Ordonnancement de taches contraintes dans le cas d'un serveur

N. Dridi, Jean-Marie Proth, S. Sedillot

INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 325

ORDONNANCEMENT DE TACHES CONTRAINTES DANS LE CAS D'UN SERVEUR

Najoua DRIDI
Jean-Marie PROTH
Simone SEDILLOT

Août 1984

ORDONNANCEMENT DE TACHES CONTRAINTES DANS LE CAS D'UN SERVEUR

Najoua DRIDI, Jean-Marie PROTH et Simone SEDILLOT.

Nous remercions

P. NEPOMIASTCHY pour son exposé sur les méthodes de pénalisation

C. LEMARECHAL pour l'aide qu'il nous a apportée dans l'utilisation de MODULOPT.

ABSTRACT.

In this paper, we consider the case of a single facility used in order to perform a set of tasks, every of them being defined as follows :
1) the beginning time of the task lies between a lower and an upper bound
2) the duration of the task is known.

Three algorithms are given. We also consider the interscheduling problem and an application in the local networks area.

RESUME.

Dans ce papier, nous considérons le cas d'une entité utilisée pour exécuter un ensemble de tâches, chacune d'elles étant définie comme suit : 1) le début de la tâche est bornée inférieurement et supérieurement 2) la durée de la tâche est connue. Nous proposons trois algorithmes. Nous examinons également le problème d'interclassement et une de ses applications au domaine des réseaux locaux.

0. INTRODUCTION.

Ce rapport s'intéresse à la recherche d'un ordre de passage de produits sur une machine connaissant, pour chaque produit : 1) le temps de début de fabrications au plus tôt 2) le temps de fin de fabrication au plus tard 3) la durée de fabrication.

Nous recherchons d'abord une solution admissible en utilisant une méthode de type "branch and bound". Nous exprimons ensuite le problème sous forme d'une fonction à optimiser, fonction destinée à pénaliser toute situation non admissible. Enfin, nous proposons un algorithme tout à fait élémentaire qui est efficace lorsque les périodes de travail admissibles des différents produits sont bien séparées. La solution pratique satisfaisante est à rechercher dans une utilisation parallèle de ces différents algorithmes.

Nous terminons en recherchant comment prendre en charge un produit nouveau lorsqu'un certain nombre de produits sont déjà ordonnés. Un cas particulier de ce problème, que l'on rencontre dans l'étude des réseaux locaux, consiste à essayer de placer un produit nouveau en éliminant un nombre minimal de produits déjà placés. Ce problème est examiné et un algorithme simple est proposé.

I. NOTATIONS.

Soient $P_1, P_2, \ldots, P_n$ n produits à fabriquer par passage sur une machine unique.

Pour chaque produit P_i $(i = 1,2,\ldots,n)$ nous connaissons :

t_0^i, instant de début de fabrication au plus tôt

t_1^i, instant de fin de fabrication au plus tard

Θ_i, durée de fabrication de P_i.

On appelle ordre de passage (ou ordonnancement) des produits sur la machine toute application biunivoque D de $E = \{1,2,\ldots,n\}$ sur E :

$$D : E \longrightarrow E$$

D(i) représente l'indice du produit dont le rang de passage est i. En d'autes termes, $P_{D(i)}$ est le i^{me} produit qui passe sur la machine.

Dans toute la suite, nous ferons la convention :

$\sum_{k=i}^{j} . = 0$ si $j < i$

On dit que l'on fabrique au plus tôt en respectant D si les produits sont fabriqués dans l'ordre donné par D, chacun d'eux étant fabriqué le plus tôt possible compte tenu de son instant de début de fabrication au plus tôt et des produits qui le précèdent sur la machine.

Dans ce cas, pour i = 1,2,..., n :

a) l'instant de début de fabrication $P_{D(i)}$ s'écrit :

$$T_0^D(i) = \max_{j=1,\ldots,i} \left[t_0^{D(j)} + \sum_{r=j}^{i-1} \Theta_{D(r)} \right] \quad (1)$$

b) l'instant de fin de fabrication de $P_{D(i)}$ s'écrit :

$$T_1^D(i) = T_0^D(i) + \Theta_{D(i)} = \max_{j=1,\ldots,i} \left[t_0^{D(j)} + \sum_{r=j}^{i} \Theta_{D(r)} \right] \quad (2)$$

Notons que, lorsqu'on fabrique au plut tôt, les contraintes de fin de fabrication au plus tard peuvent être violées, c'est à dire $T_1^D(i) > t_1^{D(i)}$ pour $i \in \{1,\ldots,n\}$.

On parlera de fabrication au plus tard en respectant D si tous les produits sont fabriqués au plus tard dans l'ordre de passage D et en respectant les instants de fin de fabrication au plus tard.

Dans ce cas, pour i = 1,2,..., n :

a) l'instant de début de fabrication de $P_{D(i)}$ s'écrit :

$$U_{0(i)}^D = \min_{j=i,\ldots,n} \left[t_1^{D(j)} - \sum_{r=i}^{j} \Theta_{D(r)} \right] \quad (3)$$

b) l'instant de fin de fabrication de $P_{D(i)}$ s'écrit :

$$U_{1(i)}^D = \min_{j=1,\ldots,n} \left[t_1^{D(j)} - \sum_{r=i+1}^{j} \Theta_{D(r)} \right] \quad (4)$$

Dans le cas d'une fabrication au plus tard en respectant D, les contraintes de début de fabrication au plut tôt peuvent être violées.

En d'autres termes, il peut exister $i \in \{1,2,\ldots,n\}$ tel que :

$$U_0^D(i) < t_0^{D(i)}$$

L'ordre de passage D est admissible s'il permet la fabrication des produits en respectant les instants de début de fabrication au plus tôt et les instants de

fin de fabrication au plus tard.

Nous reviendrons sur ce problème. Nous exposons d'abord les questions auxquelles nous cherchons à donner une réponse.

II. LES PROBLEMES A RESOUDRE.

Ils sont de deux types :

1. existe-t-il un ordre de passage admissible ? Si oui, comment l'obtenir ?
2. un ordre de passage admissible de n produits étant donné, est-il possible de prendre en compte un produit nouveau ? Nous rechercherons un algorithme "temps réel".

Nous donnons d'abord, dans le paragraphe suivant, plusieurs C.N.S. équivalentes pour qu'un ordre de passage soit admissible.

Nous nous attacherons ensuite à la recherche de solutions admissibles. Nous proposerons une approche du type "branch and bound", puis nous donnerons une heuristique conduisant à une solution admissible (si elle existe).

Nous fournissons enfin un algorithme "temps réel" de recherche d'un ordre de passage admissible lorsqu'on intègre un élément supplémentaire à un ordre de passage admissible donné.

III. ORDRE DE PASSAGE ADMISSIBLE : DEFINITIONS EQUIVALENTES.

Résultat 1.

Les propositions suivantes sont vraies :

1. D est admissible si et seulement si

$$T_1^D(i) \leq t_1^{D(i)} \quad \text{pour } i = 1,2,\ldots, n \tag{5}$$

2. D est admissible si et seulement si :

$$U_0^D(i) \geq t_0^{D(i)} \quad \text{pour } i = 1,2,\ldots, n \tag{6}$$

3. D est admissible si et seulement si :

$$T_0^D(i) \leq U_0^D(i) \quad \text{pour } i = 1,2,\ldots, n \tag{7}$$

4. D est admissible si et seulement si :

$$U_1^D(i) \leq U_1^D(i) \quad \text{pour } i = 1,2,\ldots, n \tag{8}$$

Démonstration.

1. La relation (5) indique que, lorsqu'on fabrique au plus tôt, tous les produits sont fabriqués avant leur instant de fin de fabrication au plus tard. D est donc admissible lorsque (5) est vérifiée.
   Supposons maintenant qu'il existe $i \in \{1,2,\ldots,n\}$ tel que $T_1^D(i) > t_1^{D(i)}$ (i.e. (5) n'est pas vérifiée). $T_1^D(i)$ est l'instant de fin de fabrication de $P_{D(i)}$ lorsque les fabrications sont effectuées au plut tôt en respectant les instants de début de fabrication au plus tôt. Par conséquent toutes les décisions de fabrication qui respecteraient les instants de début de fabrication au plus tôt et l'ordre D vérifieraient :

   $$T_1^{*D}(i) \geq T_1^D(i) > t_1^{D(i)}$$

   où $T_1^{*D}(i)$ est l'instant de fin de fabrication de $P_{D(i)}$.
   Donc D n'est pas admissible lorsque (5) n'est pas vérifiée.

2. La démonstration de la seconde proposition est analogue à la précédente.

3. La relation (7) exprime que les temps de début de fabrication des différents produits sont plus petits lorsqu'on fabrique au plus tôt que lorsqu'on fabrique au plus tard.
   Par conséquent, lorsque (7) est vérifiée, la fabrication au plus tôt suivant D vérifie (5).
   Lorsque (7) n'est pas vérifiée, (5) n'est pas vérifiée. D'où la conclusion.

4. L'inégalité (8) est équivalente à (7). D'où la conclusion. □

IV. RECHERCHE D'UNE SOLUTION ADMISSIBLE.

IV.1 Une approche de type "branch and bound".

Supposons choisis les D(i) pour i = 1,2,..., r, avec $r \leq n$

Nous posons :

$$E_r^D = \{D(1), D(2),\ldots, D(r)\} \tag{9}$$

Résultat 2.

Supposons choisis les r premiers éléments de l'ordonnancement ($r < n$). Une condition nécessaire pour que ces éléments puissent être complétés pour conduire à un ordonnancement admissible est que

$$\text{Max}\left[T_1^D(r),\ t_0^k\right] + \theta_k \leqq t_1^k,\ \forall k \notin E_r^D \tag{10}$$

(lorsque $r = 0$, $T_1^D(r)$ est l'instant initial).

Démonstration.

La relation (10) exprime que si nous plaçons au rang r+1 l'un quelconque des produits non encore ordonnancés, son temps de fin de fabrication sera inférieur ou égal à son temps de fin de fabrication au plus tard si nous effectuons les fabrications au plus tôt.

Supposons qu'il existe $k_1 \notin E_r^D$ tel que

$$\text{Max}\left[T_1^D(r),\ t_0^{k_1}\right] + \theta_{k_1} > t_1^{k_1} \tag{11}$$

Deux cas sont à envisager :

a) on choisit

$$D(r+1) = k_1$$

La relation (11) montre qu'alors le temps de fin de fabrication de P_{k_1} dépassera son temps de fin de fabrication au plus tard et tout ordre de passage D commençant par D(1), D(2),..., D(r), $D(r+1) = k_1$ ne peut être admissible (c.f.(5)).

b) on choisit

$$D(r+1) = k \neq k_1,\quad k \notin E_r^D$$

alors

$$T_1^D(r+1) = \text{Max}\left[T_1^D(r),\ t_0^k\right] + \theta_k \geqq T_1^D(r) \tag{12}$$

donc, d'après (11) et (12) :

$$\text{Max}\left[T_1^D(r+1),\ t_0^{k_1}\right] + \theta_{k_1} > t_1^{k_1} \tag{13}$$

Nous sommes alors conduits à choisir D(r+2)

Si $D(r+2) = k_1$, (13) conduit à la conclusion a.

Sinon on obtient, comme dans b. :

$$\text{Max}\left[T_1^D(r+2),\ t_{0}^{k_1}\right] + \Theta_{k_1} > t_1^{k_1} \qquad (14)$$

En poursuivant le raisonnement précédent, nous voyons que nous serons conduits à choisir :

$$D(j) = k_1 \quad \text{pour } j > r+1$$

avec :

$$\text{Max}\left[T_1^D(j-1),\ t_{0}^{k_1}\right] + \Theta_{k_1} > t_{k_1}^{1}$$

si bien que tout ordre de passage D dont les r premiers éléments D(1), D(2),...,D(r), sont donnés et tel que (11) est vérifié ne peut être admissible. □

Résultat 3.

Nous conservons les hypothèses du résultat 1 et nous supposons que (10) est vérifiée.

1. s'il existe $k \notin E_r^D$ tel que :

$$\text{Max}\left[T_1(r),\ t_0^k\right] + \Theta_k \leq t_1^{\ell} - \Theta_{\ell},\ \forall \ell \notin E_r^D,\ \ell \neq k \qquad (15)$$

   alors P_k est candidat au rang r+1 dans l'ordonnancement (i.e. il est possible qu'en choisissant D(r+1) = k, nous soyons conduits à un ordonnancement admissible)

2. s'il n'existe aucun $k \notin E_r^D$ vérifiant (15), alors l'ordonnancement partiel constitué des r produits déjà choisis ne peut conduire à un ordonnancement admissible.

Démonstration.

1. La relation (15) exprime qu'en commençant P_k au plus tôt après avoir fabriqué au plus tôt $P_{D(1)}$, $P_{D(2)}$,..., $P_{D(r)}$, alors P_k sera terminé avant le début au plus tard de chacun des produits non encore placés et autres que P_k. Le choix D(r+1) = k est donc envisageable.

2. La seconde partie du théorème indique que tout produit P_k, $k \notin E_r^D$, choisi au rang r+1, se terminera après l'instant de début au plus tard d'au moins un des produits restant à placer. D'où la conclusion □.

L'algorithme.

Nous désignons par $\sigma(r)$ la suite des r-1 premiers produits choisis :

$$\sigma(r) = \begin{cases} \emptyset \text{ si } r = 1 \\ \{D(1), D(2), \ldots, D(r-1)\} \text{ si } r > 1. \end{cases}$$

Les résultats que nous venons de présenter conduisent à l'algorithme suivant :

1. Faire $T_0(0) = d$, où d est l'instant initial
2. Faire r = 1
3. Utilisation de la relation (10).
   - 3.1. si (10) n'est pas vérifiée
     - 3.1.1. si r = 1
       - 3.1.1.1. Editer "Pas de solution"
       - 3.1.1.2. Fin du processus
     - 3.1.2. si r > 1
       - 3.1.2.1. Faire $\sigma(r-1) = \sigma(r) - \{D(r-1)\}$ (supp. du dernier él. de $\sigma(r)$)
       - 3.1.2.2. Faire r = r-1
       - 3.1.2.3. Aller en 3.
   - 3.2. si (10) est vérifiée, aller en 4.
4. Recherche d'un indice k vérifiant :
   - et $\begin{cases} k \notin E_r^D \\ k \text{ n'a pas encore été testé pour la suite } \sigma(r). \end{cases}$
   - 4.1. S'il n'existe aucun k vérifiant les conditions précédentes, aller en 3.1.1.
   - 4.2. S'il existe un k vérifiant les conditions précédentes
     - 4.2.1. Faire $\sigma(r+1) = \sigma(r)$ O $\{k\}$ (adjonction d'un él. à $\sigma(r)$)
     - 4.2.2. Faire r = r+1
     - 4.2.3. Si r = n (nombre de produits à placer), $\sigma(n+1) = \{D(1), \ldots, D(n)\}$ est un ordre de passage admissible, sinon aller en 3.

Nous proposons maintenant une heuristique qui conduit à une solution admissible lorsqu'elle existe.

IV.2. Approches heuristiques.

IV.2.1. Minimisation de l'instant de fin de fabrication.

Résultat 4.

Soit D un ordonnancement, $r \leq n$ et E_r^D la suite donnée par (9).

Si $t_0^{D(1)} \leq t_0^{D(2)} \leq ,\ldots, \leq t_0^{D(r)}$, alors $T_0^D(r) = \underset{U \in d'_r}{\text{Min}}\ T_1^0(r)$, où d'_r est l'ensemble de tous les ordonnancements dont les r premiers éléments sont ceux de E_r^D.

En d'autres termes, si l'on fixe les éléments qui précèdent le r^{me} élément à fabriquer, son instant de fin de fabrication est minimal lorsque les éléments qui le précèdent sont fabriqués dans l'ordre croissant de leur instant de début de fabrication au plus tôt et que la fabrication se fait "au plus tôt" au sens donné dans le paragraphe I.

Démonstration.

Considérons (2) et soit j_1 l'entier vérifiant :

$$t_0^{D(j_1)} + \sum_{k=j_1}^{r} \Theta_{D(k)} = \underset{j=1,..,r}{\text{Max}} \left[t_0^{D(j)} + \sum_{k=j}^{r} \Theta_{D(k)} \right] \qquad (16)$$

Supposons qu'il existe $j_0 < j_1$ tel que $t_0^{D(j_0)} > t_0^{D(j_1)}$ (17)

Alors :

$$t_0^{D(j_0)} + \sum_{k=j_0}^{r} \Theta_{D(k)} = t_0^{D(j_0)} + \sum_{k=j_0}^{j_1-1} \Theta_{D(k)} + \sum_{k=j_1}^{r} \Theta_{D(k)}$$

soit, en tenant compte de (17) et du fait que $\Theta_{D(k)} \geq 0\ \forall k$:

$$t_0^{D(j_0)} + \sum_{k=j_0}^{r} \Theta_{D(k)} > t_0^{D(j_1)} + \Theta_{D(k)}$$

ce qui est en contradiction avec (16).

Donc (17) n'est pas vérifiée ce qui achève la démonstration.□

Nous allons maintenant montrer qu'il est possible d'écrire notre problème sous forme d'un problème d'optimisation.

IV.2.2. Résolution de problèmes d'optimisation.

Pour i = 1,2,..., n, notons x_i l'instant de début de fabrication du produit P_i.

Posons :

$$X = (x_1, \ldots, x_n) \tag{18}$$

La contrainte suivante doit être vérifiée :

$$t_0^i \leqq x_i \leqq t_1^i - \theta_i, \quad i = 1,2,\ldots, n \tag{19}$$

Si les x_i sont choisis de manière aléatoire compte tenu des contraintes (19), il peut y avoir recouvrement, c'est à dire que deux produits différents peuvent solliciter la machine au même instant.

Considérons par exemple les produits P_i et P_j (i=1,...,n ; j=1,...,n ; $i \neq j$).

La figure 1 donne les différentes possibilités de recouvrement pour ces deux produits :

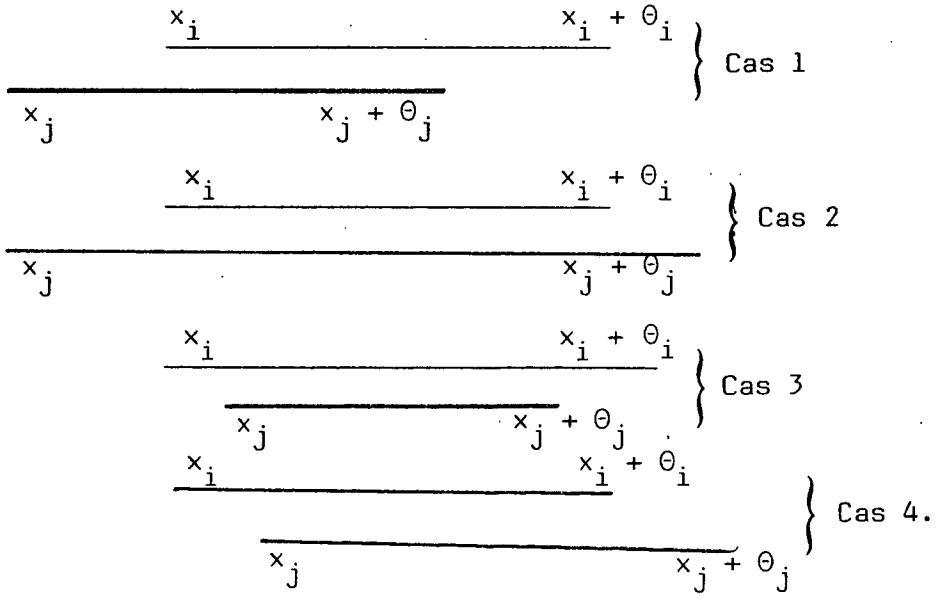

Figure 1.

Dans tous les cas de figure, nous voyons qu'il y a recouvrement lorsque :

$$\text{et} \begin{cases} x_i + \theta_i - x_j > 0 \\ x_j + \theta_j - x_i > 0 \end{cases} \tag{20}$$

Résultat 5.

Il y a recouvrement pour P_i et P_j (i=1,....,n ; j=1,..,n ; $i \neq j$) si et seulement si :

$$(x_i + \Theta_i - x_j)\ (x_j + \Theta_j - x_i) > 0$$

Démonstration.

a) La réalisation simultanée des inégalités :

$$x_i + \Theta_i - x_j < 0 \qquad (21)$$

et $$x_j + \Theta_i - x_i < 0 \qquad (22)$$

est impossible.

En effet :

(21) entrâîne :

$$x_j > x_i + \Theta_i$$

et (22) entraîne :

$$x_j < x_i - \Theta_j$$

Mais :

$$x_i + \Theta_i > x_i - \Theta_j$$

D'où la contradiction.

b) Le résultat que nous cherchons à établir est la conséquence de a) et (20).

De ce qui précède, nous tirons deux formulations possibles de notre problème :

Formulation P_1.

Soit

$$f_1(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \chi_{U_{ij} > 0}\ U_{ij}$$

avec :

$$\chi_a = \begin{cases} 1 \text{ si la condition logique } \underline{a} \text{ est réalisée} \\ 0 \text{ sinon.} \end{cases}$$

et :

$$U_{ij} = (x_i + \Theta_i - x_j)\ (x_j + \Theta_j - x_i) \qquad (23)$$

La formulation P_1 de notre problème s'écrit :

$$(P_1)\ \begin{cases} \text{trouver } X^* \text{ tel que :} \\ f_1(X^*) = \min\limits_{X} f_1(X) \\ \quad \text{sous les contraintes (19).} \\ X \text{ est solution de notre problème si et seulement si} \\ f_1(X) = 0 \end{cases}$$

Formulation P_2.

Soit :

$$f_2(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \chi_{U_{ij} > 0}\, U_{ij}^2$$

La formulation P_2 de notre problème est la suivante :

$$(P_2)\ \begin{cases} \text{trouver } X^* \text{ tel que :} \\ f_2(X^*) = \min\limits_{X} f_2(X) \\ \quad \text{sous les contraintes (19)} \\ X^* \text{ est solution de notre problème si et seulement si} \\ \quad f_2(X^*) = 0 \end{cases}$$

Les deux formulations sont proches. La seconde assure la continuité de la dérivée première aux points qui annulent les U_{ij}.

Nous proposons deux heuristiques issues des formulations (P_1) et (P_2). Nous les formulons dans les algorithmes résumés comme suit :

1. Choix du nombre m d'essais souhaités
2. Faire

   KK = 0
3. Génération au hasard de $X^0 = (x_1^0, \ldots, x_n^0)$ vérifiant les contraintes (19)
4. Recherche, à l'aide d'un algorithme de type gradient, de X^*, minimum local de $f_1(X)$ (resp. $f_2(X)$) obtenu à partir de X^0 en respectant (19).
5. Test de X^*.

   5.1. Si $f_1(X^*) > \varepsilon$ (resp. $f_2(X^*) > \varepsilon$), où ε est un réel positif petit, et qui dépend de la précision des calculs, alors :

      5.1.1. Faire KK = KK + 1

      5.1.2. Test de KK

5.1.2.1. Si KK > m, pas de solution.
Fin de processus.
5.1.2.2. Si KK ≦ m, aller en 3.

5.2. Si $f_1(X^*) \leqq \varepsilon$ (resp. $f_2(X^*) \leqq \varepsilon$), alors :
X^* est solution du problème.
Fin de processus.

Le point 4 de notre algorithme utilise des sous-programmes faisant partie de MODULOPT et développés à l'INRIA.

Nous donnons plus loin une application des différentes approches proposées sur des problèmes comportant trente produits.

IV.2.3. Une heuristique simple.

Nous proposons maintenant une heuristique extrêmement simple. Elle consiste à générer X au hasard en tenant compte de (19), puis à fabriquer au plus tôt dans l'ordre des x_i croissant. Nous verrons, lors des applications, que l'utilisation d'un tel algorithme n'est pas sans intérêt.Il peut se résumer comme suit :

1. Choix de nombre m d'essais souhaités
2. Faire KK = 0
3. Génération au hasard de $X = (x_1, \ldots, x_n)$ vérifiant les contraintes (19).
4. Soit D l'ordre donné par x_i croissants (i = 1,2,..., n).

Pour i = 1,2,.., n :

4.1 Placement au plus tôt du produit $P_{D(i)}$
4.2. Test de T_1^0 (i) (c.f.(2)).
4.2.1 Si $T_1^D(i) > t_1^{D(i)}$
4.2.1.1. Faire KK = KK + 1
4.2.1.2. Test de KK
4.2.1.2.1 Si KK > m, pas de solution. Fin.
4.2.1.2.2 Si KK ≦ m aller en 3.
4.2.2 Si $T_1^D(i) \leqq t_1^{D(i)}$, passer à la valeur suivante de i
4.3. T_0^D (i) (i=1,...,n) est une suite d'instants de début de fabrication admissible.

Remarque.

Les trois heuristiques que nous proposons débutent par un essai de fabrication au plus tôt dans l'ordre des t_0^i croissants. Cette démarche résulte de l'observation que, dans la pratique, il existe une certaine corrélation entre l'ordre des t_0^i et l'ordre des t_1^i.

V. PRISE EN COMPTE D'UN PRODUIT NOUVEAU.

n produits sont fabriqués dans l'ordre $P_1, P_2, \ldots, P_n$.

Nous désignons par u(i) et v(i) l'instant de début au plus tard et de fin au plus tôt du produit P_i. Ce sont les valeurs introduites par (3) et (2). Nous les donnons sous forme récursive :

$$\left.\begin{aligned} u(n) &= t_1^n - \Theta_n \\ u(i) &= \text{Min}\,(t_1^i,\ u(i+1)) - \Theta_i, \quad i = n-1,\ n-2,\ldots,\ 1 \end{aligned}\right\} \qquad (24)$$

et :

$$\left.\begin{aligned} v(1) &= t_0^1 + \Theta_1 \\ v(i) &= \text{Max}\,(t_0^i,\ v\,(i-1)) + \Theta_i,\ i = 2,3,\ldots,\ n \end{aligned}\right\} \qquad (25)$$

Soit à fabriquer un produit P dont :

- V_0 est l'instant de début au plus tôt
- V_1 est l'instant de fin au plus tard
- Z est le temps de fabrication.

Intercaller P dans les fabrications existantes, c'est trouver $i \in \{1,2,\ldots,\ n-1\}$ tel que :

$$\text{Min}\,(u(i+1),\ V_1) - \text{Max}\,(v(i),\ V_0) \geq Z \qquad (26)$$

Cette démarche exclut la possibilité de fabriquer P avant P_1 ou après P_n.

L'algorithme cherché s'écrit alors :

1. Calcul de u(i), i = n, n-1, n-2,..., 1 (c.f.(24))
2. Calcul de v(i), i = 1,2,..., n (c.f. (25))
3. Pour i = 1,2,3,..., n-1 :

3.1. si $u(i+1) \leq V_0$, retour en 3.

3.2. si $v(i) \geq V_1$, pas de solution. Fin de processus.

3.3. Test :

3.3.1. $a = \text{Max}\ (v(i), V_0)$

3.3.2. $b = \text{Min}\ (u(i+1), V_1)$

3.3.3. si $b-a \geq Z$, la solution consiste à intercaller P entre P_i et P_{i+1}.
Fin de processus
sinon, retour en 3.

4. Pas de solution.

Il arrive (c'est le cas en informatique) que l'on soit autorisé à supprimer certains des P_i pour pouvoir assurer le passage de P, mais alors il convient d'en supprimer le moins possible, et pas n'importe lesquels.

Soit x(i), i = 1,2,...,n un tableau défini comme suit :

$$x(i) = \begin{cases} 0 \text{ si } P_i \text{ peut être supprimé} \\ 1 \text{ sinon.} \end{cases}$$

L'algorithme qui assure qu'un nombre minimal de produits sera supprimé s'écrit :

1. Calcul de u(i), i = n, n-1,..., 1 (c.f.(24))
2. Calcul de v(i), i = 1,2,...,n (c.f. (25))
3. Pour k = 0,1,2,..., n-1

3.1 Pour i = 1,2,3,..., n-k

3.1.1 j = i+k+1

3.1.2 si j > n, q = V_1
sinon, q = u(j)

3.1.3 si $q \leq V_0$, retour en 3.1

3.1.4 si $v(i) \geq V_1$, pas de solution. Fin de boucle k.

3.1.5 Test :

3.1.5.1 si k = 0, aller en 3.1.6

3.1.5.2 sinon

3.1.5.2.1 si x(j) = 0 pour j = i+1,..., i+k
aller en 3.1.6

3.1.5.2.2 sinon, fin de boucle i.

3.1.6 Test

3.1.6.1 a = Max (v(i), V_0)

3.1.6.2 b = Min (q, V_1)

3.1.6.3 si b-a ≧ Z, la solution consiste à intercaller P entre P_i et P_{i+k+1}. Fin de processus. Sinon, fin de boucle i.

3.2. Fin de boucle k.

4. Pas de solution.

VI. APPLICATION NUMERIQUE.

Nous considérons le problème suivant, faisant intervenir 30 produits.

PRODUIT	DEBUT DE FABRI. AU PLUS TOT	DEBUT DE FAB. AU PLUS TARD	DUREE DE FAB.
1	0.0000	10.0000	6.0000
2	2.0000	10.0000	2.0000
3	0.0000	14.0000	6.0000
4	140.0000	160.0000	3.0000
5	8.0000	20.0000	4.0000
6	10.0000	30.0000	1.0000
7	20.0000	30.0000	1.0000
8	24.0000	29.0000	3.0000
9	0.0000	34.0000	6.0000
10	30.0000	50.0000	8.0000
11	30.0000	45.0000	2.0000
12	45.0000	150.0000	6.0000
13	0.0000	50.0000	6.0000
14	50.0000	54.0000	3.0000
15	15.0000	61.0000	5.0000
16	58.0000	74.0000	2.0000
17	30.0000	70.0000	4.0000
18	60.0000	75.0000	10.0000
19	75.0000	100.0000	4.0000
20	125.0000	150.0000	6.0000
21	40.0000	100.0000	10.0000

22	80.0000	96.0000	7.0000
23	95.0000	110.0000	7.0000
24	80.0000	110.0000	2.0000
25	120.0000	200.0000	3.0000
26	100.0000	130.0000	4.0000
27	40.0000	120.0000	8.0000
28	115.0000	130.0000	9.0000
29	125.0000	129.0000	3.0000
30	100.0000	140.0000	2.0000

Nous avons d'abord appliqué l'algorithme issu de l'approche type "branch and bound". Les résultats sont donnés par le tableau suivant qui fournit pour chaque produit, et dans l'ordre de leur passage, le temps de début de fabrication et le temps de fin de fabrication, et qui rappelle la durée de fabrication.

PRODUIT	T.D. FAB.	T.F. FAB.	DUREE
1	0.00	6.00	6.0
2	6.00	8.00	2.0
3	8.00	14.00	6.0
5	14.00	18.00	4.0
6	18.00	19.00	1.0
7	20.00	21.00	1.0
8	24.00	27.00	3.0
9	27.00	33.00	6.0
10	33.00	41.00	8.0
11	41.00	43.00	2.0
13	43.00	49.00	6.0
14	50.00	53.00	3.0
15	53.00	58.00	5.0
16	58.00	60.00	2.0
17	60.00	64.00	4.0
18	64.00	74.00	10.0
19	75.00	79.00	4.0
21	79.00	89.00	10.0
22	89.00	96.00	7.0
23	96.00	103.00	7.0
24	103.00	105.00	2.0
26	105.00	109.00	4.0
27	109.00	117.00	8.0
28	117.00	126.00	9.0
29	126.00	129.00	3.0
12	129.00	135.00	6.0
25	135.00	138.00	3.0
30	138.00	140.00	2.0
4	140.00	143.00	3.0
20	143.00	149.00	6.0

L'heuristique qui passe par la résolution d'un problème d'optimisation nous conduit à des solutions admissibles après un nombre d'essais compris entre 200 et 2000 suivent les exemples traités (30 produits). Le troisième algorithme nécessite de 200 à 6000 essais.

CONCLUSION.

Les algorithmes proposés se sont révélés efficaces. Ils seront naturellement étendus au cas où la machine est immobilisée pour cause de changement d'outil et au cas de plusieurs machines ("job-shop"), en supposant toujours l'existence d'un temps de début au plus tôt et d'un temps de début au plus tard pour chaque tâche. Dans la pratique, ces bornes sont souvent données par l'expérience, donc susceptibles de varier dans des limites raisonnables. Il reste donc à étudier comment intervient cette souplesse dans la recherche d'une solution admissible. Notons que ce travail s'inscrit dans une approche hiérarchisée de la gestion des systèmes de production : les contraintes sont fixées au niveau haut, afin de réduire l'aspect combinatoire du problème, et la recherche d'une solution admissible obéissant à ces contraintes se déroule au niveau bas.

Enfin, le problème d'interclassement, qui paraît essentiel dans la mise en place de protocole de gestion de réseaux locaux, n'est pas encore résolu de manière satisfaisante.

BIBLIOGRAPHIE.

[1] C. LEMARECHAL et E. PANIER "Les modules m2qnl et mqhess". Doc. INRIA

[2] R.H. BYRD, G.A. SCHULTZ "A practical class of globally convergent active set strategies for linearly constrained optimisation". Univ. of Colorado 1982 (à paraître dans Mathematical Programming).

[3] P.E. GILL, W. MURRAY, M.H. WRIGHT : Practical optimisation. Academic Press 1981.

[4] B.A. MURTAGH, R.W.H. SARGENT : A constrained minimization méthod with quadratic convergence. in : R. FLETCHER (Ed). Optimisation, Academic Press 1969.

[5] P. NEPOMIASTCHY : Méthode de pénalisation et applications. Résolution et Optimisation de modules macroéconomiques.

[6] P. WOLFE : On the convergence of gradient methods under constraints. IBM Journal of Res. and Dev. 16 (1972) 407-411.

[7] A.H.G. RINNOOY KAN : Machine scheduling problems, North Holland.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique