RESOLUTION DU PROBLEME D'ORDONNANCEMENT DE TYPE JOB-SHOP GENERALISE PAR DES HEURISTIQUES DYNAMIQUES

Fatima Ghedjati **Jean-Charles Pomerol**

LAFORIA-IBP (URA CNRS1095) Université de Paris VI, case 169, 4 Place Jussieu F-75252 Paris cedex 05
tél. (01)44.27.70.10 tél. (01)44.27.47.21
e-mail : ghedjati@laforia.ibp.fr e-mail : pomerol@laforia.ibp.fr
fax (01)44.27.70.00

RESUME

Le présent article propose une résolution du problème d'ordonnancement d'ateliers de type "job-shop généralisé" par plusieurs méthodes heuristiques statiques et dynamiques originales tenant compte de la charge potentielle des machines au fur et à mesure de la construction de la solution. Nous considérons, d'une part, des machines non identiques (ou non reliées) en parallèle pouvant effectuer les opérations de différentes pièces et, d'autre part, des contraintes de précédence quelconques entre les opérations. L'objectif de l'ordonnancement est la minimisation de la durée totale de l'exécution de toutes les pièces, autrement dit la minimisation du C_{max}. Ce problème est NP-difficile. Des expérimentations ont été effectuées avec divers types de données issues de la littérature ou générées aléatoirement. Notre approche permet de traiter d'une manière satisfaisante des problèmes relativement importants en des temps raisonnables.

Mots-clés : Ordonnancement, Job-shop généralisé, Machines non identiques en parallèle, Gamme linéaire et non linéaire, heuristiques dynamiques.

ABSTRACT

This paper proposes a solution of the generalized job-shop factory scheduling problems by several original of static and dynamic heuristics relying on the potential load of the machines. On the one hand, we consider, unrelated parallel machines which can execute the operations of the different jobs, and, on the other hand, any precedence contraints between the operations is allowed. The objective of the scheduling is to minimize the total duration of all the jobs, i.e. to minimize the C_{max}. This problem is NP-Hard. Experimental results using different types of data, taken from the literature or randomly generated benchmarks are provided. Our approach allows a satifactory resolution of relatively important problems in a raisonable time.

Key words : Scheduling, Generalized job-shop, Unrelated parallel machines, Linear and non-linear process rooting, Dynamical heuristics.

1. Introduction

Les problèmes d'ordonnancement (voir par exemple Carlier et Chrétienne 1988, Lawler et al. 1989) sont présents dans tous les secteurs de l'économie et constituent une fonction importante en gestion de production. Beaucoup de problèmes d'ordonnancement sont NP-difficiles (Garey et Johnson 1979), cela signifie que l'on n'obtient des résultats exacts qu'avec une complexité exponentielle. Deux méthodes sont envisageables pour la résolution de ces problèmes: les méthodes exactes pour une solution optimale exacte lorsque la taille du problème le permet et les méthodes approchées si une solution approchée suffit ou si l'on n'est pas capable de trouver la solution exacte. Les méthodes exactes telles que la programmation dynamique (Bellman 1954), les procédures par séparation et évaluation ("branch and bound") garantissent l'optimalité mais en une durée exponentielle (voir par exemple Roy 1970, Carlier et Pinson 1989). De manière générale, pour les problèmes de taille industrielle, on ne peut pas espérer trouver un ordonnancement optimal en un temps raisonnable (Rinnooy Kan 1976). Ceci a poussé un grand nombre de chercheurs à adopter des méthodes approchées qui ne garantissent pas l'optimalité et sont plus ou moins performantes selon le type de problèmes considéré, mais leur durée d'exécution est raisonnable. Les méthodes approchées peuvent être classées en plusieurs grandes familles. On distingue.

- les méthodes considérées dans la classification de Portmann (1987) et GHOTA (1993), à savoir les méthodes par construction (voir par exemple Ghedjati 1992 a et b), les méthodes par décomposition (voir par exemple Roy 1970), les méthodes par relaxation (voir par exemple Van de Velde 1991), les méthodes par voisinage telles que le recuit simulé (Aarts et Van Laarhoven 1987), la méthode de recherche tabou (Glover 1987, Widmer 1991);

- les méthodes liées à l'intelligence artificielle (Rayson 1985, Kusiak et Chen 1988), voir aussi Erschler (1976) pour les méthodes d'analyse sous contraintes;

- les méthodes inspirées par l'analogie avec des phénomènes biologiques : les algorithmes génétiques (Goldberg 1989) (voir aussi par exemple Alexandre et al. 1995, Ghedjati 1994 et 1996), les réseaux de neurones (Hopfield et Tank 1995).

Pour une vue d'ensemble plus détaillée de ces différentes approches voir par exemple Portmann (1987), GHOTA (1993), Ghedjati (1994).

Nous traitons d'une manière générale dans cet article, les problèmes d'ordonnancement de type job-shop généralisé (voir Woerlee 1991), dans lesquels les machines uniques du job-shop classiques (pour une description de ce type de problèmes voir par exemple Lageweg et al. 1977, Carlier et Pinson 1989, GOTHA 1993) sont remplacées par des groupes de machines parallèles. Plus précisemment, nous nous intéressons au problème d'ordonnancement d'ateliers avec plusieurs machines non identiques en parallèle (voir par exemple, Horowitz et Sahni 1976, Davis et Jaffe 1981) et contraintes de précédence entre les opérations d'un "job" (pièce). Ce problème est NP-difficile puisque le problème est déjà NP-difficile à partir de deux machines identiques et ou un "job" est restreint à une seule opération (Van de Velde 1993).

Pour une résolution approchée et efficace du problème généralisé, nous proposons différentes heuristiques nouvelles. Notre article est divisé en cinq sections, la seconde section décrit le type de problèmes auquel nous nous intéressons, dans la troisième section nous présentons les heuristiques utilisées pour résoudre notre problème, la quatrième section correspond aux résultats expérimentaux, enfin, nous concluons.

2. Description du problème d'ordonnancement considéré

On trouvera dans Ghedjati (1994) une description des différents types de problèmes d'ordonnancement ainsi que les principales références qui les concernent. Nous considérons dans cet article un problème de job-shop très général mais néanmoins réaliste puisque c'est celui que l'on rencontre généralement dans les ateliers de petite série. Nous nous intéressons donc à un atelier comportant m machines qui doivent fabriquer n pièces. Chaque pièce j consiste en une suite de k opérations $O_{1,j},\dots,O_{k,j}$. Une ou plusieurs machines sont susceptibles d'exécuter une opération $O_{i,j}$ avec des durées différentes : il existe un ensemble de machines M (non identiques en parallèle) associé à chaque opération $O_{i,j}$. Cette opération doit être exécutée sans interruption par une seule machine, elle dure $p_{i,j,r}$ lorsqu'elle est exécutée par la machine r de l'ensemble $M_,$. Nous acceptons le cas simple de la gamme linéaire et des cas plus généraux comportant n'importe quel graphe de précédence sans circuit et pas forcément connexe entre les opérations d'une pièce. Toutefois, comme il s'agit d'une même pièce physique, nous n'acceptons pas de recouvrement entre les exécutions de deux opérations de la même pièce. Le problème est statique, c'est-à-dire que tous les travaux à traiter sont connus et peuvent démarrer à la date zéro. Le critère utilisé est la minimisation de la durée totale de l'exécution de toutes les pièces (min C_{max}).

Ce problème peut être vu comme un problème de job-shop généralisé de deux manières : d'une part, parce qu'il existe plusieurs machines susceptibles d'effectuer une même opération avec des durées différentes et d'autres part, parce que les contraintes de succession entre les opérations ne constituent pas une gamme linéaire (cas du job-shop classique). Pour la notation de notre problème, nous utilisons le schéma de classification $\alpha/\beta/\gamma$ de Graham et al. (1979) et qui a été révisé par Lawler et al (1982) (voir aussi Blazewicz et al. 1994) où le paramètre α décrit l'environnement des machines, le second champ β représente les caractéristiques des travaux ou "jobs", et le troisième champ, q, spécifie le critère d'optimisation considéré. Ainsi, nous notons le problème job-shop généralisé que nous considérons par le J(R)/prec/C_{max}.

Exemple :

Considérons un exemple décrivant un problème de type job-shop généralisé dont le nombre de jobs (pièces) et le nombre de machines sont respectivement de 4 et de 5.

Chaque nœud représente une opération définie par : le nom de l'opération ($O_{1,2}$); la durée de l'opération ($p_{1,2}$); la ressource (machine) demandée ($R_{1,2}$ = M2). Dans le cas d'une opération pouvant être effectuée par plus d'une machine, par exemple O1,1, les durées et les machines associées sont notées successivement : (3,M1) (4,M3) (2,M4), ceci explique le fait que O1,1 peut s'exécuter sur l'une des trois machines M1, M2, ou M3 en des durées respectives de 3, 4, 2.

Les flèches représentent les contraintes de précédence, en ce sens que pour le job j1 nécessitant 5 opérations $O_{1,1}$, $O_{1,2}$, ..., $O_{1,5}$, il est indispensable que O1,1 soit exécutée avant O1,2 (le nombre de prédécesseur de $O_{1,2}$ est 1), $O_{1,2}$ et $O_{1,3}$ avant $O_{1,4}$ etc.

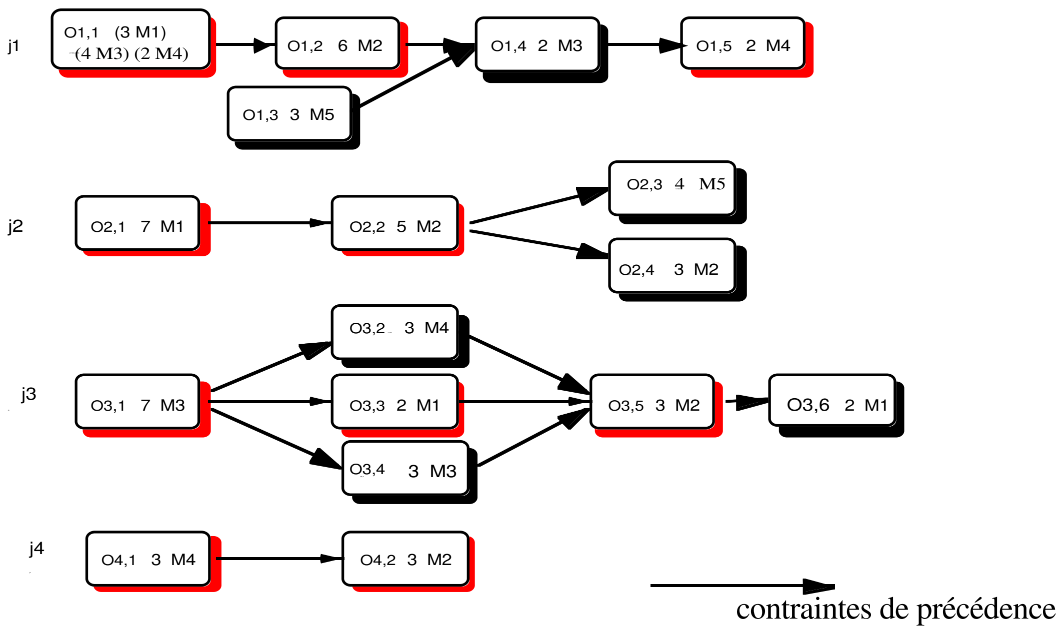


Fig.1: Un exemple de problème de type job-shop généralisé (à machines non identiques en parallèle et contraintes de précédence) de 4 jobs et de 5 machines

3. Description des heuristiques et description générale de l'algorithme

Les heuristiques ont pris une place importante dans la recherche de solutions (Baker 1974, Ibarra et Kim 1977, Panwalkar et Iskandar 1977, French 1982, Potts 1985, Fox et al. 1986, Ulusoy et Özdamar 1989, França et al. 1994), et les logiciels les plus répandus sont basés sur des heuristiques. L'objectif majeur est d'obtenir des solutions satisfaisantes en un temps raisonnable, pour des problèmes complexes d'optimisation, de taille industrielle, dont l'algorithme optimal ou exact est incapable de donner une solution acceptable. La création ou la fabrication d'une heuristique nécessite une bonne connaissance du problème et du domaine d'application. Une bonne heuristique doit être, d'une manière générale, simple, facile à implémenter et doit proposer un bon résultat, pour autant que l'on puisse savoir si l'on est proche de l'optimum et enfin, produire une solution acceptable en un temps raisonnable. Des variantes de description d'une bonne heuristique peuvent être trouvées dans la littérature, (voir par exemple, de Werra 1987, Zanakis et Evans 1981).

Pour la résolution de notre problème, nous avons construit une famille d'heuristiques basées sur l'approche ou la méthode par construction progressive. Ces différentes heuristiques consistent à prendre une décision à chaque itération de l'algorithme (générateur d'ordonnancement) concernant d'une part, l'affectation des machines aux opérations et d'autre part, l'ordonnancement des opérations sur les machines (Ghedjati 1994).

Description de l'algorithme

Nous définissons avant de commencer l'ordonnancement l'ordre dans lequel nous considérons les machines une à une (machine courante). Nous utilisons la stratégie opération par opération pour la construction des ordonnancements sans délai (Baker 1974), c'est-à-dire que nous ne laissons pas une machine inoccupée s'il existe une opération en attente d'être exécutée au pied de la machine et n'ayant pas été définitivement affectée à une autre machine. Nous pouvons également utiliser des ordonnancements actifs (Baker 1974) tout en conservant par ailleurs les mêmes techniques.

Notre algorithme de base ou générateur d'ordonnancement (Ghedjati 92a et 94) consiste à remplir les intervalles de temps (créés dynamiquement) en parallèle sur toutes les machines. Il recherche, à chaque instant t, la première machine inoccupée à liste d'opérations possibles non vide. Les opérations admissibles (candidates, possibles), sont issues d'une sélection parmi les opérations des "jobs" à effectuer par la machine en cours, et rajoutées à la liste des opérations possibles ou admissibles. Le compteur temps progresse d'une unité de temps à chaque itération. Pour chaque valeur du compteur de temps, toutes les machines sont donc examinées et des décisions sont prises pour les machines non occupées pour lesquelles il existe des opérations accessibles candidates. Les opérations candidates à un placement sur une machine donnée à un instant t sont triées selon une règle de priorité. Boucon (1991), dans ses tentatives de classification de règles, présente la règle "**SPT**" (Shortest Processing time) comme une règle ayant souvent de bonnes performances et donnant de bons résultats notamment dans le cas de l'atelier de type job-shop (même s'ils ne sont pas les meilleurs). L'auteur conclut qu'il est très difficile de faire une généralisation de classification des règles, étant donné le nombre de critères et les nombreux paramètres entrant en jeux. Nous avons retenus essentiellement les règles (**SPT**; **SPT/RANDOM** : on applique SPT et à défaut ou en cas de conflit nous faisons un choix aléatoire). Nous remarquons que la structure adaptée nous permet de tester facilement plusieurs types de règles de priorité telles que la priorité à l'opération ayant le plus grand nombre de successeurs /SPT (à égalité on applique SPT), la priorité à l'opération dont la somme de sa durée et des durées de ces successeurs (jusqu'à l'achèvement de la gamme) est la plus grande, en prenant la durée la plus courte pour les opérations successeurs à choix multiple /SPT. Deux variantes se dégagent de cette dernière règle dans le cas où l'opération à placer est à choix multiple : d'une part, la durée de l'opération à placer est celle de la machine considérée (machine courante), et d'autre part, la durée de l'opération à placer est prise sur la machine qui l'exécute le plus rapidement.

Il reste par la suite à décider, en balayant les opérations dans l'ordre considéré, laquelle on place à l'instant t sur cette machine. On décide éventuellement également lors de ce balayage de l'affectation définitive ou non d'opérations à d'autres machines. Tous ces détails sont explicités selon le cas par les différentes heuristiques décrites après les étapes de l'algorithme.

tant que toutes les opérations des pièces ne sont pas ordonnancées **faire**

tant qu'il existe une machine de libre (dans l'ordre du tableau des données organisées précédemment) sur l'intervalle de temps courant **faire**

pour chaque pièce contenant des opérations à exécuter par cette machine **faire**

sélectionner les opérations candidates et les ajouter à la liste des opérations candidates ou possibles correspondante par la procédure "sélection et ajout de candidats";

fin pour

choisir parmi les opérations candidates à un placement sur la machine courante, l'opération prioritaire (appliquer une règle de priorité entre les opérations);

si l'opération peut être exécutée par plusieurs machines **alors**

appliquer une heuristique pour le choix d'une d'entre elles et décider ou non de l'affectation définitive de l'opération considérée sur la machine choisie (si cette machine n'est pas la machine courante);

si l'opération n'a pas de choix multiple de machines ou **si** la machine choisie est la machine courante **alors**

placer l'opération dans l'intervalle de temps en cours du planning; mettre l'état de la machine à "occupée"; mettre l'état de l'opération à "placée"; retirer l'opération de la liste des opérations candidates;

fin tant que

mettre les opérations terminées dans la liste des opérations terminées de la pièce; mettre l'état des machines à "libre"; mettre l'état de l'opération à "terminée";

incrémenter l'intervalle de temps; replacer les opérations en cours non encore terminées dans le nouvel intervalle de temps (soit dans l'intervalle de temps courant) du planning;

fin tant que

étapes de l'algorithme

Heuristiques pour l'affectation des machines

Toute opération candidate à un placement se voit attribuer une machine selon une heuristique et une technique spécifique du générateur d'ordonnancement. S'il n'y a pas de choix multiple pour cette opération alors on la place dans l'intervalle de temps courant du planning de la machine courante. On met l'état de la machine à "occupé", l'état de l'opération à "placé" et on retire cette opération de la liste des opérations possibles (donc dans ce cas, on a automatiquement un placement). Si un choix multiple existe, alors on applique une des heuristiques statiques (H1, H2) ou dynamiques (H3, H4, H5, H6, H7) pour le choix d'une machine. Ces heuristiques sont détaillées dans ce qui suit.

3.1. Heuristiques statiques

Ces heuristiques sont dites statiques car elles ne tiennent pas compte des charges des machines au fur et à mesure de la construction de l'ordonnancement.

- L'heuristique H1 place la première opération de la liste des opérations candidates à un placement, sans se préoccuper des choix multiples et des charges des machines, autrement dit, l'opération la plus prioritaire (suivant les règles de priorité SPT; SPT/RANDOM décrites dans le paragraphe ci-dessus) est placée sur la première machine disponible dans l'ordre où les machines sont examinées.

- L'heuristique H2 utilise systématiquement la machine la plus rapide pour effectuer chaque opération prioritaire de la liste des opérations candidates et qui possède un choix multiple de machines.

Les heuristiques H1 et H2 ne tiennent pas compte de la charge des machines. Nous les avons retenues pour servir de comparaison par rapport aux heuristiques qui tiennent compte dynamiquement de la charge des machines et qui sont décrites par la suite.

3.2. Heuristiques dynamiques

Nous nous intéressons plus particulièrement, dans cet article, à l'introduction et à la modification dynamique d'heuristiques. Nous insistons sur le fait que la structuration des données que nous avons adaptée va permettre de greffer à plusieurs endroits des heuristiques nouvelles en particulier celles concernant le choix entre plusieurs machines multiples qui permettront de traiter de vrais problèmes de job-shop généralisé. Cette approche est opportuniste car elle contrôle l'évolution des charges des machines durant la construction de l'ordonnancement, et affecte d'une manière générale l'opération prioritaire à la machine la moins chargée dans l'état courant du système. Mais, nous remarquons que nous ne faisons pas de "backtracking", c'est-à-dire qu'une fois qu'on décide d'affecter une opération à une machine, cette décision sera définitive, même si l'on s'aperçoit plus tard que la machine supposée la moins chargée est en fait la plus chargée. Les heuristiques que nous avons conçues sont présentées ci-dessous; ce sont des heuristiques nouvelles et particulières qui sont soit basées sur la charge potentielle maximale des machines soit des heuristiques utilisant des "probabilités" d'utilisation des machines.

3.2.1. Heuristiques basées sur la charge potentielle maximale des machines

- L'heuristique H3 (Ghedjati 1992 b) affecte l'opération la plus prioritaire à la machine la moins chargée. La charge des machines est calculée de manière dynamique au fur et à mesure de la construction de la solution. A l'instant t où l'on espère placer une opération à choix multiple de machines, la charge potentielle des machines est constituée de la durée restante de l'opération en cours d'exécution augmentée de la somme des durées de toutes les opérations non encore effectuées (ou placées) et non encore affectées à une autre machine (voir Fig. 2) et qui relèvent de ce type de machine. Dans le cas où la machine courante n'est pas la moins chargée, alors l'opération est définitivement affectée à la machine choisie. Cependant, l'opération considérée ne sera placée sur la machine sélectionnée que lorsqu'elle sera prioritaire sur cette dernière, une fois qu'elle sera la machine courante.

Le calcul de la charge des machines en fonction du temps est le suivant.

<u>t = 0</u>

calcul de la charge (ch) initiale pour toute machine Mi en fonction de la durée *p(o,Mi)* de toute opération o (non encore ordonnancée) qu'elle peut effectuer :

$$Ch(Mi)\,(0) = \sum_{o \in L_{Mi}} p(o,Mi);$$

(L_{Mi}: Liste des opérations à exécuter par la machine Mi)

<u>t quelconque</u>

calcul de la charge (ch) à partir de l'instant t pour toute machine Mi (charge des machines calculée de manière dynamique) :

$$Ch(Mi)\,(t) = \text{durée restante de l'opération en cours} + \sum_{o'} p(o',Mi)$$

(o': opérations de L_{Mi} non encore placées et non affectées à une autre machine)

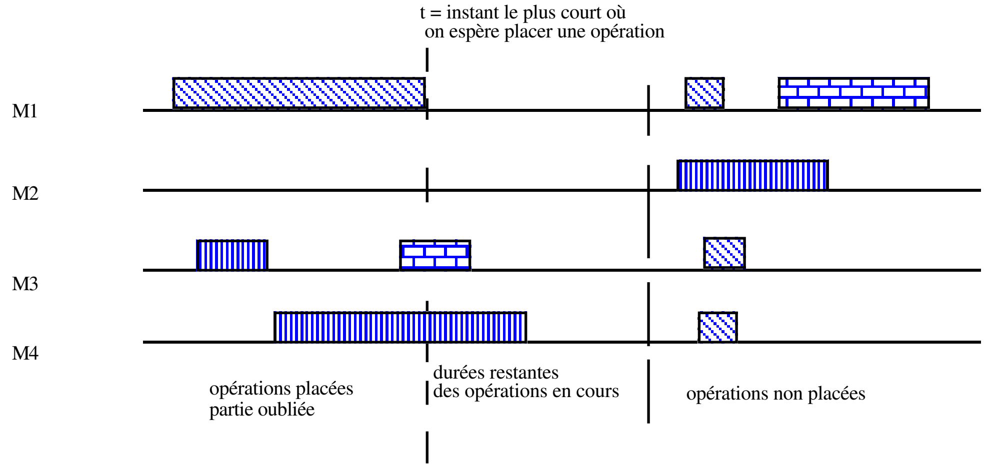


Fig. 2 Charge des machines à l'instant t

3.2.2. Heuristiques utilisant des "probabilités" (Portmann et Ghedjati, 1995)

Ces heuristiques calculent dynamiquement (pour le choix d'une machine) la charge potentielle des machines au fur et à mesure de la construction de l'ordonnancement. Elles ont été introduites et implémentées dans ma thèse sur la suggestion de M.C. Portmann. Cette charge potentielle est utilisée de manière à choisir la machine à affecter à l'opération prioritaire en effectuant ce que l'on espère être un "bon" compromis entre affecter l'opération à la machine potentiellement la moins chargée et affecter l'opération à une des machines qui l'exécutent le plus rapidement possible. Autrement dit, une opération *o* qui peut être effectuée par plusieurs machines $M_{o,1}$, $M_{o,2}$, ..., $M_{o,r}$ aura d'autant plus de chance d'être placée sur la machine *Mi*, que la durée *p(o,Mi)* est petite et que la machine *Mi* est peu chargée. Le problème est que ce raisonnement est récursif car la charge potentielle des machines dépend de la manière dont on affecte les opérations aux machines. Tant qu'une opération n'est pas affectée définitivement à une machine, on va supposer, dans le calcul de la charge potentielle, que sa durée est répartie entre les machines qui peuvent l'effectuer. La proportion de la durée revenant à chaque machine est calculée de telle sorte qu'elle est d'autant plus grande que la machine est potentiellement moins chargée et qu'elle exécute d'autant plus vite l'opération. Cette proportion sera appelée tantôt "probabilité" car la charge potentielle des machines peut être assimilée à une espérance mathématique de charge et prend des valeurs réelles comprises entre 0 et 1 dont la somme est égale à 1 et tantôt "indice d'attraction" car cette valeur va également être utilisée comme un indicateur de priorité dans les heuristiques.

Comme l'indice d'attraction est fonction de la charge potentielle des machines et que la charge potentielle des machines est fonction de l'indice d'attraction, la méthode retenue pour le calcul des indices d'attraction peut être vue comme une méthode de point fixe utilisant un processus de calcul itératif : on suppose donnée la valeur des indices d'attraction, on en déduit la charge potentielle des machines, puis en fonction de cette charge on déduit la nouvelle valeur des indices d'attraction et on réitère ce processus jusqu'à stabilisation des indices d'attraction. L'algorithme itératif que nous avons construit à la base de cette méthode (voir ci-dessous) démarre avec des indices d'attraction identiques pour toutes les machines multiples associées à une opération (équiprobabilité de choisir n'importe quelle machine pour l'exécution de l'opération) et s'arrête lorsque la variation des indices d'attraction devient inférieure à un seuil donné d'une itération à l'itération suivante. Les expérimentations que nous avons effectuées montrent que cet algorithme converge très rapidement, en général en moins de 5 itérations. Le calcul des indices d'attraction est refait de manière dynamique au fur et à mesure de la construction d'un ordonnancement chaque fois que l'on veut placer une opération à choix multiple de machines.

Le calcul des indices d'attraction à l'instant t est décrit par l'algorithme suivant.

a-1. calcul de "**l'indice d'attraction**" (inda) initial pour toute opération o non encore ordonnancée et toute machine Mi :

$$inda(o, Mi)(t) = \frac{1}{\text{le nombre de machines susceptibles d'exécuter } o}$$

a-2. calcul de "**l'espérance de charge**" (Ec) initiale pour toute machine Mi en fonction des indices d'attraction :

$$Ec(Mi)(0) = \sum_{o' \in L_{Mi}} inda\,(o',Mi) \times p(o',Mi);$$

(L_{Mi}: liste des opérations à exécuter par la machine Mi)

b- calcul à l'instant t de "**l'indice d'attraction**" pour toute opération o (à choix multiple de machines) non encore ordonnancée et toute machine Mi :

$$\mathbf{inda(o,Mi)(t)} = \frac{\dfrac{1}{\mathbf{p(o,Mi)\ x\ Ec(Mi)\ (t)}}}{\displaystyle\sum_{\mathbf{Mi'}} \frac{\mathbf{1}}{\mathbf{p(o,Mi')\ x\ Ec(Mi')\ (t)}}}$$

(Mi': machines susceptibles d'exécuter l'opération o)

c- calcul de l'espérance de charge (Ec) à partir de l'instant t pour toute machine Mi en fonction des indices d'attraction :

$$Ec(Mi)(t) = \text{durée restante de la tâche en cours} + \sum_{o'} inda(o',Mi)\,(t) \times p(o',Mi)$$

(o': opérations non encore placées et non affectées à une autre machine)

d- répéter les phases b et c jusqu'à stabilité.

Nous avons intégré ces indices d'attraction de deux manières différentes dans la construction de nos heuristiques : dans une première famille d'heuristiques en utilisant les indices d'attraction comme des probabilités d'affecter une opération à une machine (*heuristique choix des machines par tirage au sort sur les indices d'attraction*) et dans une deuxième famille d'heuristiques en utilisant les indices d'attraction comme des indicateurs de priorité des machines pour effectuer les opérations (*heuristiques choix des machines sur priorité*).

. Choix des machines par tirage au sort sur les indices d'attraction

- L'heuristique H4 (Ghedjati 1992 b) utilise les indices d'attraction comme des probabilités. L'opération la plus prioritaire est attirée vers les machines en utilisant une technique de loterie sur les indices d'attraction (la probabilité pour une machine d'être tirée au sort est proportionnelle à son indice d'attraction). La charge des machines et les indices d'attraction sont recalculés dynamiquement, chaque fois que l'on veut placer une opération à choix multiple de machines. Comme dans cette heuristique, on se demande seulement à un instant t si oui ou non on affecte l'opération la plus prioritaire à la machine courante, la technique utilisée est simple; on génère un nombre pseudo-aléatoire compris entre 0 et 1. S'il est plus petit que l'indice d'attraction de la machine courante pour l'opération considérée alors on affecte l'opération à la machine considérée. Dans le cas où le choix de la machine courante est rejeté, aucune affectation définitive de l'opération prioritaire n'est prévue, on passe à l'opération suivante dans la liste des opérations possibles. Cette heuristique faisant intervenir des choix aléatoires, peut être exécutée plusieurs fois et fournir des résultats différents. L'heuristique H4(nb) consiste à utiliser un nombre nb de fois l'heuristique H4 et à retenir le meilleur résultat obtenu. Pour que sa durée soit comparable à celles des autres heuristiques, il faut utiliser H4(1).

. Choix des machines sur priorité (priorité à la plus grande valeur des indices d'attraction)

- L'heuristique H5 (Ghedjati 1992 b) est très semblable à l'heuristique H4, simplement, au lieu d'effectuer des tirages au sort on utilise les indices d'attraction comme des indicateurs de priorité. L'opération la plus prioritaire est affectée à la machine ayant la plus grande valeur d'indice d'attraction. Dans le cas où la machine courante n'est pas prioritaire (donc non choisie), l'opération est définitivement affectée à la machine choisie, mais ne sera placée que lorsqu'elle sera prioritaire sur cette machine une fois qu'elle sera la machine courante.

3.2.3. Analyse de la décision d'affectation définitive d'une opération sur une autre machine (choisie par rapport à la machine courante)

Une opération à choix multiple de machines, prioritaire sur la machine courante mais ne pouvant être placée sur cette machine (l'heuristique appliquée a rejeté le choix de cette machine) se ramène à deux situations.

1- On affecte définitivement à cet instant l'opération à la machine qui a été choisie, mais elle ne sera placée que lorsqu'elle sera prioritaire sur cette machine une fois qu'elle sera la machine courante (cas des heuristiques H3 et H5). Dans ce cas on prend éventuellement le risque, qu'entre temps, il s'avère que ce choix ne soit pas le meilleur .

2- On évite de prendre ce risque en décidant de ne pas affecter définitivement l'opération sur la machine jugée la plus prometteuse à cet instant. Puisque l'opération ne sera placée que lorsqu'elle sera prioritaire sur la machine courante, alors on relancera à ce moment là le calcul décidant à nouveau du choix d'une machine.

Cette proposition est matérialisée par les heuristiques H6 et H7, qui permettront avec les heuristiques H3 et H5 d'élaborer une étude et une analyse de ces deux situations.

- L'heuristique H6 est très semblable à l'heuristique H3 sauf qu'elle ne décide pas de l'affectation définitive de l'opération prioritaire sur une autre machine jugée la plus prometteuse à l'instant t, si la machine courante ne l'est pas. Autrement dit, l'opération la plus prioritaire n'est affectée à la machine courante que si elle est la moins chargée. La charge potentielle des machines est recalculée de manière dynamique au fur et à mesure de la construction de la solution à chaque fois que l'on espère placer une opération à choix multiple de machines. A l'instant t, la charge potentielle des machines à choix multiples est constituée de la durée restante de l'opération en cours d'exécution augmentée de la somme des durées de toutes les opérations non encore effectuées et non encore affectées à une autre machine.

- L'heuristique H7 est très semblable à l'heuristique H5. Cependant, dans le cas où la machine courante n'est pas prioritaire à l'instant t (elle n'a pas la plus grande valeur d'indice d'attraction), on ne décide pas de l'affectation définitive de l'opération prioritaire sur une autre machine.

4. Expériences et résultats

Des expériences ont été effectuées avec divers types de données issues de la littérature ou sur des échantillons de grandes tailles (360 exemples) générés aléatoirement dans le but de comparer les heuristiques entre elles. Les résultats expérimentaux des données aléatoires sont classés en 10 familles selon différents paramètres du problème afin d'observer le comportement des heuristiques sur chaque famille d'exemples. Ils sont regroupés, soit par nombre de machines, soit par nombre de gammes (n = m , 2m ou 3m), soit encore selon la diversité des durées des opérations (petites durées, grandes durées, ou durées variables) (voir résultats détaillés dans Ghedjati 1994). Ce travail est implémenté en C, et tourne sur une station de travail SPARC10 sous le système Unix.

Nous utilisons dans les expériences les notations suivantes.

notations :

m :	nombre de machines;
n :	nombre de travaux (jobs) à ordonnancer;
m x n :	problème à m machines et n travaux;

B&B : "branch and bound" (ou procédure par séparation et évaluation);
H4(nb) : (nb) nombre d'exécutions de l'heuristique stochastique H4;
t : temps CPU en secondes et /ou en microsecondes;
PTD: petite durée;
GDD: grande durée;
CM : moyenne des durées totales obtenues (ou du C_{max});
%G : pourcentage de fois où une heuristique, pour le sous-ensemble retenu d'heuristiques, fournit la meilleure solution pour la durée totale;
ERM: l'erreur relative moyenne commise si on n'utilise que l'heuristique considérée au lieu d'utiliser la meilleure valeur trouvée par l'ensemble des heuristiques testées.

4.1. Exemples tirés de la littérature

Des exemples empruntés à la littérature et concernant des problèmes plus simples que le notre, nous ont permis de tester nos méthodes et de comparer nos résultats par rapport aux solutions optimales existantes.

Exemple de type R//C_{max} ou problème à machines non identiques en parallèle

Pour le cas des problèmes à machines non identiques (ou non reliées) en parallèles qui sont une simplification de notre problème général (pas de contraintes de précédence), nous considérons l'exemple (3 x 8) de Van de Velde (1991 et 1993). Les résultats sont portés dans le tableau suivant.

Tableau 1

travaux	méthode	3 x 8 C_{max}	t
Van de Velde 1991 et 1993	relaxation lagrangienne	20	_
nos résultats	H1	27	39374 µs
"	H2	33	49441 µs
"	H3	36	76493 µs
"	H4(1)	24	1.5 s
"	H4(5)	22	33.5 s
"	H4(10)	22	72 s
"	H5	25	3 s
"	H6	25	6 s
"	H7	25	5s

La meilleure solution est donnée par l'heuristique H4(5), elle est à 10% de l'optimum (20).

Exemples de type J//C_{max} ou job-shop classique

Nous présentons, dans ce paragraphe, les résultats d'expérimentations (voir tableau 2) de deux exemples tests (bien connus dans la littérature) des problèmes job-shop (Muth and Thompson 1963) : le problème (6 x 6) et le problème (10 x 10). Ces exemples ont résisté plusieurs années avant d'être résolus. Nous testons également deux exemples (5 x 10) de ce type de problèmes (Adams et al. 1988) et qui sont extraits de Pinson (1988) (voir tableau 2).

Nous remarquons, que pour ces problèmes, seule l'heuristique H1 est testée vu qu'il n'existe pas de choix multiples de machines dans le *job-shop* classique pour pouvoir faire des tests avec les autres heuristiques (H2, ..., H7). Mais l'heuristique H1 est une simple heuristique statique retenue ici pour avoir une bonne solution initiale pouvant être par la suite améliorée par d'autres méthodes telles que le recuit-simulé ou la méthode tabou ou encore les algorithmes génétiques.

Tableau 2

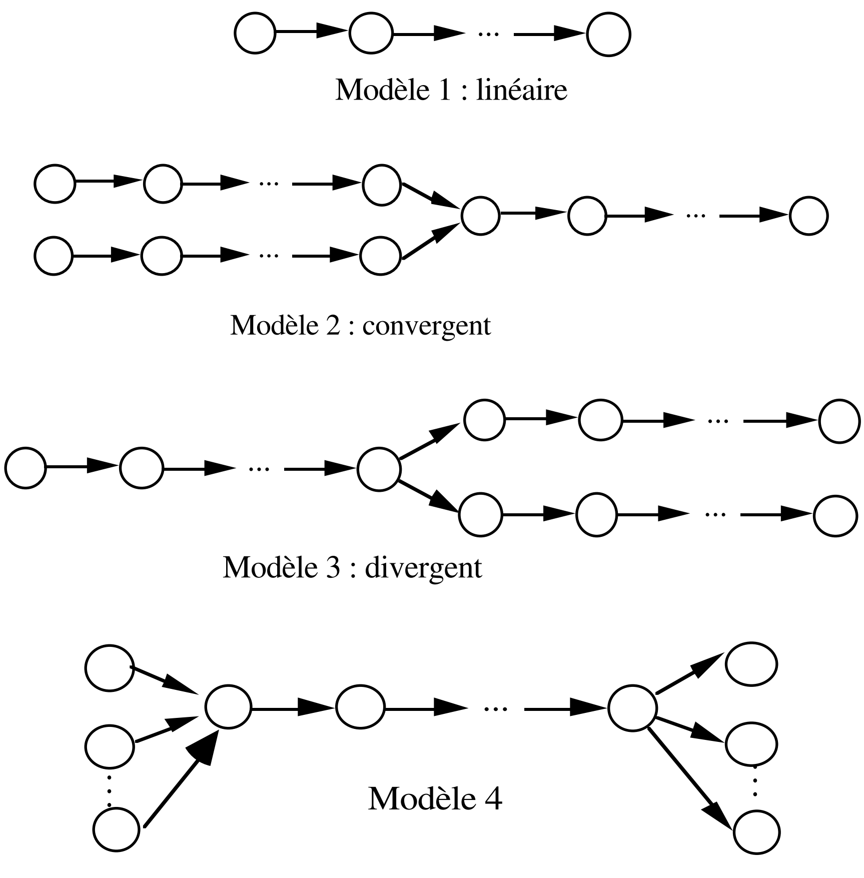
travaux	méthode	6x6 C_{max}	t	10x10 C_{max}	t	5x10 C_{max}	t	5x10 C_{max}	t
Balas 1969	B&B	55	_	1177	_	_	_	_	_
McMahon & Florian 1975	B&B	55	_	972	_	_	_	_	_
Barker & McMahon 1985	B&B	55	_	960	_	_	_	_	_
Carlier & Pinson 1989	B&B	55	_	930	_	_	_	_	_
Adams et al. 1988 extrait de Pinson 1988	B&B	_	_	_	_	666	_	593	_
nos résultats	H1	88	1125µs	1074	1s	751	388380µs	610	355045µs

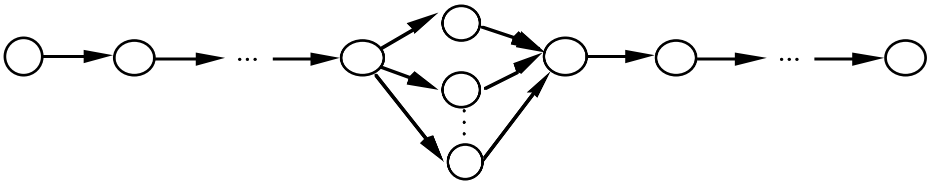
Pour ces exemples (voir tableau 2), nous sommes à 2% de l'optimum dans le meilleur des cas et à 15% de l'optimum dans le pire des cas, mais en des temps très rapides.

4.2. Exemples générés aléatoirement

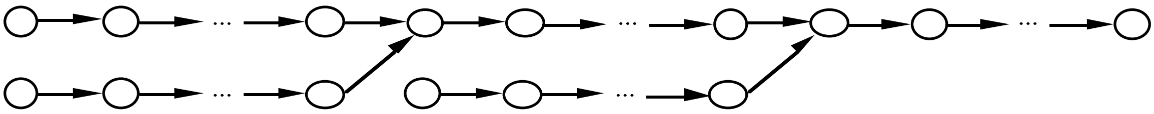
Pour comparer les heuristiques entre elles nous avons généré aléatoirement 360 énoncés de notre problème avec des gammes absolument quelconques. Nous avons alors généré 10 exemples pour chaque valeur des paramètres avec 3 valeurs pour le nombre de machines m (m = 5, 10 et 15), 3 valeurs pour le nombre de pièces n (n = m, 2m et 3m, avec le nombre d'opérations NOP = n si n = m et NOP = aléatoire si n ≠ m), 2 valeurs pour la multiplicité des machines (nombre de machines susceptibles d'exécuter une opération à choix multiple de machines) "**faible** : faible probabilité d'avoir un grand choix de machines multiples" et "**forte** : cas inverse du précédent"; et 2 valeurs pour la diversité des durées "faible" (entre 1 et 5) et "forte" (entre 1 et 50). Le graphe de précédence de chaque pièce est choisi selon 6 modèles (voir Fig. 3) : *le modèle 1* correspond à une gamme linéaire comme dans les *job-shop*s classiques; *le modèle 2* est un modèle convergent simple, deux gammes linéaires se rejoignent pour un assemblage, puis la gamme se poursuit linéairement; l*e modèle 3* est un modèle divergent simple, analogue au modèle convergent; *le modèle 4* est un modèle convergent complexe : il commence par des opérations en nombre quelconque en parallèle avant une opération d'assemblage, suivie d'une gamme linéaire et termine par des nouvelles opérations en parallèle; *le modèle 5* est une variante légèrement différente du modèle 4 : il commence par une gamme linéaire suivie d'opérations en nombre quelconque en parallèle, précédant une opération d'assemblage, elle-même suivie par une gamme linéaire; *le modèle 6* commence par deux gammes linéaires en parallèle précédant une opération d'assemblage, suivie d'une gamme linéaire, en parallèle avec une autre gamme linéaire, avant une deuxième opération d'assemblage, suivie d'une gamme linéaire. Pour plus de détails voir Ghedjati (1994).

Fig. 3 : Modèles des pièces considérés





Modèle 5



Modèle 6

4.2.1. Comparaison des heuristiques (H1, ...,H5)

L'heuristique H4 a la particularité d'avoir un comportement stochastique et donc de ne pas donner le même résultat si on l'exécute plusieurs fois. Nous avons donc mené les comparaisons avec une seule exécution de H4, 5 exécutions de H4 (en retenant la meilleure des solutions trouvées) et 10 exécutions de H4, les autres heuristiques n'étant exécutées qu'une seule fois.

Les résultats obtenus sont résumés dans les tableaux suivants :

Tableau de comparaison des heuristiques (H1,...,H5) avec 1 exécution de H4

	H1	H2	H3	H4(1)	H5
CM	262.36	244.69	284.64	243.49	**226.36**
%G	6	16	5	13	**65**
ERM	0.199	0.125	0.292	0.125	**0.029**

Tableau de comparaison des heuristiques (H1,...,H5) avec 5 exécutions de H4

	H1	H2	H3	H4(5)	H5
CM	262.36	244.69	284.64	**222.94**	226.36
%G	3	11	2	44	**47**
ERM	0.217	0.143	0.312	**0.043**	0.046

Tableau de comparaison des heuristiques (H1,...,H5) avec 10 exécutions de H4

	H1	H2	H3	H4(10)	H5
CM	262.36	244.69	284.64	**217.85**	226.36
%G	3	10	1	**56**	40
ERM	0.227	0.153	0.323	**0.028**	0.056

On constate qu'avec une et cinq exécution(s) de H4, c'est l'heuristique H5 qui est la plus performante en % de fois où elle est la meilleure. Elle est battue par H4 en moyenne que lorsqu'on exécute celle-ci cinq ou dix fois. Il faut noter aussi que les heuristiques H4(5) ou H4(10) et H5 sont bien meilleures que les heuristiques statiques et ne sont jamais très éloignées des meilleurs résultats obtenus.

4.2.2. Comparaison des heuristiques (H4, H5) avec les heuristiques (H1, H2, H3)

Nous effectuons une comparaison des heuristiques utilisant les "probabilités" ou les priorités (H4, H5) avec les heuristiques plus simplistes prenant en compte ou non les charges des machines (H1, H2, H3), dans le but de synthétiser autrement les résultats précédents. Nous voulons en fait connaître l'apport de H4 $\cup$ H5 par rapport à H1 $\cup$ H2 $\cup$ H3. D'une manière générale le principe consiste à sélectionner d'une part la meilleure solution des heuristiques (H1, H2 et H3) et d'autre part la meilleure des heuristiques (H4 et H5). Nous appelons méta-heuristique MH1 l'heuristique qui consiste à appliquer toutes les heuristiques (H1, H2, H3) et à retenir le meilleur résultat obtenu et MH2(nb) l'heuristique qui consiste à appliquer les heuristiques H4(nb) et H5 et à retenir le meilleur résultat obtenu. Ceci est formulé comme suit.

Pour chaque individu (i), $MH1_i = \min_{H1,H2,H3} (Cmax_i)$;

$$MH2(nb)_i = \min_{H4(nb),H5} (Cmax_i) ;$$

Les résultats sont portés dans les tableaux suivants.

Tableau de comparaison de (MH1, MH2) avec 1 exécution de H4

	MH1	MH2 H4(1)
CM	235.54	**223.5**
%G	26	**78**
ERM	0.080	**0.016**

Tableau de comparaison de (MH1, MH2) avec 5 exécutions de H4

	MH1	MH2 H4(5)
CM	235.54	**217.29**
%G	17	**87**
ERM	0.097	**0.007**

Tableau de comparaison de (MH1, MH2) avec 10 exécutions de H4

	MH1	MH2 H4(10)
CM	235.54	**214.44**
%G	14	**91**
ERM	0.106	**0.004**

On constate qu'utiliser successivement 10 fois H4 et une fois H5 plutôt que les heuristiques plus simples une fois chacune permet d'obtenir la meilleure solution dans 91% des cas (87% et 78% en n'exécutant H4 que 5 et 1 fois) et on commet une erreur relative de 10,6% en n'utilisant que les méthodes simples (H1, H2 et H3) plutôt que les méthodes H4 et H5.

4.2.3. Comparaison des heuristiques H3 et H6

Nous effectuons une comparaison de l'heuristique H3 prenant compte dynamiquement des charges des machines (en les surestimant, puisqu'une opération à choix multiple non affectée est intégrée à la charge de toutes les machines susceptibles de l'exécuter). Rappelons que H3 décide d'affecter une opération à choix multiple à la machine jugée la plus prometteuse à l'instant t sachant qu'elle ne pourra être placée sur cette machine qu'ultérieurement. H6 est semblable à H3 sauf qu'elle ne décide pas de l'affectation de l'opération sur une machine donnée si elle ne peut pas l'exécuter à cet instant.

Tableau de comparaison de H3 avec H6

	H3	H6
CM	284.64	**269.42**
%G	34	**73**
ERM	0.071	**0.016**

On constate que H6 est intéressante et améliore nettement H3.

4.2.4. Comparaison des heuristiques H5 et H7

La même situation que précédemment se présente pour H5 et H7. Ces deux heuristiques utilisent les priorités, mais H7 ne décide pas à un instant donné de l'affectation de l'opération (à choix multiple de machines) sur une machine si elle ne peut pas l'exécuter à cet instant.

Tableau de comparaison de H5 avec H7

	H5	H7
CM	226.36	**226.08**
%G	**68**	62
ERM	**0.019**	0.021

On remarque que H7 n'améliore pratiquement pas H5 (cela est probablement dû au fait que H5 tient mieux compte des charges "potentielles" que H3, il y a en quelque sorte une auto correction de l'affectation prématurée dans H3).

4.2.5. Comparaison de l'heuristique H6 avec les heuristiques H4 et H5

Tableau de comparaison de H6 avec H4(nb) et H5

	H4(1)	H4(5)	H4(10)	H5	H6
CM	239.07	222.05	216.68	226.36	269.48
%G	5	31	60	43	2
ERM	0.135	0.048	0.025	0.056	0.256

On constate que les heuristiques H4 et H5 demeurent les meilleures malgré le bon comportement de H6 par rapport à H3.

5. Conclusion

L'objectif de cet article est la résolution approchée de problèmes d'ordonnancement au moyen d'un algorithme général (un générateur d'ordonnancement sans délai) le plus souple possible, pouvant sans grands développements, s'adapter à de multiples variantes. Le problème choisi pour tester cet algorithme est un problème d'ordonnancement d'atelier de type *job-shop* généralisé à machines non identiques en parallèle et avec contraintes de précédence (où les gammes des travaux sont quelconques), mais notre approche peut également fonctionner pour d'autres problèmes particuliers tels que le *job-shop* classique (à gamme linéaire) ou le problème $R//C_{max}$. Le critère considéré est la minimisation de la durée totale de l'ordonnancement, mais les approches utilisées peuvent facilement s'adapter à d'autres critères du même type.

Nous avons développé de nouvelles heuristiques statiques simples et dynamiques plus ou moins complexes liées aux charges des machines. On suppose d'une part, que toute opération non encore placée participe totalement à la charge de toutes les machines qui peuvent l'exécuter. D'autre part, on essaye de tenir compte au mieux de la charge potentielle des machines par l'utilisation des "probabilités" d'affecter une opération à choix multiple sur la machine la plus rapide et la moins chargée. Notre système permet de tester rapidement différentes heuristiques et de basculer facilement et dynamiquement (au fur et à mesure de la construction de la solution) d'une heuristique à une autre sans changer l'algorithme de base.

Des exemples tests de problèmes particuliers empruntés à la littérature ont été traité d'une manière que l'on peut considérer comme satisfaisante. Le temps de calcul est raisonnable (de l'ordre de quelques secondes). Des expérimentations effectuées sur un échantillon de 360 exemples types (générés aléatoirement) de notre problème, montrent, d'une part, qu'en moyenne les heuristiques dynamiques sont toujours meilleures que les heuristiques statiques. D'autres part, l'heuristique H5 utilisant les indices d'attraction comme des indicateurs de priorité des machines pour effectuer les opérations (priorité à la machine de plus grande valeur d'indice d'attraction) est en moyenne plus performante que l'heuristique à caractère stochastique H4 si cette dernière est exécutée moins de 5 fois et que l'utilisation simultanée de H4(5) ou H4(10) avec H5 domine presque toujours les autres heuristiques et donne, dans les cas où la solution exacte est connue une approximation du résultat à 10% près.

Références

F. E.H.L. Aarts, P.J.M. Van Laarhoven (1987), Simulated Annealing : Theory and Applications. D. Reidel Publishing Company, Dordrecht, Holland.

J. Adams, E. Balas, D. Zawack (1988), The shifting bottleneck procedure for job-shop scheduling. Man. Sci., 34, 391-401.

F. Alexandre, C. Cardeira, F. Charpillet, Z. Mammeri, M. C. Portmann (1995), Compu-Search Methodologies II : Scheduling Using Genetic Algorithms and Artificial Neural Networks. To appear in Production and Scheduling of Manufacturing System, Artiba A., El Maghraby S.E., Chapman & Hall eds.

K.R. Baker (1974), Introduction to sequencing and scheduling. John Wiley and Sons, New-York.

J.R. Barker, G.B. McMahon (1985), Scheduling the general job-shop. Man. Sci., 31, 594-598.

R. Bellman (1954), The theory of dynamic Programming. Bull. Amer. Math. Soc.,n° 60, 503-515.

J. Blazewicz, K.H. Ecker, G. Schmidt, J. Weglarz (1994), Scheduling in Computer and Manufacturing Systems. Second, Revised Edition, Springer-Verlag.

D. Boucon (1991), Ordonnancement d’ateliers: aide au choix de règles de priorité. Thèse de doctorat, Ecole Nationale Supérieure de l’Aéronautique et de l’Espace.

J. Carlier, P. Chrétienne (1988), Problème d’ordonnancement / modélisation / complexité / algorithme. Masson.

J. Carlier, E. Pinson (1989), An algorithm for solving the job-shop problem. Man. Sci., 35, 164-176.

E. Davis, J.M. Jaffe (1981), Algorithms for scheduling task on unrelated processors. Journal of the Association for Computing Machinery, 28(4): 721-736, Octobre.

J. *Erschler* (1976), Analyse sous contrainte et aide à la décision pour certains problèmes d'ordonnancement. Thèse d'état, LAAS, Université Paul Sabatier, Toulouse, Novembre.

M.S. Fox, B. Allen, G. Strohm (1986), Job-Shop Scheduling: on investigation in constraint-directed reasoning. Proc. 2nd. American A.I. Conference, Pittsburg.

P.M. França, M. Gendreau, G. Laporte, F.M. Müller (1994), A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective. Computers and operations research, 205-211.

S. French (1982), Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. Wiley.

M.R. Garey, D.S. Johnson (1979), Computers and Interactability: A Guide to the Theory of NP-completeness, Freeman and Co.

F. Ghedjati (1992 a), Utilisation d'heuristiques dans un algorithme de résolution du problème d'ordonnancement d'ateliers du type job-shop, 12ème Journées Internationales "Intelligence Artificielle, Systèmes Experts, Langage Naturel", Avignon, vol. 2, p. 311-324.

F. Ghedjati (1992 b), Heuristiques pour le problème job-shop avec machines non reliées en parallèle, MOAD ' 92 : Colloque International "**M**éthodes et **O**utils d'**A**ide à la **D**écision", Bejaïa, vol. 1, p. 83-88.

F. Ghedjati (1994), Résolution par des heuristiques dynamiques et des algorithmes génétiques du problème d'ordonnancement de type job-shop généralisé (à machines non identiques en parallèle et contraintes de précédence). Thèse de l'Université Paris 6.

F. Ghedjati (1996), Genetic algorithm for the generalized job-shop scheduling problem. IPMU'96 (Information Processing and Management of Uncertainty in Knowledge-Based Systems), Granada (Spain), July 1-5, Vol. 2, 793-801.

F. Glover (1987), Tabu search methods in artificial intelligence and operations research. ORSA, Artificial Intelligence Newletter, 1, no. 2.

D. E. Goldberg (1989), Genetic Algorithms in Search, Optimisation, and Machine Learning. Addison-Wesley Publishing Company, Inc.

GOTHA (groupe de chercheurs) (1993), les problèmes d'ordonnancement. RAIRO, 27 (1), 77-150.

R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan (1979), Optimisation and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics, 5, 287-326.

J. Hopfield, J. Tank (1985), Neural computation of decisions in optimisation problems. Biological Cybernetics, 52, 141-152.

E. Horowitz, S. Sahni (1976), Exact and approximate algorithms for scheduling nonidentical processors. Journal of the Association for Computing Machinery, 23(2): 317-327, Avril.

O.H. Ibarra, C.E. Kim (1977), Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the Association for computing Machinery, 24(2): 280-289, Avril.

A.Kusiak, M. Chen (1988), Expert systems for planning and scheduling manufacturing systems. E. J. of Operational Research, 34, 113-130.

B.J. Lageweg, J.K Lenstra, A.H.G. Rinnooy Kan (1977), Job-Shop Scheduling by implicit enumeration. Man. Sci., 441-450.

E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan (1982). Recent developments in deterministic sequencing and scheduling: a survey. In M.A.H. Dempster, J.K. Lenstra, and A.H.G. Rinnooy Kan, editors, Deterministic and stochastic scheduling, 35-73, Dordrecht, The Netherlands, 1982. NATO Advanced Study and Research Institute, D. Reidel Publishing Company.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D.B. Shmoys (1989), Sequencing and scheduling: algorithms and complexity, technical paper Eindhoven University of Technologie. November.

G. McMahon, M. Florian (1975), On Scheduling with ready times and due dates to minimize maximum lateness. Opns., 23, 475-482.

J.F. Muth, G.L. Thompson (1963), Industrial Scheduling. Prentice-Hall, Englewood Cliffs, NJ.

S.S. Panwalkar, W. Iskandar (1977), A Survey of Scheduling Rules. Operations Research, 25 (1): 45-61.

E. Pinson (1988), Le problème de job-shop. Thèse de l'Université Paris 6.

M.C. Portmann (1987), Méthodes de décomposition spatiale et temporelle en ordonnancement de la production. Thèse d'état, Université de Nancy1.

M. C. Portmann, F. Ghedjati (1995), Affectation et Ordonnancement par des Algorithmes Génétiques. Journées d'Etudes "Affectation et Ordonnancement", GDR Automatique du CNRS (GT3 Ordonnancement), EIII TOURS, 18-19 septembre, 67-80.

C.N. Potts (1985), Analysis of a linear programming heuristic for scheduling unrelated parallel machines. Discrete Applied Mathmatics 10, 155-164.

P. Rayson (1985), A review of expert systems principales and their roles in manufacturing. Robotica, vol. 3.

A.H.G. Rinnooy Kan (1976), Machine Scheduling Problems: classification, complexity and computation. Nijhoff, The Hague.

B. Roy (1970), Algèbre moderne et théorie des graphes. Tome 2, Dunod, Paris.

G. Ulusoy, L. Özdamar (1989), Heuristic Performance and Network / Ressource characteristics in Ressource-Constrained Project Scheduling. J. Opl Res. Soc. vol. 40, No. 12, pp. 1145-1152.

S.L. Van de Velde (1991), Machine scheduling and lagrangien relaxation. Doctoral Thesis, CWI, Amsterdam, April.

S.L. Van de Velde (1993), Duality-based algorithms for scheduling unrelated parallel machines. ORSA Journal on Computing, vol. 5, No. 2, 192-205.

D. de Werra (1987), Design and operation of flexible manufacturing systems : The kingdom of heuristic methods. R.A.I.R.O Recherche Opérationnelle, Volume 21.

M. Widmer (1991), Job-shop scheduling with tooling constraints: a Tabu Search Approach, J. Opl Res. Soc. Vol.42, No. 1, 75-82.

A.P. Woerlee (1991), Decision support systems for production scheduling. Thèse Rotterdam, Avril.

S.H. Zanakis, J.R. Evans (1981), Heuristics "Optimisation": why, when, and how to use it. Interfaces, 11, 5, 84-91.