Production Scheduling With Limited Buffer Storage

A. Benlogab and B. Descotes-Genon
École des Mines de Nantes
4, rue Alfred Kastler, La Chantrerie
44070 Nantes Cedex 03
FRANCE

Abstract: This paper deals with the flow line control using the perturbation analysis approach. The perturbed system is analysed in order todetermine the perturbation propagation rules under the deterministic similarity assumption.
The second part focuses on carrying out a local search based method, to obtain a new jobs' sequencing when control deviation in real time becomes intolerable. This adaptive procedure is based on a simulated annealing method.
The last part concerns the robustness of scheduling policies. The expected delay due to disruptions is analysed for some particular situations, and an application to the exponentially distributed duration of the disruption is considered for illustration.

Keywords: flow shop scheduling, buffer capacities, perturbation analysis, scheduling robustness.

1 Introduction

1.1 Motivation

The complexity of manufacturing, traffic and computer systems has given an increasing importance to the investigation of discrete event systems [9, 19].
This paper deals with short-term production scheduling of the flow line cells which are particularly interesting in practice and have been broadly discussed in the literature.
However, for almost all these purposes only the cases where the intermediate buffer sizes are either all infinite or all null have been considered, although in industrial production the amount of available buffer storage has an important influence on the performance of the system.
A new heuristic method will be presented in order to determine a near optimal solution for the flow shop problem with finite capacity storage aiming to minimize the maximum jobs lateness criterion. Nevertheless, in production environment random events usually challenge any static solution, without considering the dynamic behaviour of the system, with the scheduling initially drawn up rapidly becoming useless. Therefore, to bring out a realistic solution we are incited to analyse perturbation propagation over the production line.
After all, the robustness of a scheduling policy represents the expected delay associated with an objective function due to disruptions. Although such property is important for on-line control, it is rarely considered in the literature on production scheduling. Given the nature of discrete event systems, only some particular cases are analysed in order to give basic notions and results about the robustness property of scheduling policies.

1.2 Notion of Perturbation Analysis

Perturbation Analysis (PA) is a recent technique applied in the area of discrete event dynamic systems; it has been developed for the gradient estimation of performance measures with respect to some system control variables. PA requires only knowledge of the nominal sample path, therefore it is complementary to any existing simulation [20, 12, 13].
An interesting application of PA is in the on-line monitoring of systems operations, since we can calculate sensitivities of the performance measure with respect to the system parameters in real time. This feature is reminiscent of "neighbouring optimal control" ideas in control theory where corrective action for a system can be taken based on an observed or estimated perturbation of a nominal trajectory [20, 4, 10, 11].
The change of a system control is called *pertur-*

bation and the new trajectory is called *perturbed path.*
The discrete system for production scheduling purpose is represented by means of a Gantt chart.
Our first objective is to carry out a provisional solution for the flow shop scheduling problem under static conditions and then to look for the perturbation propagation rules in order to predict effects due to the operation times changing; this will permit to reduce delays with respect to jobs' due dates.
The second objective consists in determining a new production scheduling more suitable for the perturbed system when some variables go beyond their associated bounds, and also in analysing the scheduling robustness.
The remainder of this paper is arranged as follows: in the next section, we present the flow shop scheduling problem and give its general mathematical formulation. A heuristic method is proposed and illustrated on some applications. In the third section, the perturbed system is analysed in order to carry out the perturbation propagation rules under the deterministic similarity assumption. In the fourth part, a simulated annealing based method is proposed in order to determine a new jobs sequencing taking into account the accumulated perturbations and aiming to minimize the maximum jobs lateness.
The last part focuses on the scheduling robustness analysis for some particular situations because of the complexity of the discrete event systems due to their nature.

2 Problem Description

Let there be a set N of n jobs simultaneously available $J_i \quad i = 1, 2, ..., n$ to be processed on m machines $M_j \quad j = 1, 2, ..., m$ in the same order of processing, with processing times $p(i, j)$. We assume that each job has a due date denoted $d(i)$. The set-up and shut down times are sequence-independent and so they are included in processing times.
There is a buffer B_j of some known finite capacity designated by b_j between machines M_j and M_{j+1} $j = 1, 2, ..., m-1$. b_j is interpreted as the maximum number of jobs that can be in the buffer at the same time. We allow the machine to be used for storage, therefore it will be blocked until at least one job leaves its **downstream buffer (Figure 1).**

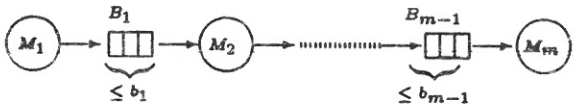


Figure 1: Production Line with Limited Buffer Capacities, of m Machines

The transport time from a machine (resp. buffer) to its downstream buffer (resp. machine) is negligible. In the multistage flow shop scheduling ($m \geq 3$), it appears that the assumption of preservation of the job sequence on machines is made for convenience, e.g. to reduce the number of feasible schedules from $(n!)^m$ to $n!$. Such assumption literally renders the problem unsolvable for any practical purposes. A study by Heller (1960), has however revealed evidence that the assumption may not only be convenient, but also may often produce near optimal solutions [7].
The objective of the problem is therefore to determine a jobs sequencing σ fulfilling the above constraints and minimizing the maximum jobs lateness criterion.
σ_i is interpreted as the identity of the job sequenced at the i^{th} position, and the job lateness equals its completion time (upon the last machine) minus its due date. Setting that $S(i, j)$ (resp. $T(i, j)$) is the starting (resp. ending) time of the job J_i on the machine M_j.
Since no interruption of processing on a machine is allowed, $S(\sigma_i, j)$ and $T(\sigma_i, j)$ are connected via :

$$T(\sigma_i, j) = S(\sigma_i, j) + p(\sigma_i, j) \qquad \forall i, j$$

The classical computation formula for the earliest start of J_i on M_j if no buffer storage constraint is taken into account is :

$$S(\sigma_i, j) = \max\{T(\sigma_i, j-1), T(\sigma_{i-1}, j)\} \quad \forall\, i, j$$

If limited buffer storage is taken into account, then it is also necessary that job J_{i-1} should be able to leave M_j, and this is guaranteed if the job J_{i-b_j-1} has left the buffer behind M_j [17, 18, 15].

Therefore, for our purpose, the departure times T^* are defined by : $\forall\, i, j$

$$T^*(\sigma_i, j) = \max\{T(\sigma_i, j), S(\sigma_{i-b_j-1}, j+1)\}$$

and the starting times then hold :

$$S(\sigma_i, j) = max\{T(\sigma_i, j-1), T(\sigma_{i-1}, j), S(\sigma_{i-b_j-1}, j+1)\} \quad \forall\, i, j$$

According to the previous considerations, a solution σ is determined by the following system :

$$\begin{cases} \min_{\sigma\in\Sigma} \max_{i\in N}\{T(\sigma_{i-1}, j) - d(\sigma_i)\} \\ s.c. \\ S(\sigma_i, j) = max\{T(\sigma_i, j-1), T(\sigma_{i-1}, j), \\ S(\sigma_{i-b_j-1}, j+1)\} \qquad \forall i, j \\ T^*(\sigma_i, j) = S(\sigma_i, j) + p(\sigma_i, j) \qquad \forall i, j. \end{cases}$$

The connection of this model with the flow line manufacturing systems enables to gradually approximate the bounds of starting times and therefore to produce a near optimal solution for the problem.

3 Heuristic Method for the Flow Shop Scheduling Problem

In order to simplify the previous modelling, variables T^* can be removed and the previous system turns to be:

$$\begin{cases} \min_{\sigma\in\Sigma} \max_{i\in N}\{S(\sigma_{i-1}, j) + p(\sigma_{i-1}, j) \\ -d(\sigma_i)\} s.c. \\ S(\sigma_i, j) = max\{S(\sigma_i, j-1) + p(\sigma_i, j-1), \\ \qquad S(\sigma_{i-1}, j) + p(\sigma_{i-1}, j), \\ \qquad S(\sigma_{i-b_j-1}, j+1)\} \quad \forall i, j \qquad (1) \end{cases}$$

The problem consists in determining a sequence σ and starting times $S(\sigma_{,}, .)$ for all the jobs on one machine, such that the maximum lateness is minimized.
Suppose that an optimal sequence is available and denoted by σ^*.
Then the corresponding maximum lateness is determined via :

$$L^*_{max} = \max_{i\in N}\{S(\sigma^*_i, m) + p(\sigma^*_i, m) - d(\sigma^*_i)\} \qquad (2)$$

Now, let us set $d^*(\sigma^*_i, j)$ the latest ending time of the job J_i on machine M_j, with respect to the maximum lateness L^*_{max}, in other words :

$$S(\sigma^*_i, j) \leq d^*(\sigma^*_i, j) - p(\sigma^*_i, j) \qquad (3)$$

We denote this upper bound $F(\sigma^*_i, j)$.

$$F(\sigma^*_i, j) = d^*(\sigma^*_i, j) - p(\sigma^*_i, j)$$

It is important in this formula to point out that the upper bound depends only on parameters associated with the jobs sequenced after J_i, because $d^*(\sigma_i, j)$ depends on $d(\sigma_i)$ (related to the last machine M_m) and the intermediate machines $M_{j+1}, \ldots, M_{m-1}$.
This remark is important as it will enable us to gradually determine the sequence starting from its last job.

3.1 Flow Line Models

In this section, we present some general issues in the analysis of production lines with limited capacity storage.
Manufacturing flow line systems consist of material, work areas, and storage areas. Each job visits each work and storage exactly once in a fixed sequence.
Material flows from work area to storage area to work area. Storage areas can hold only a finite amount of material. Since the buffers have finite capacity, blocking may occur. Different types of blocking mechanisms are of interest : blocking after service (BAS) occurs if, at the instant of completion of a part on machine M_j, the downstream buffer is full. During this time the machine is prevented from working and is said to be *blocked*. When a space becomes available in the downstream buffer, the part is immediately transferred and the machine can start process another part, if any [5, 8].
Blocking before service (BBS) occurs if a machine can start process a part only if there is a space available in the downstream buffer. Otherwise it has to wait until a space gets available.
A machine M_j is said to be blocked when buffer B_j is full.
For our purpose we only consider the BAS case also called *production blocking*.
Thus, when a machine is blocked, it is prevented from working. A machine may also be prevented from working because it has no material to work on. This phenomenon is starvation. It corresponds in the case of production blocking to the situation where the upstream buffer is empty (Figure 2).

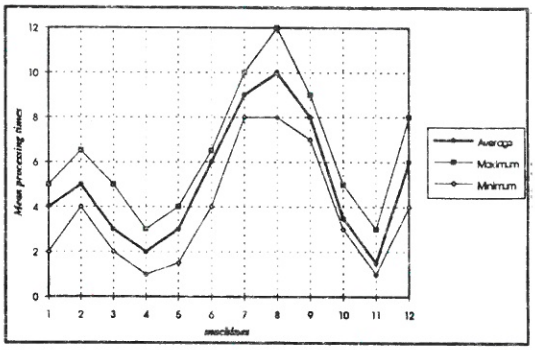


Figure 2: Mean Operation Time Tendency Over a Production Line

In the context of flow shop scheduling, these two phenomena involve gaps between ending time of a job and starting time of the next one even with respect to no idle scheduling policy.
By means of the Gantt diagram, we would like first to analyse the limit behaviour of the system wherein all jobs are considered to be identical, and considering the mean processing time associated with a machine M_j defined as:

$$\overline{p}(j) = \frac{1}{n}\sum_{i=1}^{n} p(i,j) \qquad j = 1, 2, \ldots, m$$

A steady state is indeed equivalent to the system behaviour if all its buffers were null [1, 2].
Nevertheless, the effect of the intermediate capacity storage appears in the length of the transient state and the increasing fashion of the gaps values before reaching the steady state (if there are enough jobs).
However, within the transient state, it is more difficult to determine analytically the gaps evolution in time (Figure 3).

To achieve this, we used a simulation approach, and applying the initial flow shop formula, jobs sequence is obviously useless here (jobs are identical), the gaps are determined by means of the obtained processing times. Finally, we have at our disposal a set of m gaps functions depending on the time, but it is more suitable to define them with respect to the jobs ranks,

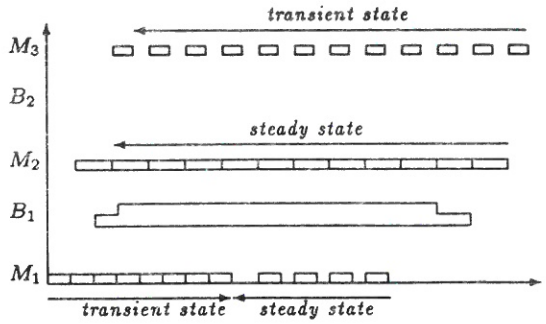


Figure 3: Gaps Evolving in Transient and Steady State

that is :

$$G(k,j) \qquad k = 1, 2, \ldots, n-1 \quad j = 1, 2, \ldots, m.$$

3.2 Starting Time Bounds

In the rest we assume that, with respect to a given machine, the variance of the job processing times is limited by a constant upper bound ϵ (which will be discussed later).
Under this assumption, the lower bound of starting times will be determined supposing that all jobs not already scheduled must pass before it (see Figure 4) :

$$r(\sigma_i, j) = \sum_{l=1}^{j-1} \overline{p}(l) + \sum_{k=1}^{i-1} [p(\sigma_k, j) + G(k,j)]$$
$$i = 1, 2, \ldots, n\ ;$$
$$j = 1, 2, \ldots, m.$$

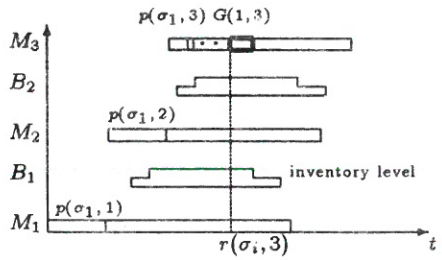


Figure 4: Lower Bound for Starting Times via the Gantt Chart

To determine an approximation of the latest

ending times used in the upper bound F introduced at the beginning of this section, let us turn back to the relationship (2) : if we set :

$$d^*(\sigma_i, m) = d(\sigma_i) + L^*_{max}$$

the following equation holds :

$$T(\sigma_i, j) \leq d^*(\sigma_i, m)$$

Thereafter, the initial problem turns into the problem consisting of determining an admissible solution σ^* which respects all the new jobs due dates $d^*(.,m)$.
Regarding the rest of machines, the latest ending times are approximated using the average gaps G(.,.) as follows :

$$d^*(\sigma_i, j) = \min\{d^*(\sigma_{i+1}, j) - p(\sigma_{i+1}, j) - G(i, j),$$
$$d^*(\sigma_i, m) - \textstyle\sum_{k=j+1}^{m}[p(\sigma_i, k) + G(i-1, k)]\}$$
$$i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m-1.$$

with

$$d^*(\sigma_i, m) = d(\sigma_i) + L^*_{max} \quad i = 1, 2, \ldots, n.$$

But the variable L^*_{max} is not known yet. To overcome this problem we use the dichotomy procedure; firstly we consider the maximum lateness α obtained during the simulation mentioned in the last paragraph and we consider the interval $I = [\alpha - |\alpha|, \alpha + |\alpha|]$.
L^*_{max} will ever be the mean value of this interval ($L^*_{max} = \alpha$ here).
If the proposed algorithm (in the next section) permits to carry out an admissible jobs sequencing σ^* then the next interval to test is $I = [\alpha - |\alpha|, \alpha]$.
Otherwise, one considers $I = [\alpha, \alpha + |\alpha|]$ and so on, until the interval length gets too small.
Once the value L^*_{max} is fixed (for some iteration), jobs are tested until finding one for which both bounds are compatible to all machines, then it is set at the end of the sequence σ.
Afterwards, one seeks the previous job (in the order $n-1$) by determining both bounds of the tested job which obviously differs from the already scheduled jobs. We check the remaining jobs until finding one which fulfils the condition for all machines. Therefore it will be scheduled in the order $n-1$ and so on.
We point out that this partial job sequencing is not definitive, in fact, if the procedure is blocked at a certain step, previously ordered job will be reviewed.

3.3 Proposed Algorithm

The proposed algorithm ought to involve an important computation time if the research of the solution is not oriented.
In order to enhance this method, we suggest to use an initial jobs sequencing based on the *Mean Slack Time (MST)* priority rule :

$$\lambda(i) = \sum_{j=1}^{m} p(i, j) - d(i) \qquad i = 1, 2, \ldots, n$$

Jobs are considered according to their priorities given by λ and starting by the last job in the list made out.
This will be used at the beginning of the algorithm, but in the following iterations (when L^*_{max} is changed) we would rather have the solution made out in the last iteration wherein L^*_{max} was successful.
However, the number of "turning backs" after a blocking in the algorithm may become very important; for this reason it has been limited by a certain bound which depends on the size of the problem instance in order to bring about a solution in a reasonable computation time.
Any iteration of the algorithm can be presented as follows :

/***Next(Input :** σ, **Output :** σ, ind) **:** *its objective is to make some order function on the set of permutations of the jobs. It gives the permutation obtained by a minimum switching starting from the end of the sequence* σ *and aiming to change the content of* si_i, *and ind is the index of the little rank changed in the new permutation.**/

Next(Input : σ, **Output :** σ, ind)
$ind = n - 1$;
While $\sigma_{ind} > \sigma_{ind+1}$ **do**
    $ind = ind - 1$
enddo
Determine $i \in \{ind + 1, \ldots, n\}$ such that :
$\min_{\sigma_i > \sigma_{ind}} (\sigma_i - \sigma_{ind})$
Permut (σ_{ind}, σ_i).
Range $(\sigma_{ind+1}, \ldots, \sigma_n)$ in the increasing order.
end Next

$i \leftarrow n$; $j \leftarrow m$; $compt \leftarrow 0$;

initial permutation σ it /*determined according to the above remarks*/
Sol_Adm $\leftarrow$ True;
Repeat
Repeat
Determine the bounds $r(\sigma_i, j)$ and $F(\sigma_i, j)$;
$j \leftarrow j - 1$;
Until $(j < 0)$ or $(r(\sigma_i, j) > F(\sigma_i, j))$;
if $(j \geq 0)$ **then**
$\sigma \leftarrow Next(\sigma, ind + 1)$;
$i \leftarrow \max\{i, ind\}$;
$compt \leftarrow compt + 1$;
endif;
$i \leftarrow i - 1$;
Until $(i < 0)$ or $(compt > B)$
if $(i \geq 0)$ then
Sol_Adm $\leftarrow$ false;
endif;

- **Applications**

The performance evaluation of the proposed method is illustrated via one of several examples made with uniformly distributed processing times.
The production line considered, is composed of 30 jobs and 15 machines , separated by buffers of capacities $b = (2, 3, 4, 1, 2, 6, 4, 3, 10, 0, 0, 0, 0, 0)$.
The sub-systems FS(n, m, b^m) consist of the $n \leq 15$ first jobs, the $m \leq 15$ first machines and their corresponding buffers sizes b^m.
Jobs $J_1, \ldots, J_n$ have to be processed on each station, all the jobs must flow in the same order and it is assumed that each job requires one unit of storage capacity.
The due dates associated with the jobs are given by :

$$d(i) = 100 + 10 * i \qquad i = 1, 2, \ldots, n$$

The performance criterion we considered is the minimum of jobs maximum lateness.
For each sub-system FS(n, m, b^m), 20 tests of the problem are generated and every time the obtained jobs sequence is collected.
In order to evaluate these results we have compared them with a set of K arbitrary permutations (including the obtained one). We considered $K = 1000$ for $n = 6$ and $K = 5000$ for $n = 10, 15$.
For every test the minimum of jobs maximum lateness, the maximum one and also the mean value of the K generated permutations, are determined.
All the results are obtained using a PC DX2-66 and are summarised in Figures 5, 6 and 7 (τ represents the computing time).

The tests carried out show that our proposed algorithm gives a relatively good solution for the flow shop problems with finite capacity storage.
The evaluation method may be considered, in fact, as a procedure which aims to ameliorate the obtained scheduling solution.
Moreover, the assumption related to the processing times variance is not really restrictive, therefore this method can be used even if the operation times are not close to each other.

4 Perturbation Propagation Rules

We denote by $\tilde{S}(\sigma_u, v)$ (resp. $\tilde{T}(\sigma_u, v)$) the new corresponding starting (resp. ending) time for each job J_{σ_u} $u = 1, 2, ..., n$ and each machine M_v $v = 1, 2, ..., m$. We also denote by $\tilde{T}^*(\sigma_u, v)$ the new departure time taking into account the perturbation.
The relationships between variables S and T in the flow shop problem with finite capacity storage applied to the new variables can be written as follows :

$$\tilde{S}(\sigma_u, v) = \max\{\tilde{T}^*(\sigma_u, v-1), \tilde{T}^*(\sigma_{u-1}, v)\}$$
$$\tilde{T}(\sigma_u, v) = \tilde{S}(\sigma_u, v) + \tilde{p}(\sigma_u, v)$$
$$\tilde{T}^*(\sigma_u, v) = \max\{\tilde{T}(\sigma_u, v), \tilde{S}(\sigma_{u-b_v-1}, v+1)\}$$

where $\tilde{p}(\sigma_u, v)$ is identical to $p(\sigma_u, v)$ except for $(u, v) = (i, j)$ where we have $\tilde{p}(\sigma_u, v) = p(\sigma_u, v) + \delta$ (δ : perturbation duration).
For production scheduling the nominal path and the perturbed path are represented by a Gantt diagram. The perturbation corresponds in general to the operation times' variation [3].
This is illustrated by an example of a flow shop with three machines and five jobs. The buffer B_1 has one unit capacity storage and the buffer B_2 is null. The perturbation occurs at the end of the operation of the job J_{σ_2} onto M_2 and is of amount Δ (see Figure 8).

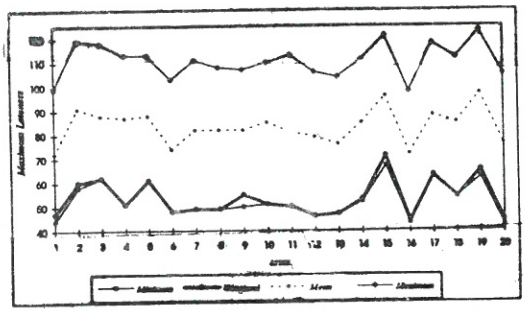


Figure 5: FS(6, 5, $b^{\#}$, L_{max}), $\tau = 1$ sec.

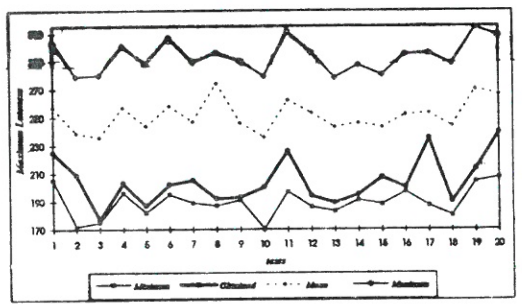


Figure 6: FS(10, 10, b^{10}, L_{max}), $\tau = 20$ sec.

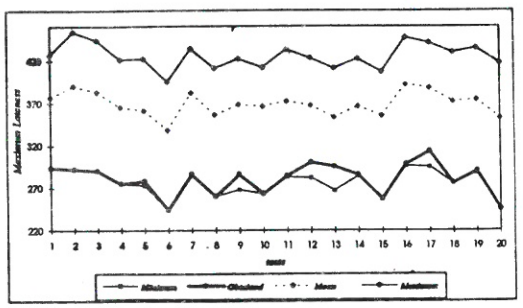
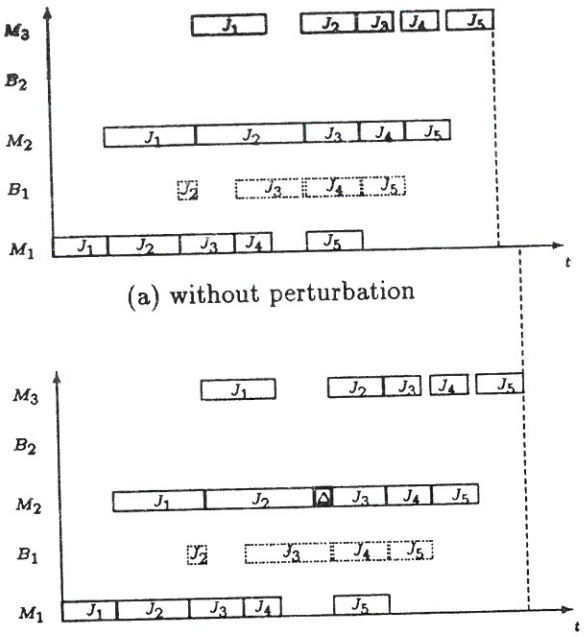


Figure 7: FS(15, 10, b^{10}, L_{max}), $\tau = 30$ sec.

Times for



(a) without perturbation

(b) with a perturbation Δ

Figure 8: Gantt Diagram for the Nominal and Perturbed Path

With respect to a given machine M_v, the perturbation will take effect as a certain job denoted uv which is already on it or will visit it after some period of time [3].

$$\tilde{S}(\sigma_u, v) = \begin{cases} \max\{\tilde{T}(\sigma_u, v-1), \tilde{T}(\sigma_{u-1}, v))\} & \text{if } u \geq u_v \\ S(\sigma_u, v) & \text{otherwise} \end{cases}$$

After the perturbation period (at the instant $\tilde{T}(\sigma_i, j)+$) every machine M_v of the production line is in one of the following situations :

case 1: M_v is blocked owing to the limitation of the downstream buffer capacity. Let M_{v^*} be the machine which finishes the blocking chain which contains M_v and $J_{\sigma_{i^*}}$ the job blocked on it, then this relationship holds :

$$u_v = i^* + b_v + ... + b_{v^*-1} + v^* - v.$$

We point out that the finishing times associated with the blocked jobs are not changed but their departure times may be affected.

Taking into account the fact that $\tilde{T}^*(\sigma_u, v) > \tilde{T}^*(\sigma_i, j)+$, it follows that :

$$\tilde{T}^*(\sigma_u, v) = \tilde{T}^*(\sigma_{u_{v+1}}, v+1)$$

and consequently :

$$\boxed{\tilde{T}^*(\sigma_u, v) = \tilde{T}^*(\sigma_{i^*}, v^*)}$$

case 2: M_v is starved.

Let M_{v^*} be the machine which starts the starving chain including M_v and $J_{\sigma_{i^*}}$ the next job which will visit it, then it is clear that :

$$u_v = i^*$$

Taking into account the fact that $\tilde{T}^*(\sigma_u, v) > \tilde{T}^*(\sigma_i, j)+$, the following relationship holds :

$$\tilde{S}(\sigma_{u_v}, v) = \tilde{S}(\sigma_{u_v}, v-1) + \tilde{p}(\sigma_{u_v}, v-1)$$

and consequently :

$$\boxed{\tilde{S}(\sigma_{u_v}, v) = \tilde{S}(\sigma_{i^*}, v^*) + \sum_{l=v^*}^{v-1} \tilde{p}(\sigma_{i^*}, l)}$$

case 3: M_v is neither blocked nor starved. The perturbation may become effective as the current job $J_{\sigma_{i^*}}$ on machine M_{v-1}, if the latter is affected by the perturbation, and also if it meets a starving period on M_v (otherwise the perturbation is absorbed before M_v). Under these conditions, we have :

$$\boxed{\tilde{S}(\sigma_{i^*}, v) = \tilde{T}(\sigma_{i^*}, v-1)}$$

We have so far considered only delays in departure times but it is possible to bring out similar rules if job operation finishes earlier than on the nominal path.

Afterwards, without mentioning it, the deterministic similarity assumption was ever used, that is : All of the accumulated perturbations of a trajectory are so small as not to change the order of events between the nominal and the perturbed trajectories. For our purpose, we use the propagation rules in order to predict the evolution of the system on the perturbed path.

By means of the provisional maximum jobs lateness, we can obtain some bounds for the operations within the current solution considered as acceptable and the deviation is tolerable.

Beyond these limits, we will be urged to look for a new jobs' sequencing more adapted to the new situation.

5 Rescheduling Via A Local Search Based Method

In the retained generalship here, one fixes the number of landings p, and the number of tests by landing q; this is to know precisely the number of explored configurations [6, 14].

The adopted decreasing procedure is a geometrical decreasing. If we denote $T_p, T_{p-1}, ..., T_1$ the temperatures (where T_p represents the initial temperature and T_1 the ending one), then we have :

$$T_i = \alpha.T_{i+1} = \alpha^i.T_1 \qquad (\alpha < 1)$$

The coefficient α is determined as follows :

$$\alpha^p = \frac{T_p}{T_1}$$

$$\alpha = \exp\left(\log\left(T_p/T_1\right)/p\right)$$

At the maximal temperature one accepts with a high probability β_p an important deterioration δ_p of the initial solution L^*_{max}.

At the ending temperature one accepts with a weak probability β_1 a little deterioration δ_1 of the initial solution.

The neighbouring configuration σ^v in our case is then obtained by switching, in a given iteration, two random jobs according to the simulated annealing method.

The maximum lateness is carried out taking into account simultaneously the current sequence and the new tested one obtained by switching jobs as mentioned above.

The initial solution is provided from the last jobs' sequencing : L^*_{max}

Algorithm:

fix a parameter T at an elevated value (the temperature), $T = T_p$

initial configuration : $L^a_{max} = L^*_{max}$

repeat:

for a given T, repeat a certain number of time $(= q)$:

put $\sigma_a = \sigma_v, L^a_{max} = L^v_{max}$

randomly generate a neighbouring configuration σ_v of the current one σ_a. put $\Delta L_{max} = L^v_{max} - L^a_{max}$

accept σ_v as a new current configuration with the probability :

$$P = \begin{cases} 1 & \text{if } \delta L_{max} < 0 \\ \exp\left((L^v_{max} - L^{best}_{max}).\delta_i . \log \beta_1/\alpha^i.\delta_1\right) & \text{otherwise} \end{cases}$$

This heuristic method does not require an important computation time, the number of landings (p) and of tests by landing (q) is determined relative to the data size, the desired accuracy of the solution and its computation time.
To illustrate this approach let us consider an example of a production line with four machines and eight jobs to be processed in the same order of processing, the buffers are supposed null.
We consider the initial solution minimizing the maximum job lateness, and thereafter a perturbation is introduced in the processing time of job J_1 during its operation onto machine M_1.

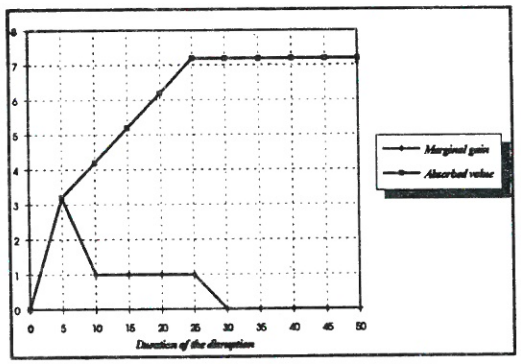


Figure 9: Variation of the Absorbed Perturbation Using the Simulated Annealing Method

The duration of the perturbation is varied from 0 to 50 by a step of 5 unit time, and every time the new maximum lateness given by the simulated annealing method is collected.
The latter takes into account the old sequencing before the end of the operation O_{11} and the new sequencing (carried out by the simulated annealing) after it on each machine.
Figure 9 presents the variation of the lateness function and the amount of the perturbation absorbed by the new sequencing solution.

6 Scheduling Robustness Analysis

Consider a flow shop system with limited buffers, and suppose that the jobs dispatching times are represented by a $nxm-$matrix $S(i,j)$, $i = 1,\ldots,n$, $j = 1,\ldots,m$, the scheduling criterion is given by the function f. If no disruption occurs, this function is valued f_0.
According to these notations, the variation $\delta(S)$ of the objective function holds :

$$\delta(S) = f(S) - f_0(S)$$

The scheduling robustness represents the expected variation of the objective function for a random disruption characterized by two parameters : the density function of its duration $g(\tau)$ and the density function of its arrival time $h(\lambda)$ [16].
The delay function $R(\tau,\lambda)$ gives the value of the scheduling criterion variation due to any disruption represented by (τ,λ) on a particular machine M_j (Figure 10).

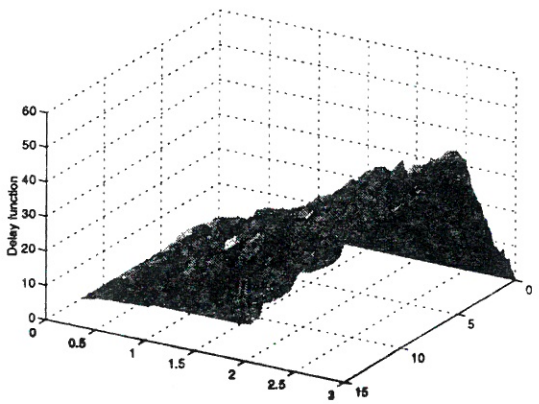


Figure 10: Delay Function With Respect to the Lateness Criterion

The expected delay then holds :

$$E[\delta(S)] = \int_{\tau=0}^{\infty} \int_{\lambda=0}^{\infty} R(\tau,\lambda)\; g(\tau)\; h(\lambda)\; d\tau\; d\lambda$$

This robustness measure is related to a particular machine M_j, and in order to obtain a similar measure for the scheduling solution on all the

system, we need combine the partial obtained solutions. This procedure is not discussed in this paper.

Definition 1

Let f be a regular scheduling criterion and R its associated delay function. The scheduling solution is represented by the matrix S.
Then, the margin corresponding to the task of the job J_i on machine M_j is given by :

$$e_f(i,j) = \max_{\alpha \in U} \alpha$$

with

$$U = \{\alpha \in \Re^+ \ tq \quad R((i,j) \leftarrow \alpha) = R(0) \ \}$$

In order to analyse the scheduling robustness, let us consider two particular cases :

• $\lambda = \overline{\lambda}$ constant :

We suppose that the perturbation duration is fixed, but its arrival time is a randomly distributed variable according to the density function $g(\tau)$.
For the maximum lateness criterion, the delay function R depends on the starting times, and is formulated according to a given machine M_j as follows :

$R(\tau, \lambda) = \tau + \overline{\lambda} - S_f(i,j) - e_f(i,j)$ for $\tau \in [a_i, b_i]$
with
$S_f(i,j)$: starting time of the operation of job i on M_j with respect to the criterion f.
$a_i = \max\{S_f(i,j) + e_f(i,j),$
$\qquad S_f(i-1,j) + p(i-1,j)\}$
$b_i = S_f(i,j) + p(i,j) + c_f(i,j)$
The definition of the operation margin, in this case, is illustrated in Figure 11.

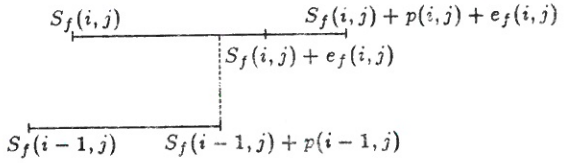


Figure 11: Task Margin of O_{ij} with Respect to the Lateness Criterion

A simple form of the robustness measure then holds :

$$E[\delta(\sigma)] = \sum_{i \in N} \int_{\lambda=0}^{\infty} \int_{\tau=a_i}^{b_i} R(\tau, \overline{\lambda})\, g(\tau)\, h(\lambda)\, d\tau\, d\lambda$$
$$= \sum_{i \in N} \underbrace{\int_{\lambda=0}^{\infty} h(\lambda)\, d\lambda}_{1} \int_{\tau=a_i}^{b_i} R(\tau, \overline{\lambda})\, g(\tau)\, d\tau$$

$$\boxed{E[\delta(S)] = \textstyle\sum_{i \in N} \int_{\tau=a_i}^{b_i} R(\tau, \overline{\lambda})\, g(\tau)\, d\tau}$$

where N is the set of operations on machine M_j.

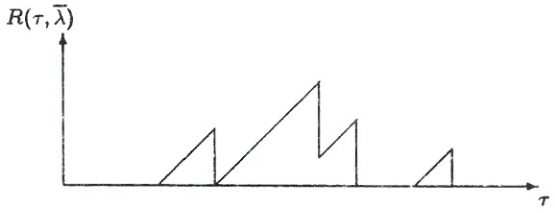


Figure 12: Delay Function with Constant Perturbation Duration

• $\tau = \overline{\tau}$ constant :
In this situation, the arrival time is supposed to be a constant. If one considers a given machine M_j, the latter date corresponds in fact to a particular operation of a certain job i^*.
In order to determine the expected delay due to this disruption, it is important to distinguish two subcases, depending on whether the duration of the disruption is greater than the operation margin of $O_{i^*,j}$.

(i) $0 < \lambda \leq e_f(i^*, j)$: the disruption is completely absorbed.

(ii) $\lambda > e_f(i^*, j)$: there is an effective delay according to the original solution.

$$E[\delta(S)] = \underbrace{\int_{\tau=0}^{\infty} g(\tau)\, d\tau}_{1} \int_{\lambda=0}^{\infty} R(\overline{\tau}, \lambda)\, h(\lambda)\, d\lambda$$

Taking into account the definition of the operation margin, the expected delay can be written simply as the following form :

$$\boxed{E[\delta(S)] = \int_{\lambda=e_f(i^*,j)}^{\infty} R(\overline{\tau},\lambda)\, h(\lambda)\, d\lambda}$$

For the makespan criterion, the delay function has a linear form for $\lambda > e_f(i^*,j)$ as indicated in Figure 12.

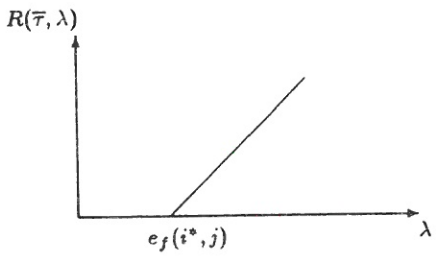


Figure 13: Delay Function with Constant Perturbation Arrival Time

Application

Let us consider a delay function R represented in Figure 14, wherein the duration of the disruption is a constant $\bar{\lambda} = 4$.
The variation of the function r with respect to the arrival time τ over the scheduling horizon $[0, 12]$.
The scheduling robustness measure is expressed as follows :

$$E[\delta(S)] = \int_{\tau=0}^{12} R(\tau,\overline{\lambda})\, g(\tau)\, d\tau$$

One supposes that the density function g of the disruption is an exponentiel distribution of mean ω $(exp(1/\omega))$.
The function $R(\tau,\overline{\lambda})\, g(\tau)$ is represented in Figure 15 for some values of ω. The latter shows that when the parameter ω increases, $R(\tau,\overline{\lambda})\, g(\tau)$ gradually becomes close to the form of $R(\tau,\overline{\lambda})$ and $g(\tau)$ looks as a constant function.

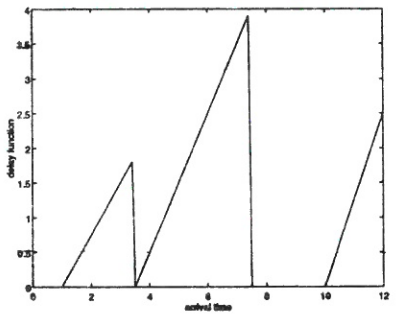


Figure 14: Delay Function $R(\tau,\bar{\lambda})$ with Constant Duration of the Disruption

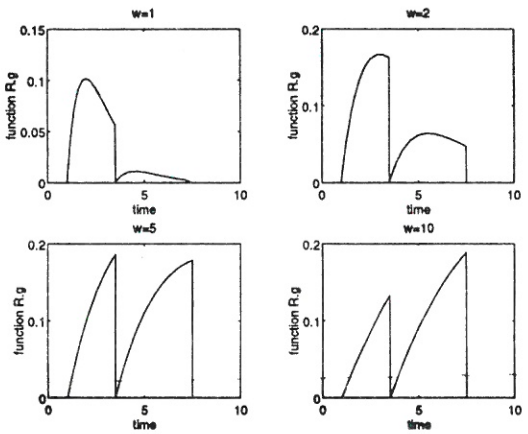


Figure 15: Function $R(\tau,\bar{\lambda})\, g(\tau)$ for $\omega = 1, 2, 5, 10$

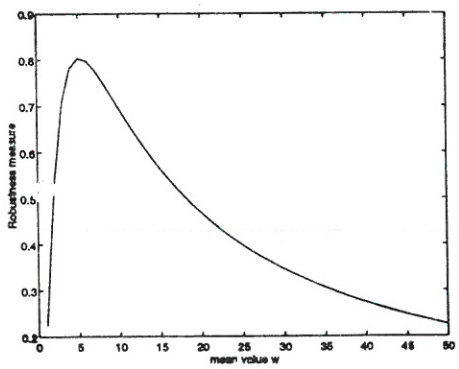


Figure 16: Variation of the Robustness Measure With Respect to ω

The variation of the robustness measure with respect to the parameter of the density function ω in Figure 16 shows that the scheduling policy which the delay function $R(\tau, \bar{\lambda})$ is associated with reaches its worst robustness (i.e. $\max_{\omega} E[\sigma(S)]$) for a disruption with density function $exp(1/\omega_0)$ with $\omega_0 = 5.2$.

7 Conclusion

The purpose of this paper concerns the flow line systems with disturbances. A solution methodology is proposed to resolve the scheduling problem under static conditions. The simulations show the results effectiveness.

Under the deterministic similarity assumption between the sample and perturbed path, the rules of perturbation propagation were pointed out.

All the mentioned rules are simple and need no significant computing time. This fact is important because otherwise the system evolution may be stopped.

In addition when the perturbation becomes more and more important, we are able to determine at what time the sequence ought to be reviewed.

The new solution is carried out by means of a simulated annealing based method. Its associated computing time is relatively weak and therefore suitable for on-line control of the system.

The robustness property of scheduling policies due to disruptions is analysed, the latter consists of examining the delay function $R(\tau, \lambda)$ associated with the problem for a simple disruption characterized by its duration $g(\tau)$ and its arrival time $h(\lambda)$.

Because of the complication of the robustness measure expression for the discrete systems, we just studied this property for only one of two parameters (duration or arrival time).

An exponentially distributed duration of the disruption is considered for illustration.

REFERENCES

[1] BENLOGAB, A. and DESCOTES-GENON, **Minimizing Maximum Lateness in Flowshops with Finite Buffer Storage**, 2-nd IFAC/IFIP/IFORS Workshop IMS'94, Vienna, Austria, April 1994.

[2] BENLOGAB, A. and DESCOTES - GENON, B., **Sequencing Flowshops with Arbitrary Intermediate Storage**, IEEE/ISIC'94, Columbus, Ohio, USA, 15-18 August 1994.

[3] BENLOGAB, A. and DESCOTES - GENON, B., **Scheduling and Rescheduling in Flowshops with Finite Capacity Storage**, IFAC/LSS'95, London, United Kingdom, July 1995.

[4] CAO, X.-R. and HO, Y.-C., **Sensitivity Analysis and Optimisation of Throughput in a Production Line with Blocking**, IEEE TRANSACTIONS ON AUTOMATIC CONTROL, AC-32(11), 1987, pp. 959-967.

[5] DALLERY, Y. and GERSHWIN, S.B., **Manufacturing Flow Line Systems: A Review of Models and Analytical Results**, Technical Report, Laboratoire MASI/CNRS, March 1991.

[6] DUPONT, L., **Algorithmes et Ordonnancements**, Ph.D Thesis, INPG, France, 1986.

[7] ELMAGHRABY, S.E., **Handbook of Operations Research**, VAN NOSTRAND REINHOLD COMPANY, 1978.

[8] GERSHWIN, S.B., **Manufacturing Systems Engineering**, PTR PRENTICE HALL, 1994.

[9] HALL, N.G., POSNER, M.E. and POTTS, C.N., **Scheduling with Finite Capacity Output Buffers**, 1994.

[10] HO, Y.-C., **A Survey of the Perturbation Analysis of Discrete Event Dynamic Systems**, ANNALS OF OPERATIONS RESEARCH, No.3, 1985, pp. 393-402.

[11] HO, Y.-C., **Performance Evaluation and Perturbation Analysis of Discrete Event Dynamic Systems**, IEEE TRANSACTIONS ON AUTOMATIC

CONTROL, AC-32, 1987, pp. 563-572.

[12] HO, Y.-C. and CAO, X.-R., **Perturbation Analysis of Discrete Event Dynamic Systems**, KLUWER ACADEMIC PUBLISHERS, 1991.

[13] HO, Y.-C. and CASSANDRAS, C., **A New Approach to the Analysis of Discrete Event Dynamic Systems**, AUTOMATICA, No.19, 1983, pp. 149-167.

[14] ISHIBUCHI, H., MISAKI, S. and TANAKA, H., **Modified Simulated Annealing Algorithms for the Flow Shop Sequencing Problem**, EUR. J. OPER. RES., No.81, 1995, pp. 388-398.

[15] LEISTEIN, R., **Flowshop Sequencing Problems with Limited Buffer Storage**, INT. J. PROD. RES., 28(11), 1990, pp. 2085-2100.

[16] LEON, V.J., WU, S.D. and STORER, R.H., **Robustness Measures and Robust Scheduling for Job Shops**, IEEE TRANSACTIONS, 26(5), September, 1994, pp. 32-43.

[17] MACCARTHY, B.L. and LIU, J., **Addressing the Gap in Scheduling Research: A Review of Optimization and Heuristic Methods in Production Scheduling**, INT. J. PROD. RES., 31(1), 1993, pp. 59-79.

[18] PROUST, G., **De l'influence des idées de Johnson dans la résolution des problèmes d'ordonnancement de type flowshop**, Proc. of Summer School on Scheduling Theory and its Applications, Bonas, France, September 1992.

[19] RODAMMER, F.A. and WHITE, K.P., **A Recent Survey of Production Scheduling**, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 18(6), December 1988, pp. 841-851.

[20] SURI, R., **Perturbation Analysis: State of the Art and Research Issues Explained Via the GI/GI/1** , Proc. of the IEEE, 77(1), January 1989, pp. 114-137.