PRISE EN COMPTE DE LA FONCTION APPRENTISSAGE DANS UN ORDONNANCEMENT

Boudhar Hamza, Boschian Valerio

9e Conférence Internationale de Modélisation, Optimisation et SIMulation - MOSIM'12
06 au 08 Juin 2012 - Bordeaux - France
« Performance, interopérabilité et sécurité pour le développement durable »

PRISE EN COMPTE DE LA FONCTION APPRENTISSAGE DANS UN ORDONNANCEMENT

Hamza BOUDHAR[A], Valerio BOSCHIAN[B]

LGIPM, équipe SdP
Université de Metz/ ENIM
Ile du Saulcy - 57000 Metz - France
A. boudharhro@gmail.com, B. boschian@univ-metz.fr

RÉSUMÉ : *Dans cette étude, nous traitons le problème de l'ordonnancement des tâches de maintenance sur lequel nous avons introduit la notion de matrice des compétences, que l'on peut trouver sous deux formes possibles, dynamique ou statique. Un concept de niveau de compétences à quatre niveaux intervient dans la sélection des ressources aptes à réaliser des opérations, soit de manière autonome, soit par apprentissage, pour les opérateurs n'ayant pas le niveau de maîtrise nécessaire. Cette dernière solution apportera davantage de flexibilité aux opérateurs. Après une formulation mathématique du problème, nous orientons notre recherche vers une solution admissible en utilisant une méta-heuristique (algorithme génétique) décrite sous plusieurs variantes.*

MOTS-CLES : *Ordonnancement, Matrice des Compétences, Ressources, Makespan, Formation par apprentissage.*

1 INTRODUCTION

Dans un monde où la technologie industrielle est devenue banale, l'aspect de la gestion des projets devient l'enjeu le plus important et le moyen le plus efficace de se distinguer de la concurrence. Ainsi les industriels sont-ils amenés à développer de nouvelles méthodes afin de mieux exploiter ses ressources, matérielles et humaines.
Dans une entreprise, il existe deux types de ressources renouvelables, les machines et les humains. Ces derniers retiennent l'attention en raison de leurs aptitudes à acquérir de nouvelles compétences et à augmenter leur niveau de maitrise, et ce jusqu'à un niveau d'expertise qui les rend capables de transmettre leur savoir-faire à d'autres personnes. Cet aspect de compétence peut jouer un rôle important sur la réalisation des projets, et faire la différence entre les concurrents, tant sur le plan de l'innovation que sur le plan technique. Pour la réalisation des tâches du projet, les opérateurs doivent disposer d'un certain savoir-faire et d'un niveau de maîtrise suffisant. Dans cette perspective, le concept de matrice des compétences a été développé afin de regrouper les informations relatives aux spécialités demandées avec les opérateurs intervenants. Ce type de données peut être exploité afin de mieux former de nouveaux personnels habiles pour la réalisation des tâches, ce qui va conduire à une réalisation plus rapide du projet. Cette formation est assurée par l'accomplissement en commun de tâches entre un formateur et un apprenant et selon des règles fixées.

Dans l'industrie on trouve plusieurs types de problèmes de gestion de projet, appelés aussi problèmes d'ordonnancement. D'une façon générale on peut les diviser en deux catégories : sans ressource, qui sont généralement traités par la méthode Pert, ou avec ressource, généralement notés RCPSP (Resource-Constrained Project Scheduling Problem). Ce dernier type de problème a fait l'objet de nombreux travaux académiques. On trouve le calcul d'une borne inférieure par la programmation linéaire en nombre entier [Stinson et al, 1978] ou bien par réduction du problème [Carlier et Latapie, 1991]. La résolution du problème avec des méthodes approchées telles que la méthode tabou par Pinson [Pinson et al, 1994], qui figure parmi les premiers à l'avoir implémentée sur le RCPSP, ou bien l'algorithme génétique de Hatmann [Hartmann, 2002]. Cependant et malgré sa classification parmi les problèmes NP-difficile, on trouve des propositions de méthodes exactes telles que le Branch and Bound de Patterson [Patterson et al, 1990] et par Sprecher [Sprecher, 2000] , avec des schémas basés sur des systèmes de branchement chronologique. De plus, on trouve des propositions de modèle linéaire en nombre entier (PLNE), parmi ceux-là, ceux de Koné [Koné, 2009], où l'auteur propose différentes variantes du modèle.

Des extensions du problème RCPSP ont été proposées dans la littérature, on le trouve avec un nombre fini de façons d'exécuter les opérations, ce qui a donné le MRCPSP (Multi mode Ressouce Constrained Project Scheduling Problem), ou bien avec des ressources multi compétentes, ce qui a donné le MSPSP (Multi Skill Project Scheduling Problem), une formulation plus réaliste du problème par rapport au problème de base. Quant aux travaux de recherche sur le problème MSPSP, Morineau [Morineau, 2006] propose un modèle linéaire au nombre entier inspiré de celui de Prisker et al [Pritsker et al, 1969] pour le RCPSP ; il propose aussi une borne inférieure et des méthodes approchées telles qu'un algorithme génétique et une méthode de type placement par

série, inspirée des travaux de Kelley [Kelley, 1963] et Klein [Klein, 2000].

Dans la littérature on trouve d'autres classifications des problèmes d'ordonnancement, que l'on appelle problème d'ordonnancement d'atelier, qui font partie d'une des extensions du RCPSP. On peut les diviser en trois grands problèmes, Flow Shop, Open Shop et le Job Shop. Ce dernier, comme pour les deux autres, a connu plusieurs extensions, dont celle du Job Shop flexible, pour laquelle la ressource qui effectue l'opération est sélectionnée parmi un ensemble de ressources candidates [Kacem, 2003]. Dupas [Dupas, 2004] propose pour ce problème un algorithme génétique avec trois critères à optimiser, le but de son travail consiste à comparer différents opérateurs de mutation.

Notre cas d'étude est un problème de type Job Shop flexible, pour ordonner des tâches de maintenance. Ces dernières sont effectuées par des ressources humaines (opérateurs), ayant un niveau de compétence dans différentes spécialités. Une matrice des compétences, statique ou dynamique, enregistre les niveaux de maîtrise de chaque opérateur pour des spécialités données. Notre double objectif (conflictuel) est la minimisation du temps total de traitement des tâches ainsi que l'augmentation des compétences des opérateurs afin qu'ils accèdent à l'autonomie pour traiter des tâches et/ou au rôle de formateur.

Nous avons choisi une équipe de maintenance industrielle car ce domaine exige de la polyvalence de la part des opérateurs. C'est également dans ce service que l'on doit optimiser le nombre de ressources pour le rendre économiquement compétitif.

Cet article est décomposé de la manière suivante : La partie 2 est consacrée à la définition de certains concepts utilisés dans ce type de problème avec les notations adoptées. Dans la partie 3, nous allons proposer une formulation mathématique du problème dans le cas d'une matrice de compétences statique et nous allons démonter la difficulté d'une résolution analytique du problème. La partie 4 sera consacrée à l'implémentation d'un algorithme génétique avec plusieurs variantes, et ayant plusieurs objectifs conflictuels à optimiser : la minimisation du temps total d'exécution (makespan), la maximisation du temps total de formation des opérateurs ou un compromis entre les deux précédents critères. Nous poursuivrons ce travail par une application numérique et une discussion sur les résultats obtenus par application de l'algorithme génétique et l'utilisation des deux variantes de matrice de compétence (statique et dynamique). Une conclusion générale terminera ce travail.

2 DEFINITION DU PROBLEME ET NOTATIONS

2.1 Matrice de compétence dédiée à un atelier de maintenance

On dispose d'une équipe composée de *p* opérateurs de maintenance, ayant chacun des compétences dans *s* spécialités complémentaires. Ces opérateurs constituent les ressources essentielles pour la réalisation des *n* tâches de maintenance, ils possèdent aussi un niveau de maîtrise pour chacune des *s* spécialités. La matrice de compétence *Comp* est composée de *s* lignes pour les spécialités et *p* colonnes pour les opérateurs, les éléments de cette matrice $Comp_{ij}$ représentent le niveau de maîtrise de chaque opérateur pour chaque spécialité, leurs valeurs peuvent varier de la façon suivante :

- $Comp_{ij} = 0$: l'opérateur *j* n'a aucune maîtrise pour la réalisation d'une opération qui demande la spécialité *i*.
- $1 \le Comp_{ij} < 2$: Niveau insuffisant de l'opérateur *j* pour la réalisation individuelle une opération qui demande la spécialité *i*, il doit être accompagné par un opérateur de niveau 3.
- $2 \le Comp_{ij} < 3$: Le niveau de l'opérateur *j* est suffisant pour effectuer d'une manière autonome une opération qui demande la spécialité *i* pour la réaliser.
- $Comp_{ij} = 3$: Le niveau de l'opérateur *j* est suffisant pour la réalisation de toute opération qui demande la spécialité *i* mais aussi l'accompagnement d'un opérateur de niveau 1 durant sa formation.
-

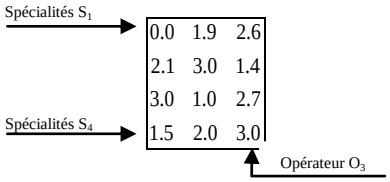

Figure 1 : Exemple de la matrice compétence *Comp* pour 4 spécialités et 3 opérateurs.

Les matrices de compétence sont largement utilisées dans le milieu automobile. Elles permettent, dans le cadre d'un plan de formation, d'indiquer quelles sont les personnes à former et dans quelles spécialités, et ceci en fonction des objectifs de production fixés par l'entreprise. Il existe deux types de matrices de compétence, statiques et dynamiques [Boschian et Stock, 2011]. Dans le cas des matrices statiques, le niveau de maîtrise des opérateurs pour chaque spécialité reste fixe tout au long de l'horizon d'ordonnancement. Dans le cas des matrices dynamiques, le passage d'un niveau de compétence à un autre s'effectue en réalisant des tâches par apprentissage. Cette différence jouera un rôle important pour la réduction du temps d'exécution total des opérations, qui est l'un des objectifs de notre étude.

Dans une spécialité donnée, un opérateur de niveau 1 pourra passer au niveau 2 par la voie de l'apprentissage, en travaillant en qualité d'apprenant, avec un opérateur de niveau 3 qui sera son formateur. Dès que son niveau de compétence, qui évolue de manière linéaire, sera obtenu, ce changement de niveau va s'effectuer sous condition de réussite à un contrôle d'aptitude. En cas

d'échec (à l'instant t_1), l'apprenant continuera son apprentissage. En cas de réussite (à l'instant t_2), il deviendra niveau 2 et pourra exécuter des opérations dans la spécialité considérée de manière autonome.
L'opérateur de niveau 2 deviendra niveau 3 grâce à son travail en autonomie dans la spécialité considérée. On suppose que l'évolution de ses compétences va s'effectuer de manière linéaire. Le passage au niveau supérieur est conditionné par un contrôle d'aptitude (instants t_3 et t_4 sur la figure 2.

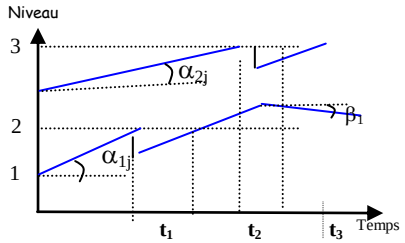

Figure 2. Evolution des compétences.

Nous poserons les hypothèses complémentaires suivantes :

- Un opérateur de niveau égal 0 dans une spécialité donnée ne pourra jamais passer à un niveau supérieur par l'apprentissage.
- Le passage d'un niveau à un autre de l'opérateur *j* s'effectue soit par accomplissement du travail avec un opérateur de niveau 3 si $1 \le Comp_{ij} < 2$, soit par l'expérience professionnelle si $2 \le Comp_{ij} < 3$ et ceci en une durée de temps connue. De plus, ces passages sont conditionnés par la réussite à un examen d'aptitude.
- Lorsqu'un opérateur de niveau $1 \le Comp_{ij} < 2$ collabore avec un de niveau 3, ils accomplissent ensemble une seule opération.
- On suppose que les opérateurs disposent de l'infrastructure nécessaire pour la réalisation des tâches.
- Les opérateurs sont considérés comme des ressources disjonctives.
- Les opérateurs sont disponibles jusqu'à la fin des opérations.
- Le temps d'exécution des opérations est connu et ne dépend pas de l'opérateur (les opérateurs) qui vont l'exécuter.
- Il n'y a pas de priorité d'exécution sur les tâches de maintenance à effectuer.

notation	*Définition*
n_T	*nombre de tâches*
s	*nombre de spécialités*
p	*nombre d'opérateurs*
K_i	*nombre d'opérations de la tâche T_i*
$T=\{T_1,\ldots, T_n\}$	*ensemble des tâches*
$Op=\{Op_1,\ldots,Op_p\}$	*ensemble des opérateurs*
$S=\{S_1,\ldots,S_s\}$	*ensemble des spécialités*
$O_i=\{O_{i1},\ldots,O_{iKi}\}$	*ensemble des opérations de la tâche T_i*
C_j	*la date de fin d'exécution de la tâche T_j*
C_{max}	*le Makespan*
tf_{ik}	*le temps de formation de l'opérateur Op_i assuré par l'opérateur Op_k*
Tf	*le temps total de formation*
R_{ij}^m	*ensemble des m ressources dont les compétences permettent l'exécution de O_{ij}*
Γ_{ij}^+	*ensemble des opérations qui succèdent à O_{ij}*
Γ_{ij}^-	*ensemble des opérations qui précédent O_{ij}*

Figure 3 : Notations utilisées pour le problème

3 MODELE MATHEMATIQUE POUR LE CAS STATIQUE

3.1 Formulation du problème

Notre formulation du problème d'ordonnancement des tâches de maintenance est inspirée de celle de Manne [Manne, 1960], reprise notamment par Vilcot [Vilcot et al, 2008] que nous avons modifiée afin de tenir compte de la compétence des opérateurs, de l'affectation des ressources et de la fonction d'apprentissage.
Les variables utilisées sont :

t_{ij} : La date de début de l'opération O_{ij}

Sous les contraintes suivantes :

$$\forall i = 1\ldots n_T, \forall j = 1\ldots n_O : t_{ij} \ge 0 \qquad (1)$$

Ce type de contrainte (1) indique que les dates de début d'exécution des opérations sont de type réel.

$$y_{cd}^{ab,k} = \begin{cases} 0 & si\ O_{ab}\ est\ effectuée\ avant\ O_{cd}\ sur\ la\ ressource\ R_k \\ 1 & dans\ le\ cas\ contraire \end{cases}$$

$$x_{ij}^k = \begin{cases} 1 & si\ O_{ij}\ est\ effectuée\ sur\ la\ ressource\ R_k \\ 0 & dans\ le\ cas\ contraire \end{cases}$$

$$\forall i = 1\ldots n_T, \forall j = 1\ldots n_O : t_{ij+1} \ge t_{ij} + p_{ij} \qquad (2)$$

Les contraintes de type (2) représentent les gammes opératoires.

$$\forall i = 1\ldots n_T, \forall j = 1\ldots n_O : \sum_{\forall R_k \in R_{ij}^m} x_{ij}^k = 1 \qquad (3)$$

Les contraintes de type (3) assurent que chaque opération est traitée par une seule ressource (binôme ou monôme).

$\forall a = 1...n_T, \forall b = 1...n_O, \forall c = 1...n_T, \forall d = 1...n_O;$

$\forall R_k \in R_{ab}^m \cap R_{cd}^m$ *avec*

$R_{ab}^m = (\sum_{t=1}^{p} R_{t,ab} \,|\, (Comp(R_{t,ab}) \geq 2) \cup (\sum_{t=1}^{p} \sum_{s=1, s\neq t}^{p} R_{t+s,ab} \,|\, (Comp(R_{t,ab}) = 1), (Comp(R_{t,ab}) = 3))$

$t_{cd} \geq t_{ab} + p_{ab} - M.y_{cd}^{ab,k} - M.(2 - (x_{ab}^k + x_{cd}^k))$ (4)

$\forall a = 1...n_T, \forall b = 1...n_O, \forall c = 1...n_T, \forall d = 1...n_O;$

$\forall R_k \in R_{ab}^m \cap R_{cd}^m$ *avec*

$R_{ab}^m = (\sum_{t=1}^{p} R_{t,ab} \,|\, (Comp(R_{t,ab}) \geq 2) \cup (\sum_{t=1}^{p} \sum_{s=1, s\neq t}^{p} R_{t+s,ab} \,|\, (Comp(R_{t,ab}) = 1), (Comp(R_{t,ab}) = 3))$

$t_{ab} \geq t_{cd} + p_{cd} - M.(1 - y_{cd}^{ab,k}) - M.(2 - (x_{ab}^k + x_{cd}^k))$ (5)

Les deux contraintes (4) et (5) symbolisent les disjonctions qui relient les opérations réalisées sur la même ressource et ceci en introduisant les variables booléennes $y_{cd}^{ab,k}$. M est une constante très grande, généralement égale à la somme des durées d'exécution de toutes les opérations. Soit $tf_{i,k}$ le temps de formation de l'opérateur i assuré par l'opérateur k. Les fonctions-objectifs à optimiser sont :

$C_{\max} \geq t_{ij} + p_{ij} \;\; \forall i = 1...n_T, \forall j = 1...n_O$ (6)

$\mathrm{Tf} = \sum_{i=1}^{p} \sum_{k=1/i\neq k}^{p} tf_{ik}$ (7)

(6) Le makespan.
(7) le temps total de formation des opérateurs.

3.2 Caractéristiques du modèle

Le modèle proposé est caractérisé par l'utilisation de deux types de variable. Des variables continues t_{ij} dont le nombre est égal au nombre total d'opération $\sum_i^{n_T} n_i$. Des variables binaires, $y_{cd}^{ab,k}$ et x_{ij}^k avec un nombre égal à $(\sum_i^{n_T} n_i)^2.(n_r)$ et $(\sum_i^{n_T} n_i).(n_r)$ pour $y_{cd}^{ab,k}$ et x_{ij}^k respectivement. Avec un nombre aussi grand de variables, le modèle est constitué d'un nombre très important de contraintes, égal à $2.(\sum_i^{n_T} n_i) + (\sum_i^{n_T} n_i - n_T)^2.n_R + 2.((\sum_i^{n_T} n_i).n_R)$. Ce qui fait augmenter la taille du modèle en augmentant le nombre de tâches, d'opérations et de ressources, et ceci rend le modèle difficile à résoudre en utilisant les solveurs.

4 METHODE APPROCHEE (ALGORITHME GENETIQUE)

Vu la difficulté pour résoudre ce type de problème à l'aide d'une méthode exacte, nous avons choisi d'adopter une méta-heuristique d'un type évolutionnaire qui est l'Algorithme Génétique (AG). Nous pouvons résumer le principe d'un Algorithme Génétique par les étapes suivantes :

```
Début :
    Génération aléatoire d'une population d'individus
    Répéter
        Evaluer chaque individu en fonction des objectifs
        Sélectionner les individus pour le croisement
        Croiser les individus sélectionnés
        Muter certains individus
        Jusque : critère(s) de fin
        Retourner le meilleur compromis
    Fin
```

Figure 4 : Le principe de AG

4.1 Génération de la population initiale et codage

Nous avons choisi une génération aléatoire de la population de départ en utilisant un codage en deux parties. Ce codage est inspiré de celui utilisé dans [Kebabla, 2008]. Dans un premier temps, nous créons, une séquence ordonnée des opérations pour chaque individu. Chaque apparition du numéro de la tâche correspond à l'une des opérations qui la constituent. La figure 4 représente une séquence de trois tâches constituées chacune de trois opérations.

O_{11}	O_{31}	O_{32}	O_{21}	O_{12}	O_{22}	O_{33}	O_{23}	O_{13}
1	3	3	2	1	2	3	2	1

Figure 5 : Exemple d'une séquence d'opérations générée aléatoirement

Dans un deuxième temps, nous affectons les ressources aux différentes opérations. Pour ce faire, nous utilisons la *matrice de disponibilité* des ressources possibles créée à l'aide de la *matrice de compétence*. Selon le type de matrice de compétence utilisée, on distingue les cas suivants :

Cas statique : pour le cas statique, les ressources possibles pour réaliser les opérations sont identiques pendant l'horizon d'ordonnancement. Donc, nous choisissons d'une façon aléatoire une des ressources (un opérateur ou un couple d'opérateur) parmi celles possibles.

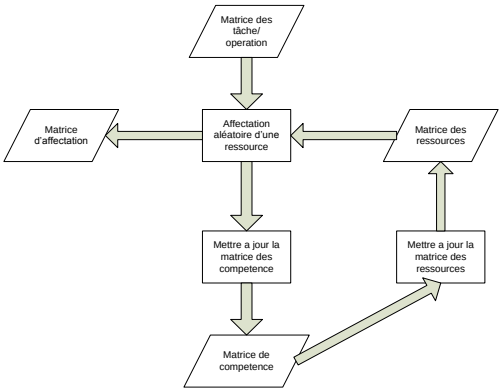

Figure 6 : Schéma de construction de la matrice d'affectation pour le cas de matrice de compétence dynamique

Cas dynamique : pour le cas dynamique, nous devons prendre en compte l'augmentation de niveau éventuel des opérateurs, après l'accomplissement des opérations auxquelles ils ont été affectés. Les ressources possibles pour réaliser chaque opération ne sont pas les mêmes après chaque opération. La matrice des ressources est mise à jour après l'achèvement de chaque opération. La

figure 5 résume la procédure d'affectation. Dans les deux cas (statique ou dynamique), le résultat de l'affectation est une matrice qu'on appelle *matrice d'affectation*, qui comporte un nombre de lignes égal au nombre d'opérateurs et un nombre de colonnes correspondant au nombre total des opérations. Pour chaque colonne, on affecte le numéro de la tâche à la ligne (aux lignes) qui correspond (ent) à l'opérateur (aux opérateurs) qui lui a (ont) été affecté. Cette affectation se fait d'une façon aléatoire, en choisissant une ressource (binôme ou monôme d'opérateur) parmi les ressources possibles. La figure 6 représente la *matrice d'affectation* de la séquence d'opérations représentées dans la figure 4.

	O_{11}	O_{31}	O_{32}	O_{21}	O_{12}	O_{22}	O_{33}	O_{23}	O_{13}
Op_1	1	3	0	0	1	2	3	2	0
Op_2	0	0	3	0	1	0	0	2	0
Op_3	0	3	0	2	0	0	3	0	1

Figure 7 : Exemple d'une matrice d'affectation pour 9 opérations et 3 ressources

Ce type de codage ne nécessite pas de correction des individus après le couplage, problème qui s'est posé dans d'autres types de codage.

4.2 Le croisement des individus

Nous avons utilisé un couplage avec un point de croisement choisi aléatoirement sur les colonnes et qui devra dépasser la moitié de la liste des opérations. De cette façon l'enfant aura les caractéristiques qui correspondent au père duquel il a pris la grande partie, et ceci sur la *séquence des opérations* et sur la *matrice d'affectation*. Donc pour chacune d'entre elles, nous coupons les deux pères à la position choisie et nous recopions la grande partie du père 1 et du père 2, sur les deux fils, respectivement fils 1 et fils 2. Puis nous recopions ce qui reste du père 1 dans le fils 2, selon l'ordre d'apparition dans le père 1, et ce qui reste du père 2 dans le fils 1 selon l'ordre d'apparition dans le père 2. L'exemple de la figure 7 montre la façon de coupler deux individus parents afin d'obtenir deux individus fils.

O_{21} O_{22} O_{11} O_{23} O_{31} O_{12} O_{32} O_{33} O_{13}

2 2 1 2 3 | 1 3 3 1

	O_{21}	O_{22}	O_{11}	O_{23}	O_{31}	O_{12}	O_{32}	O_{33}	O_{13}
Op_1	0	2	0	2	3	1	3	0	0
Op_2	2	2	0	0	0	1	0	3	1
Op_3	0	0	1	2	0	0	3	0	1

Père 1

O_{11} O_{31} O_{32} O_{21} O_{12} O_{22} O_{33} O_{23} O_{13}

1 3 3 2 1 | 2 3 2 1

	O_{11}	O_{31}	O_{32}	O_{21}	O_{12}	O_{22}	O_{33}	O_{23}	O_{13}
Op_1	1	3	0	0	1	2	3	2	0
Op_2	0	0	3	0	1	0	0	2	0
Op_3	0	3	0	2	0	0	3	0	1

Père 2

O_{21} O_{22} O_{11} O_{23} O_{31} O_{32} O_{12} O_{33} O_{13}

2 2 1 2 3 | 3 1 3 1

	O_{21}	O_{22}	O_{11}	O_{23}	O_{31}	O_{32}	O_{12}	O_{33}	O_{13}
Op_1	0	2	0	2	3	0	1	3	0
Op_2	2	2	0	0	0	3	1	0	0
Op_3	0	0	1	2	0	0	0	3	1

Enfant 1

O_{11} O_{31} O_{32} O_{21} O_{12} O_{22} O_{23} O_{33} O_{13}

1 3 3 2 1 | 2 2 3 1

	O_{11}	O_{31}	O_{32}	O_{21}	O_{12}	O_{22}	O_{33}	O_{23}	O_{13}
Op_1	1	3	0	0	1	2	2	0	0
Op_2	0	0	3	0	1	2	0	3	1
Op_3	0	3	0	2	0	0	2	0	1

Enfant 2

Figure 8 : Exemple illustratif d'un croisement de deux individus

4.3 La mutation

La mutation est une opération qui joue un rôle important pour garder la diversité de la population, et ceci en empêchant la convergence rapide vers un seul genre d'individu. Dans notre travail, nous avons adopté une mutation par changement de position (une permutation). Deux opérations sont choisies aléatoirement puis nous permutons leurs positions et ceci avec un taux de mutation faible prédéfini. Sur exemple, nous appliquons cette mutation au père 2. Nous allons permuter les opérations des positions 2 et 9.

O_{11} O_{31} O_{32} O_{21} O_{12} O_{22} O_{33} O_{23} O_{13}

1 3 3 2 1 2 3 2 1

	O_{11}	O_{31}	O_{32}	O_{21}	O_{12}	O_{22}	O_{33}	O_{23}	O_{13}
Op_1	1	3	0	0	1	2	3	2	0
Op_2	0	0	3	0	1	0	0	2	0
Op_3	0	3	0	2	0	0	3	0	1

Avant la mutation.

O_{11} O_{21} O_{31} O_{22} O_{12} O_{23} O_{32} O_{33} O_{13}

1 2 3 2 1 2 3 3 1

	O_{11}	O_{21}	O_{31}	O_{22}	O_{12}	O_{23}	O_{32}	O_{33}	O_{13}
Op_1	1	2	0	0	1	2	3	3	0
Op_2	0	2	3	0	1	0	0	0	0
Op_3	0	0	0	2	0	0	3	3	1

Après la mutation.

Figure 9 : Exemple illustratif, de la mutation

4.4 L'évaluation

Pour l'évaluation des individus, nous utilisons trois critères qui sont soit la valeur du makespan, soit le temps de formation des opérateurs, soit une combinaison

entre les deux critères précédents pour établir un compromis.

4.5 La sélection

Cette opération est très importante, du fait qu'elle détermine les individus qui vont constituer les nouvelles générations. Notre stratégie de sélection (tournoi) consiste à comparer deux individus choisis aléatoirement et de garder le meilleur pour la suite des opérations.

4.6 Critère d'arrêt.

Le critère d'arrêt que nous avons choisi pour notre algorithme génétique, consiste à fixer un nombre d'itération maximal. Dans la partie application numérique nous allons étudier l'effet du nombre d'itération sur l'efficacité des stratégies choisies.

4.7 Stratégies adaptées.

Pour l'application de l'algorithme génétique sur notre problème, nous avons utilisé trois stratégies différentes.
Stratégie 1 : elle consiste à prendre deux individus d'une manière aléatoire et à les coupler. Nous aurons la possibilité d'avoir tous les cas possibles : mauvais/mauvais, mauvais/bon et bon/bon (bon ou mauvais individu).
Stratégie 2 : identique à la première, à une seule différence, qui consiste à introduire un nouvel opérateur, appelé *opérateur d'immigration*. Il consiste à générer aléatoirement un nouvel individu, et ceci pour donner plus de diversification, et éviter la convergence rapide vers un optimum local.
Stratégie 3 : c'est une stratégie utilisée par Goncalves [Goncalves et al, 2008]. Elle consiste

- à recopier les meilleurs individus de la population présente vers la nouvelle génération avec un pourcentage fixé,
- à générer aléatoirement des individus et à les injecter dans la nouvelle génération avec un certain pourcentage fixé,
- le reste des individus constituant la nouvelle génération sont sélectionnés parmi la population générée par l'application de la *stratégie* 1 sur toute la population.

La figure 9 représente le passage d'une génération à une autre.

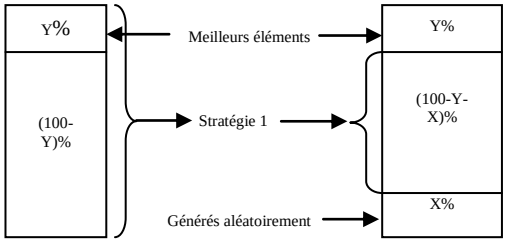

Figure 10 : Illustration de la stratégie 3

5 APPLICATION NUMERIQUE

5.1 Données numériques

Afin de valider les performances de notre méta-heuristique, nous avons choisi de réaliser une application numérique avec les valeurs numériques suivantes :
Les tâches à réaliser seront définies par un benchmark (abz5) défini par Adams [Adams et al, 1998], tout en remplaçant les ressources par les spécialités requises pour l'accomplissement des tâches.
La matrice des compétences dynamique/statique initiale comporte 10 compétences et 5 opérateurs, et est représentée sur la figure 10.

10×5				
0.0	1.5	2.4	3.0	2.7
2.7	3.0	1.7	1.0	3.0
3.0	1.9	2.8	0.0	2.2
1.0	2.0	3.0	3.0	1.2
2.4	0.0	3.0	1.3	1.2
0.0	3.0	1.0	0.0	0.0
0.0	1.3	3.0	0.0	1.7
3.0	1.0	2.1	0.0	2.5
1.3	0.0	2.3	1.8	0.0
2.5	2.3	1.4	3.0	3.0

Figure 11 : La matrice de compétence utilisée pour l'application numérique

La taille de la population initiale est comprise entre 50 et 500 individus, que nous ferons varier pour l'étude des stratégies définies dans la partie 4.7. Le taux de mutation sera de 0.08 et 0.05 et le nombre d'itérations sera compris entre 100 et 1200.

5.2 Résultats et interprétation

Nous débutons les simulations par l'application des différentes stratégies présentées dans la partie 4.7, avec une matrice de compétence statique, dans le but d'optimiser chacun des critères (Cmax et Tf) de manière indépendante. Nous y étudions l'influence des paramètres de l'algorithme génétique ainsi que la performance des différentes stratégies (S1, S2, S31, S32, S33, S34) sur les deux objectifs, afin de pouvoir sélectionner les bons paramètres. Ensuite nous étudions l'application d'une méthode multi-objectif à l'aide de la stratégie et des paramètres déterminés précédemment. Ceci nous permettra de trouver un compromis entre les deux critères à optimiser. Nous appliquerons ensuite la même stratégie et les mêmes paramètres sur une matrice dynamique, afin d'observer les changements qui interviennent sur la valeur des solutions trouvées pour le Cmax.

Notation

- S1 : stratégie 1.
- S2 : stratégie 2.
- S3 : stratégie 3, avec :

- S31 pour 1% d'individus gardés parmi les meilleurs, 1% d'individus générés aléatoirement.
- S32 pour 10% d'individus gardés parmi les meilleurs, 1% d'individus généré aléatoirement.
- S33 pour 1% d'individus gardés parmi les meilleurs, 10% d'individus générés aléatoirement.
- S34 pour 10% d'individus gardés parmi les meilleurs, 10% d'individus générés aléatoirement.

5.2.1 Le cas Statique

1. Objectif monocritère

1.1. Minimisation du makespan (C_{max}).

Dans un premier temps, nous allons fixer le nombre d'itération à 1200 et nous allons faire varier la taille de la population et ceci avec deux taux de mutation différents : 0,08 (Figure 11) et 0,05 (Figure 12).

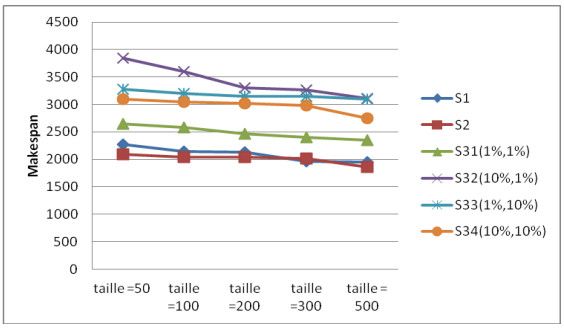

Figure 12 : Evolution de Cmax en fonction de la stratégie et de la taille de population.

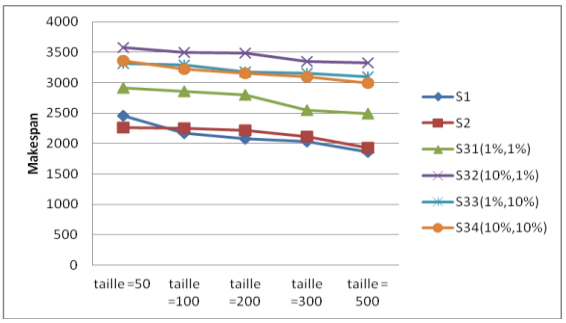

Figure 13 : Evolution de Cmax en fonction de la stratégie et de la taille de population

La figure 13 nous montre l'influence de la variation du nombre d'itérations sur la valeur du makespan avec un taux de mutation de 0.05 et une génération de 500 individus.

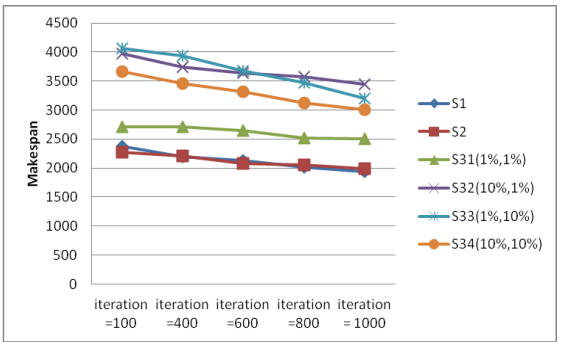

Figure 14 : Evolution de la qualité des solutions données pour le Cmax

1.2. Maximisation du temps de formation (*Tf*) :

Nous allons faire varier la taille de la population avec 1200 itérations et deux valeurs du taux de mutation (0.05 et 0.08). Nous obtenons les figures 14 et 15.

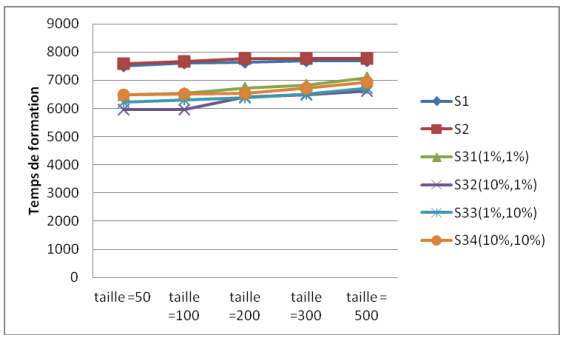

Figure 15 : Evolution de la qualité des solutions pour le Tf

Pour 1200 itération et un taux de mutation de 0.05 :

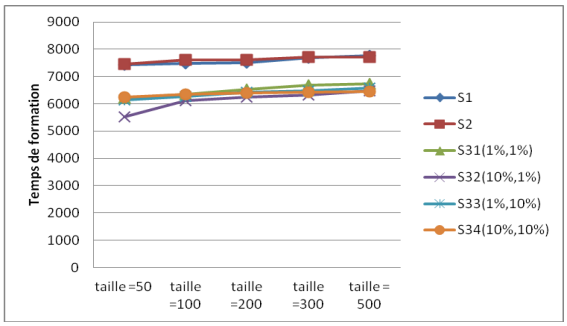

Figure 16 : Evolution de la qualité des solutions données pour le Tf

Pour la suite, nous allons adopter un taux de mutation de 0.05, une génération de 500 individus et nous faisons varier le nombre d'itérations.

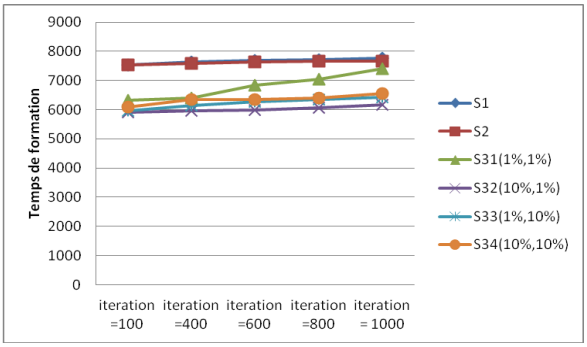

Figure 17 : Evolution de la qualité des solutions données pour le Tf

1.3. Interprétation des résultats.

Sur les figures 12 à 17, nous constatons que :

- Les deux stratégies S1 et S2 nous donnent de meilleurs résultats par rapport à la stratégie S3. L'écart entre les solutions trouvées par S1 et S2 et celles trouvées par S3, pour le temps de formation, est plus réduit que celui trouvé pour le makespan.
- L'augmentation de la taille de la population fait converger toutes les stratégies vers de meilleures solutions.
- Pour une petite taille de la population, la diversification par la génération aléatoire de nouveaux individus améliore la qualité des solutions.
- Pour la stratégie 3, un nombre important d'élites gardés pour la génération suivante, réduit la qualité des solutions sans diversification. Nous pouvons interpréter cela par une convergence rapide vers un optimum local.
- La diminution du taux de mutation peut jouer un rôle important pour l'amélioration de la qualité des solutions.

A titre indicatif, le temps d'exécution des stratégies 1 et 2 varie entre 9 et 116 secondes, en fonction de la taille de la population et du nombre d'itérations. Le temps d'exécution de la stratégie 3 est plus élevé et varie entre 100 et 5500 secondes selon la taille population et le nombre d'itérations. Nous avons exécuté ce programme sur un Intel Xeon CPU W3550 3.07 GHz.

Dans la suite de notre article, nous allons utiliser la stratégie 1 avec une taille de population de 500 individus et un nombre d'itérations de 1200.

2. Objectif multicritère.

Afin de trouver un compromis entre les deux critères déjà optimisés individuellement, nous adoptons la notion de désirabilité (Harrington, 1965). Derringer [Derringer et Suich, 1980] donne une évaluation du désir partiel de chaque critère par rapport à deux valeurs, la première fixée par le gestionnaire, elle est considérée comme limite extrême (supérieure ou inferieur, $y_{(\sup \text{ ou } \inf)}$) et la deuxième considérée comme valeur cible (y_c) obtenue par l'optimisation de chaque critère séparément. Cette évaluation est modélisée par Derringer avec une courbe dont l'équation est :

$$di = \left[\frac{y_i - y_{(\sup ou \inf)}}{y_c - y_{(\sup ou \inf)}} \right]^T \qquad (8)$$

Où : di est entre la valeur de $d_{cible} = 1$ et $d_{\sup} = 0$ resp ($d_{\inf} = 0$).

Si la réponse obtenue est égale à la valeur cible, nous dirons que l'objectif est atteint à 100% et le désir partiel vaut 1. Dans le cas où la valeur obtenue est supérieure ou égale à la limite supérieure, on dira que l'objectif du désir partiel est atteint à 0% et le désir partiel vaut 0.

T est un coefficient de forme, son rôle consiste à valoriser les réponses.

Nous disposons de deux fonctions à optimiser. La première est la minimisation du makespan, pour laquelle nous allons prendre comme valeur cible 1863 u.t et nous supposons que la valeur limite fixée par le gestionnaire est de 3300 u.t. Le deuxième critère est la maximisation du temps de formation, pour lequel nous allons prendre comme valeur cible 7773 u.t, la limite inférieure fixée par l'opérateur est de 5000 u.t.

Les courbes correspondantes à l'équation 8 pour les deux critères sont représentées sur les figures 17.

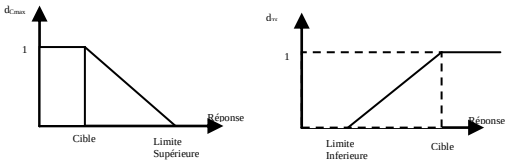

Figure 18 : Désirabilité du makespan et du temps de formation.

Le résultat de chaque équation est donné par le désir partiel d_i ($0 \le d_i \le 1$). Ces derniers sont transformés en désir global D que nous devons maximiser. Chaque désir partiel est pondéré par un poids w_i choisi par le gestionnaire de production. La fonction du désir global D est donnée par :

$$D = \sqrt[w]{d_1^{w_1}.d_2^{w_2}\ldots d_m^{w_m}} \quad \text{Avec} \quad w = \sum_{i=1}^{m} w_i$$

Les résultats obtenus en faisant varier les différents poids sont résumés dans le tableau suivant :

	Cas 1	***Cas 2***	***Cas 3***	***Cas 3***
Poids du makes-pan	4	3	2	1
Poids du temps de formation	1	2	3	4
Makespan	2478	2752	3224	3186
Temps de formation	6694	6895	7175	7416
Désir partiel du makespan	0.80	0.71	0.56	0.58
Désir partiel du	0.61	0.68	0.78	0.87

temps de formation				
Désir global D	0.76	0.70	0.69	0.80

Figure 19 : Désirabilité en fonction du poids de chaque objectif

Nous remarquons que le poids donné au désir du makespan a une influence moins importante que celui donné au désir du temps de formation sur la valeur du désir global et sur les valeurs obtenues pour les deux fonctions à optimiser.

5.2.2 Le cas Dynamique.

Dans cette partie, nous allons utiliser la matrice de compétences dynamique pour résoudre le problème dont l'objectif est de minimiser le makespan. Nous pourrons ainsi comparer les résultats trouvés avec les deux types de matrice de compétences. Les résultats trouvés pour chaque cas sont donnés dans le tableau suivant :

	Critère	*Cas statique*	*Cas dynamique*	*La réduction pour le C_{max} en %*
A la création des individus	Cmax	3983	2978	23.37%
	Tf	5267	2992	
Apres 1200 itérations	Cmax	2003	1769	11.68%
	Tf	3189	2823	

Figure 20 : L'évolution de la solution obtenue par les deux types de matrice de compétence

Nous notons la diminution de la valeur du makespan en utilisant une matrice compétence dynamique. Ceci est dû à l'augmentation du niveau de compétence de certains opérateurs. Cette augmentation des niveaux influe directement sur la disponibilité des ressources. La figure 20 nous montre l'évolution des ressources sur la matrice de compétence.

1	0.0 1.5 2.4 3.0 2.7	0.0 3.0 2.4 3.0 2.7	0.0 3.0 3.0 3.0 2.7
2	2.7 3.0 1.7 1.0 3.0	3.0 3.0 2.8 2.0 3.0	3.0 3.0 2.8 3.0 3.0
3	3.0 1.9 2.8 0.0 2.2	3.0 3.0 3.0 0.0 3.0	3.0 3.0 3.0 0.0 3.0
4	1.0 2.0 3.0 3.0 1.2	2.3 2.9 3.0 3.0 3.0	2.3 3.0 3.0 3.0 3.0
5	2.4 0.0 3.0 1.3 1.2	3.0 0.0 3.0 1.3 3.0	3.0 0.0 3.0 2.5 3.0
6	0.0 3.0 1.0 0.0 0.0	0.0 3.0 3.0 0.0 0.0	0.0 3.0 3.0 0.0 0.0
7	0.0 1.3 3.0 0.0 1.7	0.0 3.0 3.0 0.0 3.0	0.0 3.0 3.0 0.0 3.0
8	3.0 1.0 2.1 0.0 2.5	3.0 2.4 3.0 0.0 3.0	3.0 3.0 3.0 0.0 3.0
9	1.3 0.0 2.3 1.8 0.0	1.3 0.0 3.0 3.0 0.0	3.0 0.0 3.0 3.0 0.0
10	2.5 2.3 1.4 3.0 3.0	2.5 3.0 1.4 3.0 3.0	3.0 3.0 3.0 3.0 3.0
	La matrice du départ	après 50 opérations	la matrice a la fin des opérations

Figure 21 : L'évolution de la matrice de compétence

Prenons par exemple la spécialité 8 représentée en rouge sur les matrices dans la figure 20. Nous remarquons l'évolution des ressources. Les opérateurs deviennent autonomes (niveau 2) puis passent à un niveau de formateur dans cette spécialité (niveau 3). Les diagrammes de Gantt de la figure 21 montrent la variation du makespan avec les deux types de matrices.
Il nous est impossible de comparer les résultats obtenus avec ceux d'autres algorithmes (par exemple [Applegate et al., 1991]) car dans notre cas, une compétence donnée peut être réalisée par un ou plusieurs opérateurs : on peut donc avoir de multiples tâches qui sont réalisées en parallèle.

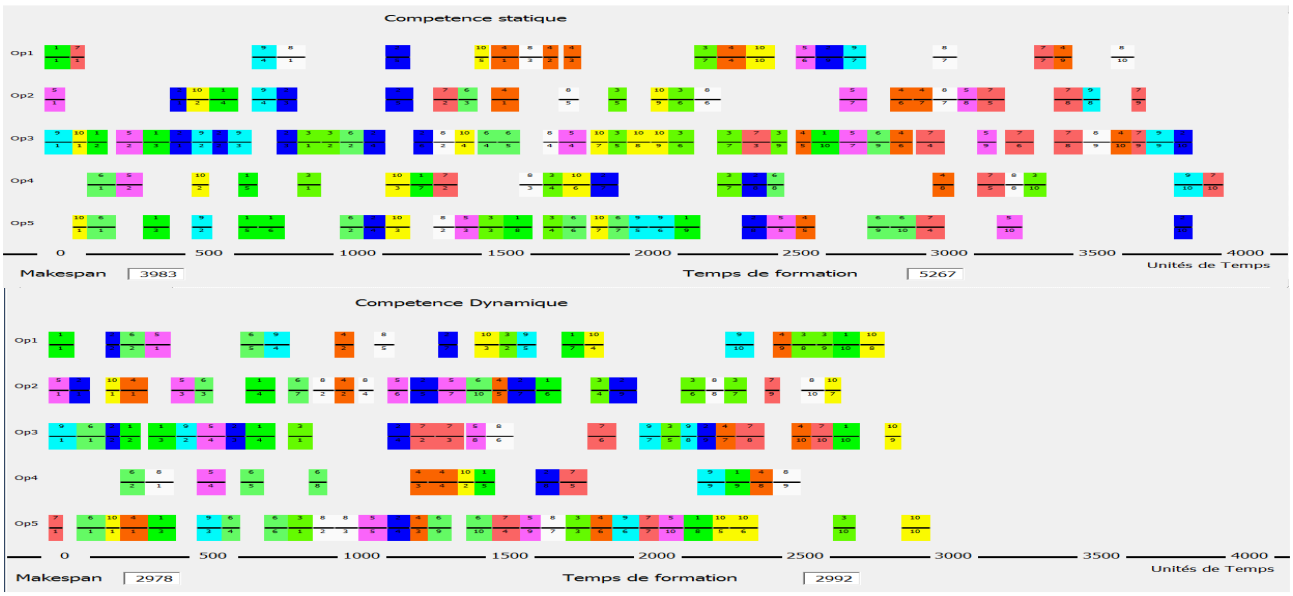

Figure 22 : Gantt pour la matrice Statique et Dynamique

6 CONLUSION

Dans cet article, nous avons traité le problème de la gestion de projet avec des ressources renouvelables multi compétentes. Notre étude a débuté par la définition des outils nécessaires à l'étude du problème ainsi que du concept de matrice de compétence. Nous avons ensuite proposé une formulation mathématique du problème considérant uniquement des compétences statiques. Vu la complexité à résoudre de manière analytique ce problème pour des instances de grandes tailles, notre choix s'est tourné vers l'utilisation d'une méta-heuristique (algorithme génétique). Nous avons ainsi pu utiliser la matrice de compétence avec ses deux variantes (statique et dynamique) avec lesquelles nous avons pu observer l'optimisation du makespan. Nous avons utilisé aussi la notion de désirabilité afin de trouver un compromis entre les deux critères d'optimisation, conflictuels, que sont le makespan et le temps de formation. Une extension future consistant à intégrer la sous-traitance est en cours de développement.

REMERCIEMENTS

Les auteurs remercient Mme Nathalie SAUER pour ses suggestions et commentaires.

REFERENCES

Adams J., Balas E. et Zawack D., 1998. The shifting bottleneck procedure for job shop scheduling. Management Science, 34, pp 391-401.

Applegate D. et Cook W., 1991. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, 3 (2), pp. 149-156.

Bellenguez-Morineau. O., 2006. Méthodes de résolution pour un problème de gestion de projet multi-compétence. Doctorat Université Tours.

Boschian-Campaner V., Stock R., « Amélioration de l'efficience d'un ordonnancement des tâches par la prise en compte des desiderata des opérateurs », 9ème Congrés International de Génie Industriel, CIGI2011, Saint-Sauveur, Québec, Canada, 12, 13 et 14 octobre 2011

Carlier J. et Latapie B., 1991. Une méthode arborescente pour résoudre les problèmes cumulatifs. RAIRO – Recherche opérationnelle, 25(3).

Derringer G. et Suich R., 1980. Simultaneous Optimization of several Response Variables. Journal of Quality Technology, 12(4), pp. 214-219.

Dupas Rémy., 2004. Amélioration de performance des systèmes de production : apport des algorithmes évolutionnistes aux problèmes d'ordonnancement cycliques et flexibles. Habilitation à Diriger des Recherches. Université d'Artois.

Goncalves J.F., Mendes J.J.M., Resende M.G.C., 2008. A genetic algorithm for the resource contrained multi-project scheduling problem. Operations Research.

Harrington E. C. Jr, 1965. The Desirability Function. Industrial Quality Control, 21 (10), pp. 494-498.

Hartmann S., 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics, 49:433–448.

Kacem I., 2003. Ordonnancement multicritère des job-shops flexibles : formulation, bornes inférieures et approche évolutionniste coopérative, Thèse de doctorat, Université de Lille 1.

Kebabla M., 2008. Utilisation des stratégies méta heuristiques pour l'ordonnancement des ateliers de type Job Shop. Magister, Génie industriel, Batna.

Kelley J.E., 1963. The Critical-Path Method: Resources Planning and Scheduling, Muth J.F. Thompson G.L. (Eds.): Industrial Scheduling. Prentice Hall, Englewood Cliffs,: 347-365.

Klein R., 2000. Scheduling of resource-constrained projects, Kluwer Academic Publishers.

Koné. O., 2009. Nouvelles approches pour la résolution du problème d'ordonnancement de projet à moyens limités. Doctorat de l'université de Toulouse.

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

Patterson J., Slowinski R., Talbot F. et Weglarz J., 1990. Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems. European Journal of Operational Research, 49: 68–79.

Pinson E., Prins C. et Rullier F., 1994. Using tabu search for solving the resource constrained project scheduling problem. Proceedings of the Fourth International Workshop on Project Management and Scheduling, pages 102–106.

Pritsker A., Watters L. and Wolfe P., 1969. Multiproject scheduling with limited resources: a zero-one programming approach, Management science, 16: 93-107.

Sprecher A., 2000. Scheduling resource-constrained projects competitively at modest memory requirements. Management Science, 46:710–723.

Stinson J. P., Davis E. W. et Khumawala B. M., 1978. Multiple resource-constrained scheduling using branch-and-bound. AIIE Transactions, 10(3):252–259.

Vilcot G., Kergosien Y., Janvier J., Billaut J.-C, 2008. Une recherche tabou et un algorithme génétique pour un problème de job shop multiressource multicritère. 7e Conférence Internationale de Modélisation et Simulation.