Séparation et Evaluation pour le problème d'ordonnancement avec blocage.

Abdelhakim Ait Zai[1], Abdelkader Bentahar[1], Hamza Bennoui[1],

Mourad Boudhar[2] et Yazid Mati[3]

[1] Faculté d'Electronique et d'Informatique, Département d'Informatique, USTHB, BP 32 El Alia Alger, Algérie.

[2]Faculté de Mathématiques, Département de Recherche Opérationnelle, USTHB, BP 32 El Alia Alger, Algérie.

[3]Université d'Elkassime Arabie Saoudite.

Résumé : Nous traitons, dans ce papier, du problème d'ordonnancement Job shop avec blocage connu pour être NP-difficile. Après une présentation du problème posé, nous proposons de le résoudre d'une manière exacte en utilisant une méthode par séparation et évaluation SEP. Pour obtenir une solution optimale en un temps raisonnable, nous avons amélioré la méthode en utilisant une technique de séparation originale basée sur les graphes alternatifs. Nous avons utilisé aussi deux méthodes différentes pour l'évaluation d'une solution de la méthode SEP. Dans la dernière partie de ce papier, nous discutons les résultats des deux méthodes.

Mots clés : Job shop, blocage, séparation et évaluation, graphe alternatif.

1. Introduction

En général, les problèmes d'ordonnancement modélisent des problèmes d'ateliers de fabrication; ces ateliers sont composés principalement de ressources (machines, ouvriers et matière première) utiles à la fabrication de plusieurs types de produits. La résolution de ces problèmes revient, dans les méthodes classiques, à déterminer les dates de début et de fin de chaque opération composant les gammes opératoire des produits. Les nouvelles méthodes de résolution, s'orientent vers la recherche de l'ordre des séquences de passage sur chaque machine. Cette technique est plus pratique et rapide pour trouver l'ordonnancement nécessaire. Ces nouvelles techniques, ne change pas la complexité du problème d'ordonnancement classé NP-difficile. Les méthodes de résolution des problèmes d'ordonnancement d'ateliers se divisent en deux principales classes :

Les méthodes exactes ont l'avantage de garantir l'optimum en parcourant toutes les combinaisons possibles d'une manière intelligente. Comme inconvénient majeur, ces méthodes sont caractérisées par un temps de réponse non raisonnable pour des instances de moyenne et grande tailles, la pratique montre que l'utilisation d'un ordinateur (machine de calcul) est insuffisant (espace mémoire, vitesse de calcul) devant une complexité de plus en plus grande.

Concernant les méthodes approchées elles sont très rapides; mais les solutions qu'elles retournent ne sont pas forcements optimales d'où le nom approchées. La validation de ces méthodes nécessite l'utilisation des méthodes exactes, donc on compare les résultats des méthodes approchées à ceux trouvés par les méthodes exactes, puis après validation on lance les méthodes approchées sur des exemples plus grands, mais rien n'assure qu'elles vont converger.

Le problème d'ordonnancement à deux machines F2\\Cmax a été résolu de façon optimale par Johnson. Cet algorithme, et ses variantes, est un moyen rapide d'optimiser l'ordonnancement de processus simples. En effet, il est facile de voir que la complexité de cet algorithme est en O(n log n). Les résultats obtenus par Johnson sont devenus des classiques dans la théorie de l'ordonnancement. Par contre, les problèmes de job shop sont en général NP-complets, même si l'atelier est simple. En effet, Graham R et al. [14] ont montré que les ateliers possédant plus de trois machines, ou un nombre de tâches supérieur ou égal à trois, sont NP-difficiles même si la préemption est permise. De même pour les problèmes à deux machines, dès qu'il y a recirculation, ils deviennent fortement NP-difficiles.

Dans ce papier nous présentons le problème d'ordonnancement avec blocage et sa résolution par une méthode exacte de type séparation et évaluation (SEP). Ce problème est un job shop qui prend en considération le cas où les machines n'ont pas d'aires de stockage, ainsi si la machine suivante d'une opération donnée n'est pas libre, cette opération va rester sur la machine courante même si elle a terminé son traitement, on dit alors que cette machine est bloqué.

2. Position du problème

L'ordonnancement d'atelier à cheminements multiples (Job Shop) est généralement formulé comme suit :
Soit un ensemble de travaux de cardinalité n noté $\mathbf{J=\{J_1,\ldots,Jn\}}$, à traiter par un ensemble de machines de cardinalité m noté $\mathbf{M=\{M_1,\ldots,M_m\}}$; chaque travail j a un sous ensemble ordonné d'opérations noté $\mathbf{Oj=\{oj_1,\ldots\ldots,oj_{pj}\}.}$
Pour cela des contraintes de précédences sont définis :

- ✓ S'il y a une contrainte de précédence entre deux opérations $\mathbf{o_i}$ et $\mathbf{o_j}$ $\mathbf{(o_i \rightarrow o_j)}$ alors $\mathbf{o_j}$ ne peut commencer avant que $\mathbf{o_i}$ se termine. Donc l'ordre de la gamme opératoire est obligatoire.
- ✓ Chaque machine ne peut traiter qu'une seule opération à la fois et un travail ne peut occuper plus qu'une machine en même temps.
- ✓ Les séquences de passage des opérations sur une même machine sont inconnues et à déterminer dans le but de minimiser la durée total de production $\mathbf{C_{max}}$ tout en respectant les contraintes du problème**.**

2.1. Les contraintes

L'atelier de fabrication est soumis à des contraintes technologiques. Ces contraintes touchent à la fois les possibilités d'utilisation des machines et les liens (contraintes) qui peuvent exister entre les opérations. Ces contraintes sont les suivantes :

a) Les machines sont indépendantes les unes des autres (pas d'utilisation d'outil commun par exemple).

b) Une machine ne peut exécuter qu'une seule opération à un instant donné.
c) Chaque machine est disponible pendant toute la durée de l'ordonnancement, en particulier, les pannes ne sont pas prises en compte dans ce modèle.
d) Une opération en cours d'exécution ne peut pas être interrompue (la préemption des opérations n'est pas permise).
e) Les travaux sont indépendants les uns des autres. En particulier, il n'existe aucun ordre de priorité attaché aux travaux.

D'autres types de contraintes peuvent êtres prises en considérations, ceci dépendra du problème posé et du cas à étudier, pour notre cas nous avons choisi les contraintes ci-dessus.
En plus de la relation de précédence on trouve encore d'autres contraintes; celles ci dépendent de l'environnement de production, dans certains ateliers de production, deux opérations peuvent se réalisées en séquentielle ou en parallèle, ceci dépend du fait de partager la même ressource; dans ce cas, une règle de priorité doit être définie entre ces opérations. Le passage séquentiel des opérations sur les machines risque d'être bloqué si celles-ci n'admettent pas d'espace de stockage pour les opérations traitées, ainsi l'opération suivante ne peut commencer que si la première quitte cette ressource partagée, cette contrainte est appelée « contrainte de blocage ».

2.2. La contrainte de blocage

Soient **i et j** deux opérations concurrentes sur la même machine M(i)=M(j) et $\sigma(i)$, $\sigma(j)$ leurs successeurs directs, respectivement. Dans le cas où la machine M n'admet pas un espace de stockage, alors le passage séquentiel de j sur M dépend de la fin de traitement de i sur la même machine, pendant ce temps, j restera bloquée jusqu'à ce que i quitte M; cette situation est représentée en reliant $\sigma(i)$ avec j, $\sigma(j)$ avec i par une paire d'arcs alternatifs *u1 et u2* tels que : *u1*=(σ(i),j) et *u2*=(σ(j),i). Chaque arc alternatif u_l représente le fait que l'opération terminale ne peut commencer avant le début de l'opération initiale, ainsi, $t_{\sigma(i)} <= t_j$ et $t_{\sigma(j)} <= t_i$.
La pondération des arcs alternatifs est souvent nulle, l'exception est faite avec la dernière opération *k* de chaque travail qui n'admet pas de successeur, donc $\sigma(k)$ n'existe pas, dans ce cas là on relie directement ces sommets (opérations) avec leurs concurrents sur la même machine en utilisant des arcs alternatifs pondérés, cette pondérations est strictement positif, dont la valeur est égale à la durée de traitement p_k du sommet initial de chaque arc alternatif.

3. Etat de l'art

Depuis la première apparition du problème de job shop dans la littérature un grand effort a été réalisé pour la conception des algorithmes B&B. Parmi les travaux remarquables dans se domaine d'optimisation combinatoire on trouve :

L'algorithme de Carlier et Pinson (1989) qui résout le problème de complexité 10x10 proposé par Fisher et Thompson, ensuite, Pinson et Carlier introduisent en 1989, toujours, la possibilité de fixer les arcs alternatifs sans séparation, ce qui rend leur algorithme très efficace. La majorité des travaux qui sont arrivés par la suite sont basés sur leurs résultats.

Concernant l'approche parallèle on trouve l'implémentation parallèle de la méthode B&B proposé par Pergaard et Clausen. Cet algorithme basé sur les résultats de Carlier et Pinson et Brucher et al.

La contrainte de blocage dans l'ordonnancement d'atelier a intéressé beaucoup de chercheurs ces dernières années à cause de ces applications dans l'industrie, l'agriculture les hôpitaux…etc.

Le problème d'ordonnancement flow shop avec 3 machines est NP-difficile au sens fort (Reddi et Ramamorthy 1972). Pour le cas de deux machines, le problème peut être réduit en un cas spécial du problème de voyageur de commerce (PVC) et sa résolution en temps polynomial avec l'algorithme de Gilmore et Gomory.

Pour m>2 le problème flow shop est considéré comme NP-difficile, pour cela plusieurs heuristiques sont développées pour résoudre ce problème : Xianpeng Wang et Lixin Tang [3] ont proposé un algorithme de recherche tabou pour résoudre le problème flow shop hybride avec capacité de stockage limitée où chaque travail doit passer par N stages et chaque stage contient m_j machines parallèles, pour construire une solutions réalisable ils ont utilisé une procédure GCP (Greedy Constructive Procedure) qui considère les espaces de stockage entres deux stages successives comme des stages contenant des machines parallèles et les temps de traitement de chaque travail sur ces machines est nul.

Un algorithme PSO hybride utilisant la procédure NEH pour construire une solution initiale est proposé par Bo Liu, Ling Wang et Yi-Hui Jin pour résoudre le problème d'ordonnancement flow shop avec capacité de stockage limitée. P. Débora Ronconi et luis R.S Henriques [9] ont étudié la minimisation du retard total pour le problème d'ordonnancement flow shop avec contrainte de blocage, un algorithme (FPD : Fitting Processing timess and Due dates) et deux méta heuristiques GRASP (Greedy Randomized Adaptative Search Procedures) sont proposés pour trouver une solution approchée.

Jozef Grabowski et Jaroslaw Pempera [8] ont utilisé la notion des blocks et anti-blocks pour construire un algorithme de recherche tabou pour la résolution du problème d'ordonnancement flow shop avec contrainte de blocage.

Ling Wang, Liang Zhang et Da-Zhong Zheng [7] ont proposé un algorithme génétique hybride pour le problème d'ordonnancement flow shop avec capacité de stockage limitée pour construire une population initiale ils ont utilisé la méthode NEH, ils ont appliqué la procédure de recherche locale sur la population avec une probabilité $(1-p_m)$ où p_m est la probabilité de mutation.

S. Martinez, S. Dauzère-Pérès, C. Guéret, Y. Mati et N. Sauer [4] ont étudié la complexité du problème flow shop avec la nouvelle contrainte de blocage où la machine k reste bloquée jusqu'à ce que le travail commence sont traitement sur la machine k+2, ce problème a beaucoup d'applications dans l'industrie et l'agriculture et peut être modélisé par un graphe disjonctif.

A. Soukhal, A. Oulamara et P. Martineau [2] ont étudié la complexité du problème d'ordonnancement flow shop à deux machines avec contrainte de transport, les travaux traités sur les deux machines sont transportés par des camions de capacité limitée. Dans leurs articles ils ont montré la difficulté de quelques cas avec la contrainte de blocage entre les machines.

A. Soukhal et P. Martineau [5] proposent un modèle mathématique et un algorithme génétique pour résoudre le problème d'ordonnancement flow shop avec un rebot relié avec toutes les machines, le rebot peut traiter un seul travail à la fois.

Débora P. Ronconi [6] a proposé une heuristique constructive appelée MinMax pour résoudre le problème flow shop avec contrainte de blocage, cette heurstique est comparée avec d'autres algorithmes comme l'algorithme PF et l'algorithme NEH…

W. Henry, Thornton et John L. Hunsucker [10] ont proposé un algorithme appelé Nis-heuristique pour la résolution du problème d'ordonnancement flow shop avec processeurs multiple et indisponibilité d'espace de stockage, les travaux doivent passer par m stages et chaque stage contient M_j machines parallèles. L'algorithme est comparé avec d'autres heuristiques qui résolvent le même problème.

Vince Craffa, Stefano Lanes, Tapan P. Bagchi et Chelliah Striskandarajah [11] ont utilisé un algorithme génétique pour résoudre le problème d'ordonnancement flow shop avec contrainte de blocage. Ils ont utilisé le lien entre la contrainte de blocage et la contrainte de sans attente pour construire une fonction d'évaluation de la fitness des éléments générés par l'algorithme génétique.

Pour le problème job shop avec contrainte de blocage : Alessandro Mascis and Dario Pacciarelli [1], ont proposé une modélisation par graphe alternatif avec quelque propriétés sur cette catégorie des graphes qui est une généralisation des graphes disjonctifs; des méthodes heuristique est une autre par séparation et évaluation sont présentées dans cet article pour le problème d'ordonnancement job shop avec contraintes de blocage et de sans attente.

Yazid Mati, Nidhal Rezg et Xialon Xie [12] ont proposé une recherche tabou pour le problème d'ordonnancement job shop avec contrainte de blocage.

Heinz Gröflin et Andreas Klinkert [13] ont développé un algorithme recherche tabou pour le problème d'ordonnancement job shop généralisé avec contrainte de blocage avec prise en compte du temps d'installation et du temps de nettoyage.

4. Modélisation du problème

Le problème job shop traditionnel suppose que l'espace de stockage est disponible et illimité.

Dans la pratique il existe plusieurs exemples dans l'industrie où la capacité de stockage est limitée ou indisponible ce qui cause la contrainte de blocage entre deux opérations successives.

La machine qui traite l'opération d'un travail donnée reste bloquée après sa fin de traitement jusqu'à ce que l'opération prochaine de ce travail commence son traitement sur la machine suivante, cette contrainte est appelée contrainte de blocage.

4.1 Modélisation par graphe disjonctif

Notons $\sigma(o_i)$ l'opération qui suit o_i dans le même travail. Les deux figures ci-dessous présentent la différence entre les arcs disjonctifs d'un problème d'ordonnancement job shop classique et les arcs alternatifs d'un problème d'ordonnancement avec contrainte de blocage :

4.1.1. Arcs disjonctifs

Nous distinguons trois types de contraintes disjonctifs entre chaque couple d'opérations qui s'exécutent sur la même machine :

a/ Arcs disjonctifs classiques

Ce type de contraintes apparaît lorsque la machine contient un espace de stockage ou dans le cas de deux opérations terminales.

Dans ce cas, les deux opérations o_i et o_j qui doivent s'exécuter sur la même machine m_k sont reliées par deux arcs disjonctifs de o_i vers o_j et de o_j vers o_i ce qui signifie que l'opération o_j ne peut commencer qu'après la fin d'exécution de l'opération o_i sur la machine ou le contraire.

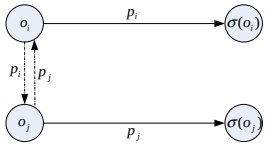


Figure 1. Arcs disjonctifs dans un job shop classique

b/ Arcs disjonctifs de blocage (arcs alternatifs)

Ce type de contraintes apparaît lorsque la machine m_k ne contient pas d'espace de stockage.

Dans ce cas, les deux opérations o_i et o_j sont reliées par deux arcs alternatifs de $\sigma(o_i)$ vers o_j et de $\sigma(o_j)$ vers o_i ce qui signifie que l'opération o_j ne peux commencer qu'après le début de l'opération $\sigma(o_i)$, ceci assure que la machine m_k est libre.

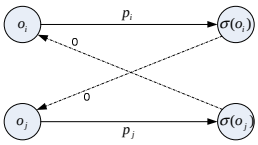


Figure 2. Arcs de blocage dans un job shop avec contrainte de blocage.

c/ Arcs croisés

Ce type de contraintes apparaît lorsque la machine m_k ne contient pas un espace de stockage et l'une des deux opérations est une opération terminale.

Soit o_i l'opération terminale : Dans ce cas, les deux opération o_i et o_j sont reliées par deux arcs alternatifs de o_i vers o_j et de $\sigma(o_j)$ vers o_i ce qui signifie que l'opération o_j ne peux commencer qu'après la fin d'exécution et de o_i, et o_j ne peux commencer qu'après le début d'exécution de l'opération $\sigma(o_i)$ ceci assure que la machine m_k est libre.

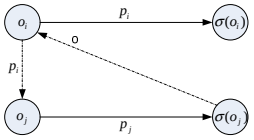
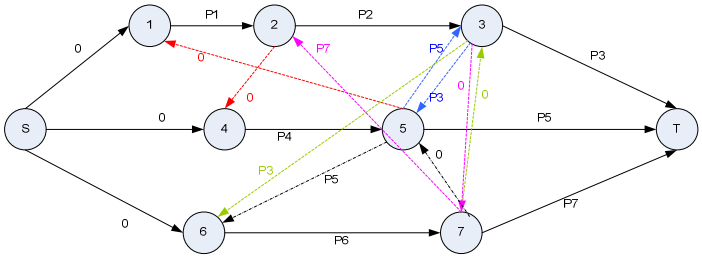


Figure 3. Arcs croisés dans un job shop avec contrainte de blocage

Exemple : Soit à ordonner 3 travaux j_1, j_2, j_3 sur 3 machines m_1, m_2, m_3

Travaux	La séquence des machines
1	1, 2,3
2	1,3
3	3, 2

Pour simplifier l'exemple nous considérons l'ensemble de toutes les opérations Numérotées de 1 à 7 :



S : est une tâche fictive qui représente la source, elle est liée à toutes les opérations qui sont prêtes à t=0.

T : est une tâche fictif qui représente le puits, elle est liée à toutes les opérations qui n'ont pas un successeur.

Remarque : nous rappelons que les poids des arcs altératifs sont nuls.

5. Résolution du problème posé

Dans cette partie nous décrivons la méthode exacte B&B que nous avons utilisé comme moyen de résolution; qui est un algorithme de création et parcourt d'un arbre n-aire « arbre de recherche », la création de l'arbre est assurée par le processus de séparation alors que le parcourt est assuré à chaque évaluation.

Dans la littérature plusieurs manières de parcourt sont citées : le parcourt en largeur, le parcourt en profondeur, le meilleur d'abord etc. On trouve même des heuristiques développées rien que pour la sélection d'un nœud parmi les autres.

5.1. La séparation

Le point de départ est un nœud qui représente la modélisation du problème. Nous avons évité de porter dans ce nœud toute information qui ne change pas pendant la séparation (les arcs conjonctifs, les données, …etc), cette manière de modéliser évite le gaspillage de l'espace mémoire; donc on a fait la différence entre deux types de données :

- Le problème initial : qui est fixe.
- Le sous-problème : représentant les contraintes sur les quelles le processus de séparation travaillera (les arcs alternatifs). Ce sont des données dynamiques qui changent avec le travail.

L'idée de séparation qu'on a utilisée se base sur l'énumération implicite de toutes les combinaisons possibles qu'on peut avoir en ordonnançant les travaux (opérations) concurrents sur chaque machine.

Le fait de séparé sur l'ordre entre les opérations donne un espace de recherche beaucoup plus petit (car lors du choix de l'opération qui doit passer en premier sur la machine concernée permet en réalité de fixer plusieurs arcs alternatifs en même temps au lieu d'un seul arc à fixer à chaque séparation, et nous évitons ainsi, tous les cas de non réalisabilité) que d'autres manières de séparation (exemple : la fixation pour chaque couple d'arcs alternatifs d'un seul arc dans les cas possibles entres les opérations concurrentes avec celle choisie). Les ordonnancements générés comportent plusieurs situations de violation des contraintes de précédences; pour cela une vérification de l'admissibilité doit se faire après chaque séparation.

Dans les problèmes de job shop idéales le circuit est un indicateur suffisant de non réalisabilité de la solution; mais dans notre cas on fait la différence entre un circuit nul est un circuit positif.

5.2. L'évaluation

Comme méthode d'évaluation nous avons utilisé l'algorithme de Bellman qui va donner à chaque opération sa date de commencement sur la machine qu'elle demande. La date de Fin d'exécution de la dernière opération représente la valeur de la fonction objectif notée C_{max}. Cette fonction ne sera appelée qu'après la mise à niveau du graphe avec une réduction (composante fortement connexe) s'il est nécessaire, assurant par ça un temps d'exécution très réduit.

Ainsi dans cette partie nous avons exploité l'algorithme de mise à niveau d'un graphe pour vérifier si le graphe n'admet pas de circuits. Dans le cas contraire on applique l'algorithme de forte connexité sur la partie restante du graphe où l'algorithme de mise à niveau c'est arrêté, la composante fortement connexe Cf détectée contiendra automatiquement un circuit, si celui-ci est de longueur positif alors on arrête la solution est non réalisable et nous stérilisons, par contre si la longueur du circuit est nulle, dans ce cas là, nous avons un cas de swapping, nous affectons, alors, à tous les sommets de la composante Cf la même pondération qui est le max des pondérations de x $\forall x \in$ Cf. Ensuite nous réduisons cette composante Cf en un seul sommet, et ainsi de suite, ceci nous permettra de garder le graphe sans circuit afin que nous puissions continuer à utiliser l'algorithme de Bellaman pour calculer la valeur du plus long chemin qui n'est rien d'autre que notre évaluation.

Une autre manière d'évaluer aussi, est l'algorithme de recherche de plus long chemin de Ford. Cet algorithme prend en considération les circuits d'une manière automatique contrairement à l'algorithme de Bellman.

5.3. La stérilisation

On stérilise dans trois situations :

Si le graphe contient un circuit positif : ce cas représente une solution non réalisable d'où il est inutile d'aller plus loin dans cette branche de l'arbre.

Le deuxième cas c'est la situation où l'évaluation du nœud représentant le sous problème donne une valeur de C_{max} supérieure ou égale à celle de la meilleure solution réalisable trouvée jusqu'à présent, qui est pour nous la borne sup.

Le troisième cas c'est la situation où la borne inférieure du sommet en cours dépasse la valeur de la borne sup.

5.4 Borne inférieure

Nous décrivons dans ce qui suit la borne inf. utilisée dans le cas d'un job shop classique, cette borne est basée sur la résolution des sous problèmes à une machine étudiée par Carlier.

a/ Le problème à une machine

Le problème à une machine est associé au problème job shop en choisissant une machine et relaxant les contraintes disjonctives concernant les autres machines.

Soit k un sous ensemble de l'ensemble de toutes les opérations O, k contient les opérations qui appartiennent à la même machine non relaxée.

Soient :

- r_i : La date de début au plutôt de l'opération i représentée par l(s,i) qui est la valeur du plus long chemin entre s et i.
- q_i : La valeur du plus long chemin de i vers T (le sommet terminal du graphe alternatif) noté aussi l(i,T).

$$H(k)= Min\{r_i / i \in k\} + \sum\{p_i / i \in k\} + Min\{q_i / i \in k\} \qquad 1$$

Proposition1 [1] : H(k) est une borne inférieure pour les makespan du problème de one-machine.

Proposition2 [1]: Soit V_k la valeur optimale pour le problème one-machine associé à la machine k.

LB= $Max\{V_k / k = 1, m\}$ est la borne inférieure pour le makespan du problème job shop.

Il est prouvé que LB= $Max\{H(k) / k \subseteq O\}$ peut être calculée en O(n log n).

Remarque : Nous avons constaté après application que dans le cas d'un job shop avec blocage cette borne inf. n'est vraiment pas très efficace. Ceci nous pousse à voir une autre borne plus efficace, en d'autre termes, la résolution des sous problèmes à une machine ne donne pas de bon résultats avec la contrainte de blocage car chaque opération ne dépend pas seulement de la machine actuelle mais dépend aussi de l'état de la machine suivante.

6. Discussion des résultats

Les résultats sont présentés dans un tableau de 6 colonnes :

La première colonne (nb_travaux x nb_machines) est la colonne de la taille du problème, ou nb_travaux est le nombre de travaux alors que nb_machines est le nombre des machines, nous rappelons que la complexité d'un problème de job shop de taille (nombre travaux x nombre machines) au pire des cas est donnée par la formule: (nombre travaux !)$^{\text{Nombre machines}}$ et elle représente la taille de l'espace de recherche (nombre de combinaisons possibles) ça en assurant que chaque ligne de données est complète i.e. que chaque travail doit demander toutes les machines.

Dans La deuxième colonne on a mis la valeur de C_{max} trouvée par le recuit simulé qu'on a développez et qui sert comme une borne supérieure d'entrée. La troisième colonne c'est le C_{max} trouvé par notre méthode de séparation et évaluation. Le temps de réponse se trouve dans la quatrième colonne, dans la 5eme colonne nous avons mis (Nb_noeuds) qui est le nombre des nœuds stérilisés puisque la fonction d'évaluation donne une valeur plus grande que la bonne solution trouvée jusqu'à présent. Le

nombre de nœuds non réalisables dans la colonne 6 c'est le nombre de nœuds stérilisés car ils portent un circuit positif, notant que la détection est immédiate à l'apparition du circuit grâce à l'algorithme de détection des composantes fortement connexes.

Nb_trav. x nb_mach.	Résultat recuit simulé (UB)	C_{max} Bellman	Cmax FORD	Temps (seconde) Bellman	Temps (seconde) FORD	Nb_noeuds LB>=UB Bellman	Nb_noeuds non réalisables Bellman
4x5	395	395	395	$188x10^{-3}$	0.031	81	105
4x10	633	633	633	4,875	0.516	332	321
4x15	957	955	955	25,578	0.531	259	429
5x10	778	681	681	77,250	7.28	1695	3031
5x15	994	-	923	-	2329,36	-	-
6x5	435	427	427	168,8	21.18	40951	119193
6x10	765	758	758	34920	1715.70	264224	343350
6x15	1247	1229	1229	26289.5	45020.96	8904	48667
7x5	679	638	638	2181	424.07	113484	416412
8x5	677	659	659	1778,4	12059.7	3347340	7516371
9x5	722	690	690	13932	2580	397858	1679816

Tableau 1. Résultats SEP avec Bellman et Ford.

Nous constatons d'après le tableau ci-dessus que l'évaluation de Ford prime sur l'évaluation de Belmann dans la majorité des exemples sauf à l'exemple 6x15 où Belmann prime sur Ford (en terme de temps d'exécution).

7. Conclusion

Dans ce papier nous avons présentez une méthode par séparation et évaluation permettant de résoudre le problème d'ordonnancement avec blocage, cette méthode utilise deux manières différentes d'évaluation, la première est basée sur l'algorithme de recherche de plus court chemin de Bellman utilisé avec les algorithmes de mise à niveau et des composantes fortement connexes, la deuxième utilise directement l'algorithme de recherche de plus long chemin dans un graphe de Ford. Une comparaison des deux méthodes est présentée, dans laquelle nous avons constaté que pour certains exemples Bellman prime pour d'autre c'est Ford qui prime. Une amélioration de notre travail peut être prise en compte en améliorant les bornes inf. et sup. du problème, ceci allègera la méthode pour des exemples un peu plus grands. Il serait aussi intéressant de voir l'effet de la parallélisation de cette SEP sur les tailles des problèmes à résoudre.

8. Références

1. A. Mascis and D. Pacciarelli. Job shop scheduling with blocking and no-wait constraints. European Journal of Operational Research 143, 498–517 (2001)

2. A. Soukhal, A. Oulamara and P. Martineau. Complexity of flow shop scheduling problems with transportation constraints. European Journal of Operational Research, 161, 32-41 (2005).

3. X. Wang and L. Tang. A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers. Computers & Operations Research, (In press).

4. S. Martinez, S. Dauzère-Pérès, C. Guéret, Y. Mati, and N. Sauer. Complexity of flowshop scheduling problems with a new blocking constraint. European Journal of Operational Research, 169, 855-864 (2006).

5. A. Soukhal and P. Martineau. Resolution of a scheduling problem in a flowshop robotic cell. European Journal of Operational Research, 161, 62-72 (2005).

6. D.P. Ronconi. A note on constructive heuristics for the flowshop problem with blocking. International Journal of Production Economics (2004).

7. L. Wang, L. Zhang and D. Zheng. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. Computers & Operations Research, 33, 2960-2971 (2006).

8. J. Grabowski and J. Pempera. The permutation flow shop problem with blocking: A tabu search approach. Omega, 35, 302-311 (2007).

9. D. P. Ronconi and L.R.S. Henriques. Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. Omega, (In press).

10. H.W. Thornton and J.L. Hunsucker. A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage. European Journal of Operational Research, 152, 96-114 (2004)

11. V. Caraffa, S. Ianes, T.P. Bagchi and C. Sriskandarajah. Minimizing makespan in a blocking flowshop using genetic algorithms. International Journal of Production Economics, 70, 101-115 (2001).

12. Y. Mati, N. Rezg and X. Xie. Scheduling Problem of Job-Shop with Blocking: A Taboo Search Approach MIC'2001 - 4th Metaheuristics International Conference, (2001).

13. H. Gröflin and A. Klinkert. A Tabu Search for the Generalized Blocking Job Shop MIC2005: The Sixth Metaheuristics International Conference, (2005).

14. R.Graham, Lawler E., Lenstra J.K., and Rinnooy Kan A., (1979).Optimization and approximation in deterministic sequenecing and scheduling theory: a survey. Annals of discrete mathematics, 5: pp.2 87-326.