7e Conférence Francophone de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 - Paris - France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination ».

# UNE RECHERCHE TABOU ET UN ALGORITHME GÉNÉTIQUE POUR UN PROBLÈME DE JOB SHOP MULTIRESSOURCE MULTICRITÈRE

**G. VILCOT, Y. KERGOSIEN, J. JANVIER, J.-C. BILLAUT**

Laboratoire d'Informatique de l'Université François-Rabelais de Tours
Département Informatique de Polytech'Tours
64 av Jean Portalis, 37200 Tours, France
{geoffrey.vilcot ; jean.billaut}@univ-tours.fr

**RÉSUMÉ :** *Nous abordons dans cet article un problème de job shop multiressource multicritère. Deux critères sont à optimiser : la plus grande date de fin et le plus grand retard algébrique. Nous proposons un programme linéaire en nombres entiers, une recherche Tabou et un algorithme génétique pour approximer le front de Pareto. Des expérimentations faites sur des instances de la littérature ainsi que sur des instances générées aléatoirement montrent l'efficacité de l'algorithme.*

**MOTS-CLÉS :** *Job shop multiressource, recherche Tabou, algorithme génétique, multicritère*

## 1 INTRODUCTION

Bien souvent dans les industries, notamment dans l'imprimerie, l'exécution d'une opération peut nécessiter plusieurs ressources simultanément. Par exemple, une opération d'impression nécessite une imprimante rotative et un opérateur humain. De plus, l'opération peut ne pas avoir besoin de toutes les ressources tout le temps. Ainsi dans notre exemple, l'opérateur humain n'est nécessaire qu'au début de l'opération pour « caler » l'imprimante. Une fois le calage effectué, l'opérateur peut aller s'occuper d'une autre machine.

On retrouve dans la littérature ce genre de problème sous l'appellation *job shop multiressource*. On peut citer (Dauzère-Pérès *et al.*, 1998) où une recherche Tabou est proposée. Dans (Chen and Lee, 1999), les auteurs considèrent un problème où chaque opération nécessite un ensemble de ressources, choisi parmi un ensemble d'ensembles de ressources qui dépend de l'opération. Une approche hiérarchique est proposée par les auteurs. Dans (Cheng *et al.*, 1999) les auteurs ont étudié le problème de flow shop à deux machines avec un opérateur chargé du montage et du démontage. On trouve dans (Shachnai and Turek, 1999) une heuristique pour résoudre un problème d'ordonnancement d'opérations multiressources dans un système informatique. Dans (Artigues and Roubellat, 2000), les auteurs ont étudié un problème de RCPSP dans le cas multimode. Dans (Dauzère-Pérès and Pavageau, 2001), les auteurs ont proposé une modélisation pour un problème de job shop flexible multiressource. On trouve un algorithme basé sur des régles de priorité pour une problème de flow shop hybride à deux étages dans (Oğuz *et al.*, 2003). Dans (Serifoğlu and Ulusoy, 2004), les auteurs ont étudié une généralisation du problème de (Oğuz *et al.*, 2003) et ont proposé un algorithme génétique.

Nous proposons une recherche Tabou et un algorithme génétique pour résoudre le problème de job shop multiressource multicritère. Notre objectif est de déterminer une approximation du front de Pareto sur les critères $C_{max}$ et $L_{max}$, où $C_{max}$ est la notation pour la plus grande date de fin et $L_{max}$ pour le plus grand retard algébrique.

## 2 DÉFINITION DU PROBLÈME ET NOTATIONS

On considère un ensemble $J$ composé de $n$ travaux. Ces travaux doivent être ordonnancés sur $m$ ressources disjonctives, l'ensemble des ressources est noté $\mathcal{R}$. On note $R_k$ la $k^{ème}$ ressource. Chaque travail $i$ est composé de $n_i$ opérations et on associe à chaque travail $i$ une date de début au plus tôt notée $r_i$ et une date de fin souhaitée notée $d_i$. L'opération $j$ du travail $i$ est notée $O_{i,j}$. Les gammes ne sont pas nécessairement linéaires, c'est-à-dire qu'une opération $O_{i,j}$ peut avoir plusieurs successeurs (l'ensemble $\Gamma_{i,j}$) et plusieurs prédécesseurs (l'ensemble $\Gamma_{i,j}^{-1}$). On suppose que chaque travail n'a qu'une seule opération finale notée $O_{i,n_i}$. Chaque opération nécessite plusieurs ressources pour son exécution. On note $R_{i,j}^m$ l'ensemble des ressources devant exécuter l'opération $O_{i,j}$. On note $O_{i,j,k}$ la sous-opération de $O_{i,j}$ s'exécutant sur la ressource $R_k$ et $p_{i,j,k}$ sa durée opératoire. On suppose donc que toutes les sous-opérations d'une opération n'ont pas nécessairement la même durée. En revanche, on suppose que toutes les sous-opérations de $O_{i,j}$ débutent à la même date $t_{i,j}$. L'opération suivante dans la gamme ne peut commencer que quand la dernière sous-opération de l'opération est terminée. On note $\Gamma_{i,j,k}^{-1}$ l'ensemble des sous-opérations précédentes dans la gamme. A chaque travail $i$ est associée une date due $d_i$.

La figure 1 donne un exemple d'ordonnancement multi-ressource.

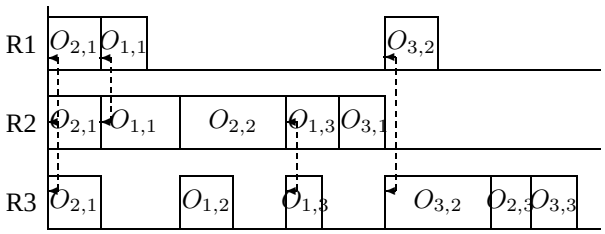


Figure 1 – Exemple d'ordonnancement multiressource

## 3 MODÉLISATION PAR UN PLNE

A notre connaissance, il n'existe pas de modèle de programmation linéaire en nombres entiers pour le problème de job shop multiressource tel que nous l'avons défini. Nous proposons d'adapter le modèle d'A.S. Manne (Manne, 1960).

Les variables de notre modèle sont :

$$t_{i,j} \quad : \quad \text{La date de début de } O_{i,j} \tag{1}$$

$$y_{c,d}^{a,b,k} \quad = \quad \begin{cases} 0 & \text{si } O_{a,b} \text{ est effectuée avant } O_{c,d} \\ & \text{sur } R_k \\ 1 & \text{sinon} \end{cases} \tag{2}$$

Les contraintes sont :

$$t_{i,j} \geq 0 \tag{3}$$
$$\forall i = 1, \ldots, n; \forall j = 1, \ldots, n_i$$

$$t_{i,j'} \geq t_{i,j} + p_{i,j,k} \tag{4}$$
$$\forall R_k \in R_{i,j}^m;$$
$$\forall i = 1, \ldots, n; \forall j = 1, \ldots, n_i - 1; \forall j' | O_{i,j'} \in \Gamma_{i,j}$$

$$t_{c,d} \geq t_{a,b} + p_{a,b,k} - HV \times y_{c,d}^{a,b,k} \tag{5}$$
$$\forall R_k \in R_{a,b}^m \cap R_{c,d}^m$$
$$\forall a = 1, \ldots, n; \forall b = 1, \ldots, n_i$$
$$\forall c = 1, \ldots, n; \forall d = 1, \ldots, n_i$$

$$t_{a,b} \geq t_{c,d} + p_{c,d,k} - HV \times (1 - y_{c,d}^{a,b,k}) \tag{6}$$
$$\forall R_k \in R_{a,b}^m \cap R_{c,d}^m$$
$$\forall a = 1, \ldots, n; \forall b = 1, \ldots, n_i$$
$$\forall c = 1, \ldots, n; \forall d = 1, \ldots, n_i$$

Les contraintes 3 indiquent que les travaux ne peuvent pas commencer avant la date zéro. L'équation 4 représente les contraintes de gammes opératoires. Pour qu'une opération puisse commencer, il faut que toutes les sous-opérations de l'opération précédente soient achevées. Les contraintes 5 et 6 permettent d'arbitrer les disjonctions sur les ressources. Dans l'équation 5, si $O_{a,b}$ s'exécute avant $O_{c,d}$ sur la ressource $R_k$ alors $O_{c,d}$ doit commencer après la fin de la sous-opération de $O_{a,b}$ sur $R_k$. Si $O_{a,b}$ s'exécute après $O_{c,d}$ sur $R_k$, alors la partie droite de l'équation est négative ce qui rend vraie l'inégalité. L'équation 6 utilise le même principe.

Les fonctions objectif sont $C_{max}$ et $L_{max}$, elles s'expriment simplement en fonction des variables $t_{i,j}$ de la façon suivante :

$C_{max} \geq t_{i,n_i} + p_{i,n_i}, \forall i = 1, ..., n$

$L_{max} \geq t_{i,n_i} + p_{i,n_i} - d_i, \forall i = 1, ..., n$

Nous avons $\#ope^2 \times m$ variables binaires et $\#ope + (\#ope - n)^2 \times m + 2 \times (\#ope^2 \times m)$ contraintes ($\#ope = \sum_i n_i$). Compte tenu du grand nombre de variables et de contraintes, il n'est pas possible de résoudre des instances réelles à l'aide d'un solver commercial. Toutefois, comme nous le verrons ultérieurement, certaines instances générées aléatoirement ont pu être résolues par CPLEX.

## 4 MODÉLISATION PAR UN GRAPHE

On peut adapter le modèle classique du graphe conjonctif au problème de job shop multiressource.

Dans ce modèle de graphe, on représente une opération par plusieurs sommets : un par sous-opération.

Formellement, soit le graphe $G = (V, E)$. L'ensemble des sommets est $V = \{O_{i,j,k}, 1 \leq i \leq n, 1 \leq j \leq n_i, R_k \in R_{i,j}^m\} \cup \{s, t\}$ avec $s$ le sommet source et $t$ le sommet puits. On a un arc entre deux sommets s'il existe une contrainte de précédence entre les opérations correspondantes. Il y a des arcs entre $O_{i,n_i,k}$ et $t$ de longueur $p_{i,n_i,k}$, où $R_k$ est la ressource sur laquelle $O_{i,n_i,k}$ s'exécute. Il y a un arc entre $s$ et $\{O_{i,j,k} | \Gamma_{i,j}^{-1} = \emptyset\}$ de longueur $r_i$. Pour les autres arcs, la longueur est égale à la durée opératoire de la sous-opération à l'origine de l'arc. Entre chaque sous-opération d'une même opération il y a des arcs de synchronisation bi-directionnels, leurs longueurs sont fixées à 0. Ce graphe ne doit pas contenir de circuit de longueur strictement positive, sinon l'ordonnancement est infaisable, et tous les arcs ont une longueur positive ou nulle. La figure 2 donne le graphe correspondant au diagramme de Gantt de la figure 1, les numéros des ressources ont été omis pour faciliter la lecture.

## 5 APPROCHE TABOU, $TS_{MPT}$

**Codage d'une solution** Une solution est représentée par le modèle de graphe présenté à la section précédente.

**Solution initiale** La génération de la solution initiale fait appel à un algorithme glouton. L'algorithme 1 décrit comment la solution initiale est générée. Ici, la fonction $choix(X)$ est une fonction qui choisit l'opération avec la plus petite marge (SLACK).

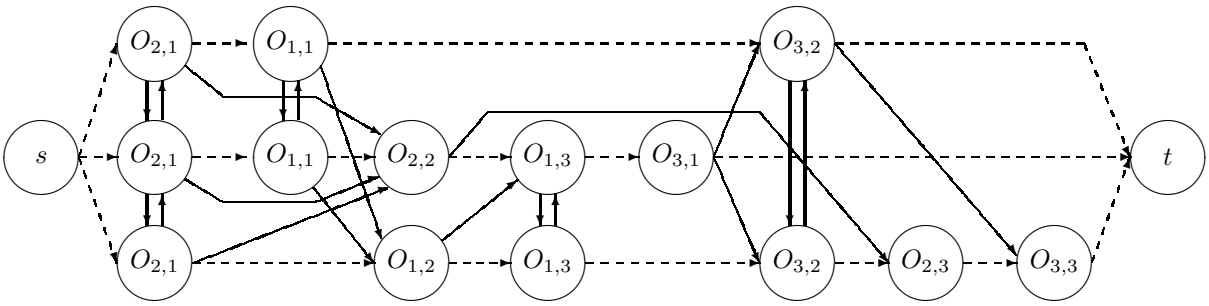


Figure 2 – Représentation sous forme d'un graphe détaillé d'un ordonnancement multiressource

**Définition du voisinage** On obtient un nouveau voisin en déplaçant une opération. Cependant, si l'opération est composée de sous-opérations, la manipulation peut devenir plus complexe. Nous avons choisi comme mouvement l'échange d'une sous-opération avec l'opération précédente sur la ressource.

Formellement, soit $x_a$ la sous-opération de $x$ qu'on souhaite déplacer entre $y$ et $z$, où $z$ est l'opération précédente à $x_a$ sur la ressource considérée. Soit $r_z$ la date de début de l'opération $z$. Pour toutes les sous-opérations $x_i$ de l'opération $x$ ($x_i \neq x_a$), on cherche à déplacer $x_i$ vers la gauche tant qu'elle n'aura pas de prédécesseur ressource ayant une date de début strictement inférieure à $r_z$ (sous réserve de compatibilité avec la gamme).

La figure 3 illustre un tel mouvement.

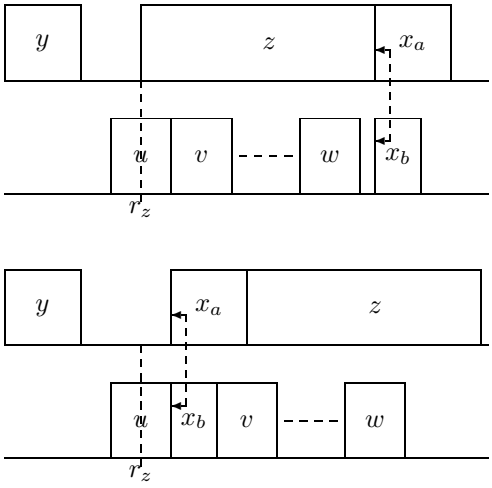


Figure 3 – Déplacement de la sous-opération $x_a$ avant $z$

Pour améliorer la convergence, on déplace une sous-opération tant que l'ordonnancement obtenu est réalisable et de meilleure qualité. De façon similaire, on peut générer un voisin en déplaçant une sous-opération vers la droite.

**Évaluation d'un voisin** La sélection du meilleur voisin se fait selon une approche epsilon-contrainte. On suppose qu'on cherche à optimiser le $C_{max}$ sous la contrainte $L_{max} < \epsilon$.

L'originalité de la recherche Tabou utilisant une approche $\epsilon$-contrainte réside dans la sélection du meilleur voisin. On distingue trois cas :

1. Il existe au moins un voisin qui respecte la borne $\epsilon$. On considère alors que le meilleur voisin est celui qui minimise $C_{max}$ parmi ceux qui vérifient $L_{max} < \epsilon$.
2. Aucun voisin ne respecte la borne $\epsilon$ et il en existe au moins un qui a un $C_{max}$ meilleur que celui de la solution courante. Alors on considère que le meilleur voisin est celui qui minimise le $L_{max}$ tout en étant au moins aussi bon sur le $C_{max}$ que la solution courante.
3. Aucun voisin ne respecte la borne $\epsilon$ et aucun n'améliore le $C_{max}$ de la solution courante. Alors on considère que le meilleur voisin est celui qui minimise le $L_{max}$.

Avec cette sélection, on cherche d'abord à obtenir une solution respectant la borne. Puis on améliore le *makespan* en respectant au maximum la borne. On s'autorise quand même à explorer, temporairement, des solutions non satisfaisantes. La solution retournée par $TS\epsilon$ est la meilleure solution respectant la borne trouvée au cours de la recherche. En appelant itérativement cette méthode, on peut approximer un front de Pareto.

**La liste Tabou** Notre liste Tabou contient deux éléments : l'indice de l'opération ou la sous opération décalée, et le nombre de décalages avec un signe désignant le sens droite ou gauche, effectué pour obtenir la solution.

Une solution est interdite si pour l'obtenir à partir de la solution courante, il faut effectuer un mouvement qui se trouve dans la liste Tabou. C'est-à-dire qui a le même nombre de décalages pour la même opération à déplacer, sans tenir compte du sens.

La taille de la liste est gérée dynamiquement. Cette gestion dynamique est basée sur un principe de détection de cycle. Ce dernier repose sur la sauvegarde d'un certain nombre des derniers mouvements effectués (deux fois le nombre d'opérations et de sous opérations) dans une liste.

**Algorithme 1** Algorithme glouton pour générer une solution pour le job-shop multiressource

**Pré-conditions:** $Choix(X)$ : Choisit un élément de la liste $X$

$\mathcal{O} = \emptyset$
**Pour** $i = 0$ à $i = m$ **Faire** $C_m[i] = 0$ **fin Pour**
**Pour** $i = 1$ à $n$ **faire**
  **Pour** $j = 1$ à $n_i$ **faire**
    $r_{i,j} = 0$
    **Si** $\Gamma_{i,j}^{-1} = \emptyset$ **Alors** Ajouter $O_{i,j}$ à $\mathcal{O}$ **fin Si**
  **fin Pour**
**fin Pour**
**Tant que** $\mathcal{O} \neq \emptyset$ **faire**
  $C^* = \infty$
  /* *Recherche de l'opération se terminant le plus tôt* */
  **Pour chaque** $O_{i,j} \in \mathcal{O}$ **faire**
    $dateFin = -\infty$
    **Pour chaque** $R_k \in R_{i,j}^m$ **faire**
      $dateFin = \max\left(dateFin; \max(r_{i,j}, C_m[k]) + p_{i,j,k}\right)$
    **fin Pour**
    **Si** $C^* > dateFin$ **alors**
      $C^* = dateFin$
      $\mathcal{R}^* = R_{i,j}^m$
    **fin Si**
  **fin Pour**
  /* *Choix d'une opération concurrente* */
  $\mathcal{L} = \{O_{i,j} \in \mathcal{O} | r_{i,j} < C^*$ et $R_{i,j}^m \cap \mathcal{R}^* \neq \emptyset\}$
  $O_{i^*,j^*} = Choix(\mathcal{L})$
  /* *Ordonnancer $O_{i^*,j^*}$ au plus tôt* */
  **Pour chaque** $R_k \in R_{i^*,j^*}^m$ **faire**
    $r_{i^*,j^*} = \max(r_{i^*,j^*}, C_m[R_k])$ /* *Trouver la date de début de l'opération* */
  **fin Pour**
  $C_{i^*,j^*} = r_{i^*,j^*} + \max_{R'_k \in R_{i^*,j^*}^m}(p_{i^*,j^*,k'}$
  **Pour chaque** $R_k \in R_{i^*,j^*}^m$ **faire**
    $C_m[k^*] = r_{i^*,j^*} + p_{i^*,j^*,k}$ /* *Mise à jour des dates de disponibilité des ressources* */
  **fin Pour**
  /* *Mise à jour des listes et des indicateurs* */
  Retirer $O_{i^*,j^*}$ de $\mathcal{O}$
  **Pour chaque** $O_{i,j} \in \Gamma_{i^*,j^*}$ **faire**
    $r_{i,j} = \max(r_{i,j}, C_{i^*,j^*})$
    **Si** Tout les prédécesseur d'$O_{i,j}$ ont été ordonnancées **alors**
      Ajouter $O_{i,j}$ à $\mathcal{O}$
    **fin Si**
  **fin Pour**
**fin Tant que**

Si à une itération donnée, deux mouvements consécutifs ont été effectués et que ces deux mouvements sont aussi dans la liste à la suite (en tenant compte du sens du déplacement), alors il y a une forte probabilité pour qu'un cycle se forme. C'est à ce moment là qu'il faut jouer sur la taille de la liste de Tabou de manière à sortir du cycle éventuel. Suite à des résultats expérimentaux, la gestion dynamique de la taille, de façon à optimiser le mieux possible la qualité des solutions, est la suivante :

– tout d'abord, la taille, à l'initialisation, est égale à un tiers du nombre d'opérations unitaires (sans les sous opérations) plus le nombre d'opérations non-unitaires.
– La taille est augmentée de 10% du nombre d'opérations et de sous opérations, lorsqu'un cycle éventuel est détecté et que la taille de la liste Tabou est inférieure ou égale à la moitié du nombre d'opérations et de sous opérations.
– La taille est diminuée de 10% du nombre d'opérations et de sous opérations, lorsqu'un cycle éventuel est détecté et que la taille de la liste Tabou est supérieure à la moitié du nombre d'opérations et de sous opérations, ou encore lorsqu'aucun voisin n'est possible à partir d'une solution. De plus, on vide la liste Tabou de façon à répartir équitablement entre tous les mouvements.

Les conditions d'augmentation et de diminution de la taille de la liste Tabou permettent de faire osciller convenablement cette taille. Cette oscillation rend encore plus dynamique la recherche de solutions, et permet d'augmenter la qualité des solutions trouvées par cette recherche Tabou.

## 6 APPROCHE PAR UN ALGORITHME GÉNÉTIQUE, $AG_{MPT}$

Pour résoudre le job shop multiressources multicritère, nous avons aussi adapté un algorithme génétique que nous avions proposé pour le job shop flexible multicritère (Vilcot and Billaut, 2007). Cette version de l'algorithme utilise le modèle NSGA-II (Deb *et al.*, 2002).

**Le codage d'un individu** On représente un individu par une matrice $\mathcal{C}$ avec $m$ lignes (une par ressource) et $|rang|$ colonnes, où $|rang|$ est le nombre de rangs dans le graphe disjonctif associé à l'ordonnancement. A la ligne $l$ et à la colonne $c$ il y a l'opération qui doit être exécutée sur la ressource $R_l$ au rang $c$ dans le graphe. Un '_' signifie qu'il n'y a pas d'opération au rang correspondant dans le graphe sur la ressource considérée. La figure 4 donne le codage associé à la figure 1.

$$\mathcal{C} = \begin{pmatrix} O_{2,1} & O_{1,1} & \_ & \_ & \_ & O_{3,2} & \_ & \_ \\ O_{2,1} & O_{1,1} & O_{2,2} & O_{1,3} & O_{3,1} & \_ & \_ & \_ \\ O_{2,1} & \_ & O_{1,2} & O_{1,3} & \_ & O_{3,2} & O_{2,3} & O_{3,3} \end{pmatrix}$$

Figure 4 – Codage d'un individu

**Le croisement** On génère un nouvel individu grâce à la méthode suivante : premièrement, on sélectionne deux individus $a$ et $b$ grâce à l'opérateur de sélection. Ensuite, on génère une matrice booléenne $\mathcal{B}$ de la même taille que $a$, les valeurs de cette matrice sont tirées aléatoirement avec une équiprobabilité. Notons que pour une opération donnée, les coefficients doivent être identiques pour chaque sous-opération. Donc lorsque la première sous-opération d'une opération est rencontrée, la valeur obtenue pour cette sous-opération est également attribuée aux autres sous-opérations. Un '1' signifie que l'opération correspondante dans $a$ sera placée à la même position dans l'enfant. A la fin de cette étape, les opérations qui ne sont pas encore placées sont insérées dans les trous dans le même ordre que dans $b$.

S'il y a un circuit dans l'enfant, l'opération venant de $b$ qui est impliquée dans le circuit est déplacée à une position qui ne crée pas de circuit, c'est-à-dire entre ses prédécesseurs et ses successeurs.

La figure 5 donne un exemple de croisement.

$$a = \begin{pmatrix} 1,1 & \_ & \_ & 2,1 & \_ & \_ & 3,2 & \_ \\ 1,1 & \_ & 1,3 & 2,1 & 2,2 & 3,1 & \_ & \_ \\ \_ & 1,2 & 1,3 & 2,1 & \_ & 2,3 & 3,2 & 3,3 \end{pmatrix}$$

$$b = \begin{pmatrix} 2,1 & 1,1 & \_ & \_ & \_ & 3,2 & \_ & \_ \\ 2,1 & 1,1 & 2,2 & 1,3 & 3,1 & \_ & \_ & \_ \\ 2,1 & \_ & 1,2 & 1,3 & \_ & 3,2 & 2,3 & 3,3 \end{pmatrix}$$

$$\mathcal{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Enfant partiel $=$

$$\begin{pmatrix} 1,1 & \_ & \_ & \_ & \_ & \_ & 3,2 & \_ \\ 1,1 & \_ & 1,3 & \_ & 2,2 & \_ & \_ & \_ \\ \_ & \_ & 1,3 & \_ & \_ & \_ & 3,2 & \_ \end{pmatrix}$$

Enfant final $=$

$$\begin{pmatrix} 1,1 & 2,1 & \_ & \_ & \_ & 3,2 & \_ & \_ \\ 1,1 & 2,1 & 1,3 & 2,2 & 3,1 & \_ & \_ & \_ \\ \_ & 2,1 & 1,3 & 1,2 & \_ & 3,2 & 2,3 & 3,3 \end{pmatrix}$$

Figure 5 – Exemple d'un croisement

**La mutation** On applique l'opérateur de mutation sur un individu. Dans l'individu à muter, on sélectionne aléatoirement une opération $x$. Cette opération $x$ est déplacée sur un rang pour lequel il n'y aura pas de circuit dans le graphe correspondant, c'est-à-dire entre le dernier prédécesseur et le premier successeur de $x$.

On calcule dynamiquement la probabilité de mutation ($Tm$) à chaque génération avec la formule suivante :

$$Tm = TM_{max} \times \frac{\#GenSansAmel}{\#MaxGenSansAmel}$$

Avec $TM_{max}$ la probabilité de mutation maximale définie par le décideur, $\#GenSansAmel$ le nombre courant de génération sans amélioration et $\#MaxGenSansAmel$ le nombre maximum de générations sans amélioration.

**La population** La génération de la population initiale réutilise l'algorithme 1. Dans ce cas, la fonction $choix(X)$ est une fonction qui choisit aléatoirement un élément de la liste $X$.

La population peut aussi être partiellement initialisée par les solutions issues de $TS_{MPT}$ lors de la recherche du front de Pareto. On nomme $AG_{MPT}$ la version avec initialisation par la Tabou et $AG_{MPT}r$ la version avec initialisation totalement aléatoire.

**NSGA-II** L'algorithme génétique utilise le schéma d'algorithme NSGA-II (Deb *et al.*, 2002). Le but de NSGA-II est d'être rapide et élitiste. Il fonctionne de la façon suivante.

Tous les individus de la population initiale (notée $P_0$ ; $|P_0| = N$) sont évalués selon la méthode des *niveaux non-dominés*. La figure 6 illustre le concept des niveaux non-dominés. Le premier niveau contient tous les individus dominants de la population. Ensuite, si on ne tient plus compte de ces individus, le nouvel ensemble d'individus non dominés constitue le second niveau. Ce processus est itéré tant que tous les individus de la population ne sont pas classés. Les niveaux forment la force (à minimiser) des individus de la population initiale.

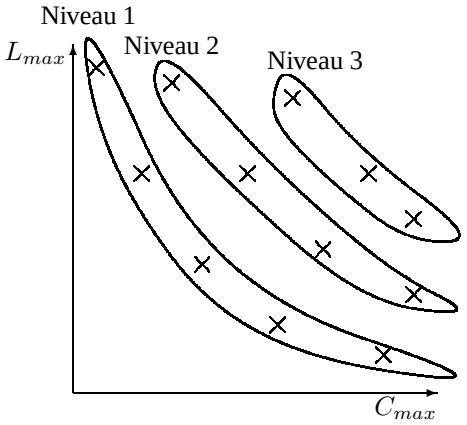


Figure 6 – Niveaux non-dominés

On utilise un tournoi binaire pour la sélection des parents. On utilise l'opérateur de croisement et de mutation pour créer le premier ensemble d'enfants $Q_0$ ($|Q_0| = N$).

A une génération donnée $t$, on définit $R_t = P_t \cup Q_t$. On recherche les niveaux non-dominés sur $R_t$. On trie les individus de $R_t$ selon leur niveau. On définit par le front $\mathcal{F}_f$ les solutions non-dominées du niveau $f$.

Les individus dans la population parente $P_{t+1}$ à la génération $t+1$ sont les solutions des fronts $\mathcal{F}_1$ à $\mathcal{F}_\lambda$ avec $\lambda$ tel que $\sum_{i=1}^{\lambda} |\mathcal{F}_i| \leq N$ et $\sum_{i=1}^{\lambda+1} |\mathcal{F}_i| > N$ plus les $N - \sum_{i=1}^{\lambda} |\mathcal{F}_i|$ premières solutions de $\mathcal{F}_{\lambda+1}$ selon l'ordre décroissant de leur distance de *crowding*. Les solutions restantes sont éliminées. Au final, la taille de $P_{t+1}$ est $N$. On utilise les opérateurs de croisement et de mutation sur les individus de $P_{t+1}$ pour former une nouvelle population enfante $Q_{t+1}$. La figure 7 illustre la création de la population $P_{t+1}$.

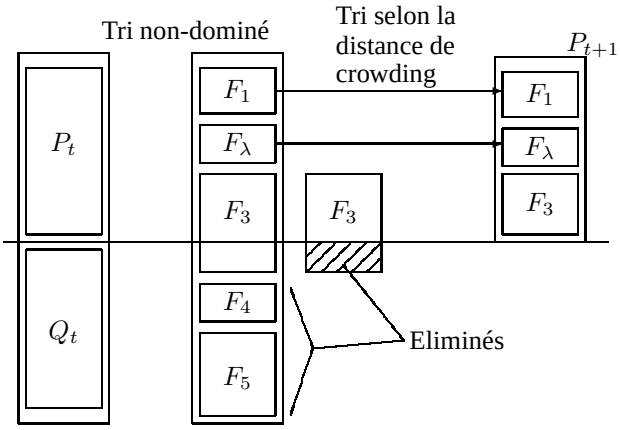


Figure 7 – Construction de la population $P_{t+1}$

La distance de *crowding* pour un individu est définie ainsi (Deb *et al.*, 2002) : c'est « une estimation du périmètre formé par le pavé utilisant les plus proches voisins comme sommets ». On assure la diversité grâce à la distance de crowding. Pour un individu, la distance de crowding est la somme des distances normalisées (dans l'espace des critères) entre les voisins *droit* et *gauche* pour chaque objectif considéré. Les solutions extrêmes ont une distance de crowding égale à l'infini. La figure 8 illustre le concept dans deux dimensions.

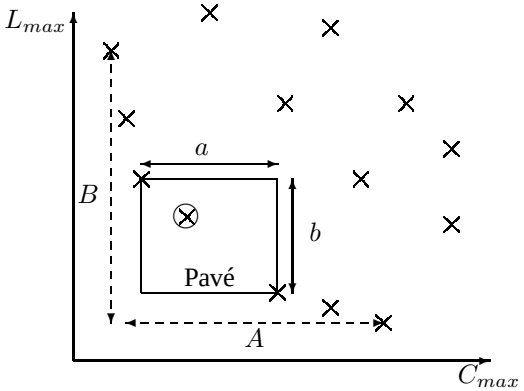


Figure 8 – Calcul de la distance de *crowding*

Dans notre exemple à deux dimensions, la distance de crowding $d_c$ de l'individu vaut : $d_c = \frac{a}{A} + \frac{b}{B}$

L'opérateur de sélection est un tournoi binaire : entre deux individus, on sélectionne celui avec le plus faible niveau. Si les deux individus sont sur le même niveau, le meilleur individu est celui avec la plus grande distance de crowding.

## 7 EXPÉRIMENTATIONS

Des expérimentations ont été réalisées pour comparer nos algorithmes.

Toutes les expérimentations ont été réalisées sur un PC équipé d'un Pentium IV cadencé à 2.8GHz et ayant 512Mo de mémoire vive. Pour l'algorithme génétique, le temps de calcul a été limité à 10 minutes.

### 7.1 Les instances

Nous considérons deux types d'instances pour nos expérimentations. Premièrement nous utilisons les instances issues de la littérature de (Adams *et al.*, 1988), dénotées par la suite par « la*xx* ». Ces instances sont pour le job shop classique, c'est-à-dire avec une seule ressource par opération. Pour ces instances, nous avons ajouté des dates de fin souhaitées avec la méthode suivante (insiprée par (Demirkol *et al.*, 1998)), pour un travail $i$ :

$$d_i \hookrightarrow \mathcal{U}\left[\mu_i \times \left(1 - \frac{R}{2}\right); \mu_i \times \left(1 + \frac{R}{2}\right)\right]$$

Avec :

$$\mu_i = \left(1 + \frac{T \times n}{m}\right) \times \sum_{j=0}^{n_i} p_{i,j}$$

$T$ et $R$ sont des paramètres que nous avons fixés à : $T = 0.3$ et $R = 0.5$.

Deuxièmement, nous avons généré des instances pour le job shop multiressource, dénotées « GJ*xx* ». A notre connaissance il n'existe pas dans la littérature d'instances pour le problème de job shop multiressource. Nous considérons quatre paramètres : $n$ le nombre de travaux, $n_i$ le nombre d'opérations par travail, $m$ le nombre de ressources et $\#sta$ le nombre d'étages. La durée opératoire d'une opération $O_{i,j}$ est tirée aléatoirement entre 1 et 130. Les dates dues sont générées avec la même méthode que celle décrite précédemment. Les ressources sont réparties en étages (1 à $\#sta$). Chaque opération $O_{i,j}$ nécessite entre 1 et $\#sta$ ressources mais ne peut nécessiter plus d'une ressource d'un même étage.

Le tableau 1 donne les instances qui ont été générées par cette méthode. Chaque ensemble est constitué de cinq instances différentes.

### 7.2 Indicateurs

Afin de comparer les fronts de Pareto, nous avons utilisé les indicateurs de l'optimisation multicritère (Gandibleux *et al.*, 2004).

| Instances | $n$ | $n_j$ | $m$ | $\#sta$ |
|---|---|---|---|---|
| GJ01 à GJ05 | 5 | 5 | 10 | 10 |
| GJ06 à GJ10 | 5 | 10 | 20 | 4 |
| GJ11 à GJ15 | 5 | 10 | 20 | 5 |
| GJ16 à GJ20 | 10 | 5 | 20 | 4 |
| GJ21 à GJ25 | 10 | 10 | 20 | 4 |
| GJ26 à GJ30 | 10 | 5 | 20 | 10 |
| GJ31 à GJ35 | 10 | 10 | 20 | 10 |

Tableau 1 – Les instances générées

**Faible dominance d'un ensemble - *Weak outperformance*** Un ensemble de solutions non-dominées $S_\times$ domine faiblement un ensemble $S_\circ$ si aucune solution dans $S_\times$ n'est dominée par une solution dans $S_\circ$ et si au moins une solution dans $S_\times$ domine une solution dans $S_\circ$. La figure 9 illustre la notion de dominance faible : l'ensemble $S_\times$ (les solutions représentées par une croix) domine faiblement l'ensemble $S_\circ$ (les solutions représentées par un cercle).

$Ow(S_\times, S_\circ)$ est égal à 1 si $S_\times$ domine faiblement $S_\circ$, 0 sinon. Notons que $Ow(S_\times, S_\circ) + Ow(S_\circ, S_\times)$ peut être égal à 0 ou 1, mais pas à 2.

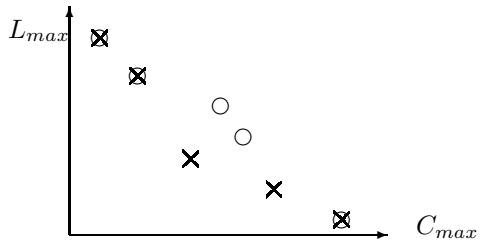


Figure 9 – Notion de dominance faible

Soit $exp_i$ et $exp_j$ deux séries d'expérimentations et $S_{exp_i}$ (resp. $S_{exp_j}$) l'ensemble des solutions non-dominées pour la série $exp_i$ (resp. $exp_j$). Pour chaque $j \neq i$, $SOw(exp_i, exp_j)$ est la somme pour chaque instance considérée des $Ow(S_{exp_i}, S_{exp_j})$.

Plus cet indicateur est grand, meilleur est $S_{exp_i}$.

**La contribution au front idéal - *Net Front Contribution*** Considérons deux ensembles de solutions non-dominées $S_\times$ et $S_\circ$, $S_\star$ représente l'ensemble des solutions non-dominées dans $S_\times \cup S_\circ$. $NFC(S_\times, S_\circ)$ est la proportion de solutions de $S_\times$ dans $S_\star$. $\overline{NFC}(exp_i, exp_j)$ dénote la moyenne de $NFC(S_{exp_i}, S_{exp_j})$ pour toutes les instances considérées.

L'indicateur $\overline{NFC}$ correspond à la moyenne des contributions au front idéal. Plus cet indicateur est grand, meilleur est $S_{exp_i}$.

Il faut noter que cet indicateur n'est pas symétrique, c'est à dire qu'on peut avoir $NFC(S_{exp_i}, S_{exp_j}) + NFC(S_{exp_j}, S_{exp_i}) \neq 100\%$. Cela est notamment vrai si les deux fronts sont identiques.

### 7.3 Expérimentations préliminaires

Des expérimentations préliminaires ont été réalisées sur quelques instances pour trouver les meilleurs paramètres pour nos algorithmes.

Pour la recherche Tabou $TS_{MPT}$, le meilleur nombre maximum d'itérations sans amélioration trouvé est $\#iter = 2000$.

Pour l'algorithme génétique $AG_{MPT}$, on trouve comme taille de population $N = 100$, nombre maximum de générations sans amélioration $\#MaxGen = 1000$ et la probabilité de mutation $Mut = 0.5$.

### 7.4 Comparaison des méthodes

Les premières expérimentations concernent la comparaison des deux méthodes pour la détermination d'un ensemble de solution non-dominées sur les critères $C_{max}$ et $L_{max}$.

**Comparaison de $TS_{MPT}$ avec $AG_{MPT}r$** Dans cette partie, nous comparons les performances de la recherche Tabou $TS_{MPT}$ avec celles de l'algorithme génétique $AG_{MPT}r$. Le tableau 2 donne les résultats de la comparaison entre les deux méthodes. Les temps de calcul moyens sont donnés en secondes.

On constate que la recherche Tabou est toujours meilleure que l'algorithme génétique aussi bien sur les instances la*xx* que sur nos instances générées. Par ailleurs, on constate que sur les instances la*xx*, $AG_{MPT}$ atteint souvent son temps de calcul maximum (10 minutes).

**Comparaison de $AG_{MPT}$ avec et sans initialisation par $TS_{MPT}$** Nous avons testé notre algorithme génétique avec et sans initialisation de la population par $TS_{MPT}$. Le tableau 3 donne les résultats, où $AG_{MPT}r$ est la « version sans initialisation » et $AG_{MPT}$ la version « avec initialisation ». Les temps de calcul pour $AG_{MPT}$ incluent le temps nécessaire pour la recherche Tabou.

On constate une nette domination de la version avec initialisation. De plus dans plusieurs cas, le temps de calcul de $TS_{MPT}$ plus celui de $AG_{MPT}$ est inférieur à celui de $AG_{MPT}r$.

**Comparaison de $TS_{MPT}$ avec $AG_{MPT}$** Dans cette partie, nous comparons les performances de la recherche Tabou $TS_{MPT}$ avec celles de l'algorithme génétique $AG_{MPT}$. Le tableau 4 donne les résultats de la comparaison entre les deux méthodes. Les temps de calcul moyens sont donnés en secondes. Les temps de calcul de $AG_{MPT}$ n'incluent pas celui de $TS_{MPT}$.

On constate que $AG_{MPT}$ donne de meilleurs résultats que $TS_{MPT}$, ce qui n'est pas surprenant dans la mesure où les solutions de $TS_{MPT}$ sont injectées comme population initiale dans $AG_{MPT}$. On peut remarquer que pour huit instances, $AG_{MPT}$ n'arrive pas à améliorer les solutions trouvées par la recherche Tabou.

### 7.5 Comparaison avec CPLEX

Notre programme linéaire, testé avec CPLEX (version 8.0), arrive à résoudre optimalement les problèmes GJxx jusqu'à GJ20 (sauf pour GJ13 où le solver n'arrive pas à trouver l'optimum). Nous avons limité le temps de calcul à 12 heures. Le tableau 5 donne les valeurs du *makespan* obtenus par CPLEX et par nos algorithmes. La colonne 'CPU(CPLEX)' correspond au temps de calcul (en secondes) nécessaire à CPLEX pour trouver l'optimum. Les colonnes $\Delta(TS_{MPT})$ et $\Delta(AG_{MPT}r)$ correspondent à l'écart relatif entre nos méthodes et l'optimum.

Dans 6 cas sur 19, $TS_{MPT}$ trouve une solution optimale et dans 4 cas pour $AG_{MPT}r$. On constate que dans le pire des cas, on est à 6.33% de l'optimum avec la recherche Tabou (15.59% pour l'algorithme génétique). Globalement on peut constater que $TS_{MPT}$ est proche de l'optimum : $\overline{\Delta(TS_{MPT})} = 1.71\%$. Par contre $\overline{AG_{MPT}r}$ est clairement moins performant : $\overline{\Delta(AG_{MPT}r)} = 6.9\%$

Remarque : nous n'avons pas présenté les résultats pour $AG_{MPT}$ car les valeurs du *makespan* sont identiques à celles de $TS_{MPT}$, sauf pour l'instance GJ17 où $C_{max}(AG_{MPT}) = 664$.

## 8 CONCLUSION

Dans cet article nous avons présenté nos travaux sur le problème de job shop multiressource multicritère. Le problème a été modélisé par un graphe conjonctif. Nous avons proposé un modèle de programmation linéaire en nombres entiers, une recherche Tabou, un algorithme génétique ainsi qu'une hybridation de ces deux dernières méthodes.

Au niveau des expérimentations, nous avons constaté une supériorité de la recherche Tabou sur l'algorithme génétique. Par ailleurs, la recherche Tabou présente des résultats proches de l'optimal pour les instances qui ont pu êtres résolues par le programme linéaire.

Pour les futures recherches, nous allons développer un algorithme permettant de résoudre un problème de job shop flexible multiressource multicritère.

## RÉFÉRENCES


Adams, J., E. Balas and D. Zawack (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science* **34**, 391–401.

Artigues, C. and F. Roubellat (2000). A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research* **127**, 297–316.

Chen, J. and C-Y. Lee (1999). General multiprocessor task scheduling. *Naval Research Logistics* **46**, 57–74.

Cheng, T.C. Edwin, G. Wang and C. Sriskandarajah (1999). One-operator-two-machine flowshop scheduling with setup and dismounting times. *Computer & Operations Research* **26**, 715–730.

Dauzère-Pérès, S., W. Roux and J. B. Lasserre (1998). Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research* **107**, 289–305.

Dauzère-Pérès, S. and C. Pavageau (2001). Différencier les durées opératoires dans une approche intégrée pour l'ordonnancement multi-ressource. In : *3éme Conférence Francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels"*.

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan (2002). A Fast and Elitist Multiobjective Genetic Algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197.

Demirkol, E., S. Mehta and R. Uzsoy (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research* **109**, 137–141.

Gandibleux, X., Sevaux, M., Sörensen, K. and T'kindt, V., Eds. (2004). *Metaheuristics for Multiobjective Optimisation*. Springer. Berlin.

Manne, A.S. (1960). On the job-shop scheduling problem. *Operations Research*.

Oğuz, C., M. Fikret Ercan, T.C. E. Cheng and Y.F. Fung (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. *European Journal of Operational Research* **149**, 390–403.

Serifoğlu, F. S. and G. Ulusoy (2004). Multiprocessor task scheduling in multistage hybrid flow-shops : a genetic algorithm approach. *The Journal of the Operational Research Society* **55**, 504–512.

Shachnai, H. and J. J. Turek (1999). Multiresource malleable task scheduling to minimize response time. *Information Processing Letters* **70**, 47–52.

Vilcot, G. and J.-C. Billaut (2007). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research*. doi :10.1016/j.ejor.2007.06.039, A paraître.

| Instances | $TS_{MPT}$ vs $AG_{MPT}r$ $SOw$ | $\overline{NFC}$ | $AG_{MPT}r$ vs $TS_{MPT}$ $SOw$ | $\overline{NFC}$ | $\overline{CPU(TS_{MPT})}$ | $\overline{CPU(AG_{MPT}r)}$ |
|---|---|---|---|---|---|---|
| la01 à la05 | 3 | 85% | 0 | 25% | 33.59 | 242.45 |
| la06 à la10 | 1 | 60% | 1 | 55% | 118.80 | 528.18 |
| la11 à la15 | 2 | 73% | 0 | 33% | 266.31 | 584.29 |
| la16 à la20 | 4 | 87% | 0 | 13% | 133.62 | 599.66 |
| la21 à la25 | **5** | **100%** | 0 | 0% | 540.58 | 600.40 |
| la26 à la30 | **5** | **100%** | 0 | 0% | 1518.12 | 600.69 |
| la31 à la35 | **5** | **100%** | 0 | 0% | 2338.61 | 601.50 |
| la36 à la40 | **5** | **100%** | 0 | 0% | 1356.68 | 600.50 |
| GJ01 à GJ05 | 1 | 60% | 3 | 93% | 4.30 | 27.91 |
| GJ06 à GJ10 | 4 | 87% | 0 | 13% | 41.06 | 134.90 |
| GJ11 à GJ15 | 4 | 90% | 0 | 10% | 65.58 | 127.30 |
| GJ16 à GJ20 | 2 | 80% | 0 | 20% | 140.51 | 283.42 |
| GJ21 à GJ25 | 3 | 70% | 1 | 30% | 522.91 | 520.35 |
| GJ26 à GJ30 | 4 | 90% | 0 | 10% | 279.03 | 303.16 |
| GJ31 à GJ35 | 4 | 90% | 0 | 10% | 1071.29 | 599.82 |

Tableau 2 – Comparaison entre $TS_{MPT}$ et $AG_{MPT}r$

| Instances | $AG_{MPT}r$ vs $AG_{MPT}$ $SOw$ | $\overline{NFC}$ | $AG_{MPT}$ vs $AG_{MPT}r$ $SOw$ | $\overline{NFC}$ | $\overline{CPU(AG_{MPT}r)}$ | $\overline{CPU(AG_{MPT})}$ |
|---|---|---|---|---|---|---|
| la01 à la05 | 0 | 25% | **5** | **100%** | 242.45 | **111.48** |
| la06 à la10 | 0 | 17% | 4 | 90% | 528.18 | **227.60** |
| la11 à la15 | 0 | 15% | 4 | 95% | 584.29 | 632.50 |
| la16 à la20 | 0 | 0% | **5** | **100%** | 599.66 | **433.17** |
| la21 à la25 | 0 | 0% | **5** | **100%** | 600.40 | 830.61 |
| la26 à la30 | 0 | 0% | **5** | **100%** | 600.69 | 1772.00 |
| la31 à la35 | 0 | 0% | **5** | **100%** | 601.50 | 2471.48 |
| la36 à la40 | 0 | 0% | **5** | **100%** | 600.50 | 1610.40 |
| GJ01 à GJ05 | 1 | 93% | 1 | 90% | 27.91 | 27.71 |
| GJ06 à GJ10 | 0 | 10% | 4 | 90% | 134.90 | **73.86** |
| GJ11 à GJ15 | 0 | 0% | **5** | **100%** | 127.30 | **116.26** |
| GJ16 à GJ20 | 0 | 10% | 4 | 90% | 283.42 | **234.56** |
| GJ21 à GJ25 | 0 | 15% | 4 | 85% | 520.35 | 695.15 |
| GJ26 à GJ30 | 0 | 0% | **5** | **100%** | 303.16 | 315.75 |
| GJ31 à GJ35 | 0 | 0% | **5** | **100%** | 599.82 | 1180.33 |

Tableau 3 – Comparaison entre $AG_{MPT}r$ et $AG_{MPT}$

| Instances | $TS_{MPT}$ vs $AG_{MPT}$ $SOw$ | $\overline{NFC}$ | $AG_{MPT}$ vs $TS_{MPT}$ $SOw$ | $\overline{NFC}$ | $\overline{CPU(TS_{MPT})}$ | $\overline{CPU(AG_{MPT})}$ |
|---|---|---|---|---|---|---|
| la01 -> la05 | 0 | 62% | 4 | **100%** | 33.59 | 92.40 |
| la06 -> la10 | 0 | 30% | **5** | **100%** | 118.80 | 157.91 |
| la11 -> la15 | 0 | 28% | **5** | **100%** | 266.31 | 458.64 |
| la16 -> la20 | 0 | 41% | 4 | **100%** | 133.62 | 345.33 |
| la21 -> la25 | 0 | 48% | **5** | **100%** | 540.58 | 440.93 |
| la26 -> la30 | 0 | 24% | **5** | **100%** | 1518.12 | 600.54 |
| la31 -> la35 | 0 | 33% | **5** | **100%** | 2338.61 | 601.42 |
| la36 -> la40 | 0 | 32% | **5** | **100%** | 1356.68 | 600.67 |
| GJ01 -> GJ05 | 0 | 60% | 3 | **100%** | 4.30 | 25.81 |
| GJ06 -> GJ10 | 0 | 80% | 2 | **100%** | 41.06 | 54.89 |
| GJ11 -> GJ15 | 0 | 24% | **5** | **100%** | 65.58 | 81.89 |
| GJ16 -> GJ20 | 0 | 23% | **5** | **100%** | 140.51 | 157.81 |
| GJ21 -> GJ25 | 0 | 27% | **5** | **100%** | 522.91 | 357.37 |
| GJ26 -> GJ30 | 0 | 30% | 4 | **100%** | 279.03 | 150.89 |
| GJ31 -> GJ35 | 0 | 0% | **5** | **100%** | 1071.29 | 503.57 |

Tableau 4 – Comparaison entre $TS_{MPT}$ et $AG_{MPT}$

| Instance | CPLEX | CPU(CPLEX) | $TS_{MPT}$ | $\Delta(TS_{MPT})$ | $AG_{MPT}r$ | $\Delta(AG_{MPT}r)$ |
|---|---|---|---|---|---|---|
| GJ01 | 533 | 0.06 | **533** | 0.00% | 544 | 2.02% |
| GJ02 | 418 | 0.05 | **418** | 0.00% | **418** | 0.00% |
| GJ03 | 487 | 0.06 | **487** | 0.00% | **487** | 0.00% |
| GJ04 | 383 | 0.03 | **383** | 0.00% | **383** | 0.00% |
| GJ05 | 460 | 0.02 | **460** | 0.00% | **460** | 0.00% |
| GJ06 | 894 | 2.52 | 898 | 0.45% | 963 | 7.17% |
| GJ07 | 900 | 4.14 | **900** | 0.00% | 918 | 1.96% |
| GJ08 | 763 | 45.39 | 794 | 3.90% | 821 | 7.06% |
| GJ09 | 767 | 1.69 | 789 | 2.79% | 892 | 14.01% |
| GJ10 | 998 | 36.03 | 1017 | 1.87% | 1100 | 9.27% |
| GJ11 | 839 | 12.97 | 891 | 5.84% | 977 | 14.12% |
| GJ12 | 1057 | 210.59 | 1100 | 3.91% | 1194 | 11.47% |
| GJ13 | # | # | 1071 | # | 1162 | # |
| GJ14 | 786 | 25.52 | 790 | 0.51% | 815 | 3.56% |
| GJ15 | 930 | 1580.28 | 934 | 0.43% | 1034 | 10.06% |
| GJ16 | 600 | 1925.77 | 614 | 2.28% | 644 | 6.83% |
| GJ17 | 656 | 42110.38 | 669 | 1.94% | 774 | 15.25% |
| GJ18 | 594 | 2590.08 | 603 | 1.49% | 608 | 2.30% |
| GJ19 | 693 | 965.81 | 698 | 0.72% | 821 | 15.59% |
| GJ20 | 681 | 13.34 | 727 | 6.33% | 760 | 10.39% |

Tableau 5 – Comparaison avec CPLEX