7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

CONTRIBUTIONS POUR LE PROBLEME A UNE MACHINE AVEC FONCTIONS TEMPORELLES DE TYPE POLYNOMIAL

F. GUEGNARD

LISA : EA 4014, IUT d'Angers-Cholet
4 bd Lavoisier, BP 42018
49016 Angers cedex
frederic.guegnard@univ-angers.fr

M. BOURCERIE

LISA : EA 4014, IUT d'Angers-Cholet
4 bd Lavoisier, BP 42018
49016 Angers cedex
marc.bourcerie@univ-angers.fr

RESUME : *Notre étude porte sur les problèmes à une machine dans lesquels les durées d'exécution des tâches ne sont plus des constantes mais dépendent du temps. Lorsque les fonctions $p_i(t)$, représentant ces durées d'exécution, sont des fonctions polynomiales d'ordre supérieur ou égal à deux, le problème à une machine $1|p_i(t)|C_{\max}$ est NP-difficile. Lorsque $p_i(t) = a_i.t^n + b_i$, le problème devient polynomial sous certaines conditions. Nous avons étudié le problème $p_i(t) = a_i.t^2 + b_i$ et nous proposons de construire un graphe temporel simple dans le cas général.*

MOTS-CLES : *ordonnancement, problèmes à une machine, fonctions polynomiales.*

1 INTRODUCTION

Dans les problèmes classiques d'ordonnancement, les durées des tâches sont des constantes. Cependant, dans de nombreux cas réels, la durée d'exécution des tâches dépend de leur date de début d'exécution. Par exemple, nous pouvons rencontrer ce type de problèmes dans la gestion des urgences médicales, des incendies ou dans l'industrie métallurgique. Dans ce dernier cas, les temps de cuisson des pièces dans un four industriel dépendent alors de la température de ce dernier.
Nous nous limiterons à l'étude d'une classe particulière des problèmes d'ordonnancement : les problèmes à une machine. Nous utiliserons les notations définies dans Garey et Johnson, 1979.
Notre travail traite donc des problèmes à une machine dans lesquels, les durées d'exécution des tâches ne sont plus des constantes mais dépendent du temps (Wagneur E. Sriskandarajah C, 1992, Gawiejnowicz S., Pankowska L., 1995, et Alidae B., NK. Womer, 1999).
On note $p_i(t)$, la fonction temporelle représentant la durée d'exécution d'une tâche *i*. Le problème $1|p_i(t), r_i|C_{\max}$ est NP-difficile lorsque $p_i(t)$ est une fonction polynomiale de degré supérieur ou égal à un ; il est possible de résoudre des instances de petites tailles à l'aide de méthode de Branch and Bound (Guegnard et al, 2004a et Guegnard et al, 2004b).
La complexité du problème $1|p_i(t)|C_{\max}$ dépend quant à lui de la nature de la fonction $p_i(t)$. Lorsque $p_i(t) = a_i.t + b_i$, le problème est polynomial (Sundararaghavan P. S. et Kunnathur A. S., 1994) ; il le reste sous certaines conditions lorsque $p_i(t) = a_i.t^n + b_i$ et devient NP-difficile lorsque $p_i(t)$ est un polynôme de degré supérieur ou égal à deux (Cheng TCE. and Q. Ding, 1998).

Nous limitons nos travaux aux problèmes où les durées des tâches sont de la forme $p_i(t) = a_i.t^n + b_i$ et en particulier le cas où $n = 2$. Nous considérons $b_i > 0$, c'est-à-dire que la durée initiale de la tâche est strictement positive et $a_i > 0$, c'est-à-dire que la durée de la tâche est croissante.

Ce problème étant à la frontière des problèmes dits « faciles » et des problèmes dits « difficiles », il est de ce fait très pertinent d'en étudier toutes les caractéristiques.

2 CAS GENERAL : $p_i(t) = a_i.t^n + b_i$

2.1 Hypothèse 1

Nous appelons *hypothèse 1*, le fait de pouvoir ordonner les tâches par b_i/a_i croissant et par a_i croissant pour le problème $1|p_i(t)|C_{\max}$ avec $p_i(t) = a_i.t^n + b_i$ et ce pour toutes valeurs de n.

2.2 Autre écriture de l'hypothèse 1

Nous avons montré qu'il était possible d'utiliser un critère basé sur la dérivée des fonctions temporelles (Guegnard F., 2006).

Dans le cas où $p_i(t) = a_i.t^n + b_i$, la dérivée de la fonction temporelle s'écrit $p_i'(t) = n.a_i.t^{n-1}$, d'où $\frac{p_i(t)}{p_i'(t)} = \frac{a_i.t^n + b_i}{n.a_i.t^{n-1}}$.

Ce rapport peut également s'écrire : $\frac{p_i(t)}{p_i'(t)} = \frac{t}{n} + \frac{b_i}{n.a_i.t^{n-1}}$.

Pour deux tâches, nous pouvons écrire notre critère : $\frac{p_1(t)}{p_1'(t)} < \frac{p_2(t)}{p_2'(t)} \Leftrightarrow \frac{t}{n} + \frac{b_1}{n.a_1.t^{n-1}} < \frac{t}{n} + \frac{b_2}{n.a_2.t^{n-1}}$. Après simplification, on peut écrire : $b_1/a_1 < b_2/a_2$.

Donc si nous pouvons ordonner les tâches par rapport $p_1(t)/p_1'(t) < p_2(t)/p_2'(t)$ et a_i croissant alors ce problème est facile.

Nous proposons dans le paragraphe suivant de poser un certain de nombre de notations dans le cas général $p_i(t) = a_i.t^n + b_i$. Ensuite, pour étudier la frontière entre les problèmes vérifiant l'*hypothèse 1* et les autres, nous allons nous intéresser dans la suite de cette étude au cas particulier $n = 2$ c'est-à-dire lorsque $p_i(t) = a_i.t^2 + b_i$.

3 $p_i(t) = a_i.t^n + b_i$ HYPOTHESE 1 VERIFIEE

Nous rappelons que lorsque l'*hypothèse 1* est vérifiée, le problème est facile.
La finalité de la *Démonstration 1* ci-dessous, n'est pas de redémontrer un résultat connu mais de poser un certain nombre de notations que nous allons utiliser par la suite.

Démonstration 1

Soit deux tâches, représentées par $p_1(t) = a_1.t^n + b_1$ et $p_2(t) = a_2.t^n + b_2$; nous pouvons alors définir les deux cas suivants :

- la tâche 1 est exécutée avant la tâche 2 :

$p_{[12]}(t) = t + a_1.t^n + b_1 + a_2.\left(t + a_1.t^n + b_1\right)^n + b_2$,

- la tâche 2 est exécutée avant la tâche 1 :

$p_{[21]}(t) = t + a_2.t^n + b_2 + a_1.\left(t + a_2.t^n + b_2\right)^n + b_1$.

En effectuant la soustraction de ces deux possibilités, nous obtenons :

$p_{[21]}(t) - p_{[12]}(t) = a_2.t^n - a_1.t^n + a_1.(t + a_2.t^n + b_2)^n - a_2.(t + a_1.t^n + b_1)^n$,

Ce qui s'écrit en factorisant :

$p_{[21]}(t) - p_{[12]}(t) = a_1.\left((t + a_2.t^n + b_2)^n - t^n\right) + a_2.\left(t^n - (t + a_1.t^n + b_1)^n\right)$

or

$x^n - y^n = (x - y).(x^{n-1} + x^{n-2}.y + ... + x.y^{n-2} + y^{n-1})$

d'ou

$p_{[21]}(t) - p_{[12]}(t) = a_1.\left(t + a_2.t^n + b_2 - t\right).A(t) + a_2.\left(t - t - a_1.t^n - b_1\right).B(t)$

avec

$A(t) = t^{n-1} + t^{n-2}.\left(t + a_1.t^n + b_1\right) + ... + t.\left(t + a_1.t^n + b_1\right)^{n-2} + \left(t + a_1.t^n + b_1\right)^{n-1}$

et

$B(t) = t^{n-1} + t^{n-2}.\left(t + a_2.t^n + b_2\right) + ... + t.\left(t + a_2.t^n + b_2\right)^{n-2} + \left(t + a_2.t^n + b_2\right)^{n-1}$

On peut alors écrire :

$p_{[21]}(t) - p_{[12]}(t) = a_1.a_2.\left[\left(t^n + \frac{b_2}{a_2}\right).A(t) + \left(-t^n - \frac{b_1}{a_1}\right).B(t)\right]$

Sous l'hypothèse initiale $\frac{b_1}{a_1} < \frac{b_2}{a_2}$ et l'hypothèse supplémentaire $a_1 < a_2$, $A(t) > B(t)$. De plus, sous l'hypothèse initiale $\frac{b_1}{a_1} < \frac{b_2}{a_2}$, $\left(t^n + \frac{b_2}{a_2}\right) > \left(t^n + \frac{b_1}{a_1}\right)$

d'où $a_1.a_2.\left[\left(t^n + \frac{b_2}{a_2}\right).A(t) + \left(-t^n - \frac{b_1}{a_1}\right).B(t)\right] > 0$.

Conclusion : sous les hypothèses $\frac{b_1}{a_1} < \frac{b_2}{a_2}$ et $a_1 < a_2$, la tâche *1* doit être ordonnancée avant la tâche *2*.

Ce problème est donc polynomial sous l'*hypothèse 1*. Nous allons donc nous intéresser aux problèmes où l'*hypothèse 1* n'est pas vérifiée et notamment pour la première valeur de *n*, c'est-à-dire *n = 2*.

4 $p_i(t) = a_i.t^2 + b_i$ HYPOTHESE 1 NON VERIFIEE

4.1 Introduction

Nous étudions dans ce paragraphe le cas où *n = 2*. Nous nous intéressons au cas où l'hypothèse 1 n'est pas vérifiée, c'est-à-dire qu'il n'est pas possible d'ordonner les tâches par rapport $p_i(t)/p_i'(t)$ et a_i croissant.
Considérons deux tâches, la tâche *1* et la tâche *2*. On suppose $p_1(t)/p_1'(t) < p_2(t)/p_2'(t)$ (si ce n'est pas le cas, il suffit de renuméroter les tâches). On suppose également $a_1 > a_2$. L'hypothèse 1 n'est donc pas vérifiée.
Dans ce cas, deux possibilités sont envisageables :
- il existe une date φ que nous appellerons date pivot où avant cette date, la tâche *1* doit être ordonnancée avant la tâche *2*. Après cette date, la tâche *1* doit être ordonnancée après la tâche *2*.
- il existe un intervalle [φ ; ϕ] que nous appellerons fenêtre pivot dans lequel la tâche *1* doit être ordonnancée avant la tâche *2*. A l'extérieur de l'intervalle, la tâche *1* doit être ordonnancée après la tâche *2*.

Démonstration 2

Nous reprenons dans cette démonstration, les notations utilisées dans la démonstration 1 donnée au paragraphe 3.
On note $f(t) = p_{[21]}(t) - p_{[12]}(t)$.
Dans le cas où *n = 2* la fonction *f(t)* s'écrit :

$$f(t)=a_1.a_2.\left[\left(t^2+\frac{b_2}{a_2}\right).\left(a_2.t^2+2.t+b_2\right)-\left(t^2+\frac{b_1}{a_1}\right)\left(a_1.t^2+2.t+b_1\right)\right],$$

ou encore :

$$f(t)=\left(a_1.a_2^2-a_2.a_1^2\right).t^4+2.a_1.a_2.(b_2-b_1).t^2+\left(2.a_1.b_2-2.a_2.b_1\right).t + a_1.b_2^2-a_2.b_1^2 .$$

La dérivée première de f(t) est :

$$f'(t)=\left(4.a_1.a_2^2-4.a_2.a_1^2\right).t^3+4.a_1.a_2.(b_2-b_1).t+2.a_1.b_2-2.a_2.b_1$$

La dérivée seconde s'écrit :

$$f''(t)=\left(12.a_1.a_2^2-12.a_2.a_1^2\right).t^2+4.a_1.a_2.[b_2-b_1]$$

Son discriminant est :

$$\Delta=192.a_1^2.a_2^2.(a_1-a_2).(b_2-b_1)$$

<u>*1er cas*</u> : $b_1>b_2$

Avec $a_1>a_2$ et $b_1>b_2$ alors le discriminant est négatif.

Comme le coefficient de t^2 est :

$\left(12.a_1.a_2^2-12.a_2.a_1^2\right)=12.a_1.a_2.(a_2-a_1)$, sous l'hypothèse $a_1>a_2$,

$f''(t)$ est toujours négatif.

$f'(t)$ est donc décroissante.

En 0, $f'(0)=2.a_1.b_2-2.a_2.b_1$, sous l'hypothèse $b_1/a_1<b_2/a_2$, $f'(0)$ est positif.

Sa limite en plus l'infini dépend du signe du coefficient de t^3 qui est négatif.

Elle passe donc d'une valeur positive à une valeur négative.

$f(t)$ est donc croissante puis décroissante.

Deux alternatives :

- si $f(0)$ est positif.

Sa limite en plus l'infini dépend du signe du coefficient de t^4 qui est négatif. Elle coupe donc bien une fois et une seule l'axe des abscisses en une valeur t_0 que nous appelons date pivot.

- si $f(0)$ est négatif. Comme $f'(0)$ est positif, la fonction est dans sa phase croissante à partir d'une valeur négative. Il n'est pas trivial de montrer que la fonction $f(t)$ va atteindre une valeur positive. Parmi les coefficients des monômes formant la fonction $f(t)$, seul celui de degré *1* est positif. Considérons la fonction $g(t)$ définie par :

$g(t)=\left(a_1.a_2^2-a_2.a_1^2\right).t^4+2.a_1.a_2.(b_2-b_1).t^2+a_1.b_2^2-a_2.b_1^2$; c'est-à-dire la fonction $g(t)=f(t)-\left(2.a_1.b_2-2.a_2.b_1\right).t$. En effectuant le changement de variable $T=t^2$, son discriminant est : $\Delta=4.a_1.a_2.\left(a_1.b_2-a_2.b_1\right)^2$ qui est toujours positif.

La fonction $g(t)$ est positive sur l'intervalle situé à l'intérieur de ses racines (car le coefficient $a_1.a_2.(a_2-a_1)$ est négatif) dont une au moins est positive. Il existe donc un intervalle sur lequel la fonction $g(t)+\left(2.a_1.b_2-2.a_2.b_1\right).t$, c'est-à-dire $f(t)$, est positive.

La fonction $f(t)$ coupe donc deux fois l'axe des abscisses en φ et ϕ, on parlera ici de fenêtre pivot $[\varphi\ ;\ \phi]$.

<u>*2ème cas*</u> : $b_1<b_2$

Avec $a_1>a_2$ et $b_1<b_2$ alors le discriminant est positif.

La fonction possède donc deux racines :

$$t_1=\frac{\sqrt{3}.\sqrt{\left((a_1-a_2).(b_2-b_1)\right)}}{3.(a_2-a_1)} \quad \text{et} \quad t_2=\frac{-\sqrt{3}.\sqrt{\left((a_1-a_2).(b_2-b_1)\right)}}{3.(a_2-a_1)}$$

La racine t_1 est négative et la racine t_2 est positive.

On peut écrire : $t_2=\sqrt{\frac{(b_2-b_1)}{3.(a_1-a_2)}}$

Comme le coefficient de t^2 est :

$\left(12.a_1.a_2^2-12.a_2.a_1^2\right)=12.a_1.a_2.(a_2-a_1)$, sous l'hypothèse $a_1>a_2$,

$f''(t)$ est positif sur l'intervalle $[0,\ t_2]$ et négatif sur l'intervalle $[t_2,+\infty]$.

$f'(t)$ est donc croissante sur l'intervalle $[0,\ t_2]$ et décroissante sur l'intervalle $[t_2,+\infty]$.

En 0, $f'(0)=2.a_1.b_2-2.a_2.b_1$, sous l'hypothèse $b_1/a_1<b_2/a_2$, $f'(0)$ est positif.

Sa limite en plus l'infini dépend du signe du coefficient de t^3 qui est négatif.

Elle passe donc d'une valeur positive à une valeur négative.

$f(t)$ est donc croissante puis décroissante. Sa limite en plus l'infini dépend du signe du coefficient de t^4 qui est négatif. Enfin, $f(0)$ est positif.

Elle coupe donc bien une fois et une seule l'axe des abscisses en une valeur φ que nous appelons date pivot.

Le paragraphe suivant s'attachera à déterminer la date pivot φ ou la fenêtre pivot $[\varphi\ ;\ \phi]$.

4.2 Détermination de la date pivot ou de la fenêtre pivot

On reprend dans ce paragraphe, les mêmes écritures que dans les *démonstrations 1* et *2*.

La date pivot et la fenêtre pivot sont définies par rapport aux racines de la fonction :

$$f(t)=a_1.a_2.\left[\left(t^2+\frac{b_2}{a_2}\right).\left(a_2.t^2+2.t+b_2\right)-\left(t^2+\frac{b_1}{a_1}\right)\left(a_1.t^2+2.t+b_1\right)\right]$$

Cette équation s'écrit également :

$$f(t)=\left(a_1.a_2^2-a_2.a_1^2\right).t^4+\left[a_2-a_2.(2.b_1.a_1+1)-a_1+a_1.(2.b_2.a_2+1)\right].t^2 + \left(2.a_1.b_2-2.a_2.b_1\right).t+a_1.b_2^2-a_2.b_1^2 .$$

La détermination de l'ensemble des solutions de $f(t)=0$ revient à l'étude d'un polynôme de degré 4.

Le paragraphe suivant propose l'utilisation de deux outils mathématiques permettant d'en trouver les solutions : la méthode de Tschirnhaus et la méthode de Cardan.

Démonstration 3

Pour la résolution de l'équation $f(t)=0$, nous allons utiliser la méthode de Tschirnhaus (Doneddu A., 1984 et Durand E., 1971). Cette méthode tente de ramener l'équation que l'on veut résoudre à d'autres équations de degré moins élevé. Nous évitons un premier changement de variable en remarquant que le polynôme correspondant à la fonction $f(t)$ n'a pas de terme de degré 3. Cette méthode permet de transformer l'équation de degré 4 en une équation bicarrée du $4^{\text{ème}}$ degré. Pour cela, nous devons résoudre une équation du $3^{\text{ème}}$ degré afin d'éliminer le terme de degré 1. Pour cela, nous utilisons la méthode Cardan (Barbeau E. J., 1989, et Goblot R., 2001).

Pour alléger l'écriture, on pose :

$$\begin{cases} a=a_1.a_2^2-a_2.a_1^2 \\ b=a_2-a_2.(2b_1.a_1+1)-a_1+a_1.(2b_2.a_2+1) \\ c=2.a_1.b_2-2.a_2.b_1 \\ d=a_1.b_2^2-a_2.b_1^2 \end{cases}$$

On a alors $f(t)=at^4+bt^2+ct+d$.

On note $(2): at^4+bt^2+ct+d=0$

Cas particulier 1

Si $a=0$, ceci est équivalent à $a_1=a_2$; dans ce cas le signe de l'expression est alors déterminé par la relation d'ordre entre les termes constants. En effet, si $b_1>b_2$, la durée d'exécution de la tâche *1* sera alors toujours plus grande que celle de la tâche *2* : il faut ordonnancer la *2* avant la tâche *1*. Si $b_1<b_2$, la durée d'exécution de la tâche *1* sera alors toujours plus petite que celle de la tâche *2* : il faut alors ordonnancer la *1* avant la tâche *2*.

Sous la l'hypothèse $a_i=a_j$, il faut ordonnancer les tâches par ordre des b_i croissant.

Fin du cas particulier 1.

On supposera par la suite $a\neq 0$, nous pouvons alors simplifier l'écriture de l'équation :

$$(2): t^4+\frac{b}{a}.t^2+\frac{c}{a}.t+\frac{d}{a}=0$$

Dans ce cas, la transformation de Tschirnhaus est alors la suivante : $y=t^2+k.t+\frac{b/a}{2}$.

La recherche de la valeur de *k*, nous amène à résoudre l'équation du 3ème degré :

$$(3): \frac{c}{a}x^3+\left(4.\frac{d}{a}-\left(\frac{b}{a}\right)^2\right).x^2-2.\frac{b}{a}.\frac{c}{a}.x-\left(\frac{c}{a}\right)^2=0$$

Cas particulier 2

Si $c=0$, la fonction $f(t)$ se simplifie :

$$f(t)=\left(a_1.a_2^2-a_2.a_1^2\right).t^4+\left[a_2-a_2.(2b_1.a_1+1)-a_1+a_1.(2b_2.a_2+1)\right].t^2 +a_1.b_2^2-a_2.b_1^2$$

En posant $T=t^2$, il s'agit alors de la résolution d'une équation du second degré. Son discriminant est :

$\Delta=4.\left(a_1^2.b_1^2-2.a_2.b_1.a_1.b_2+a_2^2.b_1^2\right).a_2.a_1$.

Il s'écrit également : $\Delta=4.(a_1.b_1-a_2.b_1)^2.a_2.a_1$.

Sous l'hypothèse du deuxième corollaire ($c=0$), il est nul, d'où la présence d'une racine double.

Cette racine est : $T=(b_1-b_2)/(a_1-a_2)$

Si T est négatif, t n'a pas de solutions réelles ; si T est positive, t a une solution positive et une solution négative.

Le signe de $f(t)$ est dans les deux cas déterminé par le signe de $a_1.a_2^2-a_2.a_1^2$. Ce que l'on peut écrire : $a_1.a_2.(a_2-a_1)$.

Sous la l'hypothèse $a_i/b_i=a_j/b_j$, il faut ordonnancer les tâches par ordre des a_i croissant.

Fin du cas particulier 2.

On supposera par la suite que $c\neq 0$, l'équation (3) est équivalente à :

$$(3'): x^3+\left(4.\frac{d}{c}-\frac{b^2}{c.a}\right).x^2-2.\frac{b}{a}x-\frac{c}{a}=0 .$$

Pour simplifier l'écriture, on pose :

$$\begin{cases} \alpha=4.\frac{d}{c}-\frac{b^2}{c.a} \\ \beta=-2.\frac{b}{a} \\ \delta=-\frac{c}{a} \end{cases} , \quad \text{ou} \quad \begin{cases} \alpha=2.\frac{(a_1.b_2-a_2.b_1)}{a_1-a_2} \\ \beta=4.\frac{(b_2-b_1)}{a_1-a_2} \\ \delta=2.\frac{(a_1.b_2-a_2.b_1)}{a_1.a_2.(a_1-a_2)} \end{cases}$$

en fonction des variables initiales.

L'équation s'écrit alors :

$(4): x^3+\alpha x^2+\beta x+\delta=0.$

Sa résolution par la méthode Cardan nous amène à poser : $x=u-\alpha/3$.

L'équation (4) est alors équivalente à :

$(5): u^3+pu+q=0$

avec $\begin{cases} p=\beta-\frac{\alpha^2}{3} \\ q=2.\frac{\alpha^3}{27}-\frac{\alpha\beta}{3}+\delta \end{cases}$.

On pose $u=y+z$, nous sommes alors ramenés à la résolution de l'équation du second degré :

$(6): k^2+qk-\frac{p^3}{27}=0$. Son discriminant réduit est :

$\Delta=\frac{q^2}{4}+\frac{p^3}{27}$.

L'équation *(6)* a comme solution $k'=y^3$ et $k''=z^3$.

En posant, $\lambda=\sqrt[3]{k'}$, on obtient le couple de solution $y_1=\lambda$ et $z_1=-\frac{p}{3\lambda}$.

Une solution pour u (équation *(5)*) est alors : $u_1=\lambda-\frac{p}{3\lambda}$ et une solution pour x (équation *(3)* et *(4)*) est $x_1=\lambda-\frac{p}{3\lambda}-\frac{\alpha}{3}$.

Cette solution nous donne la valeur de k déterminant la transformation de Tschirnhaus : $y=t^2+\left(\lambda-\frac{p}{3\lambda}-\frac{\alpha}{3}\right)t+\frac{b/a}{2}$.

Cette transformation nous permet d'obtenir une équation de degré 4 de la forme $y^4+my^2+n=0$: (7) avec

$$\begin{cases} m=\alpha x_1^2-\frac{\alpha^2}{2}+3\beta x_1+2\delta \\ n=\delta x_1^4-\frac{\alpha\beta x_1^3}{2}+\frac{\alpha^2 x_1^2}{4}-\alpha\delta x_1^2+\frac{\alpha^2\beta x_1}{4}-\beta\delta x_1+\frac{\alpha^4}{16}-\frac{\alpha^2\delta}{2}+\frac{\alpha\beta^2}{2}+\delta^2 \end{cases}$$

En effectuant le changement de variable $Y=y^2$, il est alors trivial de déterminer les quatre solutions y_1, y_2, y_3 et y_4 de l'équation (7).

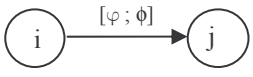
Et enfin, la résolution de $(8): t^2+\left(\lambda-\frac{p}{3\lambda}-\frac{a}{3}\right)t+\frac{b/a}{2}=y_i$ pour i=1..4, nous donne 8 solutions de l'équation initiale (2) ; quatre racines complexes inexploitables dans notre cas et quatre racines réelles dont une (φ) ou deux (φ et ϕ) sont réelles et positives : elles déterminent la date pivot φ ou la fenêtre pivot $[\varphi\ ;\ \phi]$.

Remarque : Les racines complexes inexploitables sont des racines parasites apparues lors des multiplications membre à membre effectuées pour éliminer la variable t de l'équation (2).

4.3 Graphe temporel simple

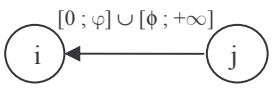
Nous utilisons dans cette partie, la notion de graphe temporel simple (ou réseau temporel simple) tel qu'il a été défini dans Dechter R., 1991.

Pour deux tâches i et j données, nous adoptons la représentation suivante :



Les deux grandeurs φ et ϕ appartiennent à l'ensemble $\mathbb{R}\cup\{-\infty;+\infty\}$.

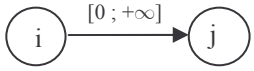
Cette représentation traduit la contrainte de précédence entre les tâches i et j : i doit être ordonnancée avant j dans la fenêtre des dates temporelles $[\varphi\ ;\ \phi]$ ou j doit être ordonnancée avant i dans la fenêtre des dates temporelles $[0\ ;\ \varphi]\cup[\phi\ ;\ +\infty]$. Ce qui peut également être représenté par :



Pour notre problème, sept possibilités sont envisageables :

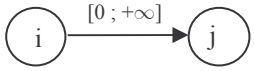
1er cas : *hypothèse 1* vérifiée.

Les tâches i et j peuvent être ordonnées par rapport b_i/a_i croissant et a_i croissant. La tâche i devra toujours être ordonnancée avant la tâche j.



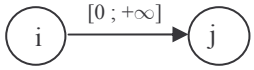
2ème cas : *hypothèse 1* non vérifiée.

- $a_i=a_j$: les tâches i et i ont le même coefficient d'ordre 2. Si $b_i<b_j$, la tâche i devra toujours être ordonnancée avant la tâche j.



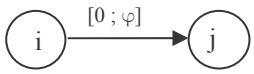
Sinon, il suffit d'inverser les tâches i et j.

- $a_i\neq a_j$ et $b_i/a_i=b_j/a_j$: dans ce cas les tâches doivent être ordonnées par a_i croissant. Si $a_i<a_j$, la tâche i devra toujours être ordonnancée avant la tâche j.

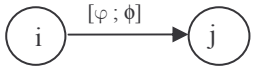


Sinon, il suffit d'inverser les tâches i et j.

- cas général : $a_i\neq a_j$ et $b_i/a_i\neq b_j/a_j$: dans ce cas, soit il existe une date pivot :



Soit il existe une fenêtre pivot :



A partir de l'ensemble des tâches, il est alors possible de créer un graphe temporel simple de n sommets (correspondant aux n tâches) et de $n(n-1)/2$ arcs valués (correspondant aux contraintes de précédence mises en évidence par le calcul de la date pivot ou de la fenêtre pivot pour chaque couple de tâches).

Exemple

Soit le problème à *4* tâches suivant :

Indice	fonctions	Rapports b_i/a_i	a_i	b_i
1	$p_1(t) = 0{,}2.t^2+10$	50	0,2	10
2	$p_2(t) = 0{,}2.t^2+8$	40	0,2	8
3	$p_3(t) = 0{,}36.t^2+8$	22,22	0,36	8
4	$p_4(t) = 0{,}1.t^2+4$	40	0,1	4

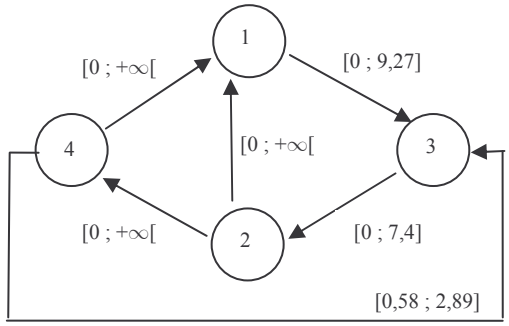
Le but de cet exemple est de présenter les différentes relations de précédences possibles entre deux tâches. Ces relations ont été mises en évidence dans la *démonstration 3*.

1er cas : *hypothèse 1* vérifiée.
Les tâches *4* et *1* peuvent être ordonnées par rapport b_i/a_i croissant et a_i croissant. La tâche *4* devra toujours être ordonnancée avant la tâche *1*.

2ème cas : *hypothèse 1* non vérifiée.

- $a_1=a_2$: dans ce cas les tâches doivent être ordonnées par b_i croissant : la tâche *2* devra toujours être ordonnancée avant la tâche *1*.
- $a_2 \neq a_4$ et $b_2/a_2=b_4/a_4$: dans ce cas les tâches doivent être ordonnées par a_i croissant. La tâche *2* devra toujours être ordonnancée avant la tâche *4*.
- cas général : entre les deux tâches *1* et *3*, il existe une date pivot $\varphi=9,27$. Il en est de même entre les tâches *3* et *2* : $\varphi=7,4$. Entre les tâches *4* et *3*, il existe une fenêtre pivot : [0,58 ; 2,89] .

Le graphe temporel simple est alors le suivant :



5 ALGORITHME

Nous avons programmé la méthode décrite ci-dessus afin de déterminer pour un problème donné, le graphe temporel simple associé.

A partir de ce graphe, nous proposons un algorithme de liste basé sur la règle suivante :

- A la suite d'une tâche *i* donnée, on ordonnance la tâche *j*, successeur de la tâche *i* dans le graphe temporel, dont l'intervalle a la plus petite borne supérieure.

Nous pouvons traduire cette règle par le fait que nous affectons la tâche suivante la plus contrainte au sens la plus « pressée » dans le temps par rapport aux différents successeurs.

Nous utilisons un algorithme de recherche en profondeur d'abord.

Algorithme

$t \leftarrow 0$
Choisir une tâche de départ *i*
Affectée (*i*)
Tâche_C $\leftarrow i$
$t \leftarrow t+p_i(t)$
`Tant Que` toutes les tâches ne sont pas affectées
  `Pour` toutes les tâches successeurs de Tâche_C
    Choisir la tâche *j* non encore affectée ayant la plus petite borne supérieure
  `FinPour`
  Tâche_Courante $\leftarrow j$
  Affectée (*j*)
  $t \leftarrow t+p_j(t)$
`FinTantQue`

Cet algorithme est en $O(n^2)$ pour un problème de *n* tâches si on fixe la tâche de départ. Si on adapte cet algorithme pour toutes les tâches de départ possible, sa complexité est alors $O(n^3)$.

6 CONCLUSION ET PERSPECTIVES

Nous nous sommes intéressés dans cet exposé aux problèmes à une machine dans lesquels, les durées d'exécution des tâches ne sont plus des constantes mais dépendent du temps. Le problème $1|p_i(t)=a_i.t^n+b_i|C_{\max}$ est polynomial sous certaines conditions. Nous limitons nos travaux au cas particulier $n=2$ lorsque celui-ci est NP-difficile.
Nous nous sommes alors intéressés à l'ordre relatif entre deux tâches au cours du temps.
La définition de la date ou de la fenêtre pivot entre deux tâches a permis de construire un graphe temporel simple représentant l'ensemble des contraintes de précédences.
Il représente, à la date $t=0$, l'ensemble des contraintes de précédences entre chaque paire de tâches.
Une solution réalisable du problème consiste alors à rechercher un chemin dans ce graphe en mettant à jour une variable temporelle à chaque passage par un sommet. En effet, le passage par un sommet *i* correspond à l'exécution de la tâche *i* ; la variable temporelle doit être augmentée de la durée d'exécution de la tâche *i* :
$t \leftarrow t+p_i(t)$ avec $p_i(t)=a_i.t^2+b_i$.
A partir de ce graphe, nous pouvons définir un algorithme de recherche de solutions réalisables. Maintenant, il est intéressant de comparer cet algorithme avec d'autres algorithmes. Plusieurs pistes sont envisageables : théorie des graphes ou intelligence artificielle.

REFERENCES

Alidae B. and NK. Womer, 1999, Scheduling with time dependent processing times : Review and extension,

Journal of the Operational Research Society, vol. 50, p. 711-720.

Barbeau E. J., 1989, Polynomials, *Springer-Verlag*, p. 18-23.

Cheng TCE. and Q. Ding, 1998, The complexity of scheduling starting time with dependent tasks with release times, *Information Processing Letters*, vol. 65, p. 75-79.

Dechter R., Meiri I., Pearl J., 1991, Temporal constraint networks, *Artificial Intelligence*, vol. 49, p. 61-95.

Doneddu A., 1984, Polynômes et algèbre linéaire, *Vuibert*, Tome 2, p. 64-74.

Durand E., 1971, Solutions numériques des équations algébriques, *Masson*, p.149-153.

Garey M. R., Johnson D. S., 1979, Computers and intractability, *Freeman*, New-York.

Gawiejnowicz S., Pankowska L., 1995, Scheduling jobs with varying processing times, Information Processing Letters, vol. 54, pp. 175-178.

Goblot R., 2001, Algèbre commutative, *Dunod*, p. 233-238

Guegnard F., 2006, Utilisation de la méthode Tabou pour la validation d'un générateur de problèmes à une machine où les durées d'exécution des tâches dépendent du temps, *META'06*, Hammamet, Tunisie.

Guegnard F., Peridy L., Rivreau D., 2004a, Single Machine with Time-Dependent Processing Time, *PMS'04*, Nancy, France, p. 135-138.

Guegnard F., Peridy L., Rivreau D., 2004b, Branch and Bound Method for the Single Machine with Time-Dependent Processing Time, *CO'04*, Lancaster, Angleterre.

Sundararaghavan P. S., Kunnathur A. S., 1994, Single machine scheduling with start time dependent processing times: Some solvable cases, European Journal of Operational Research, vol. 78, issue 3, pp. 394-403.

Wagneur E., Sriskandarajah, C., 1992, EDS with State Dependent Event Durations: Look Ahead Policies and Scheduling, Proceedings of the 31st IEEE Conference on Decision and Control, Tucson, Arizona, U.S.A, pp. 413-418.