Scheduling and Controlling Work-in-Process : An on Line Study for Shop Problems

Fabrice Chauvet, Jean-Marie Proth

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Scheduling and Controlling Work-In-Process : An On-line Study for Shop Problems

Fabrice Chauvet - Jean-Marie Proth

N ° 3950
Mai 2000

_____ THEME 4 _____

Ordonnancement et Contrôle des En-cours : Une Etude des Problèmes d'Atelier.

Fabrice CHAUVET
BOUYGUES TELECOM R&D
Europa, 30 avenue d l'Europe, 78944 Vélizy Cedex, France
fchauvet@bouyguestelecom.fr

Jean-Marie PROTH
INRIA
4 rue Marconi, 57070 Metz Technopôle, France
Jean-Marie.Proth@loria.fr

Résumé

Dans ce papier, nous nous intéressons aux systèmes de production qui ont deux caractéristiques : les temps opératoires peuvent être choisis entre deux limites données, et une opération doit débuter dès que l'opération précédente (s'il en existe une) se termine (problème sans atttente).

Plusieurs algorithmes en ligne sont proposés pou minimiser les temps de fin de fabrication dans les systèmes en ligne (flow shops) et les ateliers (jobs shops). Nous démontrons que, dans le pire des cas, l'ordonnancement obtenu avec ces algorithmes dure m fois plus que l'ordonnancement optimal, m étant le nombre de machines. Nous donnons plusieurs autres résultats des ratios qui évaluent la compétitivité du système.

Mots clés

Ordonnancement en ligne, problèmes san attente, problèmes à temps contrôlables.

Scheduling and controlling Work-In-Process: An On-line Study for Shop Problems

Fabrice CHAUVET
BOUYGUES TELECOM R&D
Europa, 30 avenue d l'Europe, 78944 Vélizy Cedex, France
fchauvet@bouyguestelecom.fr

Jean-Marie PROTH
INRIA
4 rue Marconi, 57070 Metz Technopôle, France
Jean-Marie.Proth@loria.fr

Abstract

In this paper, we address the problem of production systems having two characteristics. First, the manufacturing times can be chosen between given bounds. Such a production system is said to have controllable processing times. Second, an operation must start as soon as the previous operation on the same part (if any) is completed. A production system having this characteristic is said to be no-wait.
Several on-line schedules are considered to minimize the makespan in flow shop and job shop situations. We prove that in the worst case, the makespan provided by these schedules is *m* times longer than the optimal one (for different flow shops and job shops), *m* being the number of machines. We give several related results on competitive ratio.

Keywords

On-line scheduling, no-wait problems, controllable processing times.

1. Introduction

In this paper, we study an approach that aims at scheduling parts and controlling WIP. Hence, w are interested in the two following characteristics:

First, the processing time $P_{k,j}$ of an operation k of job j can be selected in a given time interval $[l_{k,j}; u_{k,j}]$, with $0 < l_{k,j} < +\bullet$ and $l_{k,j} \bullet u_{k,j} \bullet +\bullet$. The processing times are said to b *controllable*: processing times are decision variables.

Second, an operation starts as soon as the previous operation on the part (if any) ends: the syste considered in this paper is of *no-wait* type.

These two characteristics are apparently restrictive. In fact, they are of utmost importance to control many industrial systems, as showed hereafter.

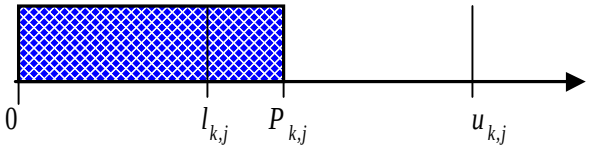

Figure 1- The duration $l_{k,j}$ is uncompressible while the rest of the processing time ($u_{k,j}$ - $l_{k,j}$) is not.

A number of papers [5,6] have dealt with the *no-wait scheduling problems* (processing times being fixed, i.e. $l_{k,j} = u_{k,j}$). A no-wait situation occurs either because of a lack of intermediate storag capacity, or because of the processing technology itself. For instance, no-wait systems exist in the hot metal rolling industry, since any interruption would preclude the maintenance of high operating temperatures.

Others papers considered the no-wait scheduling problems when the processing times are controllable and unlimited (i.e. $u_{k,j} = +\bullet$). They are called *blocking problems* [7]. Typically, it arises when we store products on machines or when we consider specific storage locations as resources. In such situations a product which has been completed may remain on the machine (or the resource) until a downstream machine becomes available. For instance, in airplane assembly lines, a plane can b moved from an area to another only if this area is empty, otherwise it would stand and "block" the machine preventing another job from being processed in the engaged place.

In others situations, it is allowed to store products on the machine during a limited time (i.e. $l_{k,j} \bullet u_{k,j} < +\bullet$). It is the case when scheduling a robot in an electroplating line [1]. The robot is used to move a part from a chemical mixture to another. In this situation, chemical tanks are the resources, th "machines" of the system. Since the product is not allowed to stay outside a bath, the robot must transport the product without delay. But products can stay for a short additional time (which is bounded) in a bath even if the chemical operation is performed. This introduces some flexibility in th scheduling process.

Generally, *blocking problems* and *no-wait problems* are considered separately. Introducing the constraint on *controllability* of processing times, we are able to study them in an unique model. This is really interesting since they have common properties as it is shown by this study. Furthermore, this model offers new possibilities. For example, if we consider each storage place as a machine with a processing time included in $]0; \upsilon]$, where $0 < \upsilon < +\bullet$, then it is possible to control WIP (Work-In-Process) through parameter υ.

Some of the propositions hereafter hold in different situations. To give the most general account, we present the different cases even if there are not of th *no-wait* type.

2. Notations and complexity results

In this paper, we focus on *flow sho*p and *job shop* problems having the two characteristics introduced in the previous section. In these two problems, we consider that n jobs have to be processed. Job j consists of $K(j)$ operations performed in series on m different machines, with $1 \bullet m \bullet K(j)$. $K = \max_{1 \le j \le n} \{K(j)\}$ is defined as the maximal number of operations performed on a job. While in the flow shop all the jobs go through each of th m machines in the same order ($m = K$), in the job shop each job has its own process routing and a job use the same machine several times, which

means that we may hav $m < K$. We denote by $M(k,j)$ the machine on which k-th operation has to b processed, and j the related job. Thus a flow shop is a special case of a job shop in which $K(j) = m$ and $M(k,j) = k$, for $1 \bullet k \bullet K(j)$ and $1 \bullet j \bullet n$.

The time a job j is available to be processed is the release date denoted by r_j. Without loss of generality, we assume that $0 = r_1$ and $r_j \bullet r_{j+1}$ for any $1 \bullet j < n$. If several requirements arriv simultaneously in the system, they received the same release date.

In traditional shop problems, a job is allowed to wait between two operations and all th processing times are fixed. We denote by $Fm||C_{\max}$ and $Jm||C_{\max}$ the problems of minimizing th makespan (i.e. the time $C_{\max}$ required to complete all the jobs) in respectively a flow shop and a job shop. When release dates exist, a flow shop problem is denoted by $Fm|r_j|C_{\max}$. We specify that a problem at hand is of no-wait type (when a job is not allowed to be stored between two consecutive operations) adding the notation "no-wait" or " nwt". We also denote a controllable problem using th notation "$l_{k,j} \bullet P_{k,j} \bullet u_{k,j}$" which means that the processing tim $P_{k,j}$ of the operation k to be performed on job j is not fixed and can be selected in the time interval $[l_{k,j}; u_{k,j}]$ for $1 \bullet k \bullet K(j)$ and $1 \bullet j \bullet n$. In what follows, we assume that $0 < l_{k,j} < +\bullet$ and $l_{k,j} \bullet u_{k,j} \bullet +\bullet$.

We use a capital letter (instead of conventional p) for the processing time since it is variable. W use a capital letter (instead of conventional p) for the processing time since it is variable. We use a capital letter (instead of conventional p) for the processing time to emphasize that it is variable.

We denote by $B_{k,j}$ and $C_{k,j}$ respectively the starting time and the completion time of the operation k on job j, for $1 \bullet j \bullet n$. We have $r_j \bullet B_{1,j}$ and $B_{k,j} + P_{k,j} = C_{k,j}$ and a set of values $\{B_{k,j}; C_{k,j}\}_{1 \bullet k \bullet K(j),\, 1 \bullet j \bullet n}$ defines a solution to a scheduling problem. In a no-wait environment, we hav $C_{k+1,j} = B_{k,j}$.

Proposition 1

F3|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$, F2|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$ and J2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$ are NP-complete in the strong sense (as well as and J2|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$).

Proof: The problems F3|no-wait|$C_{\max}$, F2|no-wait, $r_{jj}|C_{\max}$ and J2|no-wait|$C_{\max}$ are both NP-complete in the strong sense (see respectively [9], [10] and [11]). Thus, the proposition holds sinc these problems are special cases (i.e. $l_{k,j} = u_{k,j}$) of respectively F3|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$, F2|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$ and J2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$. □

Thus, in our context, there does not exist (unless P=NP) any polynomial time algorithm to solv optimally a flow shop or a job shop problem for a number of machines larger than 2. For a number of machines equal to 2, Gilmore and Gomory [5] proposed a polynomial algorithm which gives the optimal solution to F2|no-wait $C_{\max}$.

Proposition 2

The solution obtained by applying the Gilmore-Gomory algorithm on F2|no-wait, $l_{k,j} = P_{k,j}|C_{\max}$ is optimal to F2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$.

The problem F2|no-wait, $l_{k,j} = P_{k,j}|C_{\max}$ denotes the problem F2|no-wait|$C_{\max}$ for which any processing tim $P_{k,j}$ is fixed to the value $l_{k,j}$.

Proof: Let $\{B^*_{k,j}; C^*_{k,j}\}_{1 \bullet k \bullet K(j),\, 1 \bullet j \bullet n}$ b the optimal solution to F2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$. From this solution, we derive another solution $\{\underline{B}^*_{k,j}; \underline{C}^*_{k,j}\}_{1 \bullet k \bullet K(j),\, 1 \bullet j \bullet n}$ defined as follows, for $1 \bullet j \bullet n$:

$\underline{B}^*_{2,j} = \underline{C}^*_{1,j} = C^*_{1,j}$,

$\underline{B}^*_{1,j} = \underline{C}^*_{1,j} - l_{1,j}$, and

$\underline{C}^*_{2,j} = \underline{B}^*_{2,j} + l_{2,j}$.

Since $\{B^*_{k,j}; C^*_{k,j}\}_{k,j}$ is feasible to F2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$, the new solution $\{\underline{B}^*_{k,j}; \underline{C}^*_{k,j}\}_{k,j}$ is also feasible to this problem. Furthermore, since for any job j, $C^{**}_{2,j} = B^{**}_{2,j} + l_{2,j} = C^*_{1,j} + l_{2,j} \bullet C^*_{2,j}$, the makespan of this new solution is not worst than th makespan of the initial solution. As a consequence, the solution $\{\underline{B}^*_{k,j}; \underline{C}^*_{k,j}\}_{1 \bullet k \bullet K(j),\, 1 \bullet j \bullet n}$ is optimal to F2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$.

It follows that there always exists an optimal solution to F2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$ such that the processing time $P_{k,j}$ of any operation is equal to the lower processing time $l_{k,j}$. Since the Gilmore-Gomory algorithm is able to reach an optimal solution when the processing times are fixed, this algorithm provides the optimal solution to F2|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$. □

3. An On-line Context

In what follows, we consider that customer' requirements stream in the system. Requirement j enters the system at tim r_j and should be scheduled as soon as possible. Indeed, we cannot process job j before time r_j. In practice, this is used to propose to clients a reachable deadline. Due to the intensity of the production flow, we are not able to reschedule previous requirements (which have been already scheduled). So, we aim at providing algorithms to schedule on-line each new requirement. Note that we do not know what would be the future requirements.

We use competitive analysis which is a worst case analysis used to evaluate the productivity of on-line algorithms. The competitive analysis is a comparison between the performance of an on-lin algorithm and the optimal offline algorithm. Formally, an algorith H is sai α-competitive for a problem P if, for any instanc I of P, we have $C_H(I) \bullet \alpha C^*(I)$, where $C_H(I)$ and $C^*(I)$ are the values of the criterion of P for respectively the solution obtained by applying the algorithm and the optimal solution. It is a worst case analysis. Of course, we have $1 \bullet \alpha$ and our goal is to find the algorith having lowest possible competition ratio $C_H(I) / C^*(I)$, for any instance I.

Few approximation algorithms can be used on-line, since most of them assume to know the whole information before computing the solution.

Unfortunately, for F$m||C_{\max}$ and J$m||C_{\max}$ (with $2 \bullet m$), it turns out that no algorithm is better than 2-competitive [3]. This result is very strong and also holds when we have some knowledge on what would be the future requirements It is also true in the situations we are interested in when we have no knowledge on future requirements:

Proposition 3

It does not exist a 2-competitive algorithm (when we have no knowledge on what would be the future requirements) for problem Fm|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$, J$m$|no-wait, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$, F$m$|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$ and Jm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$ (with $2 \bullet m$).

Proof: The claim is proved for F2|no-wait|$C_{\max}$ and as special cas can be extended to the different problems. Assume that there exists a λ-competitive algorithm denoted by H for a flow shop problem F2|no-wait|$C_{\max}$ requiring 2 machines, such that $1 \bullet \lambda < 2$.

Consider the following instance I_0 consisting of two jobs.

The two operations to be performed on job 1 require respectively processing times $P_{1,1} = 1$ on the first machine and $P_{2,1} = \delta$ on the second one. We choose δ such that $0 < \delta < (2 - \lambda) / 3$. Any λ-competitive algorithm would schedule job 1 at a time, say x, greater than the release date $r_1 = 0$.

Consider a second job on which two operations have to be performed after tim $r_2 = r_1$. The processing times for job 2 ar $P_{1,2} = x + \delta$ and $P_{2,2} = 1$. This second job can not be scheduled before job 1 since its processing time on the first machine is too long: $P_{1,2} = x + \delta > x$. Thus, the makespan, $C_H(I_0)$, provides by our algorith H would be at least greater than $2(x + 1) + \delta$ as on Figure 2.

For this instance, the optimal makespan is obtained by scheduling job 2 before job 1. This optimal makespan is $C^*(I_0) = (x + 1) + 2\delta$. As a consequence, we have $C_H(I_0) / C^*(I_0) > \lambda$, since $(2 - \lambda)(x + 1) > \delta(2\lambda - 1)$. This contradicts the assumption that it exists λ-competitive algorithm for a flow shop problem F2|no-wait|$C_{\max}$, such that $1 \bullet \lambda < 2$.

□

Note that the proof proposed in [3] for F$m||C_{\max}$ and J$m||C_{\max}$ does not hold in "no-wait" situation. Moreover, the previous proof is much more simple than the one in [3].

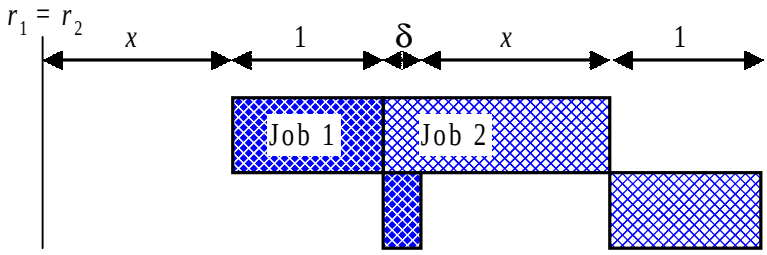

Figure 2- It does not exist any 2-competitive algorithm for this schedule.

For $m = 2$ machines, designing a 2-competitive algorithm to these problems is simple:

Schedule 1

For any new job j to be scheduled, we book the m machines (so that none of the machines can be used for another job) from the earliest date they are wholly available to the completion of the job. In other words, $B_{k,j}$ and $C_{k,j}$ being respectively the starting and the completion time of operation k on job j, job j starts at tim $B_{1,j} = \begin{cases} r_j \text{ if } j = 1 \\ \max\{r_j ; C_{K(j),j-1}\} \text{elsewhere} \end{cases}$ and is completed at $C_{K(j),j} = B_{1,j} + \sum_{k=1}^{K(j)} l_{k,j}$.

Since two jobs cannot use the same machine simultaneously, Schedule 1 provides a feasible solution (for any flow shop or job shop problems). This algorithm required O(K) operations to schedule a job (for any shop problems). Thus the complexity of Schedule 1 for n jobs is in O($K.n$).

Schedule 1 is no used in practice; but its study is simple and offers a basis from which we can extend and compare properties to others schedules.

Proposition 4

Schedule 1 is m-competitive for problems F$m|r_j|C_{\max}$, J$m|r_j|C_{\max}$, F$m|r_j, P_{k,j}=1|C_{\max}$, J$m|r_j, P_{k,j}=1|C_{\max}$, F$m$|no-wait, r_j, $l_{k,j}$• $P_{k,j}$• $u_{k,j}|C_{\max}$ and Jm|no-wait, r_j, $l_{k,j}$• $P_{k,j}$• $u_{k,j}|C_{\max}$ (even for the situation in which all the jobs are identical).

Proof: There is enough to prove the upper bound on the competitive ratio for the most general problems that are J$m|r_j|C_{\max}$ and Jm|no-wait, r_j, $l_{k,j}$• $P_{k,j}$• $u_{k,j}|C_{\max}$, whereas this bound is reached for the least general case F$m|P_{k,j} = 1|C_{\max}$.

Using this Schedule 1, job j start at tim $B_{1,j} = \max\{r_j; C_{K(j),j-1}\}$ and is completed at tim $C_{K(j),j} = B_{1,j} + \sum_{k=1}^{K(j)} l_{k,j} = \max\left\{r_j + \sum_{k=1}^{K(j)} l_{k,j-1}; C_{K(j),j-1} + \sum_{k=1}^{K(j)} l_{k,j-1}\right\}$. Therefore, the makespan obtained by applying by Schedule 1 denoted by $C_{\text{Schedule 1}}(I)$ on the n jobs of an instance I is equal to $C_{\text{Schedule 1}}(I) = \max_{j=1,\ldots,n}\left\{r_j + \sum_{h=j}^{n}\sum_{k=1}^{K(h)} l_{k,h}\right\}$. Since a machine can not perform two operations simultaneously in shop problems, the optimal makespan denoted C (I) is longer than $\max_{i=1,\ldots,m}\left\{\max_{j=1,\ldots,n}\left\{r_j + \sum_{\substack{h=j \\ \text{such that} \\ M(k,h)=i}}^{n}\sum_{k=1}^{K(h)} l_{k,h}\right\}\right\} = \max_{j=1,\ldots,n}\left\{r_j + \max_{i=1,\ldots,m}\left\{\sum_{\substack{h=j \\ \text{such that} \\ M(k,j')=i}}^{n}\sum_{k=1}^{K(h)} l_{k,h}\right\}\right\}$. Thus, whatever the instance, the makespan provided by Schedule 1 is not worst than m times the optimal one as:

$$m\,C^*(I) \bullet m \max_{j=1,\ldots,n}\left\{r_j + \max_{i=1,\ldots,m}\left\{\sum_{h=j}^{n}\sum_{\substack{k=1 \\ \text{such that } M(k,j')=i}}^{K(h)} l_{k,h}\right\}\right\} \bullet \max_{j=1,\ldots,n}\left\{r_j + m.\max_{i=1,\ldots,m}\left\{\sum_{h=j}^{n}\sum_{\substack{k=1 \\ \text{such that } M(k,j')=i}}^{K(h)} l_{k,h}\right\}\right\} \text{ and as}$$

$$m.\max_{i=1,\ldots,m}\left\{\sum_{h=j}^{n}\sum_{\substack{k=1 \\ \text{such that } M(k,j')=i}}^{K(h)} l_{k,h}\right\} \bullet \sum_{i=1}^{m}\sum_{h=j}^{n}\sum_{\substack{k=1 \\ \text{such that } M(k,h)=i}}^{K(h)} l_{k,h} \bullet \sum_{h=j}^{n}\sum_{k=1}^{K(h)} l_{k,h}\text{, for any } 1 \bullet j \bullet n.$$

Consider the case of a flow shop $Fm|P_{k,j} = 1|C_{\max}$ such that all the processing times are equal to 1 and that all the release dates are equal to 0 ($m = K$). In this case all the jobs are identical. For this instance I', the optimal makespan is $C^*(I') = n + m - 1$ while th makespan provided by Schedule 1 is $C_{\text{Schedule 1}}(I') = m\,n$. For any $\varepsilon > 0$, set $n > -m + 1 + (m^2 - m) / \varepsilon$. For such ε and m, we have $C^*(I')\,(m - \varepsilon) \bullet (n + m - 1)(m - \varepsilon) \bullet m\,n + (-\varepsilon\,n - \varepsilon\,m + \varepsilon + m^2 - m) < m\,n = C_{\text{Schedule 1}}(I')$. It turns out that Schedule 1 can provide a makespan which is m times longer than the optimal one for the flo shop $Fm|P_{k,j} = 1|C_{\max}$ and for the more general problems $Fm|r_j|C_{\max}$, $Jm|r_j|C_{\max}$, $Fm|r_j, P_{k,j}=1|C_{\max}$, $Jm|r_j, P_{k,j}=1|C_{\max}$, Fm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$ and Jm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{\max}$.

Thus, Schedule 1 is m-competitive for the specified shop problems and, therefore, optimal for th case $1 = m$. □

Actually, Schedule 1 is m-competitive when the time between the arrival of two consecutiv requirements does not exceed the time needed to perform any operation on each job (i.e. $r_{j+1} - r_j \bullet l_{k,j'}$ for any $1 \bullet k \bullet K(j)$, $1 \bullet j \bullet n$, and $1 \bullet j' \bullet n$). If the time between the arrival of two consecutive requirements is greater than the time needed to perform a job (i.e. $r_{j+1} - r_j \bullet \sum_{k=1}^{K(j)} l_{k,j}$ for any $1 \bullet j < n$), Schedule 1 is optimal and 1-competitive! Between these two utmost situations, many possible cases can happen depending on Δ_r, the minimum time between two releases dates (i.e. $r_{j+1} - r_j \bullet \Delta_r$ for any $1 \bullet j \bullet n$-1) and L, the maximum possible lower bound of a processing time (i.e. $l_{k,j} \bullet L$ for any $1 \bullet k \bullet K(j)$ and $1 \bullet j \bullet n$). For $m = 10$ machines and $L = 1$, Figure 3 gives the competitive ratio of Schedule 1 depending on Δ_r, the time between two consecutive jobs for different number ($K = 10$, 15, 20) of operations to be performed on each job.

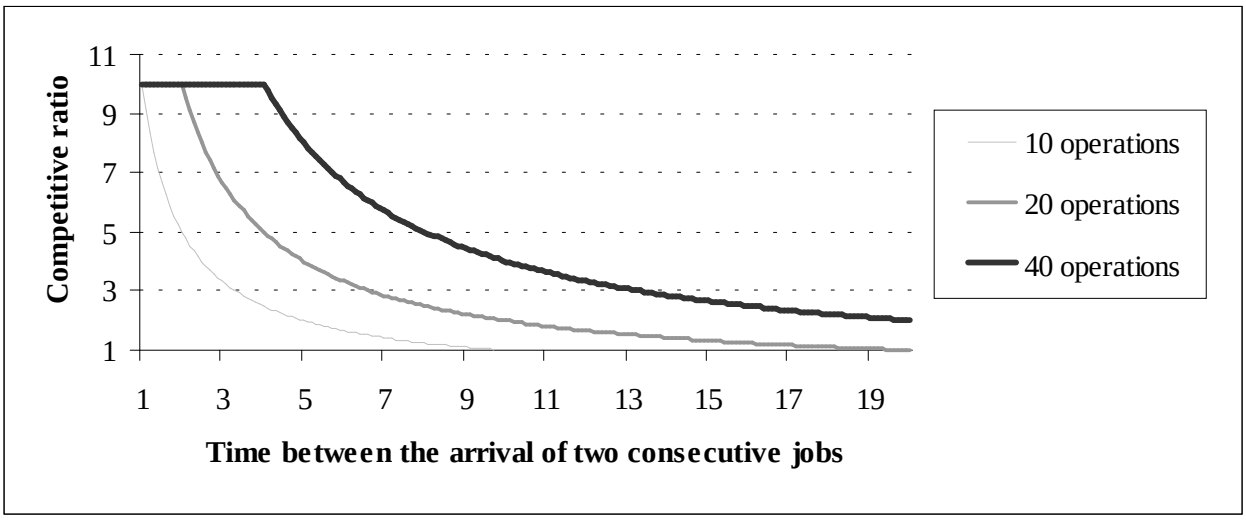

Figure 3- Representation of the competitive ratio of Schedule 1 equal to $\min\{m = 10; K.L / \Delta_r\}$.

Proposition 5

Whatever $K.L / m \cdot \Delta_r \cdot K.L$, Schedule 1 is $(K.L / \Delta_r)$-competitive for problem $Fm|r_j|C_{\max}$, $Jm|r_j|C_{\max}$, $Fm|r_j, P_{k,j}=1|C_{\max}$, $Jm|r_j, P_{k,j}=1|C_{\max}$, $Fm|$no-wait, r_j, $l_{k,j} \cdot P_{k,j} \cdot u_{k,j}|C_{\max}$ and $Jm|$no-wait, r_j, $l_{k,j} \cdot P_{k,j} \cdot u_{k,j}|C_{\max}$.

Proof: There is enough to prove the upper bound on the competitive ratio for the most general problems that are $Jm|r_j|C_{\max}$ and $Jm|$no-wait, r_j, $l_{k,j} \cdot P_{k,j} \cdot u_{k,j}|C_{\max}$, whereas this bound is reached for the least general cases $Jm|P_{k,j} = 1, r_j|C_{\max}$ and $Fm|P_{k,j} = 1, r_j|C_{\max}$.

Since each job j in a flow shop or a job shop problem has to start after tim r_j, the optimal makespan is longer than $\max_{j=1,\ldots,n}\left\{r_j + \sum_{k=1}^{K(j)} l_{k,j}\right\} \cdot r_n + \sum_{k=1}^{K(n)} l_{k,n}$. Therefore, whatever the job j $(1 \cdot j \cdot n)$ of instance I, the optimal makespan $C^*(I)$ is such that: $C(I) \cdot r_n + \sum_{k=1}^{K(n)} l_{k,n} \cdot r_j + (n-j)\Delta_r + \sum_{k=1}^{K(n)} l_{k,n}$ and sinc $\Delta_r \cdot K.L$, $C^*(I) \cdot \frac{\Delta_r}{K.L}\left[r_j + (n-j)K.L + \sum_{k=1}^{K(n)} l_{k,n}\right] \cdot \frac{\Delta_r}{K.L}\left[r_j + \sum_{h=j}^{n-1}\sum_{k=1}^{K(h)} l_{k,h} + \sum_{k=1}^{K(n)} l_{k,n}\right]$. From these inequalities, we derive $C^*(I) \cdot \frac{\Delta_r}{K.L}\max_{j=1,\ldots,n}\left\{r_j + \sum_{h=j}^{n}\sum_{k=1}^{K(h)} l_{k,h}\right\} \cdot \frac{\Delta_r}{K.L} C_H(I)$ and $\frac{C_H(I)}{C^*(I)} \le \frac{K.L}{\Delta_r}$.

Therefore, whatever the instance I, the makespan provided by Schedule 1 is not worst than $K.L / \Delta_r$ times the optimal makespan, for the considered shop problems.

Consider the following instance, say I'', of n jobs. This instance I'' is of the job shop $Jm|P_{k,j} = L, r_j = (j\text{-}1)\Delta_r|C_{\max}$ type (i.e. such that all the processing times are equal to L and that the tim between the arrival of two consecutive requirements is equal to Δ_r with $m \cdot K$ and $K.L / m \cdot \Delta_r \cdot K.L$). In addition, each first operation to perform on jobs j required machine: $M(1,j) = 1 + \left(\left\lceil \frac{j-m}{m} \right\rceil K\right) \bmod m$. $\lceil a \rceil$ denotes the smallest integer greater than a, while $[a \bmod b]$ is the integer rest obtained when dividing a y b. In other words, the first operation performed on jobs 1 to m required machine 1, the next m jobs start with machine $(1 + K \bmod m)$, and so on. The successive operations are performed on the next machine as $M(k,j) = [\,[M(1,j) + k - 2] \bmod m] + 1$. For instance, if the first operation on a job requires i, the K operations of this job j require successively machines i, i+1, i+2, …, m, 1, 2, …, (i+K-2) mod m+1. Observe that the job $j + m$ will involve the next machin $M(1,j+m) = 1 + \left[\left(\left\lceil \frac{j-m}{m} \right\rceil K\right) \bmod m + K \bmod m\right] \bmod m = [\,[M(1,j) + K - 1] \bmod m] + 1$.

Note that when the number of operations is equal to the number of machines (i.e. $K = m$), such job shop proble I'' is also a flow shop of the $Fm|P_{k,j} = L, r_j = (j\text{-}1)\Delta_r|C_{\max}$ type, since we have in this case $M(1,j) = 1 + \left(\left\lceil \frac{j-m}{m} \right\rceil K\right) \bmod m = 1 + \left[\left\lceil \frac{j-m}{m} \right\rceil (K \bmod m)\right] \bmod m = 1$ and $M(k,j) = [\,[M(1,j) + k - 2] \bmod m] + 1 = [(k - 1) \bmod m] + 1 = k$.

Look at the following schedule of this instanc I'': each job j starts at tim $B''_{1,j} = \left\lceil \frac{r_j}{L} \right\rceil L = \left\lceil \frac{(j-1)\Delta_r}{L} \right\rceil L$ and ends at tim $C''_{K,j} = B''_{1,j} + K.L = \left[\left\lceil \frac{(j-1)\Delta_r}{L} \right\rceil + K\right] L$. Since $\left\lceil \frac{(j-1)\Delta_r}{L} \right\rceil L \ge \frac{(j-1)K.L}{m}$, the following inequality holds

$B''_{1,j+m} \bullet \left\lceil \frac{(j-1)\Delta_r}{L} \right\rceil L + \left\lceil \frac{m.\Delta_r}{L} \right\rceil L \bullet \left\lceil \frac{(j-1)\Delta_r}{L} \right\rceil L + \left\lceil \frac{(j-1)\Delta_r}{L} \right\rceil \frac{L.m}{j-1} \bullet C''_{K,j}$. Furthermore, we have $B''_{1,j} = \left\lceil \frac{r_j}{L} \right\rceil L \bullet r_j$ and all the processing times are equal to L. As a consequence, this schedul is feasible. Thus, C*(I''), the optimal makespan for this instance I'' of n jobs will be better than the one obtained by this schedule $C''_{K,n}$: C*(I'') $\bullet$ $C''_{K,n} \bullet \left\lceil \frac{(n-1)\Delta_r}{L} \right\rceil L + K.L \bullet (n\text{-}1)\Delta_r + L + K.L$.

The makespan provided by Schedule 1 is $C_{Schedul\ 1}(I'') = n.K.L$. Therefore, whatever $\varepsilon > 0$, set $n > [K L^2 (1 + K) + \Delta_r (L + \varepsilon \Delta_r)] / \varepsilon \Delta_r^2$. It follows that $C_{Schedule\ 1}(I'') / C^*(I'') < (K.L / \Delta_r) - \varepsilon$. Therefore, the makespan provided by Schedule 1 can be $K.L / \Delta_r$ times longer than the optimal makespan, for the considered job shop problem and its following extensions: F$m|r_j|C_{max}$, F$m|r_j, P_{k,j}=1|C_{max}$, Fm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{max}$, J$m|r_j|C_{max}$, J$m$|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{max}$ and J$m|r_j, P_{k,j}=1|C_{max}$.

Thus, whatever $K.L / m \bullet \Delta_r \bullet K.L$, Schedule 1 is ($K.L / \Delta_r$)-competitive for the specified shop problems. For $K.L / m \bullet \Delta_r$ (see Proposition 4), Schedule 1 is m-competitive and for $\Delta_r \bullet K.L$, Schedule 1 is 1-competitive.

□

Figure 4 depicts the schedule described in the previous proof of the first $n = 6$ jobs for an instance I'' with $K = 5$ *and* $m = 3$. The makespan is equal to $C''_{K,6} = 14$ and it is optimal in this case. Th makespan provided by Schedule 1 is $C_{Schedule\ 1}(I'') = n.K.L = 6x5 = 30$. Thus, Schedule 1 provides, in this case, a makespan which is $K.L / m = \Delta_r = 5 / 3$ times longer than the optimal one.

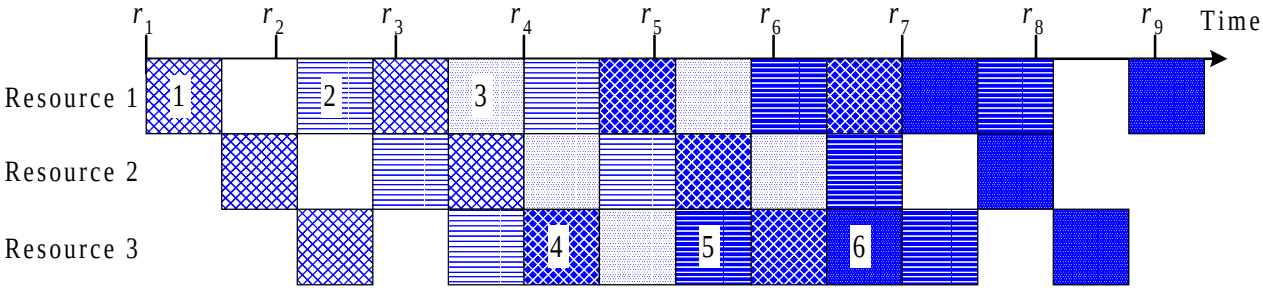

Figure 4- Optimal schedule of instance I'' for the case $n = 6$ jobs, $K = 5$ operations *and* $m = 3$.

Proposition 5 gives certain information about Schedule 1 quality, however it is not a prior analysis. It can be used to evaluate how much we can expect at the best if we reschedule all the jobs. In practice, on-line schedule aims at proposing to clients a reachable deadline, while schedule can be reconsidered at night to catch off-line a better solution.

4. Scheduling each job "a.s.a.p."

To obtain on-line a better makespan, we try to find how to complete each new requirement as soon as possible. It is an earliest ready time heuristic in which each job which enters the system is scheduled so that its makepan is minimized.

Schedule 2
For any new job j, schedule the job so that it will be completed as soon as possible.

For any schedule of Schedule 2 type, there exists, for any job j which starts (strictly) later than r_j (with $1 < j \bullet n$), a machine i required by an operation that starts on job j at the same time as an operation requiring the machine i ends on job (j-1). Otherwise, it would be possible to schedule job j earlier. This implies that in a flow shop, there exists an operation k (with $1 < k \bullet K$), that starts on job j at the same time as it ends on job (j-1), i.e. $C_{k,j-1} = B_{k,j}$. Thus, it is impossible to schedule a job in a flow

shop between two jobs which have been previously planned by applying an algorithm of the Schedule 2 type: job (j-1) is scheduled just before job j. Moreover, since two operations can not be performed simultaneously, operation k of job (j-1) ending at tim $C_{k,j-1}$, with j>1, the job j is completed at tim $C_{K(j),j} = \max\left\{ r_j + \sum_{k=1}^{K(j)} l_{k,j} ; \max_{k=1,\ldots,K(j)} \left\{ C_{k,j-1} + \sum_{f=k}^{K(j)} l_{f,j} \right\} \right\}$ in any schedule of Schedule 2 type for a flow shop.

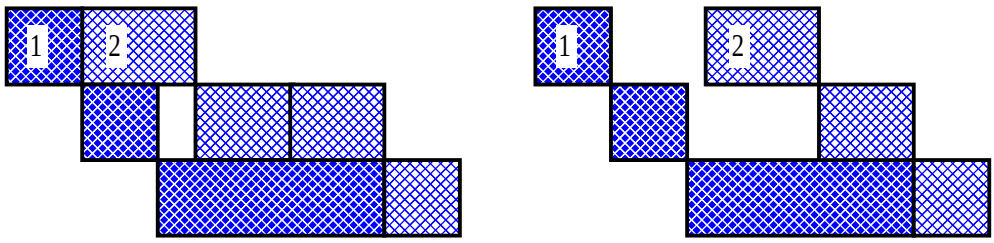

Figure 4- Job 1 being planned, both schedule are of the Schedule 2 type.

Since the processing times are not predetermined, a schedule of type 2 is not unique. This is shown on Figure 4 for a 3-operations flow shop: in the left-hand schedule, the processing time of the second operation is greater than in the right-hand schedule. Thus we obtain two schedules having the same makespan.

We distinguish two specific schedules of the Schedule 2 type:

Schedule 2a

For any new job j, schedule each operation in order to complete it as soon as possible. In others words, it is the solution of Schedule 2 such that each operation ends as soon as possible.

In flow shop situations, $B_{k,j}$ and $C_{k,j}$ being respectively the starting and the completion time of operation k on job j, job j is completed at $C_{K(j),j} = \max\left\{ r_j + \sum_{k=1}^{K(j)} l_{k,j} ; \max_{k=1,\ldots,K(j)} \left\{ C_{k,j-1} + \sum_{f=k}^{K(j)} l_{f,j} \right\} \right\}$ and operations k starts at tim $B_{k,j} = \max\left\{ r_j + \sum_{f=1}^{k} l_{f,j} ; \max_{f=1,\ldots,k} \left\{ C_{f,j-1} + \sum_{g=f}^{k} l_{g,j} \right\} ; \max_{f=k+1,\ldots,K(j)} \left\{ C_{f,j-1} - \sum_{g=k+1}^{f-1} u_{g,j} \right\} \right\} = C_{k+1,j}$

$\begin{cases} r_j \text{ if } j = 1 \\ \max\{r_j ; C_{K(j),j-1}\} \text{elsewhere} \end{cases}$ and is completed at $C_{K(j),j} = B_{1,j} + \sum_{k=1}^{K(j)} l_{k,j}$.

Schedule 2b

For any new job j, schedule each operation in order to start as late as possible, but keeping in mind that the job must be completed as soon as possible. In others words, it is the solution of Schedule 2 such that each operation starts as late as possible.

This two schedules exist (for any flow shop or job shop problems). Furthermore, each of them is unique [2]. We can find Schedule 2a by applying the critical path method on the PERT network associated to each job [8]. The algorithm provided by Ford [4] leads to the optimal solution with a time complexity equal to O(K^2), K being the number of operations to be performed on each job. dedicated approach [1] for which the complexity is O(K) can be used here. For a flow shop, th complexity of Schedule 2a for n jobs is in O(Kn), since it is impossible to schedule a job in a flow shop between two jobs which have been previously planned by applying an algorithm of the Schedule 2 type.

Moreover this approach leads to a O(Kw) algorithm when w different chronologically ordered periods are available to perform the operations on a job. Since booking periods for each new job adds less than K new periods to th w previous periods, this algorithm is able to schedule n jobs in a job

shop with $O(K^2 n^2)$ elementary operations. Similarly, it is possible to obtain Schedule 2b with the sam complexity [2], that is $O(Kn)$ for a flow shop and $O(K^2 n^2)$ for a job shop.

Proposition 6

For any instanc I of a flow shop or a job shop, the makespan $C_{Schedule\ 1}(I)$ obtained by applying Schedule 1 is not better than $C_{Schedul\ 2}(I)$ provided by one of the Schedules 2.

Proof: Consider an instance I of jobs, each of them consisting at most of K operations. We denot by I_n the n first jobs of instance I. For $n = 1$, we have $C_{Schedule\ 1}(I_1) = \sum_{k=1}^{K(1)} l_{k,1} = C_{Schedule\ 2}(I_1)$.

Assume that, f or any $n > 1$, $C_{Schedul\ 1}(I_{n-1}) \bullet C_{Schedul\ 2}(I_{n-1})$. We have $C_{Schedule\ 1}(I_n) = \max_{j=1,\ldots,n}\left\{r_j + \sum_{h=j}^{n}\sum_{k=1}^{K(h)} l_{k,h}\right\}$ (see proof of Proposition 4). It follows that $C_{Schedule\ 1}(I_n) = \max\left\{C_{Schedule\ 1}(I_{n-1}) + \sum_{k=1}^{K(n)} l_{k,n};\ r_n + \sum_{k=1}^{K(n)} l_{k,n}\right\}$. Moreover, in a schedule of 2 type, any job j is completed as soon as possible. This implies that th makespan of th n first jobs is $C_{Schedule\ 2}(I_n) \bullet \max\{C_{Schedule\ 2}(I_{n-1}); r_n\} + \sum_{k=1}^{K(n)} l_{k,n}$ (otherwise it would be possible to schedule th n-th job earlier). From the two previous inequalities, we derive that $C_{Schedule\ 2}(I_n) \bullet \max\{C_{Schedule\ 1}(I_{n-1}); r_n\} + \sum_{k=1}^{K(n)} l_{k,n} = C_{Schedule\ 1}(I_n)$.

This implies that, for any instanc I, $C_{Schedul\ 2}(I) \bullet C_{Schedule\ 1}(I)$. □

Proposition 7

Schedule 2a is better than any schedule in which any job j, with $2 \bullet j \bullet n$, is processed after $(j\text{-}1)$ for Fm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{max}$. Thus, in such situations, Schedule 2a is the best of Schedules 2.

Schedule 2a is optimal to Fm|no-wait, r_j, $l_{k,j} = l_k$, $u_{k,j} = u_k$, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{max}$.

Proof: In a flow shop Fm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{max}$, for a given sequence of jobs to be performed, Schedule 2a provides the optima makespan since it is not possible to schedule an operation earlier.

Therefore, if all jobs are identical (which is the case if $l_{k,j} = l_k$ and $u_{k,j} = u_k$), all the sequences are indistinguishable and Schedule 2a is optimal. □

Proposition 8

Schedules 2 ar m-competitive for problems F$m|r_j|C_{max}$, Fm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{max}$, J$m|r_j|C_{max}$ and Jm|no-wait, r_j, $l_{k,j} \bullet P_{k,j} \bullet u_{k,j}|C_{max}$.

Proof: Since Schedule 1 is m-competitive for these problems (see Proposition 4) and th makespan obtained by applying Schedule 1 is not better than the one obtained by Schedules 2 (see Proposition 4), the makespan obtained by applying one of the Schedules 2 is not m times longer than the optimal makespan. Unfortunately, there exists some cases (of the least general problems F$m||C_{max}$ and Fm|no-wait, $l_{k,j} = P_{k,j} = u_{k,j}|C_{max}$) for which this bound is attainable

Consider a special case of a flow shop of n jobs having the same release date (i.e. $r_j = 0$). Processing times are equal either to a given duration D for any job j such that j is an even number or to d (with $d \bullet D$) otherwise, as shown on Figure 5. Whatever n, even number of jobs, the optimal makespan of this instanc $\underline{I}$ is $C^*(\underline{I}) = (D + d)(n / 2) + D(m - 1)$ while th makespan provided by Schedule 2 is $C_{Schedul\ 2}(\underline{I}) = (D\,m - d\,(m - 2))(n / 2) + d\,(m - 1)$ (which is unique is this case). For $D = d(\tau_l + 1)$, we hav $C_{Schedule\ 2}(\underline{I}) / C^*(\underline{I}) \bullet [n(\tau_l\,m + 2) + 2(m - 1)] / [n(\tau_l + 2) + 2(m - 1)(\tau_l + 1)]$. Whatever $\varepsilon > 0$, set $n > 2\,m^2\,\tau_l / \varepsilon$. It follows that $C_{Schedule\ 2}(\underline{I}) / C^*(\underline{I}) > [(\tau_l\,m + 2) / (\tau_l + 2)] - \varepsilon$. Moreover, whatever $\varepsilon' > 0$, set $\tau_l > 2\,m / \varepsilon'$ which implies that $C_{Schedule\ 2}(\underline{I}) / C^*(\underline{I}) > m - \varepsilon'$.

Thus, Schedules 2 are *m*-competitive for the specified shop problems □

Cases in which Schedules 2 provide a makespan that is *m* times longer than the optimal makespan are not usual in practice.

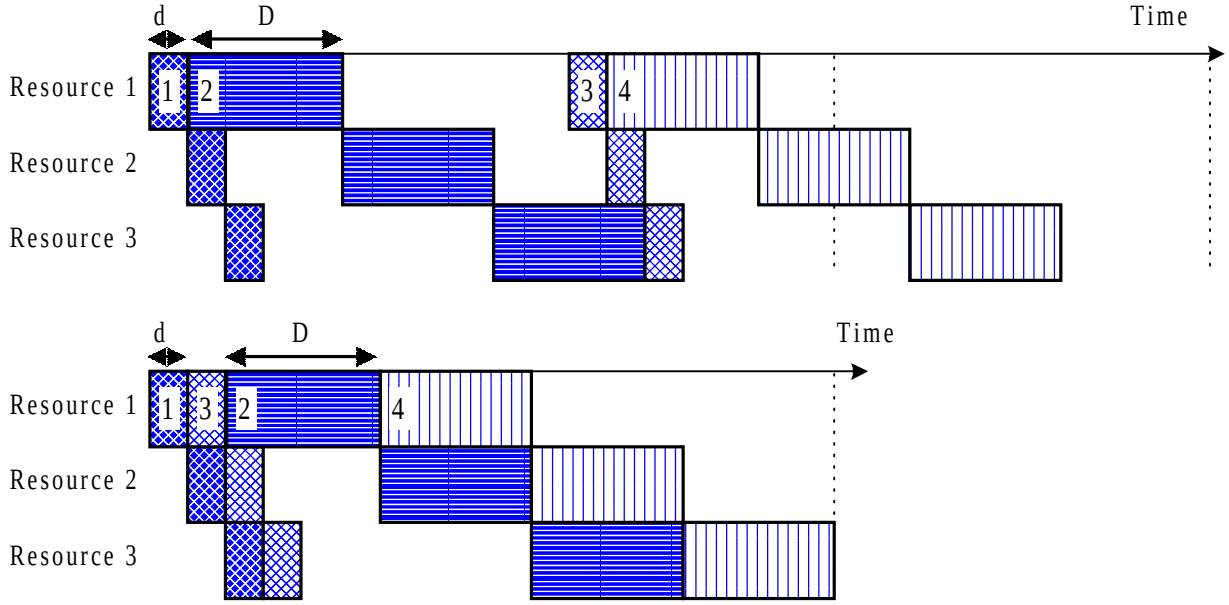

Figure 5- Schedule provided by Schedules 2 and optimal schedule of $n = 4$ jobs on which $K = m = 3$ operations are performed, for processing times equal to either D or $d = D / 4$.

5. Average case study

The average performance of the schedules is obtained by comparing their makespan on a large number of jobs. We generate $K = m$ operations to be performed on each of $n = 1000$ jobs (m is the number of machines). The bounds of processing times are randomly generated such that $0 < l_{k,j} \leq 50$ and $l_{k,j} \leq u_{k,j} \leq 150$.

We look at th *average inefficiency* of a schedule which is the ratio between th makespan $C_H(I)$ provided by algorith H and the maximal duration to process the jobs on a machine. This duration is defined by $\underline{C}(I) = \max_{i=1,\ldots,m} \left\{ \sum_{\substack{j=1 \\ \text{such that} \\ M(k,j)=i}}^{n} l_{k,j} \right\}$. We have $\alpha\, C^*(I) \leq C_H(I) \leq C^*(I) \leq \underline{C}(I) > 0$ and $\alpha \frac{C^*(I)}{\underline{C}(I)} \geq \alpha$,

where α is the competitive ratio of algorithm H. Thus, if the makespan provided by algorith H is such that $\alpha \geq \rho \frac{C_H(I)}{\underline{C}(I)}$, then we have $\alpha \frac{C^*(I)}{\underline{C}(I)} \geq \alpha \geq \rho \frac{C_H(I)}{\underline{C}(I)}$ and $\alpha \geq \frac{\alpha}{C^*(I)} \geq \rho \frac{C_H(I)}{C^*(I)}$.

In the considered shop problems, Schedule 2a and 2b ar *m*-competitive (see Proposition 8). Figure 6 shows how the average inefficiency grows as $K = m$ increases, for Schedules 2a and 2b applied on flow shops. This ratio for both schedules is tinier than m: the obtained makespan is fare from the worst case. In this situation of flow shops, we notice that Schedule 2a is better than Schedule 2b.

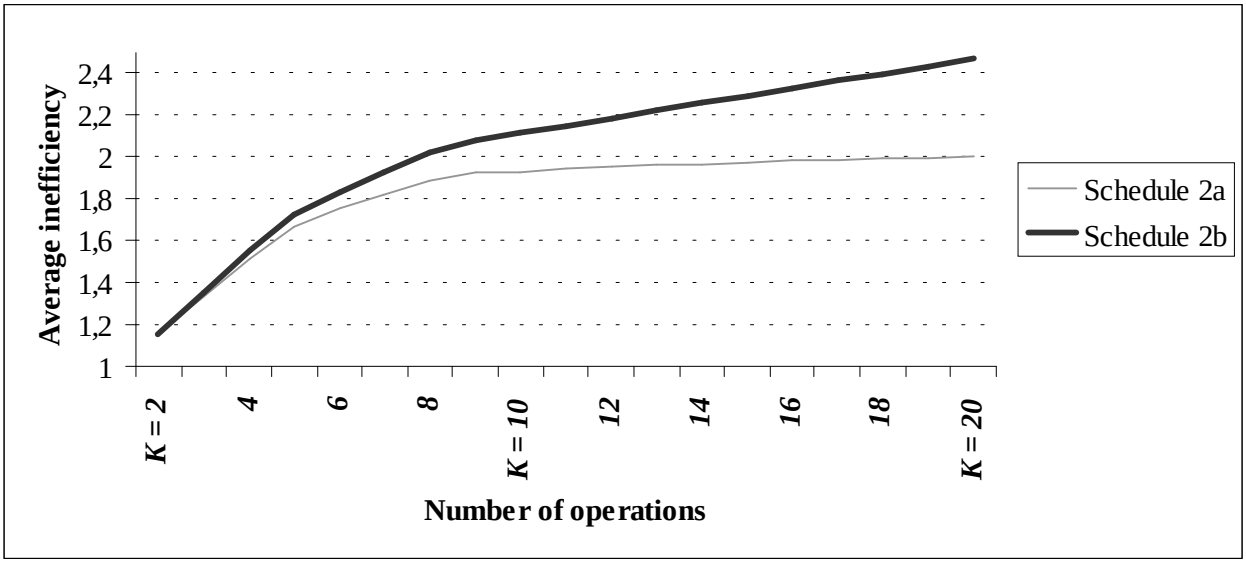

Figure 6- Average inefficiency of Schedules 2 for different numbers of operations ($K = m$).

On one hand, Schedule 2a provides a shorter makespan than Schedule 2b for flow shops (see Proposition 7). On the other hand, since Schedule 2b reduces as much as possible the time needed to perform each job compared to Schedule 2a, Schedule 2b provides a better makespan than Schedule 2a for job shops. Figure 7 depicts these two situations for jobs with $K = 10$ operations requiring $m = 1$ machines. The productivity is the number of jobs performed per unit of time. Thus, it is equal to the ratio of the number of jobs ($n = 1000$ jobs) by th makespan.

Figure 7- Productivity provided by Schedules 2 for $K = 10$ operations requiring $m = 10$ machines.

6. Conclusio

This study is devoted to shops problems having two characteristics: processing times are controllable and the process is a no-wait one. It has been explained that this approach allows bot scheduling jobs and controlling WIP. We propose several simple approaches to schedule jobs on-lin and assess the efficiency of each one in different situations. In particular, we prove that they are both m-competitive (m being the number of machines) and that no 2-competitive algorithm exists for th different flow shops and job shops we scanned. For (no-wait or classical) flow shop and job shop problems, we do not know if there exists (or not) an algorithm to schedule jobs on-line having a competitive ratio better than m and worst than 2.

This research has been extended to different situations [2]:

- Several machines of different types are required to perform an operation,

- Assembly and disassembly operations have to be performed on jobs,
- Costs are associated to processing times…

On-line methods are interesting since they are more and more popular in industry due to th increasing market competition and they can be used to refine the off-line scheduling approaches.

References

[1] Chauvet F., Levner E., Meyzin L., Proth J.-M. (2000). On-line Scheduling in a Surface Treatment System. *European Journal of Operational Research*. **120**(2).

[2] Chauvet F. (1999). Constrained Work-In-Process in On-line Scheduling. (in French) *Ph.D Thesis, INRIA, University of Metz, France,* 239 p. http://www.inria.fr/RRRT/TU-0593.html.

[3] Chen B., Woeginger G.J. (1995). A Study of On-line Scheduling two-stage shops. In D.-Z. Du P.M. Pardalos, editors, *Minimax and Applications*, 97-107. Kluwer Academic Publishers, 1995.

[4] Ford L.R.Jr (1956). Network flow theory. *The Rand Corporation*, 293.

[5] Gilmore P.C., Gomory R.E. (1964). Sequencing a One-State Variable Machine: A Solvable Case of Traveling Salesman Problem. *Operations Research* **12**, 655-679.

[6] Goyal S.K., Sriskandarajah C. (1988). No-Wait Shop Scheduling: Computational Complexity and Approximate Algorithms. *Operations Research* **25**(4), 220-244.

[7] Hall N.G., Sriskandarajah C. (1996). A Survey of Machine Scheduling Problems with Blocking and No-Wait in process. *Operations Research* **44**, 510-525.

[8] Malcolm D.G., Roseboom J.H., Clark C.E., Fazar W. (1959). Application of a technique for research and development program evaluation. *Operations Research* **7**(5).

[9] Röck, H. (1984). The Three-Machine No-Wait Flow Shop Problem is NP-Complete *Journal of the Association of Computer Machine* **51**, 336-345.

[10] Röck, H. (1984). Some new results in flow scheduling. *Z. Operations Research* **28**, 1-16.

[11] Sahni, S., Cho, Y. (1979). Complexity of Scheduling Jobs with No-Wait in Process. *Mathematics of Operations Research* **4**, 448-457.