La séquence globale minimale

Abdelghani Souilah

UNITÉ DE RECHERCHE
INRIA-LORRAINE

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1328

Programme 5
Automatique, Productique,
Traitement du Signal et des Données

LA SEQUENCE GLOBALE MINIMALE

Abdelghani SOUILAH

Octobre 1990

LA SEQUENCE GLOBALE MINIMALE
"The Global Minimal Sequence"

Abdelghani SOUILAH

Projet SAGEP
INRIA - Lorraine CESCOM
Technopôle Metz 2000
4, rue Marconi 57070 METZ
Tél : 87 20 35 10 Fax : 87 76 39 77
&
Centre de Développement des Technologies Avancées
Haut Commissariat à la Recherche
128,rue Mohammed Gacem, El-Madania
Alger, ALGERIE

RESUME :

Le problème de la Séquence Globale Minimale, connue dans la littérature sous le nom de " Plus Courte Super-séquence Commune ", est un problème NP-complet. On le rencontre dans plusieurs domaines tels que la linguistique computationnelle et la lexicographie . Nous le retrouvons ici dans un problème où il s'agit de construire une séquence minimale de machines telle qu'une série de pièces qui doivent passer sur certaines de ces machines dans un certain ordre n'ait à parcourir la séquence que dans un seul sens. Bien entendu, cette séquence peut contenir plusieurs fois la même machine, et une pièce qui parcourt la séquence peut éviter certaines machines .

Nous présentons dans ce papier une heuristique rapide permettant de trouver cette séquence. Nous développons aussi une méthode exacte nous permettant d'évaluer notre heuristique .

Mots et phrases clés :

Plus Courte Séquence Commune (Shortest Common Supersequence), NP-Complet, Job-Shop, Routages, Depth-first.

ABSTRACT :

The problem of the Global Minimal Sequence, known in the literature as the " Shortest Common Supersequence problem, " is NP-Hard. It is encountered in several areas as Computational Linguistic and Lexicography. We are looking for a sequence of machines such that : (i) the routing corresponding to each part belonging to a given set of parts can be obtained by cancelling some of the elements of the sequence, (ii) a shorter sequence having property (i) does not exist. Of course, this minimal sequence can contain the same machine several times.

In this paper we present a fast heuristic permiting to find this minimal sequence, and we also develop a precise method which is used to evaluate the heuristic .

Keywords :

Shortest Common Supersequence, NP-Hard, Job-Shop, Routing, Depth-first.

1 INTRODUCTION :

Le regroupement de pièces en familles est nécessaire si l'on veut simplifier la gestion d'un système de production du type **Job-Shop** en utilisant une approche hiérarchisée.

La construction des familles passe par la recherche d'une séquence globale minimale (**S.G.M**) . En effet, cela est nécessaire si l'on veut que nos familles regroupent des pièces qui non seulement visitent les mêmes machines, aient des temps de passage proches, mais aussi aient le même ordre de passage sur les machines .

Cette S.G.M est la plus petite séquence de machines telle que tout routage de pièce puisse en être déduit par suppression d'un ou plusieurs éléments de la séquence.

Ce problème, est connu dans la littérature sous le nom de "Plus Courte Super-séquence Commune" ou "Shortest Common Supersequence (S.C.S)" [1] et [2]. Dans [1] David Maier montre que ce problème est NP-difficile (Voir Théorème 5 dans [1]).

Nous allons donc, dans une première étape, présenter l'heuristique proposée ainsi qu'un exemple illustratif, suivi d'une procédure statistique de validation. La deuxième partie sera consacrée à la méthode exacte : algorithme et application. Avant de conclure, nous ferons une comparaison entre les résultats donnés par l'heuristique et ceux données par la méthode exacte.

Dans l'annexe A, on trouvera le programme de l'heuristique ainsi que les résultats de quelques essais. L'annexe B contient le programme de la méthode exacte et les résultats de quelques exemples.

2. UNE HEURISTIQUE POUR LA DETERMINATION DE LA SEQUENCE GLOBALE MINIMALE

2.1 Présentation du problème :

Soit un système de fabrication du type **Job-shop**, capable de produire un ensemble $P = \{ P_1, P_2, \ldots, P_q \}$ de **q** types de pièces. A chaque type de pièce correspond une séquence connue de machines appelée routage. Les routages associés aux différents types de pièces peuvent être différents. Ils forment l'ensemble $R = \{R1, R2, \ldots, Rq\}$. Un routage et les temps de passage associés constituent une gamme de fabrication. Le nombre total **m** de machines ainsi que le nombre **q** de types de pièces sont connus.

Les routages se présentent comme suit :

R_1 : $\{M_1, M_3, \ldots\ldots, M_{10}\}$

R_2 : $\{M_5, M_2, \ldots\ldots, M_2\}$

. : .....

. : .....

R_q : $\{M_9, M_{11}, \ldots\ldots, M_3\}$

Notre objectif est de trouver une S.G.M au système donné.

2.2 Propriétés de la S.G.M. :

La S.G.M. posséde les propriétés suivantes :

1 - La S.G.M. **existe** pour tout système donné,
2 - La S.G.M. **n'est jamais unique**, sauf dans le cas où parmi les routages il en existe un qui peut contenir tous les autres.

Démonstrations :

I EXISTENCE DE LA S.G.M. .

Définitions :

1 / On dit qu'une séquence vérifie la propriété $\mathcal{P}$, si tout les routages de pièces données peuvent être déduits par suppression d'un ou plusieurs éléments de la séquence (on dira aussi qu'elle englobe ces routages);

2 / On définit aussi $\mathcal{M}j$ tel que :

$\mathcal{M}j$ = { ensemble des séquences ayant j éléments et vérifiant la propriété $\mathcal{P}$ };

Propriété :

P': *i) Quels que soient i, j tels que $i \leq j$, si $\mathcal{M}i \neq \emptyset$ alors $\mathcal{M}j \neq \emptyset$.*

ii) $\mathcal{M}_{\sum ni} \neq \emptyset$, avec $\sum ni$ la somme des nombres d'éléments des routages .

Interprétation :

Cette propriété montre que s'il existe une séquence ayant i éléments qui vérifie la propriété $\mathcal{P}$ ($\mathcal{M}\, i \neq \emptyset$), alors on pourra toujours trouver une séquence à j éléments qui la vérifie aussi ($\mathcal{M}\, j \neq \emptyset$). Et que la plus évidente est celle contenant $\sum ni$ éléments.

Preuve :

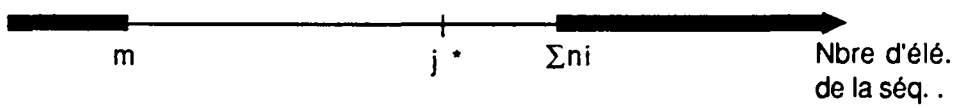
i) Soit S une séquence appartenant à $\mathcal{M}\, i$. Et Soit S' une séquence formée de la séquence S à laquelle on ajoute j - i éléments quelconques . S' vérifie la propriété $\mathcal{P}$ puisque S la vérifie et S' appartient à $\mathcal{M}\, j$ puisqu'elle possède j éléments ($j \geq i$).

D'où $\mathcal{M}j \neq \emptyset$. *c.q.f.d*

ii) On peut toujours trouver dans $\mathcal{M}_{\sum ni}$ une séquence S composée de la concatenation des q routages donnés .

"---R1---'---R2---'--- .....------'---Rq---"

la séquence S vérifie bien la propriété $\mathcal{P}$ *d'où* $\mathcal{M}_{\sum ni} \neq \emptyset$. *c.q.f.d.*



On pourra donc toujours trouver un nombre noté j^*, tel que :

$$j^* = min\,\{ m \leq j \leq \sum ni \,/\, \mathcal{M}j \neq \emptyset \}$$

j^* sera le nombre d'éléments des S.G.M. et l'ensemble des S.G.M. noté $\mathcal{M}_{SG\mathcal{M}}$ est inclus dans l'ensemble $\mathcal{M}_{j^*}$.

D'où l'existence de la S.G.M. pour tout ensemble de routages donné . *c.q.f.d.*

2 NON UNICITE DE LA S.G.M. .

On a montré ci-dessus l'existence d'un ensemble de Séquences Globales Minimales dont le cardinal est plus grand ou égal à un. On confirme cela en donnant un contre exemple pour l'unicité.

Contre exemple :

Soit le système suivant comportant deux machines M1 & M2 *et produisant deux types de pièces* P1 & P2. *On constate bien qu'aucun des deux routages ne peut contenir l'autre .*

P1 : M1 M2
P2 : M2 M1

Les deux séquences { M1 M2 M1 } et { M2 M1 M2 } *sont bien des S.G.M..*

D'où la non-unicité de la S.G.M.. *c.q.f.d.*

2.3. L'heuristique : Algorithme et application

Connaissant la nature combinatoire de notre problème nous donnons une heuristique pour la recherche d'une séquence aussi proche que possible de la S.G.M. .

L'algorithme que nous proposons est constitué, principalement, de deux parties : le traitement de données, puis la détermination de la S. G. M. proprement dite.

2.3.1 Traitement des données :

Dans cette partie on transforme les données brutes en données plus élaborées. Les étapes sont les suivantes :

a. <u>Acquisition des données :</u>

Pour chaque type de pièce on donne la succession de machines qu'il traverse

nommée routage. Dans l'algorithme on utilise les notations suivantes :

q : Nombre de types de pièces fabriqués par le système,
m : Nombre de machines dans le système,
NM(e) : Nombre de machines constituant le routage du type **e**; **e = 1, ..., q**,
I(e,i) : Indice de la **i^{ème}**machine du routage **e** ; **i = 1, ...,NM(e)** et **e = 1, ..., q**.
Les routages ainsi acquis sont schématisés comme suit :

n° du type	routage
1	[.....]
2	[.....]
3	[.....]
.	.
(q-2)	[.....]
(q-1)	[.....]
q	[.....]

b. Classement des routages par ordre croissant de leur taille :

Les routages sont classés dans l'ordre croissant du nombre de machines qu'ils contiennent. Ce classement facilite le traitement dans les étapes suivantes .

On affecte à chaque routage un numéro d'ordre qui indique sa nouvelle position. Ce numéro d'ordre est noté **A(i)**.

Comme notations supplémentaires, on utilise dans l'algorithme :

A(i) : le rang, dans le nouveau classement, du routage **i, i = 1, ... , q.**

Les routages paraîtront comme suit :

n° du type	n° d'ordre	Routage
10	A(10) = 1	[.....]
q	A(q) = 2	[.....]
.	.	.
i	A(i)	[.....]
.	.	.
9	A(9) = (q-1)	[.....]
18	A(18) = q	[.....]

c. Elimination des routages multiples :

Dans le souci de minimiser le nombre de routages à traiter, on essaye d'éliminer tous les routages qui sont contenus dans des routages plus grands. Cela exige un classement des routages dans l'ordre croissant de leur taille, d'où l'utilité de l'étape précédente .

La procédure se déroule comme suit :

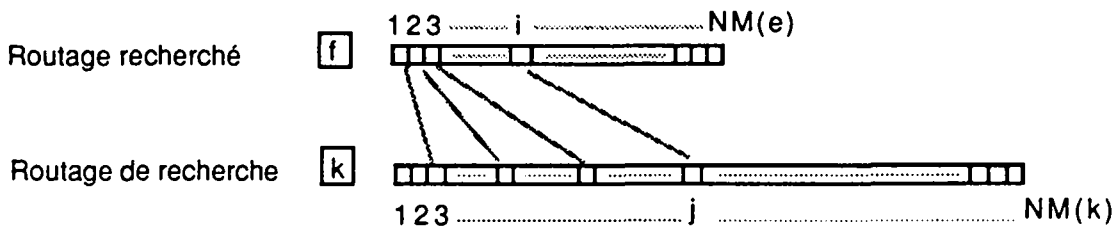
On part des routages de plus petit numéro d'ordre et on cherche s'ils sont contenus dans un des routages de numéro d'ordre supérieur.

Si oui, on annule le routage testé en mettant à zero son numéro d'ordre .

On passe ensuite au routage de numéro d'ordre suivant, et on s'arrête après avoir traité le routage de numéro d'ordre **(q-1)** .

Comparaison de deux routages :

Soit **f** le numéro d'ordre d'un routage en cours de test et **k** **(k > f)** le numéro d'ordre du routage avec lequel on effectue la comparaison. On considère les éléments de **f** dans l'ordre croissant et on les compare avec les éléments de **k** considérés également dans l'ordre croissant à partir de l'élément suivant le dernier élément de **k** trouvé identique à un élément de **f**. La figure1 illustre cette situation .



Les pointillés indiquent des éléments identiques .

L'élément **i** sera à comparer avec **j, j+1, ..., NM(k)** .

_ fig. 1 Illustration de l'algorithme de comparaison des routages _

Remarque : Si **(NM(f) - i > NM(k) - j)**, alors on peut arrêter la comparaison : le routage **k** ne peut contenir le routage **i** .

Le résultat du traitement peut se schématiser comme suit :

n° du type	n° d'ordre	Routage
10	A(10) = 1	[.....]
P	A(q) = 0	routage éliminé
2	A(2) = 3	[.....]
.	.	.
.	.	.
i	A(i) = 0	routage éliminé
.	.	.
.	.	.
18	A(18) = q	[.....]

d. Remaniement du tableau après élimination :

Aprés classement des routages dans l'ordre croissant de leur taille et des routages multiples, on procède à une organisation finale des routages, afin qu'ils soient facilement exploitables dans la suite. On conservera **q** et **A(i)** pour désigner respectivement le nombre des routages restants et le nouveau n° d'ordre.

Les routages seront utilisés sous cette forme :

n° du type	n° d'ordre	Routage
10	A(10) = 1	[.....]
2	A(2) = 2	[.....]
.	.	.
.	.	.
i+1	A(i+1)	[.....]
.	.	.
.	.	.
18	A(18) = q	[.....]

Avant de passer à l'algorithme, donnons d'abord un petit exemple permettant de bien saisir cette partie (dans les routages, les machines sont representées par leur indice) :

Acquisition des données :

type	routage
P1	2 1 4
P2	3 1 5 3 4 2
P3	5 2
P4	4 2 3 5 4
P5	1 5 4 2

Classement des routages dans l'ordre croissant de leur taille :

n° du type	n° d'ordre	routage
3	1	5 2
1	2	2 1 4
5	3	1 5 4 2
4	4	4 2 3 5 4
2	5	3 1 5 3 4 2

Elimination des routages multiples :

n° du type	n° d'ordre	routage
3	1	Rout. éliminé retrouvé dans le rout. 5
1	2	2 1 4
5	0	Rout. éliminé retrouvé dans le rout. 2
4	4	4 2 3 5 4
2	0	3 1 5 3 4 2

Organisation finale des routages :

n° du type	n° d'ordre	routage
1	1	2 1 4
4	2	4 2 3 5 4
2	3	3 1 5 3 4 2

Algorithme :

1. Acquisition des données :
   1.1. Donner le nombre **q** de types de pièces ;
   1.2. Donner le nombre **m** de machines dans le système
   1.3. **i = 1** ;
   1.4. Tant que (**i ≤ q**) faire :
   Donner le nombre **NM(i)** de machines dans le système dans le routage du type **i** ;
   Donner **RO(i)** le taux de production du type **i** ;
   j=1;
   1.4.1. Tant que (j ≤ NM(i)) faire :
   Donner l'indice **I(i,j)** de la machine **j** du routage **i** ;
   Donner **T(i,j)** le temps de passage du type **i** sur la machine **j** ;
   j=j+1;
   Fin tant que ;
   1.4.2. **i = i + 1** ;
   Fin tant que ;
2. Classement des routages dans l'ordre croissant de leur taille :
   2.1. **i = 1** ;
   2.2. Tant que (**i < Pq**) faire :
   m = NM(i) :
   j = 1 ;
   Tant que (**j ≤ P**) faire :
   a. Si (**m < NM(j)**)
   aller à b.
   Si non
   ii = j;
   m = NM(j) ;
   Fin si non
   b. **j = j + 1** ;
   Fin tant que ;
   A(i) = ii ;
   NM(ii) = 10 * q ;
   i = i + 1 ;
   Fin tant que ;
3. Elimination des routages multiples :
   3.1. n = q ;
   3.2. e = 1 ;
   3.3. Tant que (e < q) faire :
   3.3.1. k = e + 1 ;

3.3.2. i = 1 ; j = 0 ;

3.3.3. Tant que (i ≤ NM(A(e))) faire :

a. j = j + 1 ;

B = I (A(e),i) ;

C = I (A(e),j) ;

X = NM(e) - i ;

Y = NM(k) - j ;

b. Si (Y ≥ X)

k = k + 1 ;

Si (k ≤ P)

aller à 3.3.2. ;

si non

aller à 3.3.5. ;

Si non si (B ≠ C)

aller à a. ;

si non

i = i + 1 ;

Fin si ;

Fin tant que ;

3.3.4. A(e) = 0 ;

n = n - 1 ;

3.3.5. e = e + 1 ;

Fin tant que ;

3.4. Si n = 1 ;

3.4.1. Le routage restant n'est autre que la S. G. M. ;

3.4.2. Fin ;

Fin si ;

4. Organisation du tableau après élimination :

4.1. i = 1 ;

4.2. Tant que (i ≤ q) faire ;

4.2.1. Si (A(i) ≠ 0)

i = i + 1 ;

Si non

a. j = i + 1 ;

Tant que (j ≤ q) faire :

A(i + 1) = A (i) ;

q = q - 1 ;

j = j + 1 ;

Fin tant que ;

b. Aller 4.2.

Fin si non ;

Fin tant que ;

5. Construction de la S. G. M. (voir la suite après la présentation de l'heuristique) :

2.3.2. L'heuristique :

L'heuristique proposée est itérative :

Pour chaque machine **i**, on considère le rapport $\mathbf{NF^k(i)} / \mathbf{NT^k(i)}$ appelé $\mathbf{PRIMO^k(i)}$ où :

$\mathbf{NF^k(i)}$ est le nombre de routages dans lesquels **i** est la dernière machine à la $\mathbf{k^{ème}}$ itération, $\mathbf{0 \leq NF^k(i) \leq q}$; **q** étant le nombre de routages .

$\mathbf{NT^k(i)}$ est le nombre d'occurences de **i** dans l'ensemble des routages à la $\mathbf{k^{ème}}$ itération,

$$NT^k(i) = T^0(i) - \sum_{j \,/\, PRIMO^j(i) = Max_{l=1,\ldots,m} PRIMO^j(l)} NF^j(i)$$

Evidemment : quelque soit $\mathbf{i / 1 \leq i \leq M}$; $\mathbf{0 \leq PRIMO^k(i) \leq 1.}$

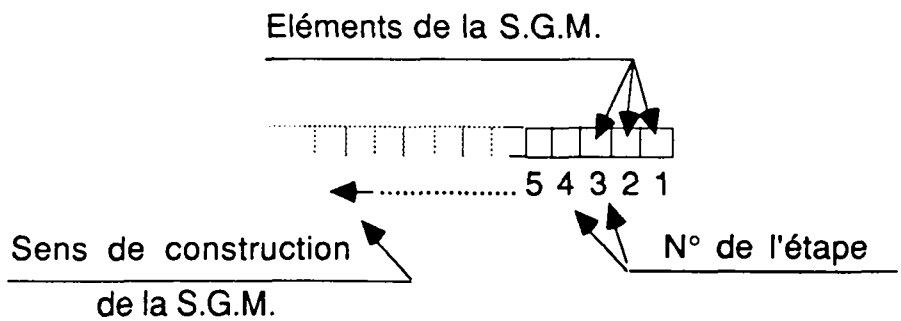
A chaque itération, on place juste avant le dernier placement effectué (ou en fin de la S.GM. si nous sommes à la première itération, voir figure 2) la machine ayant la plus grande valeur de $\mathbf{PRIMO^k}$.

$$\underset{i \in F_k}{Max} PRIMO^k(i)$$

$\mathbf{F^k(i)}$ est l'ensemble des indices des machines ayant un $\mathbf{PRIMO^k}$ non nul.

En cas d'ex aequo on retient la machine la plus souvent présente en fin de routages. Si une fois de plus il y a ex aequo le choix s'effectue au hasard .

On supprime ensuite la machine retenue de toutes les fins de routage puis on passe à l'itération suivante. On s'arrête lorsque tous les routages sont vides .



_ fig.2 Mode de construction de la S.G.M. ._

Interprétation :

La machine prioritaire sera donc celle dont le rapport entre le nombre total de ses présences à la fin des routages (NF^k) et celui de ses présences dans la totalité des routages (NT^k) restant à la $k^{ème}$ itération est le plus grand. Lorsque ce nombre est égal à un, la machine qui lui correspond sera sans aucun doute prioritaire, puisqu'elle n'est présente qu'à la fin des routages. Par contre, dans le cas où ce nombre est nul pour une machine donnée, cette dernière n'aura certainement aucune chance d'être sélectionnée. Cependant, dans le cas où ce nombre est compris entre zéro et un :

Si deux machines 'i' et 'j' sont telles que :

i) - 0 < PRIMO(i) <1 et 0 < PRIMO(j) <1,

ii) - PRIMO(i) ≥ PRIMO(j),

iii) - et NT(i) ≥ NT(j) *et* NF(i) ≥ NF(j),

la machine 'i' est prioritaire par rapport à la machine 'j'.

Dans tous les autres cas on ne peut rien dire. C'est là que l'heuristique trouvera son utilité. Puisque dans ces cas-là, le resultat ne dépend plus des nombres de présence des machines dans l'ensemble et/ou à la fin des routages, mais de leurs positions exactes dans les routages, ce qui fait la NP-Complexité du problème. On estime que le résultat que donne l'heuristique ne sera pas trop éloigné de l'optimum .

2.3.3. Exemple illustratif :

Soit le système suivant :

P1 : M4 M3 M1
P2 : M2 M3 M4
P3 : M4 M2 M1
P4 : M3 M4

La construction de la séquence minimale se fait comme suit :

Itération N°	N° type	NT	NF	PRIMO	Séq. en construction
1	1	2	2	1	
	2	2	0	0	
	3	3	0	0	
	4	4	2	1/2	1
2	1	0	0	0	
	2	2	1	1/2	
	3	3	1	1/3	
	4	4	2	1/2	2 1
3	1	0	0	0	
	2	1	0	0	
	3	3	1	1/3	
	4	4	3	3/4	4 2 1
4	1	0	0	0	
	2	1	0	0	
	3	3	3	1	
	4	1	0	0	3 4 2 1
5	1	0	0	0	
	2	1	1	1	
	3	0	0	0	
	4	1	1	1	2 3 4 2 1
6	1	0	0	0	
	2	0	0	0	
	3	0	0	0	
	4	1	1	1	4 2 3 4 2 1

Vérification finale :

S. G. M.	4	2	3	4	2	1
P1	*		*			*
P2		*	*	*		
P3	*	*				*
P4			*	*		

On constate que la 2ème machine M2 (à droite) n'est pas utilisée, elle est donc éliminée .

La *Séquence Globale Minimale* pour cet exemple sera :

4 2 3 4 1.

2.3.4 Algorithme :

1 _ Initialisation :

1.1. A partir du tableau de données, construire le vecteur des nombres d'utilisation des machines dans le système : $NT^1(i)$; $i = 1, 2, \ldots, q$.

1.2. $k = 1$;

2 _ Itérations :

Faire : 2.1. Construire le vecteur des nombres de présences des machines à la fin des routages : $NF^k(i)$; $i = 1, 2, \ldots, q$;

2.2. Calcul de la fonction de priorité : $PRIMO^k(i) = NF^k(i) + NT^k(i)$; $i=1, \ldots, q$;

2.3. Détermination de la machine prioritaire , celle qui correspond au :

$$\text{Max}_{i=1, 2, \ldots, q} PRIMO^k(i) ;$$

Si (plusieurs machines correspondent à ce $PRIMO^k$)

on prend celle qui a le plus petit indice ;

Fin si;

2.4. Réactualisation du nombre de présences de la machine prioritaire dans l'ensemble des routages : $NT^{k+1}(i) = NT^k(i) - NF^k(i)$;

i : étant l'indice de la machine prioritaire .

2.5. Elimination de la machine prioritaire des fins de routages où elle se trouvait ;

2.6. Placer cette machine dans la séquence en construction à la $k^{ième}$ position, en partant de la droite ;

2.7. Si toutes les machines d'un certain routage sont éliminées ;

$q = q - 1$;

Fin si ;

2.8 _ $k = k + 1$;

Tant que ($q \neq 0$) ;

3 - Vérification finale :

Retrouver tous les routages dans la **Séquence Globale** ainsi obtenue et finir par éliminer les machines en excès pour avoir la **S. G. M.** .

3. UNE METHODE EXACTE :

Nous allons, dans ce chapitre, trouver une méthode exacte de résolution de ce problème afin de mieux évaluer notre heuristique et d'avoir une idée plus précise de son efficacité. Afin d'examiner tous les cas possibles, nous utiliserons une méthode de prospection d'arbre du type depth-first avec amélioration de la profondeur maximale.

Dans la suite, nous allons présenter la méthode, donner l'algorithme et enfin faire une petite application illustrative. Avant de conclure, nous ferons une comparaison entre les résultats de l'heuristique et ceux de cette méthode exacte, pour une série d'exemples de taille réduite .

3.1. Présentation de la méthode :

Cette méthode consiste à explorer un arbre dont les nœuds représentent les états de la séquence en construction, son nombre d'éléments, sa composition et sa nature : globale minimale(*), globale(**) ou non globale(***). Les branches repésentent les transitions entre les nœuds, c'est à dire, l'ajout d'une machine à la séquence d'un nœud donnant naissance à un nœud de niveau supérieur .

Dans le but de réduire le nombre de transitions vers des nœuds fils possibles d'un nœud donné, on écarte toutes les machines n'ayant aucune chance de se trouver dans cette position dans la S.G.M.. On fait cela en utilisant deux propriétés de la fonction **PRIMO** définie en **2.2.2.**, qui sont les suivantes :

1. Seules les machines **i / i = 1, ..., M** ayant un $\mathbf{PRIMO^k(i) \neq 0}$ peuvent être placées dans cette position de la **S.G.M.** ,

2. Après avoir déterminé :

$$\underset{i \in F}{\text{Max}}\ {}_{k}\text{PRIMO}^{k}(i)$$

où l'indice **k** représente, cette fois-ci, le numéro du niveau du nœud pour lequel on cherche les fils, nous pouvons distinguer deux cas :

(*) Une séquence est dite globale minimale lorsqu'elle contient tout les routages du système tout en ayant un nombre minimum d'éléments.

(**) Elle est dite globale lorsqu'elle englobe tout les routages donnés sans être minimale.

(***) Elle est non globale si elle ne contient pas tous les routages donnés.

a. Max PRIMO^k(i) = 1. Dans ce cas, seules les machines correspendant à cette valeur sont considérées comme prioritaires, et seront les fils du nœud considéré.

b. 0 < Max PRIMO^k(i) < 1. Dans ce cas toutes les machines "i" ayant un PRIMO^k(i) non nul seront considérées comme prioritaires et leur nœud correspondant comme fils du nœud considéré (voir l'interprétation en 2.2.2.) .

En faisant cela, on aura réduit énormément le nombre de séparations possibles à chaque nœud atteint, d'où réduction de la taille de l'arbre a prospecter.

3.2. Réalisation :

Définitions :

On définit trois types de nœud principeaux liés les uns aux autres et sont :

Nœud père : le nœud précédant un nœud donné et avec lequel il a une liaison, est dit nœud père de ce dernier.

Nœud fils : le nœud descendant d'un nœud donné et avec lequel il a une liaison, est dit nœud fils de ce dernier.

Nœud frère: le nœud se trouvant sur un même niveau qu'un nœud donné et ayant le même père qui lui, est dit nœud frère de ce dernier.

On utilise aussi deux autres types de nœuds assez particuliers qui sont :

Nœud fils_1 : le nœud fils_1 d'un nœud donné est le nœud fils de celui-ci, classé premier dans la liste de ces fils.

Nœud frère_1 : les nœuds frères_1 d'un nœud donné sont les nœuds issus d'un même père que lui et classés après lui dans la liste des fils de ce dernier.

Pour faciliter la prospection et les calculs à chaque étape, les nœuds de l'arbre doivent contenir les informations suivantes (voir fig.3) :

- le niveau où il se trouve,
- le contenu de ses routages apres suppression de l'élément sélectionné,
- la séquence en construction,
- le nombre de fils, s'ils existent,

- l'adresse du fils aîné, s'il existe,
- l'adresse du père.

L'adresse du premier frère_1 étant égale à "sa propre adresse + 1".

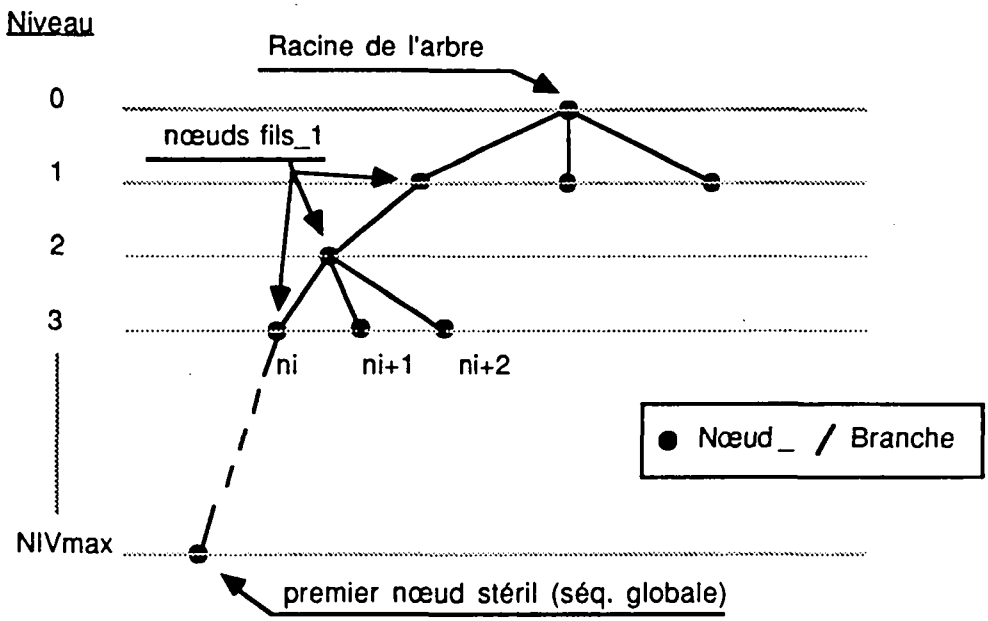


_fig.3 Définition des constituants de l'arbre _

COMMENT SE FAIT LA PROSPECTION DE L'ARBRE ?

A partir du nœud initial (*) (la racine de l'arbre), on plonge une première fois en profondeur en cherchant tous les fils de chaque nœud atteint (c'est-à-dire, toutes les machines prioritaires possibles) et en immergeant à chaque fois, à partir du fils_1 (voir fig. 3), jusqu'à l'obtention d'une séquence globale (**S. G.**) . Le nœud correspondant à cette S.G est dit stérile (**N.S.**).

Comme le nombre d'éléments de la séquence obtenue jusqu'à un nœud quelconque est égal au rang du niveau correspendant à ce nœud, le niveau de ce premier nœud stérile sera la **Profondeur maximale (P.M.)** initiale de notre arbre, **P.M. = NIVmax** (**NIVmax** étant le niveau maximum atteint).

En ce moment là, on remonte pour commencer un processus itératif fonctionnant comme suit :

En remontant, on cherche au fur et à mesure un nœud père ayant un nombre

---

(*) Ce nœud initial est bien particulier, du fait qu'à son niveau la séquence n'est pas encore commencée et qu'il n'a pas de père .

de nœuds frère_1 non nul. Par exemple, dans la figure 3 on voit que le nœud **ni** a deux nœuds frère_1 (**ni + 1**) et (**ni + 2**) , alors que (**ni + 1**) n'en a qu'un seul (**ni + 2**) . Une fois ce nœud trouvé on replonge, sauf s'il se trouve au niveau zéro .

Lors de la redescente, et à chaque nœud atteint, quatre cas peuvent se présenter (voir figure 4) :

1. le nœud atteint n'est pas stérile alors que son niveau est égal à la P.M.,
2. le nœud atteint n'est pas stérile mais son niveau est inférieur à la P.M ,
3. le nœud atteint est stérile et son niveau est inférieur à la P.M.,
4. le nœud atteint est stérile et son niveau est égal à la P.M. .

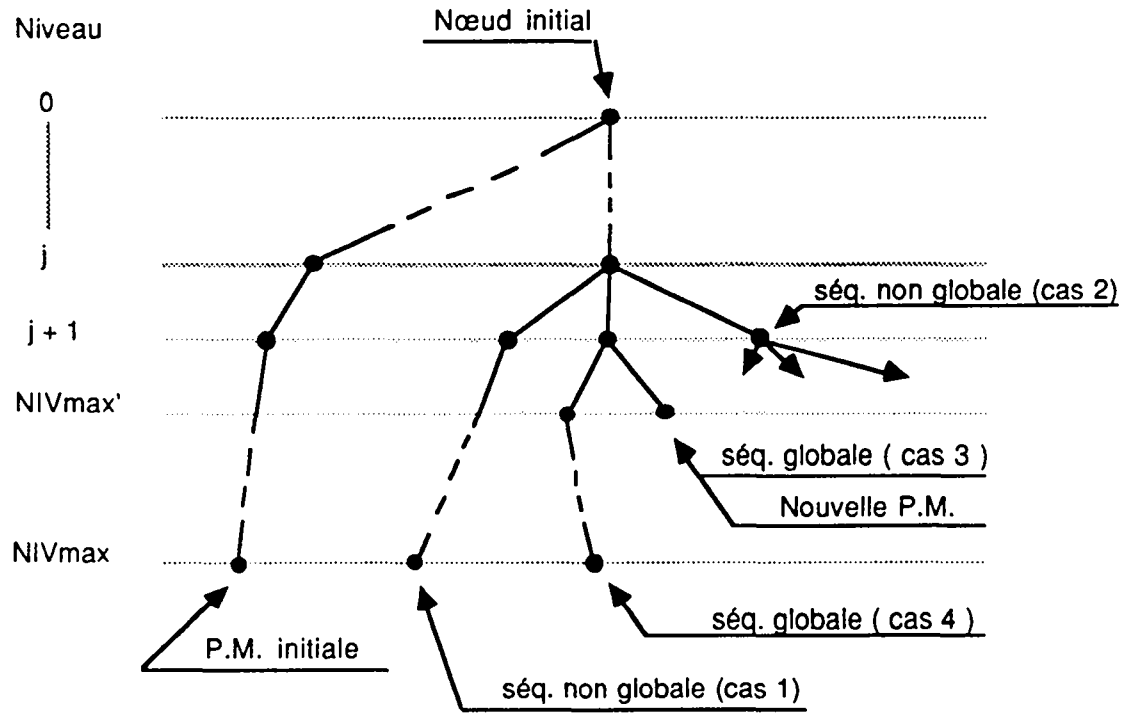


fig.4 Illustration des différents cas possibles lors de la prospection de l'arbre

ANALYSE DES DIFFERENTS CAS .

* Dans le premier et le quatrième cas, on remonte immédiatement et on reprend le processus à partir d'un nœud père ayant un nombre de nœuds frère_1 non nul, s'il existe. Sinon, fin du processus.

** Dans le deuxième cas, on poursuit notre immersion.

*** Alors que dans le troisième cas, on conserve momentanément la séquence obtenue comme séquence globale et on modifie la valeur de la **P.M.** qui sera égale désormais au numéro du niveau de ce nœud, **P.M. = NIVmax'**. On remonte alors pour reprendre la descente à partir d'un nœud père ayant un nombre de nœuds frère_1 non nul, s'il existe.

A chaque transition, on répète les opérations suivantes :

- Réactualisation du nombre de présences de la machine prioritaire concernée, dans l'ensemble des routages du nœud père,
$NT^{k+1}(i) = NT^{k}(i) - NF^{k}(i)$; **i** : indice de la machine prioritaire.

- Réduire de un le nombre d'éléments des routages où la machine prioritaire est à la fin.

- Affecter les routages ainsi obtenus au nœud fils.

- Placer la machine prioritaire à la première position dans la séquence du père et affecter cette nouvelle séquence au nœud fils.

- Si le nombre d'éléments d'un des routages est réduit à zéro, diminuer de un le nombre **q** de routages , **q = q - 1.**

Condition d'arrêt :

Le processus itératif s'arrêtera lorsque, lors d'une remontée, on n'arrive plus à trouver un nœud père ayant au moins un nœud frère_1 et cela jusqu'à atteindre le niveau zéro. On aura donc parcouru tout l'arbre.

Résultat :

La Séquence Globale Minimale (**S.G.M.**) sera la Séquence Globale (**S.G.**) du Nœud Stérile (**N.S.**) correspondant à la Profondeur maximale (**P.M.**) finale.

3.2. Algorithme :

Cet algorithme, comme le précédent, se compose de deux parties : une partie de traitement de données suivie de la partie concernant la détermination de la S.G.M.. Nous ne donnerons ici que la deuxième partie, la première étant identique à celle de l'heuristique.

1. Initialisation :
   NIV = 0 ;
   NIVmax = NIV^0 ;
   Affectation des constituants du nœud initial, en utilisant la procédure **Recherche de Machines Prioritaires** ;
2. Construction et prospection de l'arbre :
   Tant que (**NIV ≠ 0**) faire :
   2.1. On plonge en profondeur :
   Tant que (Nbre de routages restants, **q ≠ 0**) faire :
   2.1.1. **NIV = NIV + 1** ;
   2.1.2. Si (**NIV > NIVmax**) (**cas 1**)
   Aller à 2.3 ;
   Fin si ;
   2.1.3. Lors de la descente (**cas 2**), en utilisant la procédure **Recherche de Machines Prioritaires,** affecter à chaque nœud fils :
   - Le numéro du niveau ;
   - Le contenu des routages obtenus pour ce nœud ;
   - La séquence en construction ;
   - Le nombre de nœuds frère_1 ;

   Fin tant que ;
   2.2. Si (**NIV < NIVmax**) (**cas 3**)
   NIVmax = NIV ;
   La séquence globale intermédiaire sera celle du nœud considéré ;
   Fin si ;
   (**cas 4**) ;
   2.3. On remonte :
   Tant que (**NIV ≠ 0**) faire :
   2.3.1. Remonter au père avec **NIV = NIV - 1** ;
   2.3.2. Si (le père a au moins un frère_1)
   Aller en 2.4 ;
   Fin si ;
   Fin tant que ;
   2.4. On considère le premier frère_1 ;
   Fin tant que ;

3. Résultat :

La **S. G. M.** sera la **S. G.** du nœud stérile correspondant à la **P.M.** .

Procédure de recherche de machines prioritaires :

1. Nombre de présences :

Tant que (i < m) faire : (m: Nbre de machines du système)

1.2. N(i) = L(i) = 0 ; (N(i) nbre de présences de i dans le tout , L(i) nbre de présences de i en fin)

1.3. Tant que (j < q) faire : (q : Nbre de gammes)

1.3.1. Si (ce routage est éliminé):

Aller en 1.3. ;

Fin si ;

1.3.2. Calcul du nombre de présences de chaque machine dans le routage :

e = 0 ;

Tant que (e < NMRP(j)) faire : (NMRP(i) nbre de machines de la gamme i)

Si (i + 1 = I (A(j),e)) faire :

N(i) = N(i) + 1 ;

Fin si ;

e = e + 1 ;

Fin tant que ;

1.3.3. Calcul du nombre de présences de chaque machine en fin de routage :

Si (i + 1 = I (A(j), NMRP(j))) faire :

L(i) = L(i) + 1 ;

Fin si ;

j = j + 1 ;

Fin tant que ;

V(i) = L(i) + N(i) ; (V(i) l'indice de priorité)

i = i + 1 ;

Fin tant que ;

2. Détermination du Vmax

Max = 0 ;

Tant que (e < m) faire :

Si (V(e) > Max) faire

Max = V(e) ;

Fin si ;

Fin tant que ;

3. Résultats :

Si (Max == 1) faire

```
i = 0 ;
      Tant que (e < m) faire :
          Si (V(e) = Max) faire
              i=i+1;
                PRIO (i) = e + 1;
          Fin si ;
         e=e+1;
      Fin tant que ;
      Nbre de fils = i;
Fin si ;
Sinon si (Max < 1) faire
i = 0 ;
Tant que (e < m) faire :
      Si (V(e) ≠ 0) faire
           i=i+1;
           PRIO(i) = e+1;
      Fin si ;
     e=e+1;
Fin tant que ;
Nbre de fils = i;
Fin si ;
Fin
```

3.3. Application :

Nous donnons dans ce qui suit un exemple illustratif simple permettant de mieux comprendre le fonctionnement de la méthode.

Soit un Job-Shop constitué de trois machines et produisant trois types de produits. Le processus de fabrication de chaque type suit une gamme donnée. Les routages sont les suivants :

P1 : M1 M3 M2
P2 : M3 M1
P3 : M2 M1

Il s'agit donc de trouver la S. G. M. de ce système.

Chaque nœud de l'arbre comporte les informations suivantes :

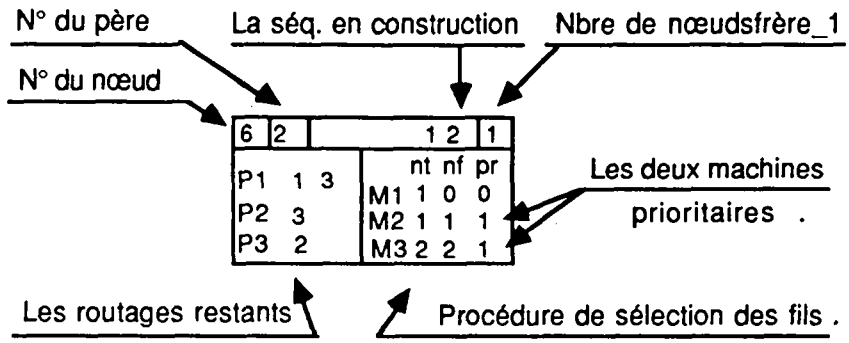


_fig. 5 Légende du contenu d'un nœud de l'arbre _

L'arbre exploré pour la détermination de la S.G.M. de ce système est le suivant :

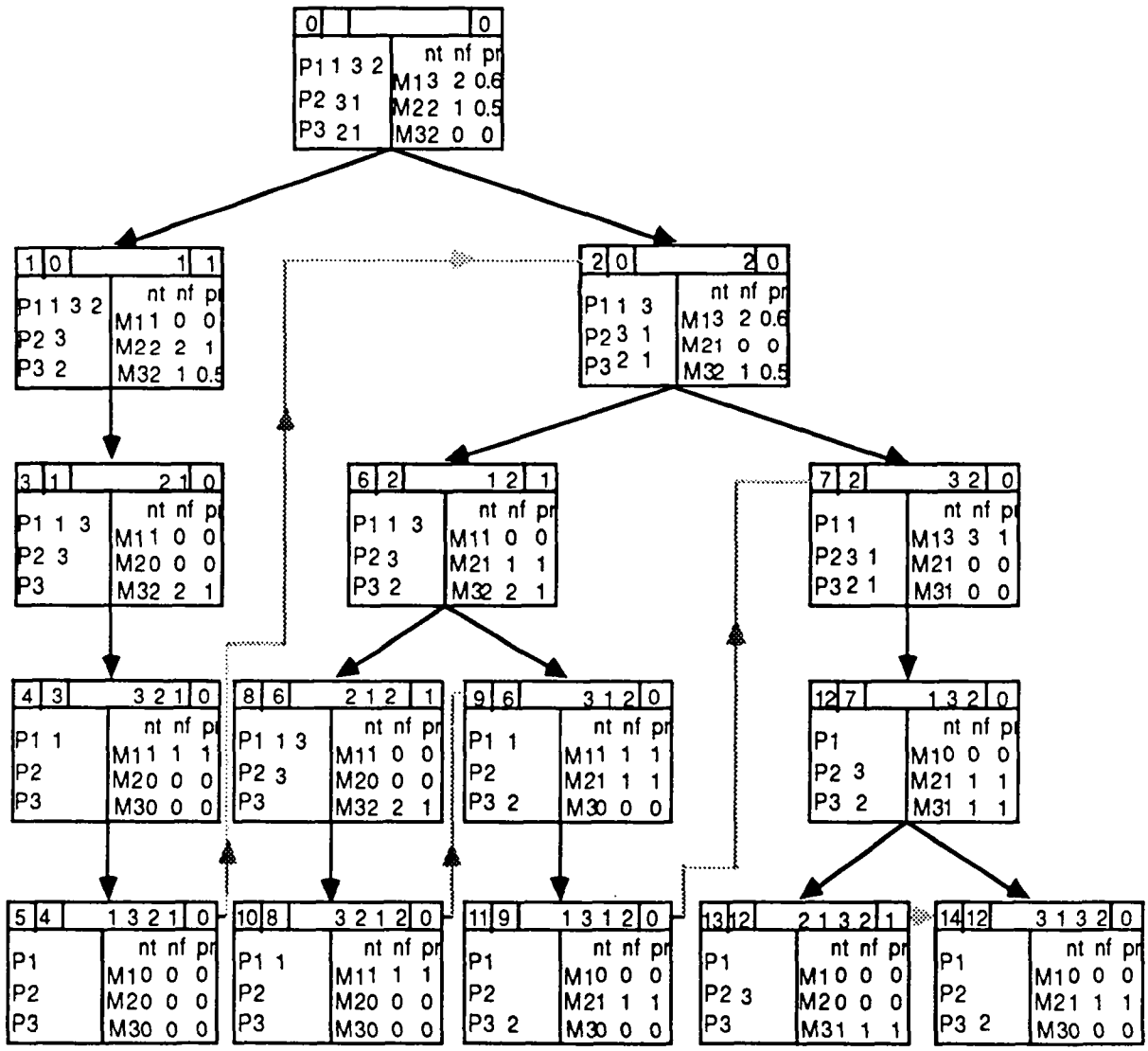


fig. 6 Arborescence de l'exemple illustratif

La S. G. M. sera la première S.G trouvée : **1 3 2 1**. On constate que, dans le cas de la figure 6, mis à part le premier nœud qui est stérile (sa séquence est globale) et qui se trouve au 4ème niveau, tous les autres ont des séquences non globales. Dans ce cas-là, l'heuristique aurait suivi la première branche à gauche, et aurait abouti directement à la seule S.G.M. qui existe en un temps nettement plus petit.

3.4. Evaluation de l'heuristique :

Passons maintenant à l'étape finale, dans laquelle nous allons procéder à l'évaluation de notre heuristique .

Pour cela nous avons traité, avec les deux méthodes, une série d'exemples bien répartis dont les paramètres sont : le nombre de machines, le nombre de gammes, le nombre de machines par gamme (pouvant être différent d'une gamme à l'autre dans un exemple donné) et le taux de répétition de machines identiques dans les gammes. Nous avons regroupé tout cela dans le tableau comparatif ci-dessous :

n°	Nbre de m/c	Nbre de g/m	Solution exacte			Heuristique		
			Nbre d'éléments	Temps d'exécution		Nbre d'éléments	Temps d'exécution	
				SUN 3/60	SUN SPARC		SUN 3/60	SUN SPARC
1	3	3	5	1	1	5	1	1
2	5	4	6	<1	<1	6	<1	<1
3	5	5	11	1	<1	11	1	<1
4	5	6	11	1	<1	12	1	<1
5	7	6	9	<1	<1	10	<1	<1
6	10	10	27	339	19	27	2	<1
7	10	18	48	3	<1	48	2	<1
8	20	30	.	-	-	105	7	5

* Les temps sont des temps CPU ((<1) = moins d'une seconde CPU).

* m/c = machine et g/m = gamme .

N.B. : Les exemples sont traités sur deux machines différentes: SUN 3/60 et SUN SPARC .

Tab. 1 Tableau comparatif

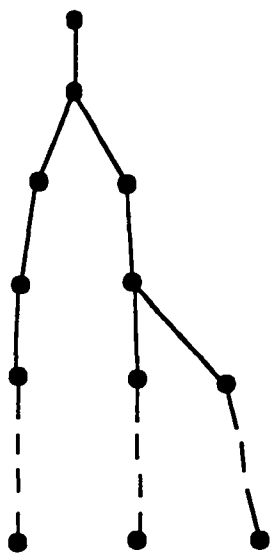
Interprétation :

Le nombre de situations possibles dépend étroitement du nombre de types d'arbres possibles. Ces derniers se caracterisent par leur largeur et leur profondeur. Dans notre cas, la profondeur de l'arbre est liée à la longueur des routages, du fait que plus il y a de machines dans les routages, plus le nombre d'éléments de la S.G.M. est grand. Par contre, la largeur est fonction du nombre de routages dans le système. Cela est dû au fait que plus il y a de routages, plus le nombre de transitions possibles à partir d'un nœud donnée est grand, d'où augmentation de la largeur de l'arbre. Nous distinguons donc trois types d'arbres :

Type a. Mince et profond (voir fig. 7.A) :

Cette situation est observée lorsque l'on a un petit nombre de routages, et que ces routages sont assez longs.

Dans ce type, la prospection est assez lente, étant donné que le passage d'un père à un fils prend beaucoup plus de temps que l'inverse, puisque dans le premier cas on doit créer le nœud fils et lui affecter tous ses constituants, alors que dans le deuxième on ne fait que remonter vers le nœud père connaissant son adresse. Malgré la minceur de l'arbre, sa longueur fera que la méthode exacte prendra plus de temps que l'heuristique. L'exemple 7 (voir tab. 1) est de ce type là.

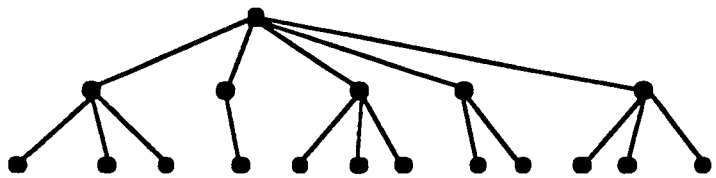


_ A _

Type b. Large et non profond (voir fig. 7.B) :

Ce cas survient souvent lorsque le nombre de routages est important, et lorsque ces routages sont courts .

Dans ce cas, la prospection sera assez rapide, puisque le travail se fait plutôt en largeur, ce qui ne nécessite peu de temps. Les temps de traitement par l'heuristique et par la méthode exacte seront très proches. les exemples n^{os} 1, 2, 3 ,4 et 5 du tableau 1sont de ce type.

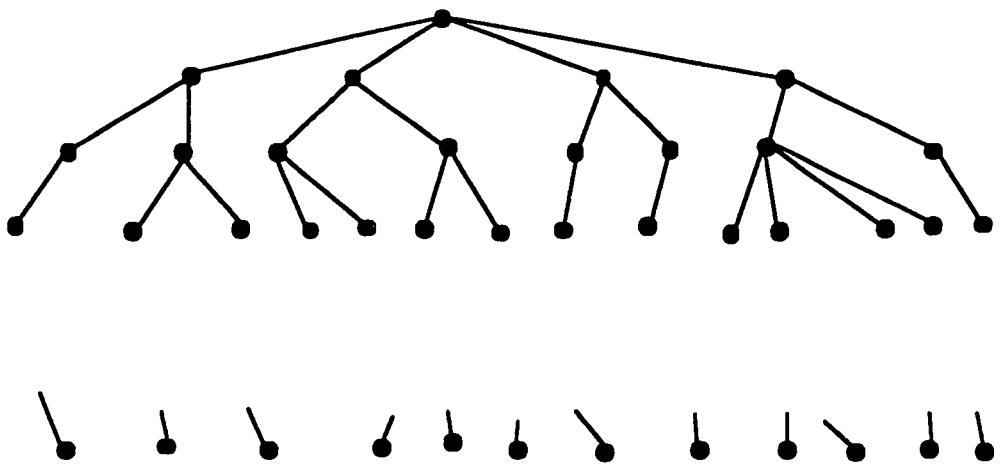


_ B _

Type c. Large et profond (voir fig. 7.C) :

Pour les mêmes raisons que précédemment, ce cas apparaît lorsque le nombre de routages et la longueur de ces derniers sont importants .

Alors l'heuristique est nettement plus rapide que la méthode exacte. Cette dernière est même parfois impossible à utiliser vus la capacité mémoire et le temps de traitement qu'elle nécessite. L'exemple 8 du tableau 1 est de ce type.



_ C _

On ne parlera pas du cas où l'arbre n'est ni large ni profond, puisque l'heuristique et la méthode exacte donneront presque toujours les mêmes

résultats, et surtout avec des délais identiques.

4. CONCLUSION :

Nous avons donc vu que la détermination de la **Séquence Globale Minimale** est un problème combinatoire. Nous avons présenter une heuristique simple qui, pour les petits et moyens systèmes, donne des résultats très proches (souvent identiques) de ceux de la méthode exacte présentée ci-dessus, et cela en un temps trés court.

Quelques améliorations peuvent être apportées a ce travail en ce qui conserne la solution exacte. Nous pouvons utiliser la méthode de Séparation Evaluation "Branch and Bound". Seulement, il faut trouver une borne inférieure permettant une évaluation efficace après séparation, en vu de réduire la mémoire utile et le temps de traitement nécessaire.

BIBLIOGRAPHIE

[1] D. MAIER '1978' " The Complexity of Some Problems on Subsequences and Supersequences " .
Journal of the Association for Computing Machinery
Vol. 25 n° 2 pp. 322 - 336 .

[2] V.G.TIMKOVSKII '1990' " Complexity of Commun Subsequence and Supersequence Problems and Related Problems ."
CYBERNETICS A translation of kibernetika
Vol. 25 n° 5 pp. 565 - 580 .

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique.

ISSN 0249-6399