteriorating jobs
Weighted completion times
Optimal release times
Multiple-machine deterministic sequencing

ABSTRACT

We consider job scheduling on a flow-line production system, which covers a wide range of real-world manufacturing situations from plastic molding, steel milling to machine maintenance, and the service industry, where the duration of a task performed on a job is an arbitrary monotone non-decreasing function of the time the job has spent in the system. Our model is set in a deterministic environment with the initial conditions (i.e., job release times $r_j$) as decision variables (determined by the parameters $\gamma_1, \gamma_2, \ldots, \gamma_n$, which control the time elapsed since the first machine becomes available). The main feature of the problem to minimize the sum of weighted completion times – as compared to, say, the problem to minimize the makespan considered earlier (Wagneur and Sriskandarajah (1993) [23]) – is that its solution depends on the rate of growth of the processing time functions. We confine our study to the two-machine case for the sake of simplicity. We derive a closed-form formula for the optimal job release times for a finite set of jobs. This result also applies to the problem to minimize the flow time as a special case.



## 1. Introduction

The state dependent scheduling problems have received some attention in the literature after Callahan [6] and Buzacott and Callahan [5], who consider a tandem soaking pits-rolling mill in steel milling in a stochastic environment. In [4], queuing theory is applied for the determination of an efficient queue service policy for a service unit where the service time of a customer depends on the length of time he has waited. A few years later, Melnikov and Shafransky [18] devise an $O(n \log n)$ algorithm to solve the single-machine minimum makespan problem with affine processing time functions, i.e. $p_j = a_j + f(t)$, where $f$ is an arbitrary function. The case where $f$ is non-decreasing is studied by Gupta [12] for the scheduling of heating treatment of parts in a furnace. Here the state of a part corresponds to its temperature.

In [17], Kubiak and Van de Velde study the single-machine minimum makespan scheduling problem. They show that when the processing time of job $j$ grows by $w_j$ with each time unit its release is delayed beyond a common critical date $d$, then the scheduling problem is NP-hard. They devise a pseudopolynomial algorithm for this problem.

The two-machine case was considered in Sriskandarajah and Wagneur [21] for the minimum makespan problem. As soon as the first machine becomes available, the decision maker has to decide whether to release the next job, or to differ its release until the job will not have to wait between the machines (the no-wait case), or else to opt for a compromise between these extreme policies. The minimum makespan ($C_{\max}$) scheduling problem in the case where the processing time of a job is an affine function of the time elapsed since its release to the shop is shown to be NP-hard in the strong sense. The scheduling problem may be decomposed into two steps. In the first step, the release times of the jobs are optimized for any

* Corresponding author. Tel.: +1 5143406053x6056; fax: +1 5143405655.
*E-mail addresses:* Edouard.Wagneur@gerad.ca (E. Wagneur), Edwin.Cheng@inet.polyu.edu.hk (E. Cheng), Daniel.Ng@inet.polyu.edu.hk (D. Ng).

0166-218X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2011.04.003

given schedule. Then ordinary scheduling techniques (such as the well-known Johnson's rule [16]) can be used. A closed-form solution for the optimal release was proposed in [22] for the two-machine minimum makespan problem, when the job processing times (the penalty functions) are affine functions of the time that jobs have spent in the shop. This solution was generalized in [23] to the $m$-machines permutation flow shop with $n$ jobs and arbitrary non-decreasing processing penalty functions.

This problem has also been considered by Finke and Jiang (cf [9]). The approach has been extended to treat algorithmically the total completion time problem, i.e., $\sum C_j$ by Oulamara in [20]. This is generalized here to the minimum weighted total completion time ($\sum w_jC_j$) problem on the one hand, and to the formulation of a closed-form solution on the other hand. Also, we consider arbitrary non-decreasing penalty functions.

Variants of the original problem with various objective functions have been dealt with by many researchers, in particular, for single-machine problems. Gawiejnowicz and Pankowska [11] devise two polynomial time algorithms for the minimum makespan problem with linear increasing processing time functions of job starting times. Janiak and Bachman [13] show that the single-machine maximum lateness minimization problem with linear deteriorating jobs is NP-hard, then they prove in [2] the strong NP-hardness of the minimum makespan problem for two different models of job processing times. Bachman et al. [3] prove that the minimum weighted sum of completion times problem is NP-hard. Janiak and Kovalyov [14] show that the minimum makespan problem with arbitrary job release times is strongly NP-hard with strongly increasing functions, the maximum lateness minimization problem with decreasing functions is strongly NP-hard, and the total weighted completion time minimization problem is ordinary NP-hard.

Another approach whereby jobs start deteriorating simultaneously at the same time has also been considered in the literature. In this simpler approach, a job is released as soon as the first machine becomes available. In [19], Mosheiov devises polynomial algorithms for the two-machine flow shop and open shop, i.e., F2 and O2, respectively, to minimize makespan ($C_{\max}$), and the corresponding three-machine F3 and O3 problems are shown to be NP-hard. This type of deteriorating jobs is also considered by Chen [7] in an environment with parallel machines. The approach considered here is well-suited to maintenance operations (think for example of aircraft maintenance) since, for any given sequence of maintenance operations, the longer the maintenance is delayed, the longer it takes to perform it. As a consequence, maintenance will take more time on the next machine, which further delays the next maintenance operations. The $\sum C_j$ (or $\sum w_jC_j$) criterion will ensure a better utilization of the resources used for maintenance.

The interested reader will find a review and some extensions of such problems in [1,8]. Also, two chapters of the book by Janiak (Chapters 2 and 3 of [15]) are dedicated to scheduling problems related to deteriorating jobs, and the recent book by Gawiejnowicz [10] provide a more recent update of the literature on the topic.

This paper is organized as follows: in Section 2 we formulate the model, introduce the notation, and briefly recall the results of Wagneur and Sriskandarajah [23], which will be used in the sequel. We introduce two examples and compute the optimal release times for the $C_{\max}$ criterion, for both the wait and no-wait cases. Then in Section 3, we derive and prove the results for the $\sum w_jC_j$ problem. The examples in Section 2 are revisited, and the optimal sequences for the $\sum C_j$, $\sum w_jC_j$, $C_{\max}$, and no-wait cases are compared.

## 2. The shop model, notation and preliminary results

We recall here the model and the results developed in Wagneur and Sriskandarajah [23], to which we refer the reader for details. The notation, formulas, and propositions will be used in the sequel.

Let $J = \{1, \ldots, n\}$ and $K = \{1, 2\}$ stand for the sets of jobs and machines, respectively. Each job must be processed consecutively by the two machines. We restrict here to permutation schedules, with $1, \ldots, n$ the sequence of processing the jobs on both machines. We assume unlimited storage capacity between the machines.

The notation follows the main scheduling literature: the first subscript refers to a job, and the second to a machine. For instance, $C_{jk}$ stands for the completion time of job $j$ on machine $k = 1, 2$, and $r_j$ is a decision variable, which stands for the release time of job $j$. The start time is assumed to be 0, this is the date when all the jobs are ready. Since withholding the first job would induce unnecessary delay, this is also the release time of the first job and job $j$ may be released at any time $r_j \geq 0$.

We write $x_{j,k}$ for the time elapsed between job $j$'s release time $r_j$ and its starting time on machine $k$. Then $g_{j,k}(x_{j,k})$ stands for the processing time function of job $j$ on machine $k$. We assume that for $j = 1, \ldots, n$, $k = 1, 2$, $g_{j,k}$ is continuous and non-decreasing.

Our assumptions are summarized in the following:

- A1. The release of the jobs can be delayed at no cost.
- A2. Storage between the machines is unlimited.
- A3. The processing sequence is the same on all the machines.
- A4. For $j = 1, \ldots, n$, $k = 1, 2$, $g_{j,k}$ is a non-decreasing continuous function of $x_{jk}$ with a constant growth rate.

We have:

$r_{j+1} \geq r_j + g_{j,1}(x_{j1})$, $j = 1, \ldots, n-1$, and since jobs never wait for machine 1:

$x_{j,1} = 0$, $\forall j \in J$. Set $t_{j,1} = g_{j,1}(0)$, $j = 1, \ldots, n$.

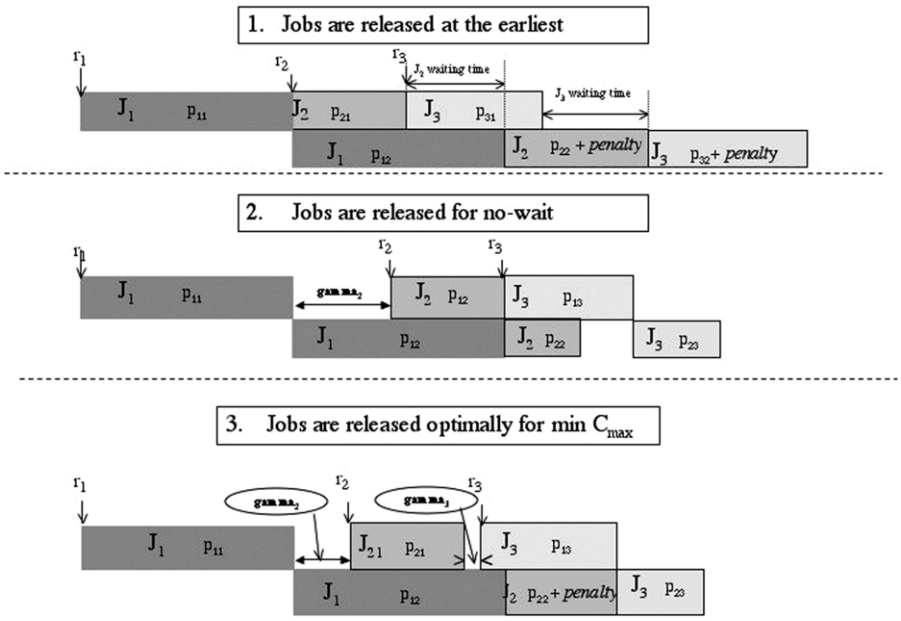


**Fig. 1.** Policies for the release of jobs to the shop.

The decision variable $\gamma_j$ associated with job $j$ consists in withholding this job after machine 1 becomes available. Our control variable $\gamma_{j+1}$ is then defined by:

$$r_{j+1} = r_j + t_{j,1} + \gamma_{j+1}, \quad j = 1, \ldots, n-1.$$

The problem consists in choosing the decision variable $\gamma_{j+1}$ optimally in the interval $[0, \max\{0, g_{j,2}(x_{j,2}) - t_{j+1,1}\}]$, whose right bound depends on the choice of $\gamma_j$.

Since machine 1 remains idle for this period of time, $\gamma_{j+1}$ also corresponds to the idle time of machine 1 between the processing of jobs $j$ and $j+1$, and

$$r_{j+1} = r_j + t_{j,1} + \gamma_{j+1} = \sum_{i=1}^{j} (t_{i,1} + \gamma_{i+1}), \quad j = 1, \ldots, n-1.$$

Since job 1 never waits ($\delta_1 = 0$), the release of job 1 is never delayed, i.e., we have $\gamma_1 = 0$ for the optimal policy.

Let $\delta_j$ stand for the waiting time of job $j$ between the two machines.

The dynamics of the system is governed by the following recursive formulas:

$$x_{j,1} = 0, \qquad C_{j,1} = r_j + t_{j,1}, \quad j \in J, \qquad C_{1,2} = x_{1,2} + g_{1,2}(x_{1,2}). \tag{2.1}$$

$$\delta_j = \max\{0, C_{j-1,2} - C_{j,1}\}, \quad j = 2, \ldots, n. \tag{2.2}$$

$$x_{j,2} = t_{j,1} + \delta_j, \quad j = 2, \ldots, n. \tag{2.3}$$

$$C_j = C_{j,2} = r_j + x_{j,2} + g_{j,2}(x_{j,2}), \quad j = 2, \ldots, n. \tag{2.4}$$

The maps $\gamma_{(j)} = (\gamma_1, \ldots, \gamma_j) \mapsto x_{j,k}$ are well defined for all $j = 2, \ldots, n$, $k = 1, 2$, and so are the maps $(\gamma_1, \ldots, \gamma_j) \mapsto C_j$. Hence, for any given regular performance criterion $\mathbb{R}^n \xrightarrow{P} \mathbb{R}^+$, $\gamma \mapsto P(\gamma) = P(C_1(\gamma), \ldots, C_n(\gamma))$, we associate with it the following control problem:

*Control problem*

For any sequence $\sigma \in S_n$, find a control vector $\gamma(\sigma) = (\gamma_2(\sigma), \ldots, \gamma_n(\sigma)) \in \mathbb{R}^{n-1}$ that optimizes the criterion $P$.

We first study the optimal control problem for the no-wait case, i.e., the control $\gamma^o$ associated with $\delta_j = 0, j = 1, \ldots, n$, which will be the basis of a renormalization of the functions and variables.

The three policies for the release of jobs, namely "earliest release", "no-wait", and "optimal control", are illustrated in Fig. 1.

### 2.1. The no-wait case

The no-wait case is characterized by: $\delta_j = 0, \forall j \in J$, with $\gamma^o$ being the optimal control vector corresponding to this case. Since by (2.3), we have $\forall j \in J, x_{j,2}(\gamma_j^o) = t_{j,1}$, we let $t_{j,2} = g_{j,2}(t_{j,1}), j \in J$.

Note that $t_{j,2}$'s are independent of the schedule. We call them *intrinsic* processing times, and since a job may be released alone to the shop, we assume that they are known for every job.

The results on this case (except for Proposition 3) for the two-machine flow shop for any given (deterministic) sequence can be found in [22]. We have:

**Proposition 1.** *For any regular performance criterion, the optimal no-wait control is given by:*

$$\gamma_j^o = \max\{0, t_{j-1,2} - t_{j,1}\}, \quad j = 2, \ldots, n. \quad \square \tag{2.1.1}$$

By Proposition 1, and [21], $C_{\max}(\gamma^o)$ and $\sum w_j C_j(\gamma^o)$ correspond to the classical $F_2|\text{no wait}|C_{\max}$ and $F_2|\text{no wait}|\sum w_j C_j$ solutions, respectively.

**Proposition 2.** *We have*

$$\begin{aligned} C_{\max}(\gamma^o) = C_n(\gamma^o) &= \sum_{j=1}^{n} t_{j,1} + \sum_{j=2}^{n} \gamma_j^o + t_{n,2} \\ &= t_{1,1} + t_{n,2} + \sum_{j=2}^{n} \max\{t_{j,1}, t_{j-1,2}\}. \quad \square \end{aligned} \tag{2.1.2}$$

Let $W_j = \sum_{i=j}^{n} w_j$.

By Proposition 2, we have:

$$C_j(\gamma^o) = t_{1,1} + t_{j,2} + \sum_{i=2}^{j} (\max\{t_{i,1}, t_{i-1,2}\}), \quad j = 2, \ldots, n;$$

hence

$$\begin{aligned} \sum_{j=1}^{n} w_j C_j(\gamma^o) &= w_1(t_{1,1} + t_{1,2}) + \sum_{j=2}^{n} w_j \left( t_{1,1} + t_{j,2} + \sum_{i=2}^{j} \max\{t_{i,1}, t_{i-1,2}\} \right) \\ &= \sum_{j=1}^{n} W_j t_{j,1} + \sum_{j=1}^{n} w_j t_{j,2} + \sum_{j=2}^{n} W_j \max\{t_{j,1}, t_{j-1,2}\}. \end{aligned}$$

So we have the following statement:

**Proposition 3.** *For the weighted sum of completion times, we have:*

$$\sum_{j=1}^{n} w_j C_j(\gamma^o) = W_1 t_{1,1} + \sum_{j=1}^{n} w_j t_{j,2} + \sum_{j=2}^{n} W_j \max\{t_{j,1}, t_{j-1,2}\}. \tag{2.1.3}$$

### *2.2. Processing times as functions of waiting times*

The no-wait solution is not optimal in general because it creates machine idle times, which could be used efficiently for the (increased) processing times eventually arising from earlier release times. In fact, the optimal $\gamma_j$'s are such that all the available idle time on machine 2 is completely utilized. This idea, which was explained in detail in [22] for the problem **F2|deteriorating jobs**, **linear functions**$|C_{\max}$, and in [23] for the problem

**Fm|*deteriorating jobs continuous non − decreasing functions*|$C_{\max}$,**

was used in [9] and in [20] for the design of an algorithm for an optimal placement of the jobs for the $C_{\max}$ and $\sum C_j$ problems, respectively. Our closed-form formulas for the $\sum w_j C_j$ problem yield the optimal solution more efficiently.

Note that waiting time $\delta_j$ induces an increase in state $x_{j,2}$ and in processing time $g_{j,2}(x_{j,2})$. This will increase $C_j$, and therefore, eventually also the waiting time $\delta_{j+1}$ of the next job, and so on. Thus, in the general case, there should be a trade-off between machine idle times and job waiting times.

The processing time functions $f_j(\cdot)$ depend on the waiting times $\delta_j$:

$$f_j(\delta_j) = g_{j,2}(x_{j,2}) - t_{j,2}, \quad j = 2, \ldots, n \tag{2.2.1}$$

$\forall j \geq 2, f_j: \mathbb{R}^+ \to \mathbb{R}^+$ is a non-decreasing continuous function satisfying $f_j(0) = 0$.

The extended penalty functions $F_j()$ are then defined by:

$$F_j(\delta_j) = \delta_j + f_j(\delta_j), \quad j = 2, \ldots, n. \tag{2.2.2}$$

The completion times are given by:

$$C_j = \sum_{i=1}^{j} t_{i,1} + \sum_{i=2}^{j} \gamma_i + t_{j,2} + F_j(\delta_j), \quad j = 1, \ldots, n, \tag{2.2.3}$$

and the differences $C_{j-1} - C_{j,1}$ by:

$$t_{j-1,2} - t_{j,1} + F_{j-1}(\delta_{j-1}) - \gamma_j, \quad j = 2, \ldots, n. \tag{2.2.4}$$

Then (2.2) becomes:

$$\delta_j = \max\left\{0, t_{j-1,2} - t_{j,1} + F_{j-1}(\delta_{j-1}) - \gamma_j\right\}, \quad j = 2, \ldots, n. \tag{2.2.5}$$

For all $j$ $f_j$ is a monotone non-decreasing functions of $\delta_j$, and $F_j(x) = x + f_j(x) \Rightarrow \frac{F(x+h)-F(x)}{h} = 1 + \frac{f(x+h)-f(x)}{h} \geq 1$ (where $h$ is small). Hence $F(\delta_j)$ is an increasing function of $\delta_j$, which increases at least as fast as the identity mapping $\imath$, thus it is a decreasing function of $\gamma_j$, which decreases at least as fast as $-\imath$. Also, for $a, b \geq 0$, we have $F_j(\max\{a, b\}) = \max\{F_j(a), F_j(b)\}$, and similarly for the min operator. Moreover, its inverse $F_j^{-1}$ exists on $\mathbb{R}^+$, and since $F_j(0) = 0$ and $F_j^{-1}(0) = 0$, it may be extended to the whole of $\mathbb{R}$ by:

$$F_j^{-1}(x) = F_j^{-1}(\max\{0, x\}) \quad \forall x \in \mathbb{R},\ j = 2, \ldots, n. \tag{2.2.6}$$

We have the following key statement:

**Lemma 2.2.7** (*Wagneur and Sriskandarajah [23]*). *Let* $G(x) = \max\{0, \omega + F(\max\{0, u - x\})\}$, *with* $F(0) = 0$, *and* $F^{-1}(x) = 0$, *for all* $x \leq 0$, *then* $\arg\min_{x\geq 0}\big(x + G(x)\big) = \arg\min_{x\geq 0} G(x), x^* = \arg\min_{x\geq 0} G(x) = \max\{0, u - F^{-1}(-\omega)\}$, *and* $G(x^*) = \max\{0, \omega\}$.

### 2.3. The minimum makespan problem with two machines

We briefly recall and adapt to the case $m = 2$ the results of Wagneur and Sriskandarajah [23] for the makespan problem. The optimal control $\gamma_j^C$ for this problem will then be shown to be a lower bound for the optimal control $\gamma_j^*$, for the $\sum w_j C_j$ (Proposition 5). They are necessary for the computation in the next Section 2.4. Moreover, the development in this subsection will help the reader to follow the main idea of the proof of Theorem 1 in Section 3. For $j = 2, \ldots, n$, we write $t_{j-1,2} - t_{j,1} = \alpha_j$.

The problem with two machines may be written as:

$$\min_{\gamma\geq 0} C_n = \sum_{j=1}^{n} t_{j,1} + t_{n,2} + \min_{\gamma\geq 0}\left(\sum_{j=2}^{n} \gamma_i + F_n(\delta_n)\right). \tag{2.3.1}$$

Up to constant terms, problem (2.3.1) may be re-written as:

$$\min_{\gamma_2\geq 0}\left(\gamma_2 + \min_{\gamma_3\geq 0}\big(\gamma_3 + (\cdots + \min_{\gamma_n\geq 0}(\gamma_n + F_n(\delta_n))\cdots)\big)\right). \tag{2.3.2}$$

(2.3.2) is solved by backward induction, starting with $\min_{\gamma_n\geq 0}(\gamma_n + F_n(\delta_n))$. It is easy to see (cf. [23]) that all the steps of the backward induction for (2.3.2) yield a function $\gamma_j$ satisfying the conditions of the map $G$ in the Lemma 2.2.7. The sequence of $\beta_j$'s is defined inductively by:

$$\beta_n = \alpha_n, \qquad \beta_j = \alpha_j - F_{j+1}^{-1}(-\beta_{j+1}). \tag{2.3.3}$$

We have the following statement.

**Proposition 4** (*[23]*). *The optimal control for the* $C_{\max}$ *problem is given by:*

$$\gamma_j^C = \max\{0, \beta_j\}, \quad j = 2, \ldots, n. \quad \square \tag{2.3.4}$$

We refer the reader to [23] for the generalization of Proposition 4 to $m$ machines and for interpretation of the optimal control $\gamma_j^C$.

We show below how the approach can be generalized to the case of minimizing $\sum w_j C_j$. However, as will be seen, we will have to take into account the growth rate of the functions $w_j F_j$ and $w_j F_j + w_{j+1} F_{j+1} \circ F_j$.

In the next section we compute the optimal controls $\gamma_j^o$ and $\gamma_j^C$ for two examples. The computation will then be compared with the optimal $\gamma_j^*$ for the $\sum w_j C_j$ problem, with the same examples, and some choices of the weights for the completion times.

### 2.4. Examples

We write $H_2(\gamma)$ for the variable cost of $\sum w_jC_j$.

#### 2.4.1

There are five jobs, with intrinsic processing times given by: $\begin{pmatrix}1&4&3&4&5\\6&4&1&2&3\end{pmatrix}$. The functions $f_j$ are all the same and are given by a linear function: $f(x) = 2x$. Then $F(x) = (1+a)x = 3x$, and $b = 3$, $b^{-1} = \frac{1}{3}$. Also let $w_1 = 2$, $w_2 = 5$, $w_3 = 2$, $w_4 = 1$, $w_5 = 3$. Then $W_2 = 11$, $W_3 = 6$, $W_4 = 4$, $W_5 = 3$, and $\beta_5^o = -3$, $\beta_4^o = -3$, $\beta_3^o = 1$, $\beta_2^o = 2$.

The no-wait policy is given by: $\gamma_5^o = 0$, $\gamma_4^o = 0$, $\gamma_3^o = 1$, $\gamma_2^o = 2$, and $H_2(\gamma^o) = 28$.

With the $C_{\max}$ policy, we have:

$\gamma_5 = \max\{0, \alpha_5+F_4(\delta_4)\} = \max\{0, -3+F_4(\delta_4)\} \Rightarrow F_4(\delta_4) = 3$, and $\gamma_5^C = 0$, $\beta_4^1 = -3$, and $\gamma_4 = \max\{0, \beta_4^1+F_3(\delta_3)\} = \max\{0, -3+F_3(\delta_3)\} \Rightarrow F_3(\delta_3) = 3$, and $\gamma_4^C = 0$, $\beta_3^2 = -\frac{1}{3}$, $\gamma_3 = \max\{0, \beta_3^2+F_2(\delta_2)\} = \max\{0, -\frac{1}{3}+F_2(\delta_2)\} \Rightarrow F_2(\delta_2) = \frac{1}{3}$, and $\gamma_3^C = 0$, $\beta_2^3 = \frac{17}{9}$, and $\gamma_2 = \max\{0, \beta_2^3\} = \frac{17}{9}$, with $\delta_2 = 0$. Then, with this policy, we have $H_2(\gamma^C) = 31\frac{4}{9}$.

#### 2.4.2

Let the processing times be given by $\begin{pmatrix}2&2&4&4&5\\6&3&3&1&3\end{pmatrix}$, with the $f_j$ linear functions given by $a_2 = a_3 = \frac{1}{4}$, $a_4 = \frac{1}{3}$, $a_5 = \frac{1}{2}$. Also, let $w_1 = w_2 = w_3 = 1$, $w_4 = 12$, $w_5 = 1$. We have: $\beta_5^o = -4$, $\beta_4^o = -1$, $\beta_3^o = -1$, $\beta_2^o = 4$, and $W_2 = 15$, $W_3 = 14$, $W_4 = 13$.

The no-wait policy yields $\gamma_5^o = \gamma_4^o = \gamma_3^o = 0$, and $\gamma_2^o = 4$, with $H_2(\gamma^o) = 60$.

With the $C_{\max}$ policy, we get $\gamma_5^C = 0$, with $\delta_5 = 0$, $\gamma_4^C = 0$, with $F_4(\delta_4) = -\beta_5^o = 4$, $\gamma_3^C = 0$, with $F_3(\delta_3) = -\beta_4^1 = 4$, and $\gamma_2^o = \max\{0, \beta_2^3\} = \frac{16}{25}$, and $F_2(\delta_2) = -\beta_3^2 = 4\frac{1}{5}$, with $H_2(\gamma^C) = 65\frac{4}{5}$.

## 3. The $\sum w_jC_j$ problem with two machines

In this section, the rates of growth $\rho(F_j)$, or $\rho(w_jF_j)$ or $\rho(w_jF_j + w_{j+1}F_{j+1} \circ F_j)$ are always meant to be in the interval $[0, \delta_j(\gamma_j^C)]$. Indeed, we must have $\gamma_j^* \le \gamma_j^o$, since $\gamma_j^* > \gamma_j^o$ would yield completion time $C_j > C_j(\gamma_j^o)$. On the other hand, the optimal control vector $\gamma^C$ will usually fail to be optimal for the $\sum w_jC_j$ problem, since it is designed to minimize $C_n$ only.

The following proposition states that the optimal control $\gamma_j^*$ lies between the optimal control for makespan and the no-wait control.

**Proposition 5.** *We have* $\gamma_j^* \in [\gamma_j^C, \gamma_j^o]$ $j = 1, \ldots, n$.

**Proof.** We just need to show that $\gamma_j^C \le \gamma_j^*$ for all $j$. We have:

$$\sum_{j=1}^{n} w_jC_j = w_1C_1 + \sum_{j=2}^{n} w_j \sum_{i=1}^{j} t_{i,1} + \sum_{j=2}^{n} w_jt_{j,2} + \sum_{j=2}^{n} w_j \left( \sum_{i=2}^{j} \gamma_j + F_j(\delta_j) \right) = D + H_2(\gamma_2, \ldots, \gamma_n),$$

where $D$ stands for the constant part of the criterion, and for $k = n, \ldots, 2$, $H_k$ is defined by $H_k(\gamma_k, \ldots, \gamma_n) = \sum_{j=k}^{n}(W_j\gamma_j + w_jF_j(\delta_j))$.

Clearly $\arg\min \sum w_jC_j = \arg\min H_2(\cdot)$ and $\min_{\gamma_2 \ge 0, \ldots, \gamma_n \ge 0} H_2(\cdot)$ splits into the subprograms:

$$\min_{\gamma_2 \ge 0} \Big( W_2\gamma_2 + w_2F_2(\delta_2) + \cdots + \min_{\gamma_n \ge 0}(w_n[\gamma_n + F_n(\delta_n)]) \ldots \Big),$$

which may be written as:

$$\min_{\gamma_2 \ge 0} \Big( W_2\gamma_2 + w_2F_2(\delta_2) + \cdots + \min_{\gamma_j \ge 0, \ldots, \gamma_n \ge 0} H_j(\gamma_j, \ldots, \gamma_n) \Big). \tag{3.1}$$

Note that, up to the multiplicative constant $W_n$, the first step of the backward induction program (3.1):

$\min_{\gamma_n \ge 0} W_n[\gamma_n + F_n(\delta_n)]$ is the same as the first step of the program for $C_{\max}$. We get: $\gamma_n = \max\{0, \alpha_n + F_{n-1}(\delta_{n-1})\}$, $\delta_n = 0$, and the next step will be: $\min_{\gamma_n - 1 \ge 0} H_{n-1}(\gamma_{n-1}, \gamma_n)$, i.e.,

$$\min_{\gamma_{n-1} \ge 0} \big( W_{n-1}\gamma_{n-1} + w_{n-1}F_{n-1}(\delta_{n-1}) + W_n \max\{0, \alpha_n + F_{n-1}(\delta_{n-1})\} \big). \tag{3.1.1}$$

In step $n-1$ for makespan, the program reads $\min_{\gamma_{n-1} \ge 0}(\gamma_{n-1} + \max\{0, \alpha_n + F_{n-1}(\delta_{n-1})\})$. Write $\min_{\gamma_{n-1} \ge 0} \Gamma_{n-1}$ for this program. Now step $H_{n-1}(\gamma_{n-1}, \gamma_n)$, as given by (3.1.1), may be written as: $\min_{\gamma_{n-1} \ge 0} \big( w_{n-1}(\gamma_{n-1} + F_{n-1}(\delta_{n-1})) + W_n(\gamma_{n-1} + \max\{0, \alpha_n + F_{n-1}(\delta_{n-1})\}) \big)$, or $\min_{\gamma_n - 1 \ge 0} H_{n-1}(\gamma_{n-1}, \gamma_n) = \min_{\gamma_n - 1 \ge 0} \big( w_{n-1}(\gamma_{n-1} + F_{n-1}(\delta_{n-1}) + W_n\Gamma_{n-1}) \big)$.

Since $\Gamma_{n-1}$ and $W_n\Gamma_{n-1}$ reach their minima simultaneously and $w_{n-1}(\gamma_{n-1}+F_{n-1}(\delta_{n-1}))$ is a decreasing function of $\gamma_{n-1}$, we have; $\gamma^C_{n-1} \le \gamma^*_{n-1}$.

The (backward) induction step is similar, and we leave it to the reader to work out the details. □

We introduce the following notation: $\beta_j^{-1} = 0$, $\beta_j^k = \alpha_j - F_j^{-1}(-\beta_{j+1}^{k-1})$, and $\gamma_j^{[k]} = \max\{0, \beta_j^k + F_{j-1}(\delta_{j-1})\}, j = 2, \ldots, n, k = 0, \ldots, n-j$.

We also use the following conditions:

$[A_j^k]$: $\beta_j^k \ge 0$,
$[B_j]$: $\rho(w_jF_j) \ge W_j$, and
$[C_j]$: $\rho(w_jF_j + w_{j+1}F_{j+1} \circ F_j) \ge W_j$.

Note that $C_{n-1}$ holds, since $\rho(w_{n-1}F_{n-1} + w_nF_n \circ F_{n-1}) \ge w_{n-1} + w_n = W_{n-1}$.

We have the following statement:

**Theorem 1.** *For $j = 2, \ldots, n-1$, and $0 \le k \le n-j$, we have:*

$$\gamma_{j+1} = \gamma_{j+1}^{[k]} \Rightarrow$$

$$\gamma_j = \begin{cases} \gamma_j^{[0]} & A_{j+1}^k \vee B_j \vee \left[A_{j+1}^o \wedge C_j\right] \\ \gamma_j^{[1]} & \neg A_{j+1}^k \wedge \neg A_{j+1}^0 \wedge \neg B_j \wedge C_j \\ \gamma_j^{[k+1]} & [j < n-1] \wedge \neg A_{j+1}^k \wedge \neg B_j \wedge \neg C_j, \end{cases} \quad \text{and}$$

$$\gamma^*_{j+1} = \max\{0, \beta^k_{j+1}\}.$$

We first show that in the statement of the theorem all the cases are considered.

We have $\neg\left[A_{j+1}^k \vee B_j \vee \left[A_{j+1}^o \wedge C_j\right]\right] = \neg\left[\left[A_{j+1}^k \vee B_j \vee A_{j+1}^o\right] \wedge \left[A_{j+1}^k \vee B_j \vee C_j\right]\right] = \left[\neg A_{j+1}^k \wedge \neg B_j \wedge \neg A_{j+1}^o\right] \vee \left[\neg A_{j+1}^k \wedge \neg B_j \wedge \neg C_j\right]$.

**Proof.** We prove the statement by backward induction on $n$.

**INITIAL STEP**: $j = n$

We have $\gamma_n = \max\{\beta_n^o + F_{n-1}(\delta_{n-1})\} = \gamma_n^{[0]}$, and $\delta_n = 0$.

Although we need only to prove the induction step, we also prove the case $n-1$, since it will help the reader to better follow the proof of the induction step.

**SUPPLEMENT STEP for $j = n-1$**

3.1.1.1 $\boxed{A_n^o}$

$$\begin{aligned} &\min_{\gamma_{n-1}\ge 0} (W_{n-1}\gamma_{n-1} + w_{n-1}F_{n-1}(\delta_{n-1}) + w_n \max\{0, \beta_n^o + F_{n-1}(\delta_{n-1})\}) \\ &\quad = \min_{\gamma_{n-1}\ge 0} (W_{n-1}\gamma_{n-1} + w_{n-1}F_{n-1}(\delta_{n-1}) + w_n[\beta_n^o + F_{n-1}(\delta_{n-1})]) \\ &\quad = \min_{\gamma_{n-1}\ge 0} (W_{n-1}[\gamma_{n-1} + F_{n-1}(\delta_{n-1})] + w_n\beta_n^o). \end{aligned}$$

As a straightforward application of 2.2.7 with $w = 0, F = F_n, u = \beta_{n-1}^o + F_{n-2}(\delta_{n-2})$, and $x = \gamma_{n-1}$, we get $\gamma_{n-1} = \max\{0, \beta_{n-1}^o + F_{n-2}(\delta_{n-2})\}$, together with $\gamma_n^* = \beta_n^o$.

3.1.1.2 $\boxed{\neg A_n^o}$

Clearly $H_{n-1}(\gamma_{n-1}, \gamma_n)$ is decreasing until $\max\{0, \beta_n^o + F_{n-1}(\delta_{n-1})\}$ reaches its minimum, which occurs when $\beta_n^o + F_{n-1}(\delta_{n-1}) \le 0$, i.e. $F_{n-1}(\delta_{n-1}) \le -\beta_n^o$, or $\delta_{n-1} = \max\{0, \beta_{n-1}^o + F_{n-2}(\delta_{n-2}) - \gamma_{n-1}\} \le F_{n-1}^{-1}(-\beta_n^o) \Rightarrow \beta_{n-1}^o - F_{n-1}^{-1}(\beta_n^o) + F_{n-2}(\delta_{n-2}) \le \gamma_{n-1} \Leftrightarrow \beta_{n-1}^1 + F_{n-2}(\delta_{n-2}) \le \gamma_{n-1} \Rightarrow \begin{cases} \gamma_{n-1} \ge \max\{0, \beta_{n-1}^1 + F_{n-2}(\delta_{n-2})\} \\ \text{and} \\ \gamma_n^* = 0. \end{cases}$

It follows in particular (from the case $\neg A_n^o$) that $\gamma_n^* = \max\{0, \beta_n^o\}$ and (3.1.1) becomes:

$$\min_{\gamma_{n-1}\ge 0} (W_{n-1}\gamma_{n-1} + w_{n-1}F_{n-1}(\delta_{n-1}))\,.$$

$\boxed{B_{n-1}}$ ($w_{n-1}F_{n-1}$ decreases faster than $-W_{n-1}\iota$). In this case,

$$\gamma_{n-1} = \max\{0, \beta_{n-1}^o + F_{n-2}(\delta_{n-2})\}, \quad \text{with } \delta_{n-1} = 0.$$

$\boxed{\neg B_{n-1}}$. Then $H_{n-1}$ is increasing for $\gamma_{n-1} \geq \max\{0, \beta^1_{n-1} + F_{n-2}(\delta_{n-2})\}$, hence $\gamma_{n-1} = \gamma^{[1]}_{n-1}$, and

$$\begin{aligned}\delta_{n-1} &= \max\{0, \beta^o_{n-1} + F_{n-2}(\delta_{n-2}) - \max\{0, \beta^1_{n-1} + F_{n-2}(\delta_{n-2})\}\}\\ &= \max\{0, F^{-1}_{n-1}(-\beta^o_n) + \min\{0, \beta^1_{n-1} + F_{n-2}(\delta_{n-2})\}\}.\end{aligned}$$

**BACKWARD INDUCTION STEP**: $j+1 \mapsto j$

Assume that the statement holds for every integer larger than $j$ and let $k$ be arbitrary ($0 \leq k \leq n-j-1$).
By the induction hypothesis, $\gamma^*_{j+2}, \ldots, \gamma^*_n$ are constant, hence the induction step in (3.1.1) becomes:

$$\min_{\gamma_j \geq 0}\left(W_j\gamma_j + w_jF_j(\delta_j) + W_{j+1}\gamma_{j+1} + w_{j+1}F_{j+1}(\delta_{j+1})\right).$$

Since $\gamma_{j+1} = \max\{0, \beta^k_{j+1} + F_j(\delta_j)\}$, by assumption, we have:

$$\begin{aligned}\delta_{j+1} &= \max\{0, \beta^o_{j+1} + F_j(\delta_j) - \max\{0, \beta^k_{j+1} + F_j(\delta_j)\}\}\\ &= \max\{0, \beta^o_{j+1} + F_j(\delta_j) + \min\{0, -\beta^k_{j+1} - F_j(\delta_j)\}\}\\ &= \max\{0, F^{-1}_{j+1}(-\beta^{k-1}_{j+2}) + \min\{0, \beta^k_{j+1} + F_j(\delta_j)\}\},\end{aligned}$$

and the program becomes:

$$\min_{\gamma_j \geq 0}(W_j\gamma_j + w_jF_j(\delta_j)) + W_{j+1}\max\{0, \beta^k_{j+1} + F_j(\delta_j)\} + w_{j+1}F_{j+1}(\max\{0, F^{-1}_{j+1}(-\beta^{k-1}_{j+2}) + \min\{0, \beta^k_{j+1} + F_j(\delta_j)\}\}). \quad (3.2)$$

3.2.1 $\boxed{A^k_{j+1}}$ The program becomes:
$\min_{\gamma_j \geq 0}\left(W_j(\gamma_j + F_j(\delta_j))\right) + W_{j+1}\beta^k_{j+1} + w_{j+1}\max\{0, -\beta^{k-1}_{j+2}\}$.
Thus $\gamma_j = \max\{0, \beta^o_j + F_{j-1}(\delta_{j-1})\} = \gamma^{[0]}_j$, $\delta_j = 0$, and $\gamma^*_{j+1} = \beta^k_{j+1}$.

3.2.2 $\boxed{\neg A^k_{j+1}}$ $H_j(\cdot)$ is decreasing until $\beta^k_{j+1} + F_j(\delta_j) = 0$ and in this case:
$F_j(\delta_j) = -\beta^k_{j+1} \Rightarrow \delta_j = \max\{0, \beta^o_j + F_{j-1}(\delta_{j-1}) - \gamma_j\} = F^{-1}_j(-\beta^k_{j+1})$.
Hence $\gamma_j \geq \max\{0, \beta^o_j + F_{j-1}(\delta_{j-1}) - F^{-1}_j(-\beta^k_{j+1})\} = \max\{0, \beta^{k+1}_j + F_{j-1}(\delta_{j-1})\}$.

Now, for $\gamma_j \geq \max\{0, \beta^{k+1}_j + F_{j-1}(\delta_{j-1})\}$, $\delta_{j+1} = \max\{0, \beta^o_{j+1} + F_j(\delta_j)\}$ and $\gamma^*_{j+1} = 0$. Hence $\gamma^*_{j+1} = \max\{0, \beta^k_{j+1}\}$, which proves this part of the induction step. The program $\min H_j(\cdot)$ may now be stated as:

$$\min_{\gamma_j \geq 0}\left(W_j\gamma_j + w_jF_j(\delta_j) + w_{j+1}F_{j+1}(\max\{0, \beta^o_{j+1} + F_j(\delta_j)\})\right).$$

3.2.2.1 $\boxed{B_j}$ $H_j(\cdot)$ is decreasing and reaches its minimum for $\gamma_j = \max\{0, \beta^o_j + F_{j-1}(\delta_{j-1})\} = \gamma^{[0]}_j$. Then $\delta_j = 0$.

3.2.2.2 $\boxed{\neg B_j}$

$\boxed{C_j}$

$\boxed{A^o_{j+1}}$ $H_j(\cdot)$ is decreasing and reaches its minimum for $\gamma_j = \max\{0, \beta^o_j + F_{j-1}(\delta_{j-1})\} = \gamma^{[0]}_j$. Then $\delta_j = 0$.

$\boxed{\neg A^o_{j+1}}$ $H_j(\cdot)$ is decreasing until $\gamma_j = \beta^o_{j+1} + F_j(\delta_j) = 0$, and increasing afterward. Hence $\gamma_j = \max\{0, \beta^1_j + F_j(\delta_j)\} = \gamma^{[1]}_j$.

$\boxed{\neg C_j}$ $H_j(\cdot)$ is increasing for $\gamma_j \geq \max\{\beta^{k+1}_j + F_{j-1}(\delta_{j-1})\}$. Hence $\gamma_j = \gamma^{[k+1]}_j$.

**Summing up**: $\gamma_j = \gamma^{[0]}_j$ when: either $A^k_{j+1}$; or $\neg A^k_{j+1}$ and (either $B_j$; or [$\neg B_j$ and $C_j$ and $A^o_{j+1}$]).

Formally, the logical conditions on the r.h.s may be written as:

$$A^k_{j+1} \vee \left\{\left(\neg A^k_{j+1}\right) \wedge \left(B_j \vee \left[\neg B_j \wedge C_j \wedge A^o_{j+1}\right]\right)\right\}.$$

But $\forall X$ and $Y$, we have: $X \vee \left[\neg X \wedge Y\right] = X \vee Y$.

Thus:

$$A^k_{j+1} \vee \left\{\neg A^k_{j+1} \wedge \left(B_j \vee \left[\neg B_j \wedge C_j \wedge A^o_{j+1}\right]\right)\right\} = A^k_{j+1} \vee \left(B_j \vee \left[C_j \wedge A^o_{j+1}\right]\right),$$

i.e., $\gamma_j = \gamma^{[0]}_j$ in case $A^k_{j+1} \vee \left(B_j \vee \left[C_j \wedge A^o_{j+1}\right]\right)$ holds.

**Similarly**: $\gamma_j = \gamma_j^{[1]}$ when $\neg A_{j+1}^k \wedge \neg B_j \wedge C_j \wedge \neg A_{j+1}^o$ holds,
**and** $\gamma_j = \gamma_j^{[k+1]}$ in case $\neg A_{j+1}^k \wedge \neg B_j \wedge \neg C_j$ holds. □

### 3.3. Remark

Notice that $\gamma_{j+1}$ is completely determined after we solve the program for $\gamma_j$. This interesting property was mentioned in [22]. This property does not hold for $\delta_j$, except $\delta_{j-1} = 0$. The generic case is a sequence of embedded $\delta_k$'s:

$$\begin{aligned}\delta_{j+1} &= \max\{0, F_{j+1}^{-1}(-\beta_{j+2}^o + \min\{0, \beta_{j+1}^1 + F_j(\delta_j)\})\}\\ &= \max\{0, F_{j+1}^{-1}(-\beta_{j+2}^o + \min\{0, \beta_{j+1}^1 + F_j(\max\{0, F_j^{-1}(-\beta_{j+1}^o + \min\{0, \beta_j^1 + F_{j-1}(\delta_{j-1})\})\})\})\} = \cdots.\end{aligned}$$

This is illustrated in Example 3.4.1, where even if $\gamma_4$ is known, $\delta_4$ cannot be determined before we know that $\gamma_3 = \gamma_3^{[o]}$, hence $\delta_3 = 0$.
Finally, as a straightforward application of Theorem 1, we have the following statement:

**Theorem 2.** *$\gamma_j^* = \max\{0, \beta_j^{k_j}\}, j = 2, \ldots, n$, where $k_n = 0$, and for $j < n$, $k_j$ is given by the sequence $\gamma_n^{[o]}, \gamma_{n-1}^{[k_{n-1}]}, \ldots, \gamma_{j+1}^{[k_{j+1}]}$, $(0 \le k_j \le n - j)$, $\delta_2 = \max\{0, \min\{\beta_2^o, F_2^{-1}(-\beta_3^{k_2-1})\}\}$, and, for $j = 3, \ldots, n-1$, $\delta_j = \max\{0, \min\{\beta_j^o, F_j^{-1}(-\beta_{j+1}^{k_j-1})\} + F_{j-1}(\delta_{j-1})\}$.* □

In Section 2, we recalled the solution to the minimum makespan problem: $\gamma_j^C = \max\{0, \beta_j^{n-j}\}$.

This means that, for every $j$, we compute how much machine 2 idle time will be available for the next jobs. The optimal release times aim at using all machine 2 idle time that will be available while the next jobs are processed. This eventually reduces makespan at the expense of $C_j$, i.e., $C_j$ increases, while $C_n$ decreases. It follows that this policy will generally fail to be optimal for the $\sum w_j C_j$ criterion.

Note that the computation of $\gamma_j^C$ requires the computation of $\gamma_n^C, \gamma_{n-1}^C, \ldots, \gamma_{j+1}^C$.

The interpretation of the $\gamma_j^*$'s for the $\sum w_j C_j$ is similar to that of the $\gamma_j^C$'s, i.e., they seek to utilize as much as possible the machine 2 idle time.

Also, the computation of the $\gamma_j^*$'s proceeds by backward induction from that of $\gamma_n^*, \gamma_{n-1}^*, \ldots, \gamma_{j+1}^*$.

### 3.4. The examples revisited

#### 3.4.1. Example 2.4.1

First consider the weights: $w_1 = 2, w_2 = 5, w_3 = 2, w_4 = 1, w_5 = 3 \Rightarrow W_2 = 11, W_3 = 6, W_4 = 4, W_5 = 3$. Also $D = 140$.

For the optimal policy, we have $\gamma_5 = \gamma_5^{[o]}$, with $\gamma_5^* = \max\{0, \beta_5^o\} = 0$, then $\boxed{\neg A_5^0}$ holds, and $\neg B_4$ holds. Also, since $\boxed{C_{n-1}}$ always holds, $\boxed{C_4}$ holds, i.e., we have that $\neg A_5^0 \wedge \neg B_4 \wedge C_4$ (line 2 in the statement of Theorem 1). Hence $\gamma_4 = \gamma_4^{[1]}$, with $\gamma_4^* = 0$, and $\delta_4 = \max\{0, F_4^{-1}(-\beta_5^o) + \min\{0, \beta_4^1 + F_3(\delta_3)\}\}$.

Similarly, from line 1 in the statement of Theorem 1, since $\rho(w_3 F_3) = 6 = W_3$, $\boxed{B_3}$ holds, then we have $\gamma_3 = \gamma_3^{[o]}$, hence $\gamma_3^* = \max\{0, \beta_3^0\} = 1$, with $\delta_3 = 0$. Thus $\delta_4 = \max\{0, \beta_4^o\} = 0$. Now, since $\gamma_3 = \gamma_3^{[o]}$, and $\boxed{A_3^0}$ holds, we have $\gamma_2 = \gamma_2^{[o]}$, and $\gamma_2^* = \max\{0, \beta_2^o\} = 2$, with $\delta_2 = 0$.

Thus, the optimal policy is no-wait and $\sum w_j C_j = 28$.

#### 3.4.2

Consider the previous example with weights given by: $w_2 = w_3 = w_4 = 1$, and $w_5 = 10$. The optimal $C_{\max}$ and no-wait policies remain unchanged. On the other hand, we have: $\gamma_5 = \max\{0, \beta_5^o + F_4(\delta_4)\} = 0$ and

$\boxed{\neg A_5^o} \wedge \boxed{\neg B_4} \Rightarrow \gamma_4 = \gamma_4^{[1]} = \max\{0, \beta_4^1\} = 0$. Similarly,

$\boxed{\neg A_4^1} \wedge \boxed{\neg B_3} \wedge \boxed{\neg C_3} \Rightarrow \gamma_3 = \gamma_3^{[2]} = \max\{0, \beta_3^2\} = \max\{0, -\frac{1}{3}\} = 0$. Finally:

$\boxed{\neg A_3^3} \wedge \boxed{\neg B_3} \wedge \boxed{\neg C_2} \Rightarrow \gamma_2 = \gamma_2^{[3]} = \max\{0, \beta_2^3\} = \max\{0, \frac{17}{9}\} = \frac{17}{9}$.

Thus, the optimal policy is the $C_{\max}$ policy. This is no surprise since, in contrast to the weights in 3.4.1, the last job now has a relatively large weight, hence the optimal policy will tend to minimize $C_5$, which is precisely what the $C_{\max}$ policy does.

#### 3.4.3

When all the $w_j$'s are equal, the problem is to minimize the total completion time. The no-wait and $C_{\max}$ solutions remain the same, with $H_2(\gamma^o) = 11$ and $H_2(\gamma^C) = 13\frac{8}{9}$.

For the $\sum C_j$ problem, we have $\gamma_5^* = 0$ (and $\delta_5 = 0$), and $B_4$ holds. Then $\gamma_4 = \gamma_4^{[o]}$, hence $\gamma_4^* = 0$, and $\delta_4 = 0$. Since $C_3$ holds, we also have $\gamma_3^* = \gamma_3^{[o]}$, with $\gamma_3^* = 1$, and $\delta_3 = 0$. As above, we also have $\gamma_2 = \gamma_2^{[o]}$, hence $\gamma_2^* = 0$, and $\delta_2 = 0$. The optimal solution is (again) no-wait.

*3.4.4. Example 2.4.2*

Consider the weights: $w_1 = w_2 = w_3 = 1$, $w_4 = 12$, $w_5 = 1$.

For the $\sum w_jC_j$ policy, we have $\gamma_5^* = 0$ (and $\delta_5 = 0$), and $\boxed{C_4}$. Hence $\gamma_4^* = \gamma_4^{[o]}$, and $\delta_4 = 0$, $\gamma_4^* = 0$. Also $\boxed{C_3}$ $\Rightarrow \gamma_3^* = \gamma_3^{[o]}$, with $\delta_3 = 0$, and $\gamma_3^* = 0$. Clearly, we have $\boxed{\neg A_3^o} \wedge \boxed{\neg B_2} \wedge \boxed{\neg C_2}$. Hence $\gamma_2 = \max\{0, \beta_2^1\} = 3\frac{1}{5}$, and $F_2(\delta_2) = -\beta_3^o = 1$, and we have $H_2(\gamma^*) = 49$.

For this problem, and for the sequence given, we have $\gamma_2^C < \gamma_2^* < \gamma_2^o$, while $\gamma_j^C = \gamma_j^* = \gamma_j^o$ for $j = 3, 4, 5$.

Note also that (for instance) $\gamma_j^C = \gamma_j^o \not\Rightarrow \delta_j^C = \delta_j^o$ (as would be the case if $\gamma_j^C = \gamma_j^o$, $j = 2, \ldots, n$).

## Acknowledgments

We are grateful to the anonymous referees for their helpful comments on earlier versions of our paper. They have greatly helped to improve the quality of our paper and update our references. This research was supported in part by the NRC grant RGPIN-143068-05 and by The Hong Kong Polytechnic University under grant number A628 from the *Area of Strategic Development in China Business Services*. This research was carried out while the first author was visiting The Hong Kong Polytechnic University.

## References

[1] B. Alidae, N.K. Womer, Scheduling with time dependent processing times: review and extensions, Journal of the Operational Research Society 50 (1999) 711–720.
[2] A. Bachman, A. Janiak, Scheduling jobs with position-dependent processing times, Journal of the Operational Research Society 55 (2004) 257–264.
[3] A. Bachman, A. Janiak, M.Y. Kovalyov, Minimizing the total weighted completion time of deteriorating jobs, Information Processing Letters 81 (2002) 81–84.
[4] J.A. Buzacott, The effects of queue discipline on the capacity of queues with service time dependent on waiting times, Infor 12 (2) (1974) 174–185.
[5] J.A. Buzacott, J.R. Callahan, The capacity of the soaking pit-rolling mill complex in steel production, Infor 9 (2) (1971) 87–95.
[6] J.R. Callahan, The nothing hot delay problem in the production of steel, Thesis, Department of Industrial Engineering, University of Toronto, 1971.
[7] Z.L. Chen, Parallel machine scheduling with time dependent processing times, Discrete Applied Mathematics 70 (1995) 81–93.
[8] T.C.E. Cheng, Q. Ding, B.M.T. Lin, A concise survey of scheduling with time-dependent processing times, European Journal of Operational Research 152 (2002) 1–13.
[9] G. Finke, H. Jiang, A variant of the permutation flow-shop model with variable processing times, Discrete Applied Mathematics 76 (1997) 123–140.
[10] S. Gawiejnowicz, Time Dependent Scheduling, Springer Verlag, 2008.
[11] S. Gawiejnowicz, L. Pankowska, Scheduling with varying processing times, Information Processing Letters 54 (3) (1995) 175–178.
[12] J.N.D. Gupta, S.K. Gupta, Single facility scheduling with non-linear processing times, Computers and Industrial Engineering 14 (1988) 387–393.
[13] A. Janiak, A. Bachman, Minimizing maximum lateness under linear deterioration, European Journal of Operational Research 126 (3) (2002) 557–566.
[14] A. Janiak, M.Y. Kovalyov, Scheduling deteriorating jobs, in: A. Janiak (Ed.), Scheduling in Computer and Manufacturing Systems, WKL, Warszawa, 2006.
[15] A. Janiak, M.Y. Kovalyov, Job sequencing with exponential functions of processing times, Informatica 17 (1) (2006) 13–24.
[16] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Research Logistics Quarterly 1 (1954) 61–68.
[17] W. Kubiak, S. Van de Velde, Minimizing makespan of deteriorating jobs on a single machine, Naval Research Logistics Quarterly 45 (1998) 511–523.
[18] O.I. Melnikov, Y.M. Shafransky, Paremetric problem of scheduling theory, Cybernetics 15 (1980) 352–357.
[19] G. Mosheiov, Complexity analysis of job-shop scheduling with deteriorating jobs, Discrete Applied Mathematics 117 (2002) 195–209.
[20] A. Oulamara, Flowshop avec détérioration des tâches et groupement des tâches. Thesis, Université Joseph Fourier, Grenoble, 2001.
[21] C. Sriskandarajah, E. Wagneur, Hierarchical control of the two-processor flow-shop with state dependent processing times: complexity analysis and approximate algorithms, Infor 29 (3) (1991) 193–205.
[22] E. Wagneur, C. Sriskandarajah, The 2-machine permutation flow-shop with state-dependent processing times, Naval Research Logistics Quarterly 40 (1993) 697–711.
[23] E. Wagneur, C. Sriskandarajah, Optimal control of a class of discrete event systems, Journal of Dynamic Discrete Event Systems 3 (1993) 397–425.