Abstract

Since the flowshop scheduling problem has found to be an NP-complete problem, the development of heuristic algorithms that give better solutions become necessary. In this paper we discuss how to resolve the flowshop problem using heuristic methodes, we were based on the study of [RAJ 91] in which three heuristic algorithms with the objective of minimizing total flowtime are proposed. In our paper we have detailed the concept introduced and given proofs of the principle formulas used in the algorithms presented. Our paper can be viewed as a contribution to understand better how to develop heuristic solution choosing the best minimizing criteria.

Key-words: The flowshop problem, Heuristics, Job scheduling, Total flowtime.

Une étude d'approches heuristiques pour l'ordonnancement des jobs dans le "flowshop"*

Tayeb LEMLOUMA †

† *INRIA Rhône Alpes,*
Zirst - 655 avenue de l'Europe - Montbonnot - 38334 Saint Ismier Cedex - France
Tel +33 4 76 61 52 81
E-Mail : Tayeb.Lemlouma@inrialpes.fr

Résumé

Puisque l'ordonnancement des jobs dans les problèmes de flowshop représente un problème NP-complet, le développement des algorithmes heuristiques qui donnent des solutions approchées devient nécessaire. Cet article discute la résolution du problème flowshop en utilisant les méthodes heuristiques, il est axé sur l'analyse de l'article [RAJ 91] de Rajendran et Chaudhuri dans lequel trois nouveaux algorithmes sont proposés dans le but de fournir des solutions proches des solutions optimales et meilleures de celles trouvées par les méthodes qui existent déjà. Dans notre étude, nous détaillons les concepts introduits et nous donnons quelques preuves concernants les principales formules mathématiques utilisées dans les algorithmes présentés. Notre article peut être vue comme une contribution pour comprendre les mécanismes de base du développement des solutions heuristiques.

Mots clés

Le problème de flowshop, Heuristiques, Ordonnancement de job, Flowtime total.

1 Introduction

Généralement, le mot "*heuristique*" est utilisé pour décrire un algorithme ou une procédure qui se base sur des expériences pratiques. Ce mot a été utilisé dans le passé pour dénoter la programmation heuristique [SIM 58, SIM 61] qui s'appliquait dans les problèmes de gestions [KAR 64, KUE 63, SIM 60, TON 61]. Dans [KUE 63] on trouve une interprétation détaillée du concept d'heuristique et d'algorithmique.

Les techniques heuristiques sont utilisées généralement dans le but de donner des solutions approchées et raisonnables (du point de vue temps) pour des problèmes

dont la résolution avec les méthodes conventionnelles nécessite un temps énorme voire infini.

Cet article est axé principalement sur l'étude de l'article [RAJ 91] de Rajendran et Chaudhuri, l'article introduit une approche efficace d'ordonnancement des jobs pour le problème du flowshop et la compare avec les méthodes qui existe déjà. Nous allons détailler les concepts introduits et nous donnons quelques preuves concernants les principales formules mathématiques utilisées dans les algorithmes présentés.

2 Les heuristiques et le problème du "flowshop"

De nombreuses contributions ont été consacrées à la conception d'algorithmes d'optimisation et d'heuristiques pour le problème du flowshop. Etant donné que l'ordonnancement des jobs dans un problème de flowshop représente un problème NP-complet, le développement des algorithmes heuristiques qui donnent des solutions approchées devient nécessaire. L'article [RAJ 91] de Rajendran et Chaudhuri s'inscrit dans cette direction. Dans cet article trois algorithmes heuristiques sont proposés afin de garantir des solutions consistantes qui soient proches des solutions optimales et meilleures de celles trouvées par les méthodes qui existent déjà.

Le critère de minimisation, appelé aussi la fonction objective, du *flowtime*, ou du temps total de l'ordonnancement des jobs dans un problème de flowshop, est le cœur de tout algorithme d'heuristique. Ce critère est utilisé pour réduire, de la meilleure façon, les coûts du processus d'ordonnancement [GUP 71, PAN 73]. L'article étudié introduit trois critères de minimisation, les algorithmes qui les utilisent sont simples et efficaces, on les a comparé avec les heuristiques de Miyazaki, Nishiyama et Hashimoto, l'algorithme "MINIT" de Gupta et l'heuristique de Ho et Chang, qui représentent d'après une étude de littérature des problèmes d'ordonnancement, les heuristiques les plus connus et les plus efficaces qui existent.

3 Terminologie et principes générales

Nous adoptons dans tout ce qui suit la terminologie suivante :

n :	le nombre de jobs à ordonner.
m :	le nombre d'étages du problème flowshop.
σ :	la séquence de jobs ordonnés.
t_{ij} :	le temps de traitement du job i à l'étage j.
[i] :	le job d'ordre i dans la séquence ordonnée.
π :	l'ensemble des jobs non encore ordonnés.
b :	le dernier job de l'ensemble σ.
a, c :	jobs de l'ensemble π.
$q(\sigma, j)$:	le temps de terminaison de l'ordre partiel σa à l'étage j quand a est rajouté à σ.

W_{ba} : la somme des durées d'attentes du job a quand il suit le job b dans la séquence d'ordre partielle σ.

Soit σa une séquence d'ordre partielle, i.e. un sous ensemble (de π) de jobs ordonnés. Le temps de terminaison de la séquence σa qui correspond à un étage j, q (σa, j) peut être calculé par la formule suivante :

$$q(\sigma a, j) = \max[q(\sigma, j), q(\sigma a, j-1)] + t_{aj}. \qquad (I)$$

La quantité q(σa, j) peut être calculée d'une manière récursive en prenant comme conditions initiales : (i) $q(\sigma a, 0) = 0$.

(ii) $q(\phi, j) = 0 \ \forall \ j \in \{1,2,\ldots,m\}$.

La condition (i) signifie que le temps de terminaison d'une séquence σa de jobs en ne passant par aucun étage est nulle. La deuxième condition signifie que le temps de terminaison d'une séquence vide de jobs est nulle pour n'importe quel étage. Notons que q(σ, j) et q(σa, j-1) sont incomparables, c'est pour cela dans (I) on prend le maximum des deux quantités. La figure suivante illustre les deux cas possibles, i.e. le cas $q(\sigma, j) \leq q(\sigma a, j-1)$ et le cas $q(\sigma, j) > q(\sigma a, j-1)$. Les deux situations dépendent du fait que le temps de terminaison du dernier job de la séquence σ (x dans la figure 1) qui correspond à l'étage j, soit supérieur ou pas au temps de terminaison de *a* dans l'étage j-1.

Calculons maintenant q(σa, 1), sachant que σ est une séquence quelconque de jobs ordonnés. D'après la formule (I), nous avons :

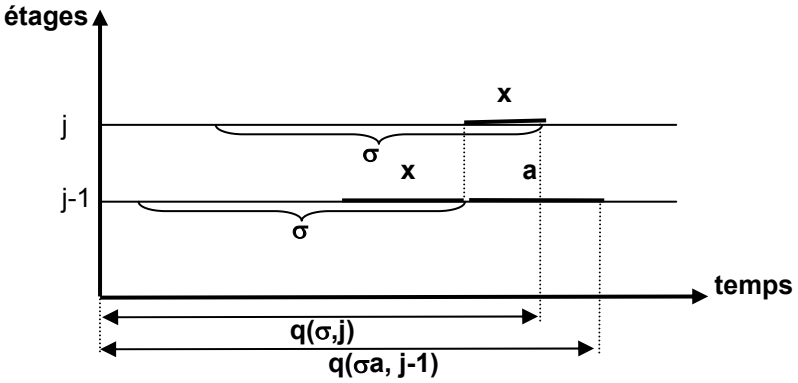
$$q(\sigma a, j) = \max[q(\sigma, 1), q(\sigma a, 0)] + t_{a1}.$$
$$= q(\sigma, 1) + t_{a1}. \ (*) \ (\text{car } q(\sigma a, 0) = 0 \text{ et } q(\sigma, 1) \text{ est positif})$$

De (*), et sachant que $q(\phi, 1) = 0$ (d'après les conditions initiales) on peut conclure que :

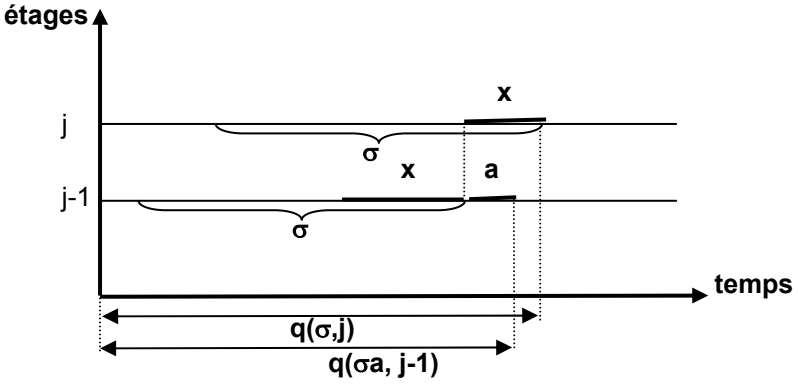
$$\forall \sigma \ \ q(\sigma, 1) = \sum_{x \in \sigma} t_{x1}. \qquad (R1)$$

De la formule (I), on peut conclure que le flowtime totale, $F_{\sigma a}$, des jobs d'une séquence σa est égal au flowtime des jobs de la séquence σ, plus le temps de terminaison du job a correspondant au dernier étage d'exécution(qui peut se calculer en utilisant (I)). Formellement on a :

$$F_{\sigma a} = F_{\sigma} + q(\sigma a, m). \qquad (II)$$



(a) Cas où q(σ, j) ≤ q(σa, j-1).



(b) Cas où q(σ, j) >q(σa, j-1).

Figure 1: Les deux cas concernant la terminaison de (σa, j-1) et (σ, j).

4 Principes des algorithmes heuristiques

Si un job *a* ne fait aucun retard durant l'exécution des différents étages, son temps de terminaison correspondant au dernier étage m, (sachant que *a* suit le job *b* dans la séquence σ) est :

$$q(\sigma a, m) = q(\sigma, 1) + \sum_{j=1}^{m} t_{aj}. \qquad \text{(III)}$$

La figure 2 montre cela. Si le job *a* fait des attentes durant son exécution, le temps de terminaison devient :

$$q(\sigma a, m) = q(\sigma, 1) + W_{ba} + \sum_{j=1}^{m} t_{aj}. \qquad \text{(IV)}$$

La quantité W_{ba} représente l'attente que fait le job *a* durant son exécution s'il suit le job *b* dans l'ordre. Cette quantité se calcule comme suit :

$$W_{ba} = \sum_{j=2}^{m} \max[q(\sigma, j) - q(\sigma a, j-1), 0] \qquad \text{(V)}$$

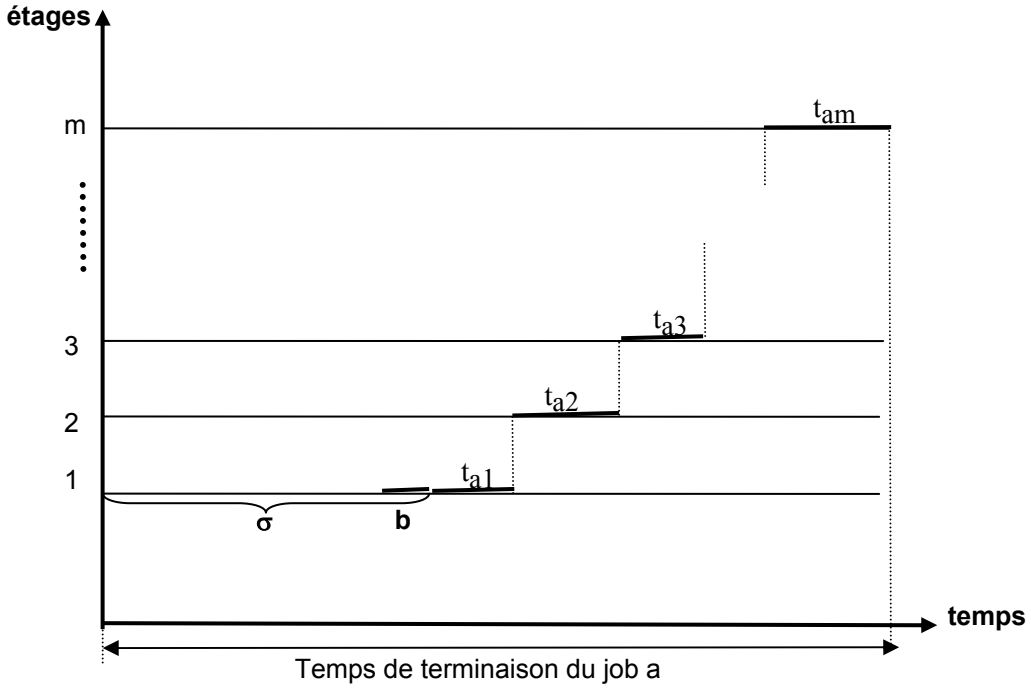
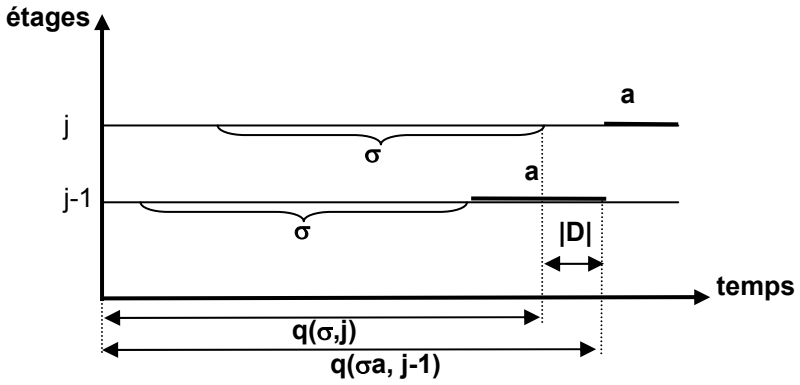
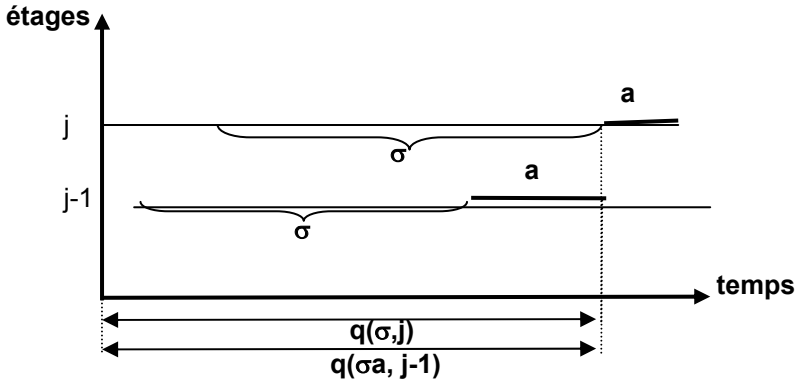


Figure 2: Le temps de terminaison d'un job.

Cette formule est basée sur le calcul du maximum entre q(σ, j) - q(σa, j-1) et 0, car la quantité Q = q(σ, j) - q(σa, j-1) peut être négative, nulle ou positive. Dans le cas où il n'y aurait pas de temps d'attente, Q sera négative ou nulle comme le montre la figure suivante :



(a) Cas où D < 0.



(b) Cas où D = 0.

Figure 3: L'absence d'attente du job *a* durant son exécution de l'étage j.

Dans le cas où D serait positive, cela signifie que le job *a* attend une certaine durée avant qu'il entame l'étage j. Cette durée est exactement égale à D.

En utilisant le résultat (R1) que nous avons démontré, la formule (IV) peut être réécrite de la façon suivante :

$$q(\sigma a, m) = \sum_{i=1}^{n'} t_{[i]1} + W_{ba} + \sum_{j=1}^{m} t_{aj}. \qquad \text{(VI)}$$

Rappelons que [*i*] représente le job d'ordre *i* dans la séquence σ, et *n*' est le nombre de job dans cette séquence.

Calculons maintenant le flowtime total quand tous les jobs du système deviennent ordonnés. On a d'après la formule (II) :

$\forall\sigma \quad F_{\sigma a} = F_{\sigma} + q(\sigma a, m)$.
On peut la réécrire sous la forme suivante :

$$\forall\sigma \quad F_{\sigma} = q([1], m) + q([1][2], m) + \ldots + q([1]\ldots[n-1], m) + q([1]..[n], m).$$
$$= q([1], m) + \sum_{i=2}^{n} q([1]..[i], m).$$

Avec *n* est la taille de la séquence σ, et [1]..[k] représente la sous séquence ordonnée des k premiers jobs de σ.
D'un autre coté on a : $q([1], m) = q(\phi[1], m)$. Donc d'après (VI) on a :

$q([1], m) = \sum_{j=1}^{m} t_{[1]j}$. Car le job [1] est le premier dans la séquence ordonnée et donc il n'existe aucun job (*b*) qui le précède, ce qui justifie le fait que la quantité $\sum_{i=1}^{0} t_{[i]1} + W_{b[1]}$ soit nulle.

Par conséquent on a :

$$\forall \sigma \quad F_\sigma = \sum_{j=1}^{m} t_{[1]j} + \sum_{i=2}^{n} q([1]..[i], m)$$

$$= \sum_{j=1}^{m} t_{[1]j} + \sum_{i=2}^{n} \left(\sum_{j=1}^{i-1} t_{[j]1} + W_{[i-1][i]} + \sum_{j=1}^{m} t_{[i]j} \right)$$

$$= \sum_{j=1}^{m} t_{[1]j} + \sum_{i=2}^{n} \sum_{j=1}^{i-1} t_{[j]1} + \sum_{i=2}^{n} W_{[i-1][i]} + \sum_{i=2}^{n} \sum_{j=1}^{m} t_{[i]j}$$

$$= \sum_{i=2}^{n} \sum_{j=1}^{i-1} t_{[j]1} + \sum_{i=2}^{n} W_{[i-1][i]} + \sum_{i=1}^{n} \sum_{j=1}^{m} t_{[i]j}$$

$$= \sum_{i=1}^{n-1} (n-i)\, t_{[i]1} + \sum_{i=2}^{n} W_{[i-1][i]} + \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} \qquad \text{(VII)}$$

Notons que la quantité F_σ cumule les durées d'attentes ($W_{[i-1][i]}$) des jobs [i] durant les différents étages. Afin de minimiser le temps de traitement, il est préférable d'ordonner les jobs les plus courts avant les jobs qui nécessitent plus de temps de traitement [CON 67, SZW 83]. Dans le cas où les jobs qui consomment un grand temps de traitement sont ordonnés au début, les jobs qui suivent dans l'ordre auront de grandes durées d'attente ce qui augmente le flowtime totale.

Rajendran et Chaudhuri proposent dans [RAJ 91] trois critères heuristiques :

1) $\sum_{j=2}^{m} \max[q(\sigma a, j-1) - q(\sigma, j), 0]$.

2) $\sum_{j=2}^{m} abs[q(\sigma a, j-1) - q(\sigma, j)]$.

3) $\sum_{j=2}^{m} abs[q(\sigma a, j-1) - q(\sigma, j)] + \sum_{j=1}^{m} q(\sigma a, j)$.

Le premier critère représente la somme des *durées de repos*. Une durée de repos (idle time) pour un job *a* et un étage j, est égal au maximum de $q(\sigma a, j-1) - q(\sigma, j)$ et zéro sachant que σ est la séquence des jobs déjà ordonnés. La durée de repos représente la durée d'inoccupation d'un étage *j* avant qu'un job *a* entamera cet étage. Le deuxième critère considère la somme des durées de repos et d'attentes. Le troisième critère considère, en plus des facteurs utilisés dans le deuxième critère, le temps de terminaison de l'ordre partiel résultant à chaque étage.

5 Algorithmes heuristiques

I) Premier algorithme : Le premier algorithme présenté dans [RAJ 91] consiste à exécuter les étapes suivantes :

Etape 1 : Initialiser σ (la séquence des jobs ordonnés) à vide et *n*' (le cardinal de

σ) à zéro.
Etape 2 : Incrémenter *n*' de 1.
Etape 3 : Calculer q(σc, j) de j = 1 à *n* et pour tout job c appartient à π (l'ensemble des jobs non encore ordonnés).
Etape 4 : On utilise dans cette étape le premier critère heuristique de la manière Suivante :
Une séquence σa est préférée que σc si :

$$\sum_{j=2}^{m} \max[q(\sigma a, j-1) - q(\sigma, j), 0] \leq \sum_{j=2}^{m} \max[q(\sigma c, j-1) - q(\sigma, j), 0]. \qquad \text{(VIII)}$$

A la fin de cette étape un job *a* est choisi, par conséquent la séquence σa est la meilleure séquence qui vérifie le premier critère proposé.
Etape 5 : Rajouter le job *a* à la séquence σ et appeler la séquence résultante σ, mettre à jour les durées de terminaison pour chaque étage et éliminer le job *a* de l'ensemble des jobs non encore ordonnés.
Etape 6 : Si *n*' = *n*-1 alors aller à l'étape suivante, sinon retourner à l'étape 2.
Etape 7 : Rajouter le dernier job qui reste à la séquence σ et calculer sa durée de terminaison pour chaque étage. Calculer le flowtime total de l'ordre final.

II) Deuxième algorithme : Le deuxième algorithme est exactement le même que premier sauf que là, l'étape 4 utilise le deuxième critère heuristique proposé.

III) Troisième algorithme : Les étapes de cet algorithme sont les mêmes que le celles du premier algorithme à l'exception de l'étape 4 qui utilise le troisième critère heuristique proposé au lieu du premier.

6 Illustration numérique

Afin de comprendre l'approche appliquée dans les algorithmes proposés nous allons reprendre l'exemple introduit dans [RAJ 91], considérons un problème de flowshop à quatre étages avec cinq jobs à ordonner. Les temps de traitement des différents jobs sont donnés comme suit :

		Étage			
		1	**2**	**3**	**4**
Job	**1**	4	18	1	23
	2	4	20	6	12
	3	19	9	11	6
	4	18	6	23	13
	5	3	23	16	15

Tableau 1 : Les temps d'exécution des jobs.

La résolution de cet exemple est faite en appliquant le troisième algorithme proposé, le tableau suivant montre les étapes du processus d'ordonnancement :

σ	σc		q(σc, j) pour j = 1	2	3	4	La somme des durées de terminaison du job c	Durée d'attente du job c	Durée de repos des étages	Somme totale	Le job *a* choisi
ϕ		{1}	4	22	23	46	95	0	49	144	1
		{2}	4	24	30	42	100	0	58	158	
		{3}	19	28	39	45	131	0	86	217	
		{4}	18	24	47	60	149	0	89	238	
		{5}	3	26	42	57	128	0	71	199	
	{1}		4	22	23	46					
	{1}	{12}	8	42	48	60	158	14	21	193	
		{13}	23	32	43	52	150	3	10	163	3
		{14}	22	28	51	64	165	0	10	175	
		{15}	7	45	61	76	189	15	37	241	
	{13}		23	32	43	52					
	{13}	{132}	27	52	58	70	207	5	15	227	2
		{134}	41	47	70	83	241	0	31	272	
		{135}	26	55	71	86	238	6	31	275	
	{132}		27	52	58	70					
	{132}	{1324}	45	58	81	94	278	7	11	296	4
		{1325}	30	75	91	106	302	22	38	362	
	{1324}		45	58	81	94					
	{13245}		48	81	97	112					

Tableau 2 : Les étapes de calcul du troisième algorithme.

Comme le montre le tableau 2, la séquence d'ordre final qui correspond au problème précédent est σ = {1,3,2,4,5}. Le flowtime totale de cet ordre est égal à :

F_σ = q({1},4) + q({13},4) + q({132},4) + q({1324},4) + q({13245},4) = 374. La solution optimale du problème est la séquence ordonnée {23145} dont le flowtime est égal à 371. Le pourcentage d'erreur de la solution calculée est donc (374-371)*100/371 = 0.8086 %.

7 Résultats expérimentaux

Les trois algorithmes d'heuristique proposés et les algorithmes déjà existants (i.e. l'heuristiques de Miyazaki, Nishiyama et Hashimoto, l'algorithme "MINIT" de Gupta et l'heuristique de Ho et Chang) ont été programmés en FORTRAN sous le système 7.580 E SIMENS Mainframe. Les expérimentations ont été faites en deux phases :

Dans la première phase, 670 problèmes de 5, 6, 7, 8, 9 et 10 jobs ont été générés et résolus en utilisant les algorithmes heuristiques, et d'une manière optimale en appliquant l'algorithme de Bansal [BAN 77]. Les durées de traitement des jobs ont été générées aléatoirement [CAM 70, DAN 77] dans l'intervalle [1..99]. Les procédures de test et de comparaison sont appliquées en se basant sur la déviation de la solution calculée par rapport à la solution optimale. Le pourcentage de déviation est calculé en utilisant la formule suivante :

$$\text{Pourcentage de déviation} = \frac{(\text{solution heuristique} - \text{solution optimale})*100}{\text{solution optimale}}$$

Nous proposons dans ce qui suit, un schéma qui compare les pourcentages de déviations moyennes des différents algorithmes proposés par rapport aux algorithmes qui existent déjà.

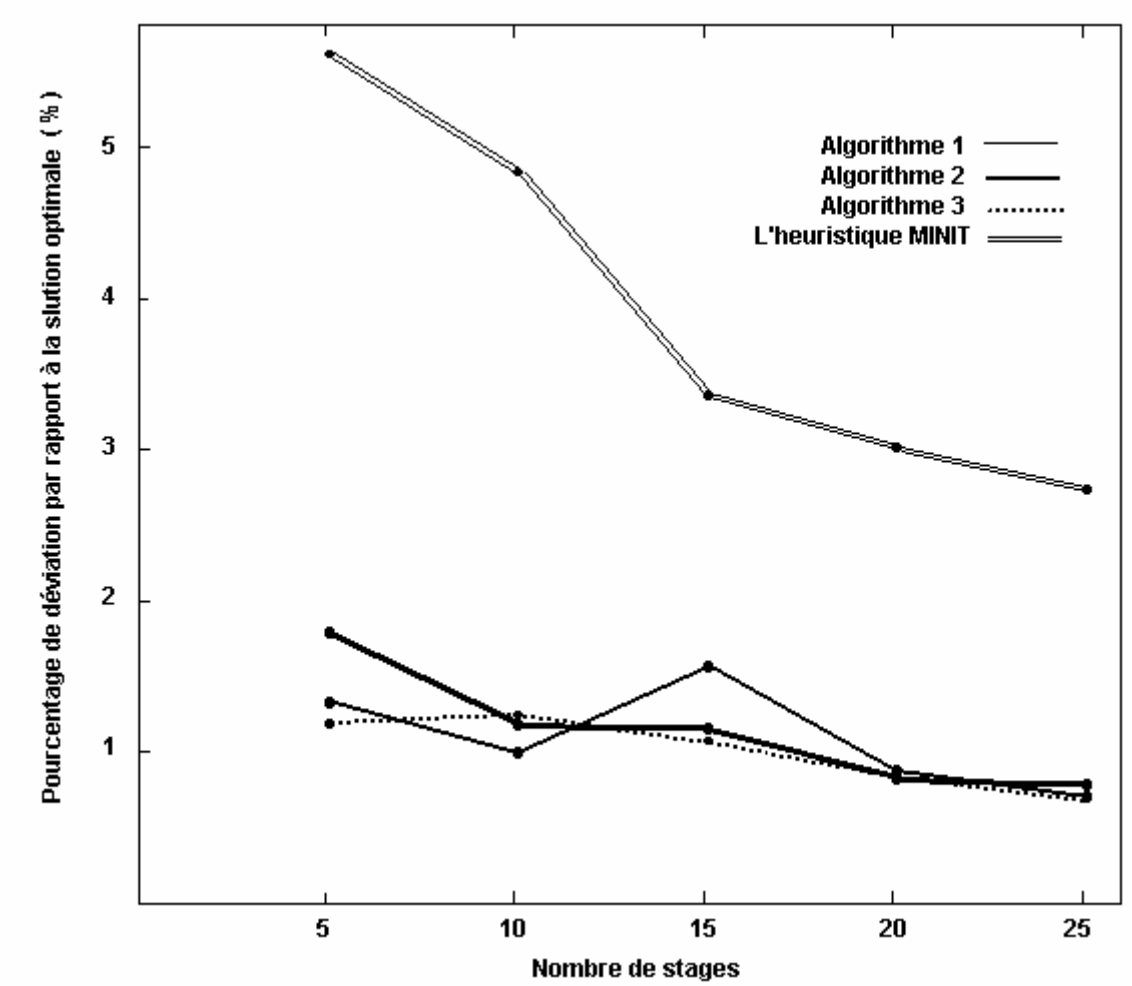


Figure 4: Résultats du calcul la déviation de la solution optimale par rapport aux solutions obtenues (# de job = 7, # de problèmes = 30).

La deuxième phase d'expérimentations faites par Rajendran et Chaudhuri, considère 900 problèmes de flowshop ayant 12, 18, 24, 30, 40 et 50 jobs. Dans le but d'éviter la comparaison entre les résultats heuristiques, les solutions obtenues en appliquant les différents algorithmes, ont été comparés avec une borne inférieure (un minorant) qui s'obtient de la manière suivante :
On considère l'équation (VII) qui donne le flowtime totale des jobs :

$$F_\sigma = \sum_{i=1}^{n-1} (n-i)\, t_{[i]1} + \sum_{i=2}^{n} W_{[i-1][i]} + \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij}$$

Le premier terme de cette expression peut être minimisé en ordonnant les différents jobs dans le premier étage, selon leurs temps d'exécution. Examinons maintenant le deuxième terme, on peut voir que l'expression :

$$\sum_{j=2}^{m} \max[q(b, j-1) - q(ba, j-1), 0].$$

Représente un minorant pour la somme W_{ba} des durées d'attentes du job *a*, quand il est rajouté à n'importe quel ordre partiel σ qui se termine par le job *b*. Formellement on a :

$$\forall \sigma \sum_{j=2}^{m} \max[q(\sigma, j) - q(\sigma a, j-1), 0] \geq \sum_{j=2}^{m} \max[q(b, j) - q(ba, j-1), 0], \text{ tel que } \sigma \text{ se termine par b.}$$

ou :

$$\forall \sigma \quad W_{ba} \geq \sum_{j=2}^{m} \max[q(b, j) - q(ba, j-1), 0].$$

$$\text{Posons } W'_a = \min \{ \sum_{j=2}^{m} \max[q(i, j) - q(ia, j-1), 0], i = 1,2,\ldots,n \text{ et } i \neq a\} \qquad \text{(IX)}$$

Cette quantité représente la somme minimale des durées d'attente pour le job a. Pour construire la borne inférieure, on suppose que tous les jobs de l'ordre final ont des durées d'attente minimales. L'expression suivante représente donc la borne inférieure des sommes des durées d'attente, i.e. de $\sum_{i=2}^{n} W_{[i-1][i]}$:

$$\sum_{i=1}^{n} W'_i - \max[W'_i, i = 1, 2, \ldots, n]$$

Notons que le terme $\max[W'_i, i = 1, 2, \ldots, n]$ est enlevé, car le premier job ordonné, a une durée d'attente nulle, donc pour que l'expression précédente soit minimale on a enlevé la valeur maximale des durées d'attente calculées. Le troisième terme de (VII) est constant.

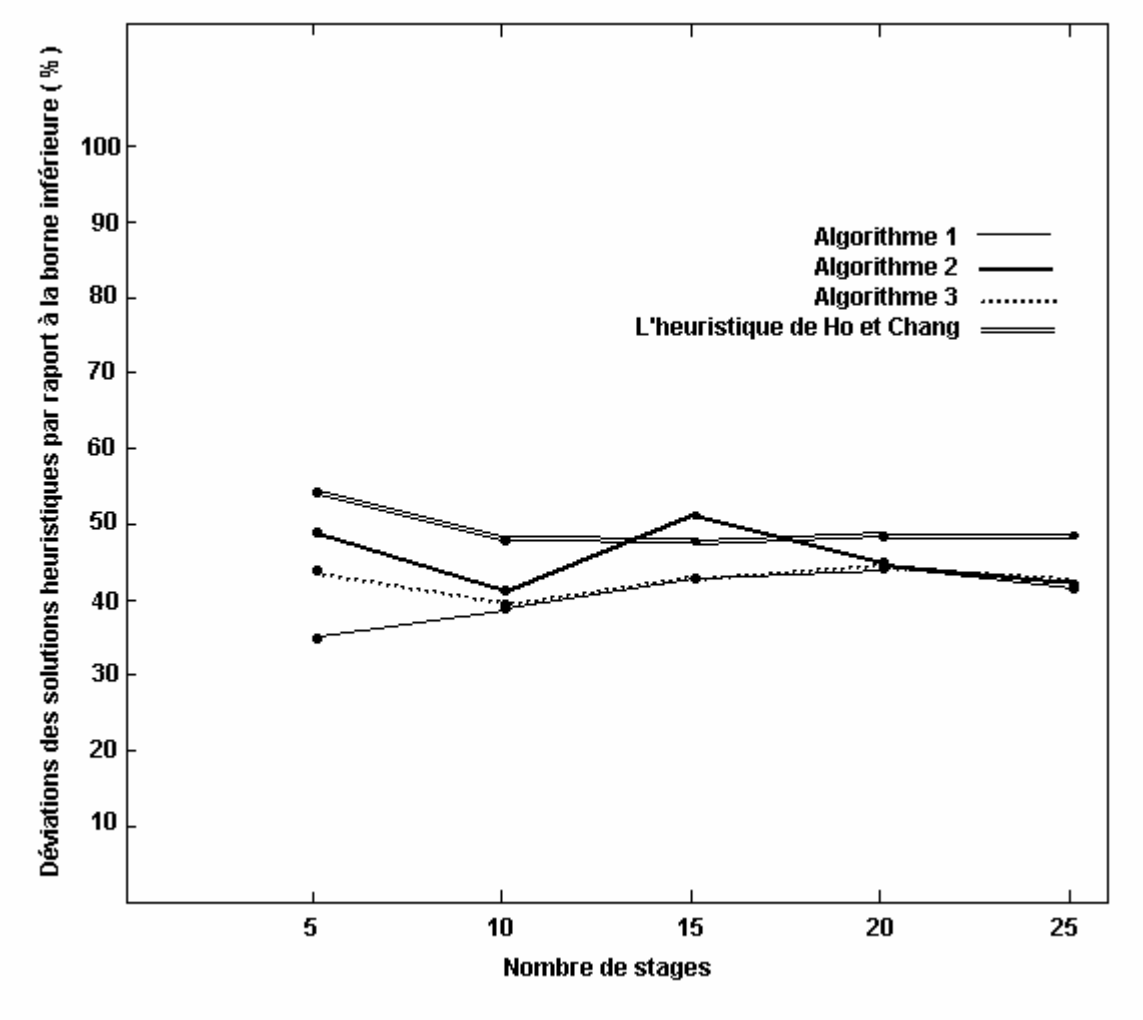
Une évaluation de la déviation de la solution optimale par rapport au minorant proposé, montre pour 520 problèmes de 6, 7, 8, 9 et 10 jobs, l'efficacité du minorant proposé, la valeur maximale des déviations moyennes calculées ne dépasse pas la valeur 16,6 %.

Dans cette phase d'expérimentation, les solutions heuristiques sont évaluées en calculant la déviation par rapport à la borne inférieure de la manière suivante :

$$\text{Pourcentage de déviation} = \frac{(\text{solution heuristique - borne inférieure})*100}{\text{borne inférieure}}$$

Le nombre de problèmes évalués est égal à 900 avec 12, 18, 24, 30, 40 et 50 jobs et un nombre d'étages qui varie entre 5 et 25. Nous représentons dans la figure suivante, les résultats d'expérimentations du calcul de la déviation des solutions heuristiques par rapport à la borne inférieure proposée.

Figure 5: Résultats du calcul la déviation des solutions heuristiques par rapport à la



borne inférieure (# de job = 40, # de problèmes = 30)

Dans le but de tester les algorithmes proposés par rapport au temps d'exécution, Rajendran et Chaudhuri donnent une évaluation - sous le même environnement - de plusieurs problèmes dont le nombre de jobs varie entre 7 et 50. Cette évaluation montre clairement la rapidité des algorithmes proposés par rapport à ceux qui existent déjà.

Les résultats d'évaluation des trois algorithmes introduits dans [RAJ 91], montrent que les heuristiques proposées donnent des solutions proches des solutions optimales, et qui sont nettement meilleures par rapport à celles obtenues en utilisant les procédures qui existent déjà que ce soit de point de vue qualité ou traitement.

8 Conclusion

Rajendran et Chaudhuri ont proposé dans [RAJ 91] trois algorithmes d'ordonnancement du problème flowshop, dont l'objectif est de minimiser le flowtime totale de l'ordre final des jobs. D'après les résultats d'évaluation, les heuristiques proposées assurent, d'une manière rapide, de bonnes solutions qui sont nettement meilleures par rapport à celles calculées par les heuristiques qui existent déjà. En effet, les trois algorithmes étudiés fournissent des résultats d'une manière simple et efficace et minimisent le flowtime total en minimisant les durées de repos et d'attente des jobs ce qui réduit les coûts du processus d'ordonnancement [GUP 71].Ce résultat encourage l'utilisation pratique des algorithmes proposés dans la résolution du problème flowshop.

Références

[BAN 77] Bansal S.P.
"Minimizing the sum of completion times of n-jobs over M-machines in a flowshop - A branch and bound approach ". AIEE Transactions 9, pp 306-311, 1977.

[CAM 70] Campbell H.G., Dudek R.A., and Smith M.L.
"A heuristic algorithm for the n-job, M-machine sequencing problem". Management Science 16 B630-B637, 1970.

[CON 67 Conway R.W., Maxwell W.L. and Miller L.W.
"Theory of Scheduling". Addision-Wesley, Reading, MA, 1967.

[DAN 77] Dannenbring D.G.
"An evaluation of flowshop sequencing heuristics". Management Science 23, pp 1174-1182, 1977.

[GUP 71] Gupta J.N.D., and Dudek R.A.
"Optimality criteria for flowshop schedule". AIIE Transactions 3, pp 199-205, 1971.

[KAR 64] Karg R.L. and Thompson G.L.
"A Heuristic Approach to Solving Traveling Salesman Problems". Management Science, Vol. 10, No. 2, pp 255-248, January 1964.

[KUE 63] Kuehn A.A. and Hamburger M.J.
"A Heuristic Program for Locating Warehouses". Management Science, Vol. 9, No 4, pp 643-666, July 1963.

[PAN 73] Panwalkar, S.S., Dudek R.A. and Smith M.L.
"Sequencing research and the industrial scheduling problem".

Proceeding of the Symposium on Theory of Scheduling and its Applications (May 1972), Springer-Verlag, Berlin, 1973.

[RAJ 91] Rajendran C. and Chaudhuri D.
"An efficient heuristic approach to the scheduling of jobs in flowshop". European Journal of Operational Reseach 61, pp 318-325, 1991.

[SIM 58] Simon H.A. and Newell A.
"Heuristic Problem Solving : The next Advance in Operations Research". Operations Research, Vol. 6, No. 1, pp 1-10, January-February 1958.

[SIM 60] Simon H.A.
"The new Science of Management Decision". Hareper & Bros., 1960

[SIM 61] Simon H.A. and Newell A.
"Computer Simulation of Human Thinking and Problem Solving". Computers and Automation, Vol. 10, No 4, pp 18-26, April 1961.

[SZW 83] Szwarc, W.
"The flowshop problem with mean completion time". IIE Transactions 15, pp 172-176, 1983.

[TON 61] Tonge, Fred.
"The use of Heuristic Programming in Management Science". Management Science, pp 231-237, April 1961.