On-line Scheduling in Assembly Processes

Fabrice Chauvet, Jean-Marie Proth

To cite this version:

Fabrice Chauvet, Jean-Marie Proth. On-line Scheduling in Assembly Processes. [Research Report] RR-3395, INRIA. 1998, pp.14. inria-00073294

HAL Id: inria-00073294

https://inria.hal.science/inria-00073294

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On-line Scheduling in Assembly Processes

Fabrice Chauvet - Jean-Marie Proth

N° 3395
Mars 1998

THÈME 4

Rapport de recherche

Les rapports de recherche de l'INRIA
sont disponibles en format postscript sous
ftp.inria.fr (192.93.2.54)

si vous n'avez pas d'accès ftp
la forme papier peut être commandée par mail :
e-mail : dif.gesdif@inria.fr
(n'oubliez pas de mentionner votre adresse postale).

par courrier :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

INRIA research reports
are available in postscript format
ftp.inria.fr (192.93.2.54)

if you haven't access by ftp
we recommend ordering them by e-mail :
e-mail : dif.gesdif@inria.fr
(don't forget to mention your postal address).

by mail :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

On-line Scheduling in Assembly Processes

Fabrice CHAUVET* and Jean-Marie PROTH* and **

ABSTRACT

The assembly system under consideration is composed with several machines, and some of these machines may be identical or able to perform the same operations. The manufacturing system is fully automated and, semi-finished products or components are not stored during the process. A limited flexibility exist since the manufacturing times can be extended within certain limits at the expense of the unavailability of the resource. There are no conflicts between the resources; in other words, the same machine cannot be used to perform different operations for a same product. Due to the intensity of the flow of products to be manufactured, it is not allowed to reschedule products which have been previously scheduled. Thus, when a new product requirement arrives in the system, we have to take advantage of the idle time windows. The goal is to complete the product as soon as possible. We give a real-time scheduling algorithm which guarantees an optimal makespan to any product which arrives in the assembly system. A numerical example is provided to illustrate this approach.

KEYWORDS
Makespan optimization, Real-time scheduling, On-line scheduling, Assembly system.

* INRIA Lorraine, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, France
** Institute for Systems Research, University of Maryland, College Park, MD 20742, USA

Processus d'assemblage en temps réel

Fabrice CHAUVET* and Jean-Marie PROTH* and **

RESUME

Le système d'assemblage considéré est composé de plusieurs machines. Certaines de ces machines peuvent être identiques ou différentes mais en mesure d'exécuter les mêmes opérations. Le système est automatisé, et il n'est pas possible de stocker des composants ou des produits semi-finis au cours de la fabrication. Une flexibilité limitée du système existe : il est possible d'étendre les temps opératoires dans certaines limites, mais dans ce cas la ressource concernée n'est pas disponible pour exécuter une autre opération. Il n'y a pas de conflit entre les ressources ; en d'autres termes, une même machine n'est jamais utilisée pour exécuter plus d'une opération sur le même produit. Du fait de la fréquence d'arrivée des produits à fabriquer, il n'est pas possible de remettre en question l'ordonnancement des produits déjà ordonnancés. Par conséquent, lorsqu'une nouvelle commande arrive dans le système, il faut tirer avantage au mieux des périodes de vacuité des ressources. L'objectif est d'exécuter la commande au plus tôt. Nous proposons un algorithme qui garantit l'obtention d'un temps de fin de fabrication (makespan) aussi court que possible. Cet algorithme est illustré par un exemle numérique.

MOTS CLES

Ordonnancement temps réel, makespan, système d'assemblage.

* INRIA Lorraine, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, France
** Institute for Systems Research, University of Maryland, College Park, MD 20742, USA

1. INTRODUCTION

The assembly system considered in this paper is fully automated. Customers' requirements stream in the system, and each new requirement should be scheduled as soon as it enters the system, the goal being to reduce its completion time as much as possible. In other words, the goal is to minimize the makespan of each product considered individually. Due to the importance of the production flow, it is impossible to reschedule a product previously scheduled. In other words, when a customer's requirement arrives in the system, and this can happen at any time, the goal is to take advantage of the idle periods available on the resource to complete the product as soon as possible.

Two types of operation have to be performed, that is the processing operations, which transform the state of components, and the assembly operations, which assemble two or more components, in order to obtain another component or the finished product.

We assume that a component is not allowed to wait in front of a resource: the next operation to be performed on the component should start as soon as the current ends. This constraint is very strong, mainly since several component which have to be assembled should be completed exactly at the same time. This constraint aims at reducing the work-in-process (WIP) as much as possible.

Nevertheless, limited flexibility exists: the processing time associated with a given operation may be extented within certain limits, but the resource on which an operation is performed is unavailable for another operation as long as the current operation is not completed. In particular, we can find this kind of limited flexibility in chemical processes.

Numerous papers have been dedicated to no-wait or blocking problems. In these problems, it is assumed that no intermediate buffers exist between the resources.

Callahan [1971] uses a queuing model to study a no-wait processing in the steel industry. Chu and all [to appear], and Chauvet and all [1997] consider the surface treament which is a no-wait problem. McCormick and all [1989] study a cyclic flowshop with buffers which can be transformed into a blocking problem by considering buffers as resources with totally flexible processing times, i.e. processing times which can take any value between 0 and $+\infty$. Hall and Sriskandarajah [1996] make a survey on scheduling problems with blocking and no-wait, while Rachamadugu and Stecke [1994] classify the FMS scheduling procedures. This paper should be considered as an intermediate approach: some flexibility exists in the model, but this flexibility can be limited.

2. NOTATIONS

Each operation of the assembly process is defined by its minimal and maximal processing times. For instance, operation i is characterized by θ_i, minimal processing time, and $\theta_i + \delta_i$, maximal processing time. Two kinds of operations can be found in an assembly process, that is:

(i) The processing operations, which transform the state of a component or a semi-finished product. A processing operation has at most one predecessor and one successor.

(ii) The assembly operations, which assemble several components. The last operations performed on these components are the predecessors of the assembly operation. In an assembly process, an assembly operation has at most one successor. It has no successor when the result of the assembly operation is the finished product.

Each operation i is characterized by the set of its predecessors denoted by Γ_i^- and the set of its successor denoted by Γ_i^+. Note that, in an assembly process, Γ_i^- may contain several elements or may be empty. Γ_i^+ contains at most one element.

Thus, an assembly process **A** is characterized by the list $\{1, 2, \ldots, n\}$ of its operations, each operation $i \in \{1, 2, \ldots, n\}$ being characterized by four elements: $\{\theta_i, \delta_i, \Gamma_i^-, \Gamma_i^+\}$. We denote by v_i the starting time of operation i, $i \in \{1, 2, \ldots, n\}$. Since operation j where $j \in \Gamma_i^+$ and $\Gamma_i^+ \neq \varnothing$, should start as soon as the current operation i ends, v_j is the completion time of operation i.

3. PROBLEM FORMULATION

The manufacturing system is composed with several machines, and some of these machines may be identical or able to perform the same operations. Since other products have been scheduled previously, the machines are partially busy. The manufacturing system being fully automated, semi-finished products or components are not stored during the process. The only flexibility available in the system is given by the δ_i' values, which express the fact that manufacturing times can be extented. This is in some way equivalent to the storage of semi-finished products or components, but makes the machines unavailable for other operations.

For each operation i, we have a series of available windows denoted by: $[\alpha_1^i, \beta_1^i]$, $[\alpha_2^i, \beta_2^i]$, ..., $[\alpha_{q_i}^i, \beta_{q_i}^i]$. These windows may be available on different machines. They are ordered, for each operation i, in the increasing order of the α_k^i, $k = 1, 2, \ldots, q_i$. When two α_k^i are equal, the order is the increasing order of the β_k^i. There are no conflicts between the resources; in other words, the same machine cannot be used to perform different operations for a same product. The last window q_i associated with each operation i, is such that $\beta_{q_i}^i = +\infty$.

A feasible solution to the problem associated with a set $[\alpha_{r_i}^i, \beta_{r_i}^i]$ of idle windows (where $i = 1, 2, \ldots, n$ and $r_i = 1, 2, \ldots, q_i$) is a set of operation starting times $\{v_i\}_{i=1,2,\ldots,n}$ which verify, for $i = 1, 2, \ldots, n$:

$$\begin{cases} \alpha_{r_i}^i \leq v_i & (1) \\ v_i + \theta_i \leq \beta_{r_i}^i, \text{ if } \Gamma_i^+ = \varnothing & (2) \\ v_j \leq \beta_{r_i}^i, \ \forall j \in \Gamma_i^+, \text{ if } \Gamma_i^+ \neq \varnothing & (3) \\ v_i + \theta_i \leq v_j, \ \forall j \in \Gamma_i^+, \text{ if } \Gamma_i^+ \neq \varnothing & (4) \\ v_j \leq v_i + \theta_i + \delta_i, \ \forall j \in \Gamma_i^+, \text{ if } \Gamma_i^+ \neq \varnothing & (5) \end{cases}$$

Note that inequalities (4) and (5) allowed to assign to operation i a processing time $w_i \in [\theta_i, \theta_i + \delta_i]$ such that $v_i + w_i = v_j$, $j \in \Gamma_i^+$. Note also that if $\Gamma_i^+ = \emptyset$, then $w_i = \theta_i$ since the goal consists in minimizing the makespan. A feasible solution is optimal if $\max_{i/\Gamma_i^+=\emptyset}(v_i + \theta_i)$ is minimal.

The goal is to schedule the operations required to manufacture a product P whose assembly process is **A**, in order to minimise the makespan, i.e. to complete the product as soon as possible. This scheduling has to be performed as soon as the product arrives in the system. Due to the intensity of the flow of products, it is not allowed to reschedule products which have been previously scheduled.

NB. The scheduling problems with no-wait and blocking are two particular cases of the problem presented in this paper. In the first case (no-wait problem) $\delta_i = 0$ while $\delta_i = +\infty$ in the blocking problem.

4. COMPUTATION OF AN OPTIMAL SOLUTION: ALGORITHM 1

The idea behind the algorithm is quite simple and can be explained considering figure 1. Let us assume that the window assigned to each operation is known.

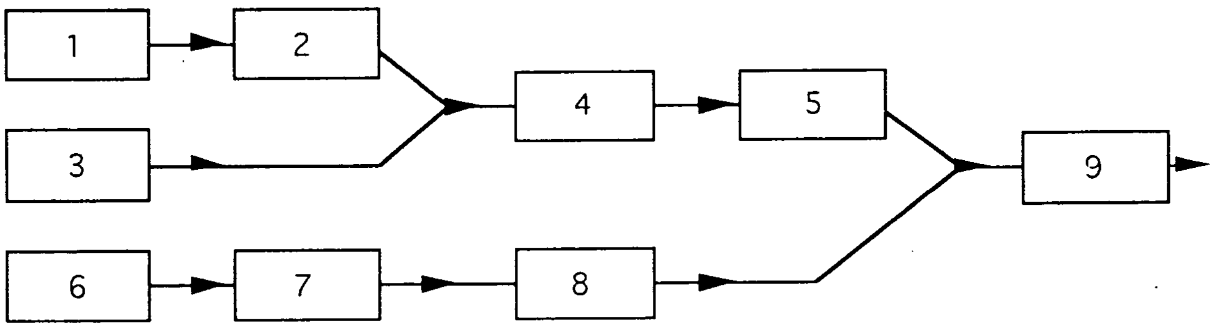


Figure 1. An assembly process composed by 9 operations

In this assembly process, there are two assembly operations: operation 4 and operation 9, the other ones being processing operations. We call "branch" a series of operations which are succesors or predecessors from each other, each operation of a branch having at most one predecessor if its predecessor belong to the same branch. The process represented in figure 1 has five branches: $B_1=\{1, 2\}$, $B_2=\{3\}$, $B_3=\{4, 5\}$, $B_4=\{6, 7, 8\}$ and $B_5=\{9\}$. B_1 and B_2 are called "predecessor branchs" of B_3.

To schedule this process, we start considering branchs which are predecessors of the same branch and which either have no predecessor, or whose predecessors are already been considered. In this case, we first consider B_1 and B_2 and we compute, for each of them, the minimal completion time. The maximal value of these times is the minimal starting time of B_3. Assume that the window where operation i should be located is $[\alpha_{r_i}^{i}, \beta_{r_i}^{i}]$, for $i \in \{1, 2, \ldots, 9\}$.

We compute successively u_i, lower bound of the minimal starting time of operation i, for $i \in \{1, 2, ..., 9\}$:

- operation 1: $u_1 = \alpha^1_{r_1}$,
- operation 2: $u_2 = \max(\alpha^2_{r_2},\ u_1 + \theta_1)$,
- operation 3: $u_3 = \alpha^3_{r_3}$,
- operation 4: $u_4 = \max(\alpha^4_{r_4},\ u_2 + \theta_2,\ u_3 + \theta_3)$,
- operation 5: $u_5 = \max(\alpha^5_{r_5},\ u_4 + \theta_4)$,
- operation 6: $u_6 = \alpha^6_{r_6}$,
- operation 7: $u_7 = \max(\alpha^7_{r_7},\ u_6 + \theta_6)$,
- operation 8: $u_8 = \max(\alpha^8_{r_8},\ u_7 + \theta_7)$,
- operation 9: $u_9 = \max(\alpha^9_{r_9},\ u_5 + \theta_5,\ u_8 + \theta_8)$.

Thus, the minimal completion time of operation 9 is $u_9 + \theta_9$. The optimal makespan if the set of windows under consideration can lead to a feasible solution is $z = u_9 + \theta_9$. We then apply a backward process starting from u_9. The goal of this backward process is to define the starting times of the operations, keeping in mind that the completion time of one operation is the starting time of its succecessors. So, we obtain successively v_i, the minimal starting time of operation i, for $i \in \{1, 2, ..., 9\}$:

- operation 9: $v_9 = u_9$,
- operation 5: $v_5 = \max(u_5,\ v_9 - \theta_5 - \delta_5)$,
- operation 4: $v_4 = \max(u_4,\ v_5 - \theta_4 - \delta_4)$,
- operation 2: $v_2 = \max(u_2,\ v_4 - \theta_2 - \delta_2)$,
- operation 1: $v_1 = \max(u_1,\ v_2 - \theta_1 - \delta_1)$,
- operation 3: $v_3 = \max(u_3,\ v_4 - \theta_3 - \delta_3)$,
- operation 8: $v_8 = \max(u_8,\ v_9 - \theta_8 - \delta_8)$,
- operation 7: $v_7 = \max(u_7,\ v_8 - \theta_7 - \delta_7)$,
- operation 6: $v_6 = \max(u_6,\ v_7 - \theta_6 - \delta_6)$.

If the operation i cannot be completed before $\beta^i_{r_i}$, we replace window $[\alpha^i_{r_i},\ \beta^i_{r_i}]$ by $[\alpha^i_{r_i+1},\ \beta^i_{r_i+1}]$, and we restart the process. Better, we can replace the window $[\alpha^i_{r_i},\ \beta^i_{r_i}]$ by the first window $[\alpha^i_{r_i+k_i},\ \beta^i_{r_i+k_i}]$ (i.e. k_i is as low as possible and $k_i \geq 1$) such that the operation i can be completed before $\beta^i_{r_i+k_i}$.

Otherwise the v_i' values are the solution of the problem. This simple example illustrate the general algorithm proposed hereafter.

Algorithm 1 - Assembly

```
1. Set $r_i = 1$,   for   $i \in \{1, 2, ..., n\}$
2. Foreward process
    2.1. Set $E = \{1, 2, ..., n\}$
    2.2. While $E \neq \varnothing$
        2.2.1. Select $i \in E$ such that $\Gamma_i^- \cap E = \varnothing$
        2.2.2. If $\Gamma_i^- = \varnothing$ then set $u_i = \alpha_{r_i}^i$, else set $u_i = \max\left(\alpha_{r_i}^i,\ \max_{j\in\Gamma_i^-}(u_j + \theta_j)\right)$
        2.2.3. Set $E = E \backslash \{i\}$
    2.3. Set $z = \max_{i/\Gamma_i^+=\varnothing}(u_i + \theta_i)$, z is the optimal makespan if there exists a feasible
         solution for the selected windows
3. Backward process
    3.1. Set $E = \{1, 2, \dots, n\}$
    3.2. While $E \neq \varnothing$
        3.2.1. Select $i \in E$ such that $\Gamma_i^+ \cap E = \varnothing$
        3.2.2. If $\Gamma_i^+ = \varnothing$ then set $v_i = u_i$, else set $v_i = \max\left(u_i,\ \max_{i\in\Gamma_i^-}(v_j) - \theta_i - \delta_i\right)$
        3.2.3. Set $E = E \backslash \{i\}$
4. Test
    4.1. Set $E = \{1, 2, \dots, n\}$
    4.2. For any $i \in E$ such that $\Gamma_i^+ = \varnothing$,
        4.2.1. While $v_i + \theta_i > \beta_{r_i}^i$, set $r_i = r_i + 1$
    4.3. For any $i \in E$ such that $\Gamma_i^+ \neq \varnothing$,
        4.3.1. Set j, such that $\{j\} = \Gamma_i^+$
        4.3.2. While $v_j > \beta_{r_i}^i$, set $r_i = r_i + 1$
    4.4. If none of the $r_i$ has been modified, then z is the minimal makespan and $v_i$ are the
         earliest starting time of the operations corresponding to z, else return to step 2.0.
```

Results 1 and 2 presented hereafter guarantee that the solution obtained by applying this algoritm is optimal.

Result 1

Algorithm 1 leads to a feasible solution.

Proof:

a. We prove that $\alpha_{r_i}^i \leq v_i,\ \forall i \in \{1, 2, \dots, n\}$.

According to step 2.2.2 in algorithm 1, $\alpha_{r_i}^i \leq u_i,\ \forall i \in \{1, 2, \dots, n\}$. Furthermore, with considering step 3.2.2, $u_i \leq v_i,\ \forall i \in \{1, 2, \dots, n\}$. (6)

Finally: $\alpha_{r_i}^i \leq v_i$: this proves inequalities (1).

b. We prove that $v_i + \theta_i \leq v_j,\ \forall i \in \{1, 2, \dots, n\}\ \forall j \in \Gamma_i^+$.

According to step 3.2.2 in algorithm 1, for any $i = 1, 2, \dots, n$ such that $\Gamma_i^+ \neq \varnothing$, $v_i + \theta_i = \max\left(u_i + \theta_i,\ \max_{i\in\Gamma_i^-}(v_j) - \delta_i\right)$. Thus, $v_i + \theta_i \leq \max\left(u_i + \theta_i,\ \max_{j\in\Gamma_i^+}(v_j)\right)$. (7)

But according to step 2.2.2, whatever $j \in \Gamma_i^+$, $u_i + \theta_i \leq u_j$. (8)

Using inequality (8) in (7), we obtain: $v_i + \theta_i \leq \max\left(\max_{j\in\Gamma_i^+}(u_j),\ \max_{j\in\Gamma_i^+}(v_j)\right)$ and using (6), $v_i + \theta_i \leq v_j$, $\forall j \in \Gamma_i^+$: this proves inequalities (2).

c.. We prove that $v_j \leq v_i + \theta_i + \delta_i$, $\forall i \in \{1, 2, \ldots, n\}$ $\forall j \in \Gamma_i^+$.

From step 3.2.2 in algorithm 1, we derive: $v_i + \theta_i + \delta_i = \max\left(u_i + \theta_i + \delta_i,\ \max_{j\in\Gamma_i^+}(v_j)\right)$. As a consequence: $v_j \leq v_i + \theta_i + \delta_i$, $\forall j \in \Gamma_i^+$: this proves inequalities (3).

d. We prove that $v_i + \theta_i \leq \beta^i_{r_i}$, $\forall i \in \{1, 2, \ldots, n\}$ such that $\Gamma_i^+ = \varnothing$.

Since we set $r_i = r_i + 1$, when $v_i + \theta_i > \beta^i_{r_i}$ (see step 4.2.1 in algorithm 1), and since the last window q_i associated with each operation i, is such that $\beta^i_{q_i} = +\infty$, then the first solution obtained verifies: $v_i + \theta_i \leq \beta^i_{r_i}$: this proves inequalities (4).

e. We prove that $v_j \leq \beta^i_{r_i}$, $\forall i \in \{1, 2, \ldots, n\}$ $\forall j \in \Gamma_i^+$.

Since we set $r_i = r_i + 1$, when $v_j > \beta^i_{r_i}$ (see step 4.3.2 in algorithm 1), and since the last window q_i associated with each operation i, is such that $\beta^i_{q_i} = +\infty$, then the first solution obtained verifies: $v_j \leq \beta^i_{r_i}$: this proves inequalities (5).

QED

Result 2

In the first feasible solution obtained by applying algorithm 1, all the completion times, and thus the starting times of the operations are minimal. As a consequence, this solution is optimal.

Proof:

a. For a given set of idle windows, we prove that, if it exits a feasible solution, the solution obtained by applying steps 2 and 3 in algorithm 1 is such that no other solution leads to a set of completion times which are lower.

Considering step 2.2.2 in algorithm 1, and keeping in mind that θ_i is the minimal processing time for operation i, we see that the u_i' values cannot be reduced. As a consequence, considering step 3.2.2, and keeping in mind that $\theta_i + \delta_i$ is the maximal processing time for operations i, we claim that the v_i' values cannot be reduced either.

As a consequence if the solution obtained by applying step 2 and 3 of algorithm 1 to a given set of windows $\left\{\left[\alpha^i_{r_i},\ \beta^i_{r_i}\right]\right\}_{i=1,2,\ldots,n}$ does not verify constraint (2), or (3) for at least one $i \in \{1, 2, \ldots, n\}$, no feasible solution exists for this set of windows. Furthermore, if the solution obtained is feasible, it is composed with the lower possible starting times of the operations. A consequence of this proof in that the completion times of the operations can not be reduced since they are starting times of the next operation. The ones which do not have a successor are also minimal since their processing time is set at this minimal value.

This completes the first part of the proof.

b. We prove that the first feasible solution obtained by applying algorithm 1 is optimal.

If the solution derived from the set of windows having the minimal lower bounds $\alpha^i_{r_i}$ is feasible, it is optimal, and the starting time of the operations are minimal for this set of windows (see a.). Furthermore, since the completion times increase with the $\alpha^i_{r_i}$' values, the solution is optimal and the starting time are minimal for any set of idle windows $\left\{\left[\alpha^i_{r_i}, \beta^i_{r_i}\right]\right\}_{i=1, 2, \ldots, n}$.

Moreover, if the operation i is such that $\Gamma_i^+ \neq \emptyset$, $\{j\} = \Gamma_i^+$ and $v_j > \beta^i_{r_i}$, then any set of windows containing $\left[\alpha^i_{r_i}, \beta^i_{r_i}\right]$ will lead to the same inequality and thus will not lead to a feasible solution. In the same way, if the operation i is such that $\Gamma_i^+ = \emptyset$ and $v_i + \theta_i > \beta^i_{r_i}$, then any set of windows containing $\left[\alpha^i_{r_i}, \beta^i_{r_i}\right]$ will not lead to a feasible solution.

This completes the proof.

QED

Note that step 4 in algorithm 1 is executed at most q times, where $q = \sum_{i=1}^{n} q_i$ is the number of idle windows. Thus, the complexity of the algorithm 1 is o(n.q).

5. REDUCTION OF THE CYCLE TIME: ALGORITHM 2

The optimal solution obtained using algorithm 1 does not lead to a minimal use of the resources. The goal of this section is to show how to proceed to reduce the cycle time and, as a consequence, the global use of the resources.

We propose algorithm 2 and we prove that the cycle time obtained by applying this algorithm is minimal.

Algorithm 2 - Reduction of the cycle time

We order the windows $\left[\alpha^i_{s_i}, \beta^i_{s_i}\right]$, for each operation i, in the increasing order of the β^i_k, $k = 1, 2, \ldots, n_i$. When two β^i_k are equal, the order is the decreasing order of the α^i_k. The v_i' values are those computed in algorithm 1.

```
1. Set $s_i = n$,   for   $i \in \{1, 2, \ldots, n\}$
2. Foreward process
    2.1. Set $E = \{1, 2, \ldots, n\}$
    2.2. While $E \neq \emptyset$
        2.2.1. Select $i \in E$ such that $\Gamma_i^- \cap E = \emptyset$
        2.2.2. If $\Gamma_i^- = \emptyset$ then set $w_i = +\infty$, else set $w_i = \min_{k \in \Gamma_i^-}(x_k)$
        2.2.3. If $\Gamma_i^+ = \emptyset$ then set $x_i = \min\left(v_i + \theta_i, \beta^i_{s_i}, w_i + \theta_i + \delta_i\right)$,
               else $x_i = \min\left(\beta^i_{s_i}, w_i + \theta_i + \delta_i\right)$
        2.2.4. Set $E = E \backslash \{i\}$
3. Backward process
    3.1. Set $E = \{1, 2, \ldots, n\}$
    3.2. While $E \neq \varnothing$
        3.2.1. Select $i \in E$ such that $\Gamma_i^+ \cap E = \varnothing$
        3.2.2. If $\Gamma_i^+ = \varnothing$ then set $y_i = \min(x_i - \theta_i,\ w_i)$,
               else $y_i = \min\left(\min_{j \in \Gamma_i^+}(y_j) - \theta_i,\ x_i - \theta_i,\ w_i\right)$
        3.2.3. Set $E = E \setminus \{i\}$
4. Test
    4.1. Set $E = \{1,\ 2,\ \ldots,\ n\}$
    4.2. For any $i \in E$
        4.2.1. While $y_i < \alpha^i_{s_i}$, set $s_i = s_i - 1$
    4.3. If none of the $s_i$ has been modified, then $y_i$ are the latest starting time of the
         operations corresponding to z, else return to step 2.0.
```

Result 3

Algorithm 2 transforms the optimal solution obtained by applying algorithm 1 into the optimal one in which the starting time of the operations are maximal. In other words, the solution obtained by applying algorithm 2 is optimal and minimize the use of resources.

Proof:

a. We prove that $\alpha^i_{s_i} \le y_i$, $\forall i \in \{1,\ 2,\ \ldots,\ n\}$.

Since we set $s_i = s_i - 1$, when $y_i < \alpha^i_{s_i}$ (see step 4.2.1 in algorithm 2), and since the window r_i associated with each operation i in algorithm 1 is such that a feasible solution exists, then the first solution obtained verifies: $\alpha^i_{s_i} \le y_i$: this proves inequalities (1).

b. We prove that $y_i + \theta_i \le \beta^i_{s_i}$, $\forall i \in \{1,\ 2,\ \ldots,\ n\}$ such that $\Gamma_i^+ = \varnothing$.

According to step 3.2.2 in algorithm 2, for any i = 1, 2, ..., n such that $\Gamma_i^+ = \varnothing$,
$y_i \le x_i - \theta_i$. (9)

Furthermore, considering step 2.2.3, $x_i \le \beta^i_{s_i}$.

Finally: $y_i + \theta_i \le \beta^i_{s_i}$ which proves inequalities (2).

c. We prove that $y_j \le \beta^i_{s_i}$, $\forall i \in \{1,\ 2,\ \ldots,\ n\}$ $\forall j \in \Gamma_i^+$.

According to step 2.2.3 in algorithm 2, $x_i \le \beta^i_{s_i}$, $\forall i \in \{1,\ 2,\ \ldots,\ n\}$.

Furthermore, considering step 2.2.2, $w_j \le x_i$, $\forall j \in \Gamma_i^+$. (10)

According to step 3.2.2, $y_j \le w_j$, $\forall j \in \{1,\ 2,\ \ldots,\ n\}$. (11)

Finally: $y_j \le \beta^i_{s_i}$, $\forall j \in \Gamma_i^+$: this proves inequalities (3).

d. We prove that $y_i + \theta_i \le y_j$, $\forall i \in \{1,\ 2,\ \ldots,\ n\}$ $\forall j \in \Gamma_i^+$.

From step 3.2.2 in algorithm 2, we derive: $y_i + \theta_i \le \min_{j \in \Gamma_i^+}(y_j)$. As a consequence: $y_i + \theta_i \le y_j$, $\forall j \in \Gamma_i^+$: this proves inequalities (4).

e. We prove that $y_j \le y_i + \theta_i + \delta_i$, $\forall i \in \{1,\ 2,\ \ldots,\ n\}$ $\forall j \in \Gamma_i^+$.

According to step 3.2.2 in algorithm 2, for any $i = 1, 2, \ldots, n$ such that $\Gamma_i^+ \neq \varnothing$, $y_i + \theta_i + \delta_i = \min\left(\min_{j \in \Gamma_i^-}(y_j) + \delta_i,\ x_i + \delta_i,\ w_i + \theta_i + \delta_i \right)$.

Thus, $\min\left(\min_{j \in \Gamma_i^-}(y_j),\ x_i,\ w_i + \theta_i + \delta_i \right) \leq y_i + \theta_i + \delta_i$. (12)

But according to step 2.2.3, $x_i \leq w_i + \theta_i + \delta_i$. (13)

Using inequality (13) in (12), we obtain: $\min\left(\min_{j \in \Gamma_i^-}(y_j),\ x_i \right) \leq y_i + \theta_i + \delta_i$ and using relations (10) and (11), $y_j \leq y_i + \theta_i + \delta_i$, $\forall j \in \Gamma_i^+$, since the successor j is unique: this proves inequalities (5).

At this point, since relations (1) - (5) are satisfied, we can claim that the solution $\{y_i\}_{i = 1, 2, \ldots, n}$ is feasible.

f. We prove that $y_i \leq v_i$, $\forall i \in \{1, 2, \ldots, n\}$ such that $\Gamma_i^+ = \varnothing$.

According to step 2.2.3 in algorithm 2, for any $i = 1, 2, \ldots, n$ such that $\Gamma_i^+ = \varnothing$, $x_i - \theta_i \leq v_i$. But according to inequality (9), $y_i \leq v_i$.

We know hat the v_i' values are minimal and optimal (see result 2). Since the y_i' values lead to a feasible solution, and since $y_i \leq v_i$ for any $i = 1, 2, \ldots, n$ such that $\Gamma_i^+ = \varnothing$, the criterion associated with $\{y_i\}_{i = 1, 2, \ldots, n}$ is at least as less as the criterion associated with $\{v_i\}_{i = 1, 2, \ldots, n}$. Thus $\{y_i\}_{i = 1, 2, \ldots, n}$ is an optimal solution.

g. For a given set of idle windows, we prove that the solution obtained by applying steps 2 and 3 in algorithm 2 is such that no other solution leads to a set of starting times which are greater.

Considering step 2.2.3 in algorithm 2, and keeping in mind that $\theta_i + \delta_i$ is the maximal processing time for operation i, we see that the x_i' values, which are upper bounds of the finishing times, cannot be increased. As a consequence, according to step 2.2.2, we see that the w_i' values, which are upper bounds of the starting times, cannot be increased. Thus, considering step 3.2.2, and keeping in mind that θ_i is the minimal processing time for operation i, we claim that the starting times y_i of operations i cannot be increased either.

h. We prove that the first feasible solution obtained by applying algorithm 2 is the optimal one in which the starting time of the operations are maximal.

If the solution derived from the set of windows having the maximal upper bounds $\beta_{s_i}^i$ is feasible, it is optimal, and the starting time of the operations are maximal for this set of windows (see g.). Furthermore, since the starting times decrease with the $\beta_{s_i}^i$' values, the solution is optimal and the starting time are maximal for any set of idle windows $\left\{\left[\alpha_{s_i}^i,\ \beta_{s_i}^i\right]\right\}_{i=1, 2, \ldots, n}$.

Moreover, if the operation i is such that $y_i < \alpha_{s_i}^i$, then any set of windows containing $\left[\alpha_{s_i}^i,\ \beta_{s_i}^i\right]$ will lead to the same inequality and thus will not lead to a feasible solution.

This completes the proof.

QED

Note that step 4 in algorithm 2 is executed at most q times, where $q = \sum_{i=1}^{n} q_i$ is the number of idle windows. Thus, the complexity of the algorithm 2 is o(n.q).

6. EXAMPLE

The assembly process represented in Figure 1 is composed with 9 operations. Each operation is defined by θ_i, its minimal processing time, and $\theta_i + \delta_i$, its maximal processing time. Since the system is fully automated, components cannot be stored in front of the resources. In this example, operations (except operations 6 and 7, which are chemical treatments) can be extented as much as necessary, but in this case, machines are unavailable for other operations until the current operation stops.

Table 1 - Processing times

i	1	2	3	4	5	6	7	8	n = 9
θ_i	3.	2.	2.	3.	2.	3.	2.	3.	2.
δ_i	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0.	0.	$+\infty$	$+\infty$

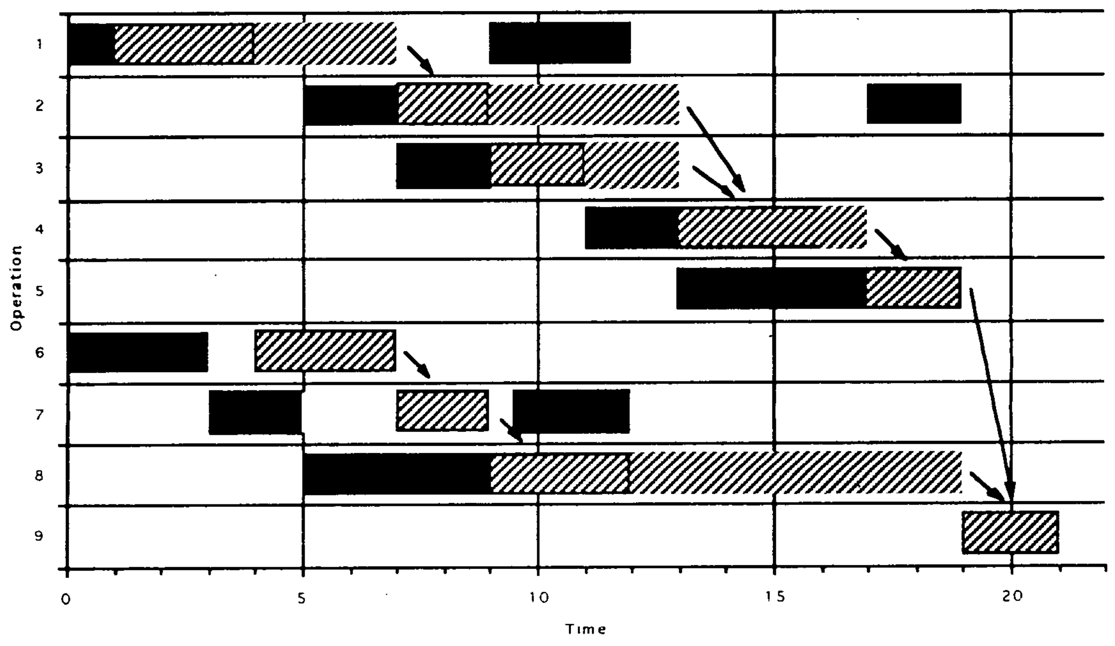
Each operation is assigned to a given machine. Since severals operations are previously scheduled, these 9 operations must be performed in one of the idle periods $\left[\alpha^i_{r_i}, \beta^i_{r_i}\right]$ available on the resource.

Table 2 - Available windows

i	1	2	3	4	5	6	7	8	9
q_i	2	3	2	2	2	1	3	2	1
$[\alpha^i_1, \beta^i_1]$	[1, 9]	[0, 5]	[0, 7]	[0, 11]	[0, 13]	[3, ∞[	[0, 3]	[0, 5]	[0, ∞[
$[\alpha^i_2, \beta^i_2]$	[12, ∞[	[7, 17[	[9, ∞[	[13, ∞[	[17, ∞[		[5, 9.5]	[9, ∞[	
$[\alpha^i_3, \beta^i_3]$		[19, ∞[					[12, ∞[		

By applying algorithm 1, we obtain the optimal solution in which the starting times of operations are minimal.

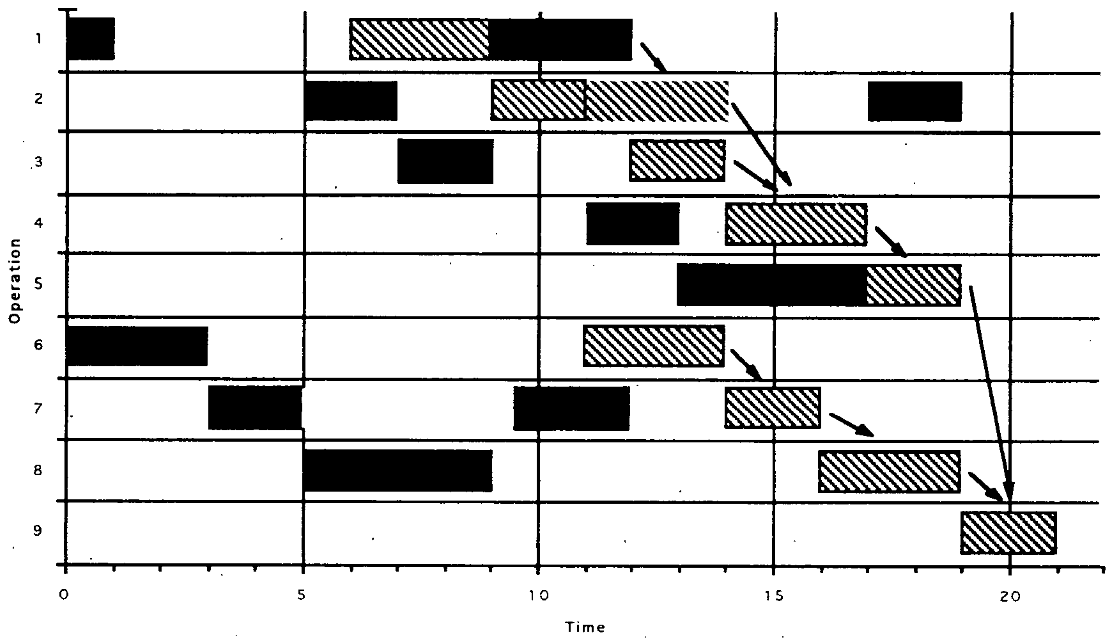
Chart 1 - Minimal starting times



Busy periods	■
Operations scheduled by applying algorithm 1 (minimal processing time)	▨
Operations scheduled by applying algorithm 1 (resulting processing time)	▨

By applying algorithm 2, we obtain the optimal solution in which the starting times of operations are maximal.

Chart 2 - Maximal starting times



Busy periods	■
Operations scheduled by applying algorithm 2 (minimal processing time)	▨
Operations scheduled by applying algorithm 2 (resulting processing time)	▨

As we can see in this example, algorithm 2 refine the schedule by pushing to the right, as much as possible, the starting times of the operations. The consequence is an overall reduction of the use of the resources.

7. CONCLUSION

The algorithms presented in this paper are real time scheduling algortihms which allow to schedule new assembly process as soon as possible. As a consequence, the first idle periods used are the ones which are the closest to the current time. This guarantees a good use of the resource. Furthermore, by controling the flexibility of the system (i.e. the δ_i' values), it is possible to reduce the WIP at the expense of the use of resources.

REFERENCES

[Callahan, 1971] CALLAHAN J.R., "*The Nothing Hot Delay Problems in the Production of Steel*", PhD. Thesis, Department of Industrial Engineering, University of Toronto, Canada, 1971

[Chauvet and all, 1997] CHAUVET F., LEVNER E., MEYZIN L.K., PROTH J.M., "*On-line Part Scheduling in a Surface Treatment System*", INRIA research reports, 1997, N. 3318, INRIA, Le Chesnay, France

[Chu and all, to appear] CHU C., PROTH J.M., WANG L., "*Improving job-shops schedules through critical pairwise exchanges*", International Journal of Production Research, to appear

[Hall and Sriskandarajah, 1996] HALL N.G., SRISKANDARAJAH C., "*A survey of machine scheduling problems with blocking and no-wait in process*", Operations Research, 1996, V. 44, pp. 510-525

[McCormick and all, 1989] MCCORMICK S.T., PINEDO M.L., SHENKER S., WOLF B., "*Sequencing in an Assembly Line with Blocking to MinimizeCycle Time*", Operations Research, 1989, V. 37, pp. 925-935

[Rachamadugu and Stecke, 1994] RACHAMADUGU R. and STECKE K., "*Classification and review of FMS scheduling procedures*", Production Planning and Control, 1994, V. 5, N. 1, pp. 2-20

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399

RR- 3395