Une amélioration de la méthode exacte pour trouver le meilleur des cas dans un ordonnancement de groupes

Zakaria Yahouni, Nasser Mebarki, Zaki Sari

Une amélioration De La Méthode Exacte pour Trouver Le Meilleur Des Cas Dans Un ordonnancement De Groupes

Zakaria YAHOUNI[a,b] , Nasser MEBARKI[a] , Zaki SARI[b]

[a] *LUNAM, Université de Nantes, IRCCyN, Institut de Recherche en Communications et Cybernétique de Nantes, 44300 France*
[b] *MELT, Manufacturing Engineering Laboratory of Tlemcen, 13000, Algérie*

Abstract

Le problème d'ordonnancement d'atelier à cheminement multiple est NP-difficile. Il est résolu généralement par des méthodes dites prédictives comme l'optimisation discrète qui cherche à trouver une solution faisable qui satisfait les contraintes du problème donné. Ce type de problème peut être aussi résolu par les méthodes réactives comme les méthodes de contrôle en temps réel, ces méthodes construisent progressivement la solution en temps réel. Les méthodes proactives-réactives combine les deux avantages des deux méthodes précédentes (i.e., une meilleure performance ainsi une réactivité en temps réel). L'ordonnancement de groupes est une des méthodes proactives-réactives la plus étudiée dans la littérature. Le but de cette méthode est de fournir une flexibilité séquentielle durant l'exécution de l'ordonnancement, elle garantit une qualité minimale qui correspond à la qualité du pire ordonnancement possible. Le meilleur ordonnancement d'un ordonnancement de groupes a été aussi abordé par [1] en utilisant un algorithme exact de séparation et d'évaluation (Branch and Bound) pour les objectifs réguliers.
Dans cet article on s'intéresse à la qualité du meilleur des cas d'un ordonnancement de groupes. Nous proposons une amélioration de l'algorithme exact de séparation et d'évaluation pour les objectifs réguliers. Deux méthodes de séparation ont été proposées et expérimentées sur des instances utilisées comme benchmark dans la littérature du *job shop*. Les résultats ont montré l'efficacité desdeux méthodes proposées.

Keywords: Ordonnancement de Groupes; Job shop; Méthodes proactives-réactives; Séparation et évaluation ; Makespan.

Nomenclature

O_i	Opération i ou tâche i.
r_i	Date de disponibilité d'une tâche.
t_i	Date de début d'une tâche.
p_i	Temps d'exécution d'une tâche.
C_i	Date de fin d'une tâche.
Γ-, Γ+	Prédécesseur et successeur d'une tâche.
M_k	Machine k.
C_{max}	MaxC_i : représente la durée totale de l'ordonnancement (*makespan*).
$G_{l,k}$	Groupe d'opérations permutables qui s'exécute en $l$ème position sur M_k.
θ_i	Borne inférieure de la date de début d'une tâche.
X_i	Borne inférieure de la date de fin d'une tâche.
$\gamma_{gl,k}$	Borne inférieure de la date de fin du groupe $G_{l,k}$.

1. Introduction

Le problème du *job shop* avec contraintes de précédence multiples ($J / r_i , \Gamma^- / f$ d'après la classification de [2]) est un problème d'optimisation combinatoire composé de ressources,

d'opération et de contraintes. Les opérations (O_i) sont exécutées sur les ressources M_k, (aussi appelées machines) pendant un temps d'exécution p_i avec des contraintes de précédences (les prédécesseurs (resp. successeur) de O_i sont données par Γ^- (resp. Γ^+)). Une ressource ne peut exécuter qu'une opération à la fois. Une opération a une date de disponibilité r_i, sa date de début est notée t_i et sa date de fin est notée C_i.

Généralement, le *job shop* utilise une fonction objective régulière qui est une fonction monotone des C_i. Le *makespan*, noté C_{max}, qui correspond au temps total de l'exécution de l'ordonnancement, est un objectif régulier classique. Dans cet article, nous nous intéressons exclusivement à cet objectif.

En pratique, les problèmes d'ordonnancement sont souvent soumis à des incertitudes sur les données : incertitudes sur les durées et les dates de disponibilité des opérations, opérations urgentes ou imprévues à réaliser, disponibilité incertaine des ressources [3]. Pour pallier ces incertitudes, une méthode d'ordonnancement reposant sur l'introduction de flexibilité séquentielle dans les opérations a été proposée [4]. Ceci est réalisé en déterminant, plutôt qu'un ordonnancement unique, un ordonnancement de groupes, c'est-à-dire en définissant une séquence de groupes d'opérations permutables sur chaque machine. L'évaluation d'un ordonnancement de groupes s'obtient par rapport à la qualité dans le pire des cas de l'ensemble d'ordonnancement réalisable. Cette approche a été largement étudiée au cours des dernières années [5; 6; 7; 8; 9; 10; 11; 1; 12]**.**

Mais la qualité dans le meilleur des cas d'un ordonnancement de groupes peut également être intéressante. Elle donne des informations sur l'ordonnancement avant son exécution. Elle peut également être utile pour évaluer une décision pendant l'exécution de l'ordonnancement. [1] propose une approche basée sur une procédure de séparation et d'évaluation (branch and bound) pour calculer de manière optimale le meilleur des cas d'un ordonnancement de groupes. Cette procédure utilise des bornes inférieures calculées en temps polynomial et décrites dans [12].

Dans cet article, nous proposons deux nouvelles méthodes de séparation pour la recherche du meilleur des cas. Les résultats obtenus montrent l'efficacité de ces deux nouvelles méthodes par rapport à la méthode proposée dans [1].

Ce papier est organisé comme suit : la section suivante donne une courte description de l'ordonnancement de groupes, la troisième section définit la méthode de séparation et d'évaluation, dans la section 4 nous présentons notre contribution. La section 5 décrit les expérimentations réalisées sur des instances utilisées comme benchmark dans la littérature *du job shop*. La dernière section conclut cet article.

2. L'ordonnancement de groupes

L'ordonnancement de groupes a été introduit par le laboratoire LAAS-CNRS de Toulouse, [4], cette approche est utilisée dans le progiciel ORDO. L'objectif de cette méthode est de fournir au décideur de la flexibilité séquentielle lors de l'exécution de l'ordonnancement et d'assurer une certaine qualité mesurée par le pire des cas.

L'ordonnancement de groupes est un ensemble de groupes noté $G_{l,k}$ (k représente l'indice de la machine, et l représente l'indice du groupe dans cette machine), chaque groupe est composé d'un ou plusieurs opérations à exécuter sur une certaine machine dans un ordre arbitraire $G_{l,k} = \{O_1 , O_2 , \ldots, O_n\}$.Le groupe contenant l'opération O_i est noté aussi $G(i)$. n est le nombre d'opération de ce groupe, le nombre de permutations possibles qui peuvent être obtenus à partir de ce groupe est égale à $n!$. Un ordonnancement de groupes est dit réalisable s'il satisfait toutes les contraintes du problème donné. En fait, une séquence de groupes décrit un ensemble d'ordonnancements valides sans les énumérer. La qualité d'un ordonnancement de groupes est mesurée par le pire ordonnancement semi-actif qui peut être obtenu [5].

L'exemple suivant permet d'illustrer cette définition :

Tableau 1. Exemple d'un problème de *Job Shop*

O_i	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}
M_k	M_2	M_3	M_1	M_3	M_2	M_1	M_3	M_1	M_2	M_1	M_2	M_3
Γ^-	...	O_1	O_2	...	O_4	O_5	...	O_7	O_8	...	O_{10}	O_{11}
p_i	4	1	5	4	1	1	3	3	2	1	1	5

Le tableau 1 représente un problème de *Job Shop* avec 4 jobs et 3 machines, la figure 1 représente un ordonnancement de groupes réalisable qui résout ce problème, les axes x et y de la figure 1 représente respectivement le temps et le numéro de machine. Cet ordonnancement de groupes est composé de 3 groupes de deux opérations et 6 groupes d'une seule opération. La figure 2 présente les ordonnancements semi-actifs qui peuvent être générées à partir de cet ordonnancement de groupes. Les ordonnancements semi-actifs générés donnent différentes valeurs pour la qualité dans le meilleur et dans le pire des cas, avec un C_{max} =15 pour le meilleur des cas et un C_{max}=18 pour le pire des cas (Figure 2). L'exécution de cet ordonnancement de groupes consiste à séquencer les groupes avec plus d'une opération. Dans notre exemple, trois décisions doivent être prises : séquencer l'opération 6 avant ou après l'opération 10, séquencer l'opération 7 avant ou après l'opération 2, séquencer l'opération 3 avant ou après l'opération 8.

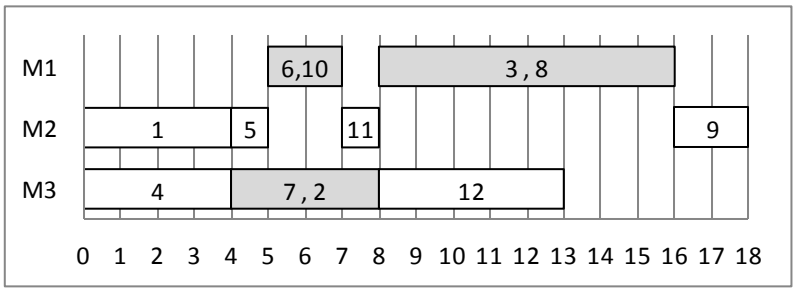
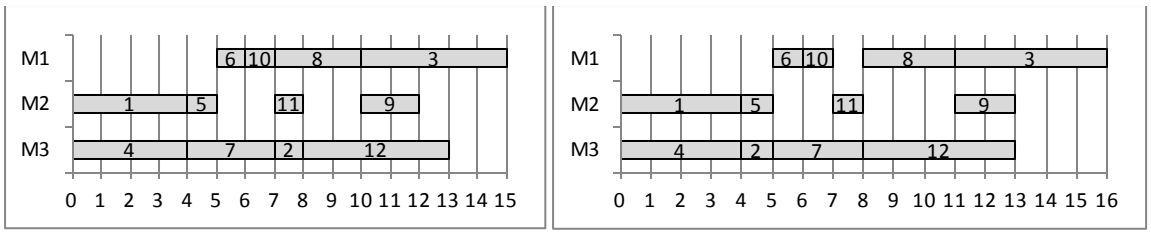


Fig. 1. Ordonnancement de groupes

L'ordonnancement de groupes a une propriété très intéressante ; le pire des cas est calculé en temps polynomial [8; 13; 14]. Il est donc possible de calculer le pire des cas d'un ordonnancement de groupes pour des problèmes de grande taille en temps réel.

[1; 10] ont montré l'intérêt du meilleur des cas comme indicateur d'aide à la décision pour choisir l'opération à séquencer dans un groupe composé de plusieurs opérations.



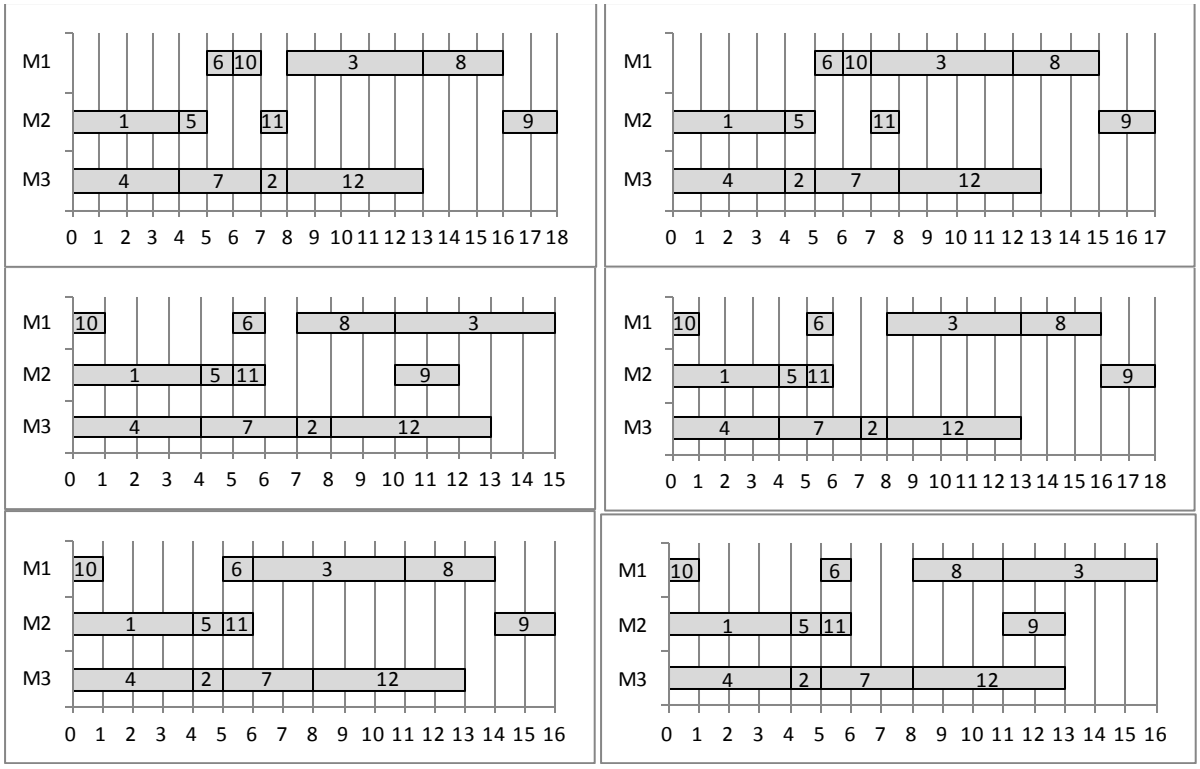


Fig. 2. Enumération des ordonnancements semi-actifs

3. La méthode de séparation et d'évaluation pour le meilleur des cas dans un ordonnancement de groupes

[1] propose une méthode de séparation et d'évaluation pour le meilleur des cas dans un ordonnancement de groupes en se basant sur des bornes inférieures du meilleur des cas de cet ordonnancement de groupes.

3.1 Calcul des bornes inférieures

La borne inférieure pour le meilleur des cas est d'une complexité polynomiale. Elle repose sur une relaxation des ressources en prenant l'hypothèse que chaque ressource a une capacité infinie. Dans ce cas, la borne inférieure du meilleur des cas pour la date de début d'une opération O_i est donnée comme le maximum de la borne inférieure de la date de fin de tous ses prédécesseurs ; pour une opération O_i, ses prédécesseurs incluent les prédécesseurs de cette tâche (Γ^-) ainsi que le groupe prédécesseur de cette tâche sur la même machine, noté $G^-(i)$. En effet, une opération d'un groupe donné ne peut pas être exécutée avant la fin de toutes les opérations du groupe prédécesseur sur la même machine. Par conséquent, une opération ne pourra commencer au plus tôt qu'après le *makespan* optimal du groupe prédécesseur. [12] propose le calcul des bornes inférieures comme suit :

$$\begin{cases} \theta_i = \max(r_i, \gamma_{G^-(i)}, \max\limits_{j \in \Gamma^-} \chi_i) \\ x_i = \theta_i + p_i \\ \gamma_{g_{l,k}} = C_{\max} of 1 \,|\, r_i \,|\, C_{\max}, \forall o_i \in G_{l,k}, r_i = \theta_i \end{cases}$$

Dans le tableau 2 nous présentons les bornes inférieures du problème décrit dans le tableau1 et la figure1 :

Tableau 2. Calcul des bornes inférieures

O_i	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}
θ_i	0	4	6	0	4	5	4	7	10	0	5	8
χ_i	4	5	11	4	5	6	7	10	12	1	6	13
γ_{g_i}	4	8	14	4	5	6	8	14	12	6	6	13

Ces bornes inférieures sont utilisées dans un algorithme de séparation et d'évaluation qui permet de calculer le meilleur des cas d'un ordonnancement de groupes.

3.2 L'algorithme de séparation et d'évaluation

[1] propose une condition suffisante pour diminuer l'espace de recherche tout en conservant la solution optimale. Une séquence d'opération d'un groupe est choisie comme solution dominante de toutes les séquences possibles de ce groupe si :

- Le séquencement ne dégrade pas la fonction objective.
- Le séquencement n'interfère pas avec les dates de début au plus tôt des autres opérations non séquencées.

Un nœud de l'espace de recherche est représenté par une séquence d'opérations dans le même groupe. Une solution est un ordonnancement de groupes avec une seule opération par groupe.

4. Amélioration de la méthode de séparation pour l'algorithme exacte

La procédure de séparation (Branching) génère des nœuds, mais l'ordre d'exploration de ces nœuds influe sur les performances de l'algorithme : le plus tôt le meilleur des cas est trouvé, meilleure sera la borne supérieure, ce qui réduira l'espace de recherche et par voie de conséquence le temps de résolution.

[1] ordonnance les groupes selon l'ordre de précédence entre les opérations de ces groupes. Si aucune relation de précédence n'est trouvée entre deux groupes, priorité est donnée au groupe qui a la plus petite date de début. Dans la suite, cette approche sera notée *PredOrder*.

Dans cet article, nous proposons deux nouvelles méthodes de séparation basées sur l'ordre du traitement des groupes. Les groupes composés de plusieurs opérations sont appelés des voisins. Dans ces deux nouvelles approches , priorité est donnée au groupe ayant le moins de relations avec les autres voisins.

Dans la première approche proposée, le nombre de relations d'un groupe donné est calculé selon les relations de précédence (directe et indirecte) ou de succession (directe et indirecte) des opérations de ce groupe avec les opérations de tous les autres groupes. Cette approche est notée *NeighborIndirectRel1* et est décrite dans l'algorithme suivant :

L(G) :={G_1,G_2,...,G_n} (G_i est un groupe contenant plus d'une opération) ;
Pour chaque groupe G_i de *L(G)* **faire**
    List_de_Relations(G_i) :={} ;
    Pour chaque opération O_i de G_i **faire**
        J_i représente le travail (*Job*) de l'opération O_i;
        Pour chaque opération O'_i de J_i tel que O'_i différente de O_i **faire**
            G'_i est le groupe de l'opération O'_i ;
            Si (G'_i appartient à *L(G)*) et (G'_i not in *List_de_Relations(G_i)*)
                Ajouter G'_i à *List_de_Relations(G_i)* ;
            Fin si ;
        Fin pour ;

Fin pour ;
Fin pour ;
Ordonnancer les groupes selon la cardinalité des *List_de_Relations(G_i)* ;

Pour le *Job Shop* présenté en exemple, l'ordre des groupes obtenus avec *PredOrder* et *NeighborIndirectRel1* sera comme suit :
PredOrder : $(O_7, O_2), (O_6, O_{10}), (O_3, O_8)$.
NeighborIndirectRel1 : $(O_6, O_{10}), (O_7, O_2), (O_3, O_8)$.

La deuxième approche proposée constitue une variante de la méthode précédente, et est notée *NeighborIndirectRel2*. Cette méthode ordonnance les groupes selon l'algorithme de NeighborIndirectRel1, mais si la condition suffisante décrite dans la section 3 est satisfaite dans un groupe donné, alors priorité est donnée à ce groupe sans poursuivre l'algorithme.

5. Expérimentation

Nous utilisons un ensemble d'instance très utilisée dans la littérature du *job shop*, nommé la01 à la40, dites instances de Lawrence [15]. Ce sont des instances de *job shop* classiques, avec *m* opérations pour chaque travail (*m* le nombre de machines), chaque opération d'un travail s'exécutant sur une machine différente. Cet ensemble est composé de 40 instances de tailles différentes (5 instances pour chaque taille).

Pour chaque instance, nous générons un ordonnancement de groupes avec une solution optimale connue et une borne de qualité dans le pire des cas représentée dans l'équation (2). Pour générer ces ordonnancements, nous utilisons un algorithme glouton qui fusionne deux groupes successeurs en fonction de différents critères jusqu'à ce qu'il n'y ait plus de fusion de groupes possible. Cet algorithme commence avec un ordonnancement de groupes à une opération par groupe calculé par l'algorithme exact décrit dans [13] (donc, par construction, le *makessan* optimal de ces ordonnancements de groupes est le *makespan* de cet ordonnancement de groupes à une opération par groupe). L'algorithme glouton est décrit dans [8].

$$\text{Le pire des cas} \leq \text{solution initiale} * (1+\Delta)\ (0\% \leqslant \Delta \leqslant 100\%) \qquad (2)$$

La technique de recherche en profondeur [14] est utilisée pour la méthode de séparation et d'évaluation. Cette méthode va directement à une solution où les nœuds sont traités dans un ordre de gauche à droite. Dans ce mode, il est très important d'ordonnancer les sous nœuds selon un ordre croissant de leurs bornes inférieures, cela permettra de trouver le meilleur des cas le plus tôt possible.

Une comparaison des trois méthodes présentées dans cet article (*PredOrder*, *NeighborsIndirectRel1*, *NeighborsIndirectRel2*) est présentée dans la section suivante. Les expérimentations ont été faites sur un Intel(R) Core™ i5 CPU 2.53GHZ.

6. Résultats

Cette section décrits les résultats obtenus. Les variables Ni utilisées dans les tableaux des résultats sont définies comme suit :
N1 : La flexibilité obtenue de l'ordonnancement de groupes ((nbre d'opérations – nbre de groupes) / (nbre d'opérations – nbre de machines)).
N2 : La borne inférieure initiale.
N3 : Le meilleur des cas obtenu.
N4 : L'écart par rapport au meilleur résultat obtenu pour le nombre de nœuds parcourus pour trouver le meilleur des cas :

- $nbBestCase1_i$ est le nombre de nœuds parcourus pour trouver le meilleur des cas en utilisant la méthode i (i = 1..3).

$$N4 = nbBestCase1_{\text{méthode courante}} - \text{Min}\ (nbBestCase1_i)\ (i=1...3)$$

N5 : Le temps en milli-secondes pour trouver le meilleur des cas.

N6 : L'écart par rapport au meilleur résultat obtenu pour le nombre de nœuds parcourus pour prouver l'optimalité du meilleur des cas.

- $nbBestCase2_i$ est le nombre de nœuds parcourus pour prouver l'optimalité du meilleur des cas.

$$N6 = (nbBestCase1 + nbBestCase2) - Min\ (nbBestCase1_i + nbBestCase2_i)\ (i=1...3)$$

N7 : Le temps pour trouver le meilleur des cas ainsi prouver son optimalité. Equation

Tableau 3 Résultats des trois méthodes

Δ = 10%				PredOrder				NeighborIndirectRel1				NeighborIndirectRel2			
	N1	N2	N3	N4	N5	N6	N7	N4	N5	N6	N7	N4	N5	N6	N7
La01	44,44	650	650	0	0,025	0	0,025	0	0,03	0	0,03	0	0,031	0	0,031
La02	22,22	655	655	0	0,005	0	0,005	0	0,006	0	0,006	0	0,006	0	0,006
La03	33,33	588	597	0	0,008	0	0,008	0	0,008	0	0,008	0	0,01	0	0,01
La04	33,33	588	590	0	0,013	**1**	0,014	0	0,015	0	0,015	0	0,018	0	0,018
La05	46,67	593	593	0	0,011	0	0,011	0	0,014	0	0,014	0	0,015	0	0,015
La06	51,43	926	926	0	0,038	0	0,038	0	0,038	0	0,038	0	0,049	0	0,049
La07	42,86	890	890	0	0,026	0	0,026	0	0,046	0	0,046	0	0,035	0	0,035
La08	45,71	863	863	0	0,101	0	0,101	0	0,117	0	0,117	0	0,108	0	0,108
La09	48,57	951	951	0	0,03	0	0,03	0	0,041	0	0,041	0	0,043	0	0,043
La10	54,29	958	958	0	0,118	0	0,118	0	0,138	0	0,138	0	0,139	0	0,139
La11	62,11	1222	1222	0	0,249	0	0,249	0	0,296	0	0,296	0	0,298	0	0,298
La12	55,79	1039	1039	0	0,141	0	0,141	0	0,151	0	0,151	0	0,167	0	0,167
La13	55,79	1150	1150	0	0,16	0	0,16	0	0,176	0	0,176	0	0,194	0	0,194
La14	67,37	1292	1292	0	0,286	0	0,286	0	0,349	0	0,349	0	0,352	0	0,352
La15	51,58	1207	1207	0	0,095	0	0,095	0	0,103	0	0,103	0	0,125	0	0,125
La16	22,22	945	945	0	0,019	0	0,019	0	0,022	0	0,022	0	0,03	0	0,03
La17	22,22	761	784	0	0,019	**550**	0,962	0	0,02	0	0,285	0	0,029	**67**	0,357
La18	21,11	848	848	0	0,019	0	0,019	0	0,018	0	0,018	0	0,025	0	0,025
La19	16,67	842	842	0	0,017	0	0,017	0	0,017	0	0,017	0	0,022	0	0,022
La20	16,67	901	902	**3**	0,018	0	0,018	0	0,016	**103**	0,189	0	0,021	**66**	0,108
La21	23,57	1046	1046	0	0,061	0	0,061	0	0,072	0	0,072	0	0,101	0	0,101
La22	22,86	927	927	0	0,074	0	0,074	0	0,075	0	0,075	0	0,1	0	0,1
La23	21,43	1032	1032	0	0,064	0	0,064	0	0,069	0	0,069	0	0,094	0	0,094
La24	21,43	934	935	0	0,067	0	0,067	0	0,065	**15**	0,096	0	0,087	0	0,087
La25	22,86	976	977	0	0,061	**2800**	6,063	0	0,071	0	0,128	0	0,101	**11**	0,224
La26	30,53	1218	1218	0	0,245	0	0,245	0	0,3	**102**	0,609	0	0,361	0	0,361
La27	26,84	1252	1252	0	0,176	0	0,176	0	0,195	0	0,195	0	0,3	0	0,3
La28	31,05	1273	1273	0	0,184	0	0,184	0	0,229	0	0,229	0	0,373	0	0,373
La29	28,42	1196	1202	0	0,175	**420**	2,47	0	0,205	0	2,02	0	0,31	0	3,281
La30	27,89	1355	1355	0	0,219	0	0,219	0	0,241	0	0,241	0	0,317	0	0,317
La31	46,55	1784	1784	0	4,646	0	4,646	0	6,049	0	6,049	0	6,013	0	6,013
La32	48,97	1850	1850	0	1,221	0	1,221	0	2,084	0	2,084	0	2,83	0	2,83
La33	43,45	1719	1719	0	1,046	0	1,046	0	1,61	0	1,61	0	1,98	0	1,98
La34	42,41	1721	1721	0	1,053	0	1,053	0	1,373	0	1,373	0	2,067	0	2,067
La35	42,76	1888	1888	0	2,533	0	2,533	0	1,366	0	1,367	0	1,87	0	1,87
La36	18,57	1267	1268	0	0,345	**2632**	23,096	0	0,165	0	0,185	0	0,234	**61**	0,672
La37	18,10	1395	1397	0	0,325	0	0,325	0	0,147	**12**	0,195	0	0,323	0	0,323
La38	17,14	1196	1196	0	0,287	0	0,287	0	0,129	0	0,129	0	0,195	0	0,195
La39	16,19	1232	1233	0	0,304	**317**	3,175	0	0,132	0	0,17	0	0,216	0	0,276
La40	14,76	1222	1222	0	0,11	0	0,11	0	0,107	0	0,107	0	0,147	0	0,147

Les trois méthodes trouvent le meilleur des cas en moins d'une seconde dans la plupart des instances : 77.5% des instances pour *PredOrder*, 85% pour *NeighborIndirectRel1* et *NeighborIndirectRel2*.

La moyenne du temps d'exécution de l'algorithme exact de *PredOrder* pour toutes les instances est de 1.23 secondes, 0.47 secondes pour *NeighborIndirectRel1* et 0.59 secondes pour *NeighborIndirectRel2*.

Lorsqu'on augmente la flexibilité de l'ordonnancement de groupe, pour $\Delta = 15\%$, le temps moyen pour résoudre toutes les instances avec *PredOrder* est de 15.04 secondes, 29.9 secondes pour *NeighborIndirectRel1*, à cause de l'instance La29 qui a été résolu en 1164 secondes, 8.77 secondes pour *NeighborIndirectRel2*. Le nombre de nœuds parcourus pour les trois méthodes, avec $\Delta = 15\%$, est représenté figure 3.

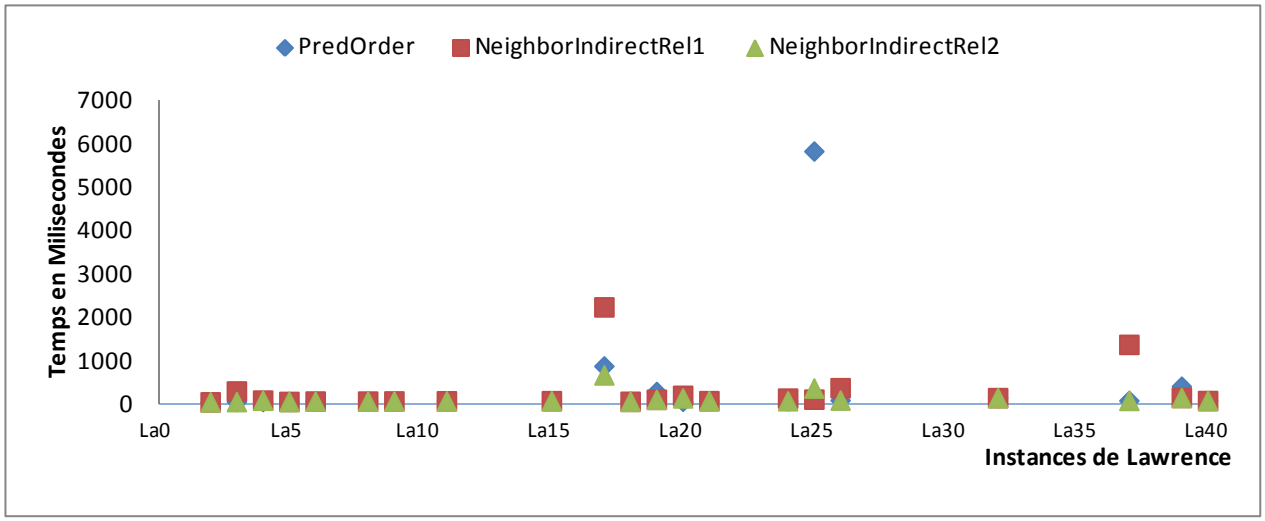


Fig. 3. Comparaison du nombre de nœuds parcouru pour chaque méthode pour terminer l'algorithme de séparation et d'évaluation ($\Delta = 15\%$)

Pour toutes les instances, au moins une des deux nouvelles méthodes proposées domine les résultats pour N6 (écart par rapport au meilleur résultat obtenu pour prouver l'optimalité du meilleur des cas) à l'exception de l'instance La20. Pour cette variable, les deux méthodes proposées ont donné de meilleurs résultats par rapport à la méthode *PredOrder* pour la plupart des instances. Par exemple pour La25, le nombre de nœuds parcourus pour terminer l'algorithme pour *PredOrder* est de 2849, pour *NeighborIndirectRel1* il est de 49 et pour *NeighborIndirectRel2* de 60. Pour la méthode *PredOrder*, la moyenne de N6 calculée sur toutes les instances est de 221.8, ce qui est environ plus de trois fois supérieur aux moyennes de N6 calculés sur les méthodes *NeighborIndirectRel1* et *NeighborIndirectRel2* (respectivement 59.6 et 58.92).

Cette amélioration des résultats est dû au fait que le nombre de nœuds à traiter pour prouver l'optimalité de la solution obtenue sera plus faible si le nombre des premiers sous-nœuds de l'arbre de la méthode exacte est plus petit. En effet, aux premiers niveaux de l'arbre, les bornes inférieures proposés par [12] ne sont pas très précises. Même si la solution optimale est trouvée plus tôt, les nœuds sur la droite de l'arbre doivent être traités afin de prouver l'optimalité de cette solution. Avec la méthode *NeighborIndirectRel1*, les nœuds ayant le plus grand nombre de relations avec les autres groupes sont traités à la fin tandis que ceux ayant le plus petit nombre de relations sont traités en premier. Cela réduira la largeur de l'arbre dans les premiers niveaux, où les bornes inférieures sont moins précises. Par conséquent, cela réduira l'espace de recherche pour prouver l'optimalité de la meilleure solution trouvée, cette propriété est illustrée dans la figure suivante :

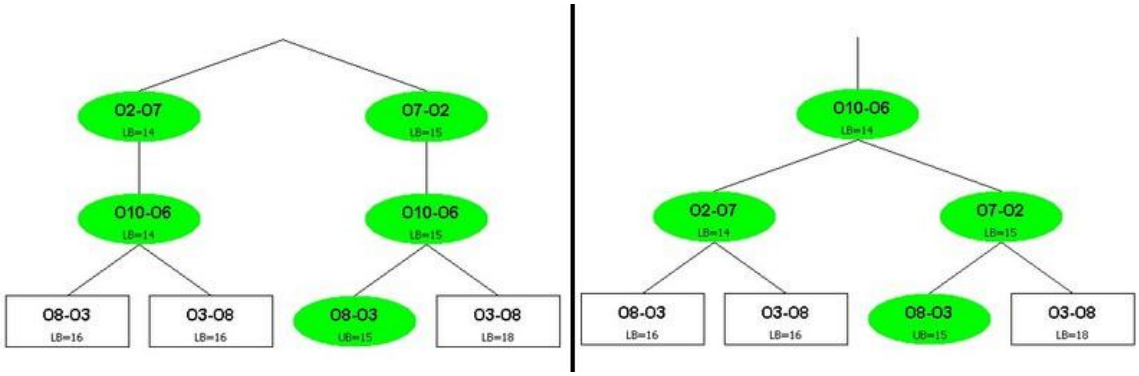


Fig. 4. Branch and Bound pour PredOrder et NeighborsInderectRel1 resp.

La figure 4 montre les arbres de résolution de notre exemple avec *PredOrder* et *NeighborsIndirerctRel1*. La méthode de recherche en profondeur est utilisée avec une borne supérieure initiale égale à 15 (qui est une solution optimale). *PredOrder* trouvera la solution optimale après la visite de cinq nœuds, par contre *NeighborsIndirectRel1* trouvera la solution optimale après avoir visité quatre nœuds seulement. Cette amélioration des résultats est due à la largeur de l'arbre au premier niveau ; le nombre des premiers sous nœuds de l'arbre de *PredOrder* est supérieur a celui de *NeighborsIndirectRel1*. Cette amélioration de la méthode de séparation a réduit l'espace de recherche pour trouver (et prouver) la solution optimale.

7. Conclusion

Cet article traite le problème du meilleur des cas dans un ordonnancement de groupes pour le *makespan*. Cet indicateur permet d'aider le décideur d'un système de coopération homme-machine à choisir l'opération à exécuter dans l'atelier en lui offrant différent paramètres relatives à son choix comme le pire des cas et le meilleur des cas. [1] a proposé une résolution du meilleur des cas pour les objectifs réguliers en utilisant un algorithme exact de séparation et d'évaluation. Le processus de séparation et donc le temps de résolution dépend de l'ordre de traitement des groupes. Nous avons présenté dans ce papier deux nouvelles approches de séparation afin de résoudre le problème du meilleur des cas de manière plus efficace pour les objectifs réguliers.

Des expérimentations ont été faites sur des instances de Lawrence utilisées comme benchmak dans la littérature du *job shop*, les résultats obtenus ont montré l'efficacité des deux méthodes proposées.

Un futur travail consiste à étudier l'impact de la flexibilité d'un ordonnancement de groupes sur les méthodes de séparation proposés.

Acknowledgment

Ce travail a été réalisé dans le cadre du projet Européen « EU-METALIC ERASUS MUNDUS »

"This work has been funded with support of the European Commission. This communication reflects the view only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. "

Références

[1]. **Pinot, G. and Mebarki, N.** , 2009, *An Exact Method for the Best Case in Group Sequence: Application to a General Shop Problem*. Moscow, Russia : 13th IFAC Symposium on Information Control Problems in Manufacturing.
[2]. Graham, Lawler ,Lenstra ,Rinnooy Kan , 1979, *Optimization and approximation in deterministic sequencing and scheduling: a survey*. s.l. : Annals of Discrete Mathematics,5, 287-326.
[3]. Artigues, C. 2004, *Optimisation et robustesse en ordonnancement sous contraintes de ressources.* Avignon. Habilitation à diriger des recherches.
[4]. Erschler and Roubellat. 1989, *An approach for real time scheduling for activities with time and resource constraints*. s.l. : In Slowinski, R. and Weglarz, J., editors, Advances in project scheduling. Elsevier.
[5]. ALLOULOU, M.A. and Artigues, C. 2007, *Worst-Case Evaluation of Flexible Solutions in Disjunctive Scheduling Problems*. Malaysia : Gervasi, Osvaldo; Gavrilova, Marina, August 26-29, 2007, Computational Science and Its Applications - ICCSA 2007 International Conference, Kuala Lumpur, , August 26-29. Proceedings, Part III, pp. 1027-1036.
[6]. Artigues, C. and Billaut, J.C. and Esswein, C. 2005, *Maximization of solution flexibility for robust shop scheduling*. European Journal of Operational Research, Vol. 165, pp. 314-328.
[7]. Cardin, O. and Mebarki, N. and Guillaume, P. 2013,*A study of the robustness of the group scheduling method using an emulation of a complex FMS*. International Journal of Production Economics, Vol. 146, pp. 199-207.
[8]. Esswein, C. 2003, *Un apport de flexibilité séquentielle pour l'ordonnancement robuste*. Université François Rabelais Tours (France).
[9]. Logendran, R. and Sara CARSON and Erik HANSON. 2005, *Group scheduling in flexible flow shops*. International Journal of Production Economics, Vol. 96, pp. 143-155.
[10]. Mebarki, N. Cardin, O. Clément, G. 2013 **,** *Evaluation of a New Human-Machine Decision Support System for Group*. Las Vegas, : Analysis, Design, and Evaluation of Human-Machine Systems, August 13-15, Vol. 12, pp. 211-217.
[11]. Pinot, G. and Cardin, O. and Mebarki, N. 2007, *A study on the group sequencing method in regards with transportation in an industrial FMS*. MONTREAL - CANADA : The IEEE SMC 2007 International Conference.
[12]. Pinot, G. and Mebarki, N. 2008, *Best-case lower bounds in a group sequence for the job shop problem*. Seoul, Korea : 17th IFAC World Congress.
[13]. Brucker, P. and Jurisch, B. and Sievers, B. 1994, *A branch and bound algorithm for the job-shop scheduling problem*. Discrete Applied Mathematics, Vol. 49, pp. 107-127.
[14]. Brusco, M.J. and Stahl, S. 2005, *Branch-and-Bound Applications in Combinatorial Data Analysis*. s.l. : Springer New York. Vol. 221.
[15]. **Lawrence, S.** 1984, *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*. Carnegie-Mellon University, Pittsburgh, Pennsylvania : s.n..