7e Conférence Internationale 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.».

ALGORITHME GÉNÉTIQUE AVEC CROISEMENTS HYBRIDES UTILISANT LA PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS

Arnaud ZINFLOU, Caroline GAGNÉ et Marc GRAVEL

Département d'Informatique et de Mathématique
Université du Québec à Chicoutimi
555, Boul. de l'Université
Chicoutimi, Québec, Canada G7H 2B1
arnaud_zinflou@uqac.ca, caroline_gagne@uqac.ca, marc_gravel @uqac.ca

RÉSUMÉ : *Dans cet article, nous présentons trois approches hybrides intégratives permettant de résoudre efficacement le problème théorique d'ordonnancement de voitures. Ces approches reposent principalement sur un algorithme génétique intégrant deux croisements utilisant la programmation linéaire en nombres entiers. La nature différente des deux croisements hybrides proposés a permis de les combiner efficacement et de montrer que l'algorithme résultant permettait d'obtenir des résultats compétitifs, en terme de qualité de solution, par rapport à un algorithme génétique utilisant une recherche locale. L'ensemble des travaux réalisés démontre bien l'intérêt de concevoir des approches hybrides exploitant les forces de différentes méthodes.*

MOTS-CLÉS : *ordonnancement, algorithme génétique, programmation linéaire, hybridation, car sequencing.*

1. INTRODUCTION

L'hybridation consiste à combiner les avantages de différentes méthodes d'optimisation en un seul algorithme. Selon Talbi (2002), ce type d'approches a permis d'obtenir de très bons résultats dans une grande variété de problèmes d'optimisation combinatoire théoriques tels le problème du voyageur de commerce et le problème d'assignation quadratique ainsi que dans des contextes réels. De nombreuses approches hybrides ont alors vu le jour, la plupart hybridant des métaheuristiques à base de populations avec des métaheuristiques à solution unique. Selon Blum et al. (2005), l'hybridation de métaheuristiques est la voie la plus prometteuse pour l'amélioration de la qualité des solutions dans beaucoup d'applications.

Récemment, un nombre croissant de travaux proposent de faire coopérer des métaheuristiques et des méthodes exactes (Barichard, 2003 ; Basseur, 2004). D'une manière générale, les approches de résolution par des méthodes exactes se limitent à de petites instances pour les problèmes d'optimisation combinatoire difficiles. Par contre, la combinaison métaheuristique /méthode exacte peut devenir une alternative très efficace, car les deux méthodes ont des particularités bien différentes et associent leurs avantages pour produire de meilleurs résultats (Basseur, 2004). Malheureusement, selon Jussien et Laburthe (2004), ce type d'hybridation n'est pas une tâche aisée autant pour des raisons culturelles que technologiques.

Dans cet article, nous proposons d'explorer certains mécanismes d'hybridation entre une métaheuristique à base de population et une méthode exacte pour solutionner le problème théorique d'ordonnancement de voitures sur une chaîne d'assemblage (*car-sequencing problem*). L'organisation de l'article est comme suit : à la section suivante, nous présentons les principales classifications des stratégies d'hybridation métaheuristiques/méthode exactes; à la Section 3, nous présentons brièvement le problème théorique d'ordonnancement de voitures et les principales approches de solution retrouvées dans la littérature; à la Section 4, nous décrivons la forme d'hybridation proposée entre un algorithme génétique et la programmation linéaire en nombres entiers; à la Section 5, nous présentons les résultats numériques obtenus sur les problèmes retrouvés dans CSPLib. Finalement, quelques conclusions et remarques sont présentées à la dernière section.

2. CLASSIFICATION DES STRATÉGIES D'HYBRIDATION MÉTAHEURISTIQUES / MÉTHODES EXACTES

Les études sur l'hybridation entre métaheuristiques et méthodes exactes sont moins nombreuses que celles portant sur l'hybridation de deux métaheuristiques. Toutefois, l'intérêt grandissant des chercheurs pour l'hybridation entre métaheuristiques et méthodes exactes et le nombre croissant de méthodes proposées a récemment amené certains auteurs à proposer des classifications. Parmi celles-ci, on peut citer la classification de Dumitrescu et Stützle (2003), celle Basseur (2004) ou encore celle de Puchinger et Raidl (2005).

Dans leur classification, Dumitrescu et Stützle (2003) distinguent cinq types d'approches de coopération métaheuristiques/méthodes exactes et détaillent une application trouvée dans la littérature pour chacune d'elles. La classification introduite par les auteurs sépare les approches selon la méthode utilisée pour la coopération. Les cinq classes proposées par Dumitrescu et Stützle regroupent :

- les approches utilisant un algorithme exact pour explorer de larges voisinages dans un algorithme de recherche locale;
- les approches utilisant une méthode heuristique afin de réduire l'espace de recherche de la méthode exacte;
- les approches exploitant les bornes de la méthode exacte pour une heuristique constructive;
- les approches utilisant les informations fournies par les relaxations des problèmes linéaires pour orienter un algorithme de recherche locale ou constructif; et
- les approches utilisant une méthode exacte pour remplacer une fonction spécifique de la métaheuristique.

Cependant, on note que, même si l'état de l'art de Dumitrescu et Stützle survole les principales stratégies de coopération entre méthodes exactes et métaheuristiques, la plupart des exemples proposés par les auteurs utilise des méthodes de recherche locale ou des algorithmes de recherche avec tabou comme métaheuristiques.

Dans sa thèse de doctorat, Basseur (2004) propose une classification plus détaillée pour catégoriser les méthodes hybridant métaheuristiques et méthodes exactes. Cette classification, de type taxonomie, s'inspire de celle proposée par Talbi (2002) pour classifier les métaheuristiques hybrides selon leur niveau d'hybridation. Ainsi, une hybridation est de *bas niveau* lorsque les éléments fonctionnels qui constituent la métaheuristique sont modifiés pour être remplacés par une méthode exacte et inversement. À l'opposé, une hybridation est dite de *haut niveau* lorsque l'intégrité des méthodes hybridées est conservée. Ces deux niveaux peuvent par la suite être subdivisés en deux modes : le *mode relais* et le *mode simultané*. En mode relais, les deux méthodes hybridées opèrent l'une à la suite de l'autre en utilisant la sortie de la précédente comme entrée à la manière d'un pipeline. En mode simultané par contre, les deux approches explorent en parallèle l'espace de recherche.

La dernière classification présentée est celle proposée par Puchinger et Raidl (2005). Dans celle-ci, les auteurs divisent les approches hybridant métaheuristiques et méthodes exactes en deux classes : les hybridations de type collaboratives et les hybridations de type intégratives. Les approches hybrides de type collaboratives regroupent les stratégies d'hybridation où les méthodes hybridées échangent de l'information de manière séquentielle, parallèle ou entrelacée tout en conservant chacune leur intégrité. En ce qui concerne les stratégies d'hybridations intégratives, par contre, une des deux approches est modifiée afin d'incorporer l'autre sous un modèle maître/esclave.

3. LE PROBLÈME THÉORIQUE D'ORDONNANCEMENT DE VOITURES ET LES PRINCIPALES APPROCHES DE SOLUTION RETROUVÉES DANS LA LITTÉRATURE

Le problème d'ordonnancement de voitures (POV) consiste à déterminer la séquence de production d'un ensemble de *nb_cars* voitures sur une ligne d'assemblage composée de trois ateliers: la tôlerie, la peinture et le montage. Toutefois, le problème théorique est généralement abordé dans la littérature en ne considérant que l'atelier de montage et la séquence déterminée est alors appliquée à l'ensemble de la chaîne (Parello et al., 1986; Parello, 1988). Au montage, de nombreux éléments sont ajoutés à la carrosserie peinturée pour terminer l'assemblage de la voiture. Chaque voiture est caractérisée par les options qu'elle requiert parmi un ensemble O d'options nécessitant des opérations lourdes (traction intégrale, freins anti-blocage, etc.). Ces options exigent la dispersion des voitures dans la séquence de production de façon à lisser la charge de travail. Cette dispersion est exprimée par un ratio r_o/s_o signifiant que, pour toute séquence consécutive de s voitures, au plus r voitures peuvent posséder l'option o. Si à un endroit dans la séquence une contrainte d'espacement n'est pas respectée, on dira alors qu'il y a conflit. L'optimisation de ce problème consiste donc à minimiser le nombre de conflits liés aux contraintes d'espacement. De plus, il doit être préciser que les voitures identiques, c'est-à-dire celles possédant exactement les mêmes options, sont regroupées en classes de voitures. Pour chacune des v classes ainsi formées, on connaît le nombre de voitures à produire n_i.

Le Tableau 1, tiré de Smith (1997), présente un exemple d'une instance d'un *POV*. Pour cette instance, on a 5 options (O), 12 classes de voitures (v) et $\sum_{i=1}^{v} n_i = nb\_cars = 25$.

			Classes de voitures											
O	***r***	***s***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	1	2	0	1	1	0	0	0	1	1	1	0	0	1
2	2	3	1	0	1	1	1	0	1	0	0	0	1	1
3	1	3	0	1	0	0	0	0	1	0	1	1	1	0
4	2	5	0	0	0	1	0	1	0	1	0	0	0	1
5	1	5	0	1	0	0	1	0	0	0	0	0	0	0
		n_i	3	1	2	4	3	3	2	1	1	2	2	1

Tableau 1. Instance d'un *POV*

Une solution d'une instance d'un POV est représentée par un vecteur de longueur *nb_cars* contenant une et une seule classe de voiture dans chacune de ses cases. On peut construire une matrice *S* de valeurs binaires de taille *O* par *nb_cars* à partir des informations contenues dans le vecteur de la solution dans le but d'évaluer cette dernière. On a $S_{oj} = 1$ si la classe de voiture assignée à la position *j* du vecteur de la solution demande l'option *o*, 0 sinon.

La Figure 1 présente une solution partielle pour l'instance du Tableau 1. Les options 1 et 3 ne causent aucun conflit dans cette partie de solution. Par contre, pour l'option 2, il existe un conflit aux positions *j+1* à *j+3* car la contrainte d'espacement 2/3 n'est pas respectée. Il en va de même pour l'option 4 aux positions *j* à *j+4* et pour l'option 5 aux positions *j+1*à *j+5*. On peut conclure qu'il existe 3 conflits dans cette partie de la solution.

Solution partielle pour l'instance du Tableau 1 :

Position : …	*j*	*j*+1	*j*+2	*j*+3	*j*+4	*j*+5	…
Classe : …	8	5	4	3	6	9	…

Matrice *S* construite à partir de la solution partielle:

Option 1 (1/2) :	1	0	0	1	0	1
Option 2 (2/3) :	0	1	1	1	0	0
Option 3 (1/3) :	0	0	0	0	0	1
Option 4 (2/5) :	1	0	1	0	1	0
Option 5 (1/5) :	0	1	0	0	0	1

Figure 1. Évaluation d'une solution

La problématique de l'ordonnancement de voitures considérant les contraintes d'espacement à l'atelier de montage a été définie comme NP-difficile au sens fort (Kis, 2004). Lopez et Roubellat (2001) ont réalisé une revue de la littérature sur cette problématique et ont recensé des méthodes exactes, des méthodes arborescentes, des méthodes constructives et des méthodes de recherche dans le voisinage comme approches de solution. Dans les récents travaux, on retrouve la recherche dans le voisinage (Davenport et Tsang, 1999 ; Putcha et Gottlieb, 2002 ; Gottlieb et al., 2003), le recuit simulé (Chew et al., 1991), les algorithmes génétiques (Warwick et Tsang, 1995 ; Terada et al., 2006 ; Zinflou et al., 2007a), les réseaux de neurones artificiels (Smith et al., 1996) les algorithmes d'optimisation par colonies de fourmis (Gottlieb et al., 2003 ; Gagné et al., 2006 ; Gravel et al., 2005 ; Gagné et al., 2007 ; Morin et al., 2007) et les approches par programmation linéaire (Drexl and Kimms, 2001; Gravel et al., 2005, Estellon et al., 2007) comme approches de solution à ce problème.

Dans la prochaine section, nous présentons une hybridation intégrative, selon la classification Puchinger et Raidl (2005), entre l'algorithme génétique proposé par Zinflou et al. (2007a) et le modèle linéaire de Gravel et al. (2005).

4. ALGORITHME GÉNÉTIQUE HYBRIDE

Les opérateurs de croisement classiques, comme le croisement binaire ou encore le croisement réel, ne sont pas adaptés à la nature particulière des contraintes d'espacement du POV. À preuve, les algorithmes génétiques proposés par Warwick et Tsang (1995) et Terada et al. (2006) ont produit des résultats de piètre qualité sur les ensembles tests de CSPLib (http://www.csplib.org/). Par contre, Zinflou et al. (2007a) ont proposé des opérateurs de croisement conçus spécifiquement pour le POV qui ont permis d'obtenir des résultats fort intéressants et qui s'avèrent les meilleurs résultats connus de la littérature. Ces opérateurs exploitent la notion d'intérêt qui vise à privilégier les classes de voitures n'entraînant pas de nouveaux conflits dans la solution ou encore utilisent l'information sur les positions non conflictuelles contenues dans les parents pour créer un nouvel individu. Dans cet article, nous nous concentrons à présenter la stratégie d'hybridation proposée et le lecteur peut consulter Zinflou et al. (2007a) pour connaître le fonctionnement général de l'AG et les caractéristiques des opérateurs de croisement *NCPX* et *IBX*.

La stratégie d'hybridation proposée vise à intégrer la programmation linéaire en nombres entiers lors du processus de croisement pour ainsi créer un *croisement hybride*. Deux croisements hybrides sont proposés à partir respectivement des opérateurs *NCPX* et *IBX*.

La première étape du *croisement hybride A* consiste à sélectionner k_{mov} positions dans le premier parent x_1 pour les transférer dans l'enfant y_1 en cours de création (Voir Figure 2). Ces positions sont marquées comme positions libres, ce qui veut dire que les classes de voitures situées à ces positions peuvent, par la suite, être déplacées dans la séquence. Dans la seconde étape, l'opérateur de croisement *NCPX* est ensuite utilisé pour compléter les positions restantes de l'enfant y_1 sans tenir compte des k_{mov} positions marquées. À l'étape 3, on cherche, parmi les voisins potentiels de y_1, celui qui est à moindre coût. Pour cela, on résout le PLNE proposé par Gravel et al. (2005) où $|x|$- k_{mov} variables de la séquence sont fixées. De manière générale, la formulation du PLNE consiste à affecter une classe de voitures à chacune des positions de la séquence en cherchant à minimiser le nombre de conflits générés par les contraintes d'espacement. Pour ce croisement, le PLNE détermine les classes de voitures à placer dans les positions libres de la séquence. Le lecteur peut consulter Gravel et al. (2005) pour la modélisation complète. Les mêmes étapes sont réalisées avec le deuxième parent de manière à créer un autre enfant.

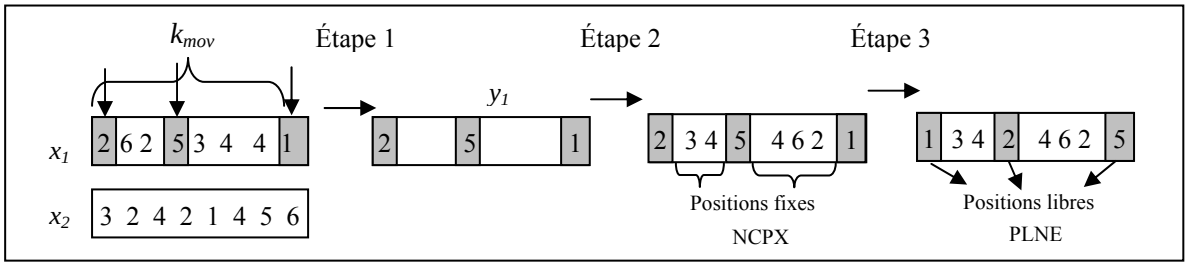

Figure 2. Illustration du fonctionnement du croisement hybride A

L'efficacité de ce type de croisement dépend essentiellement de deux éléments : le nombre de positions k_{mov} du sous-problème à résoudre et le temps maximum T_{max} alloué au solveur pour résoudre le PLNE. En effet, un k_{mov} trop élevé ou un T_{max} trop petit risque de faire échouer la résolution du PLNE dans le temps imparti. D'un autre côté, un k_{mov} trop faible ne laisse pas assez de latitude au PLNE pour améliorer la solution courante. Afin de contourner ce problème, l'idée utilisée consiste à initialiser k_{mov} à une faible valeur et d'accroître par la suite cette valeur de manière constante au fil des générations. Lorsqu'une résolution échoue, la valeur de k_{mov} est décrémentée de manière à revenir à la dernière valeur ayant fourni une solution. Notons ici que dès que la valeur de k_{mov} est décrémentée le nombre de positions à déplacer est stabilisé à cette nouvelle valeur et ne variera plus jusqu'à la fin de l'algorithme. On veut ainsi permettre le plus grand choix de positions libres possibles pour une valeur T_{max} donnée.

Un autre point crucial, pour la réussite de ce croisement, concerne le choix des positions libres. Deux stratégies sont utilisées : la première consiste à choisir aléatoirement un pourcentage de positions faisant partie d'un conflit et de compléter avec des positions choisies aléatoirement tandis que la seconde stratégie consiste tout simplement à choisir aléatoirement les k_{mov} positions de manière à ce que 2 positions j et j' sélectionnées soient séparées d'au moins s_{max} positions, où s_{max} correspond au plus grand dénominateur parmi toutes les contraintes d'espacement. Cette propriété a été à l'origine proposée par Estellon et al. (2005) et permet d'affirmer que la solution optimale du problème linéaire réduit est une solution entière. Il est aussi important de noter ici que, lors de la sélection des positions libres, on s'assure qu'au moins une des positions choisies fasse partie d'un conflit.

Le *croisement hybride B* consiste à déterminer deux points de coupure aléatoirement dans chacun des parents sélectionnés x_1 et x_2 (Voir Figure 3). Afin de créer le premier enfant, les classes de voitures comprises entre les deux points de coupure de x_1 sont directement copiées dans l'enfant y_1. La deuxième étape consiste à sélectionner aléatoirement un des deux points de coupure et à réorganiser les classes de voitures présentes à l'extérieur de ce point à l'aide de l'opérateur de croisement *IBX*. Finalement, à l'étape 3, les k positions situées à l'extérieur de l'autre point de coupure sont marquées comme libres et celles-ci sont ensuite réorganisées avec les classes de voitures restantes en utilisant le PLNE. Les mêmes étapes sont réalisées avec le deuxième parent de manière à créer un autre enfant.

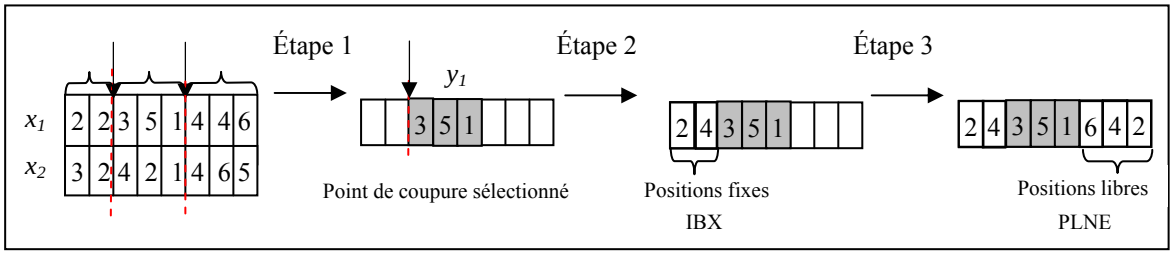

Figure 3. Illustration du fonctionnement du croisement hybride B

La Figure 4 présente le fonctionnement général de l'algorithme génétique hybride. Pour le cas de l'algorithme génétique sans l'hybridation proposée, seule la phase 2 est alors requise. Dans le fonctionnement de l'AG hybride, la création de la population d'enfants à la génération t (Q_t) se fait en 3 phases à partir de la population de parents (P_{t-1}) : une *phase mixte*, une *phase de diversification* et une *phase d'intensification* de la recherche. La Phase 1 (phase mixte) consiste à créer, selon une probabilité $Prob_{ILP}$ fixée à 50%, des enfants en utilisant le croisement hybride (*Croisement Hybride A ou B)* pour le premier 10 % de la population et avec le croisement *NCPX* ou *IBX* selon le cas pour le reste de la population. La Phase 2 (phase de diversification), de son côté, génère des enfants uniquement en utilisant le croisement *NCPX* ou *IBX* selon le cas. Finalement, la Phase 3 (phase d'intensification) de l'algorithme intensifie la recherche en générant des enfants avec le Croisement Hybride *A ou B* uniquement.

Les différents algorithmes ont tous été implémentés en C^{++} en utilisant CPLEX 10.0 comme librairie de programmation linéaire en nombres entiers. La machine utilisée pour les expérimentations numériques est un ordinateur équipé de deux processeurs Itanium 64 bits cadencés chacun à 1.4 GHz avec 4 Go de mémoire vive

et tournant sous Windows Server 2003. Les paramètres N, λ, p_c, p_m, *NbGen,* qui représentent respectivement la taille de la population de parents, la taille de la population d'enfants, la probabilité de croisement, la probabilité de mutation et le nombre maximum de générations de l'algorithme sont fixés respectivement à 250, 200, 0.8, 0.09, 700 pour tous les problèmes testés. La phase 1 se termine à la 300 ième génération, la phase 2 se termine à la 650 ième génération et les 50 dernières générations constituent la 3 ième phase. Il faut toutefois noter que l'algorithme s'arrête lorsqu'une solution sans conflit est trouvée. Finalement, on note que le temps maximum alloué à CPLEX pour résoudre un PLNE est fixé à trois secondes dans la Phase 1 de l'algorithme et à dix secondes dans la Phase 3. On veut ainsi laisser plus de temps à CPLEX pour améliorer la solution courante dans la phase d'intensification.

Création aléatoire de la population initiale P_0
Évaluer chaque individu $x \in P_0$ et trier P_0
Tant qu'aucun critère d'arrêt rencontré
  Tant que $|Q_t| < \lambda$
    Si p_c **alors**
      En Fonction de la phase
        Cas (*Phase 1*)
          Si $|Q_t| < 10$ % de λ **et** $\text{Prob}_{ILP} \leq 0.5$ **alors**
            Sélectionner un parent et créer un enfant à l'aide du Croisement Hybride *A ou B*
          Sinon
            Sélectionner les parents et créer 2 enfants x et y avec le *NCPX ou IBX*
        Cas (*Phase 2*)
            Sélectionner les parents et créer 2 enfants x et y avec le *NCPX ou IBX*
        Cas (*Phase 3*)
            Sélectionner un parent et créer un enfant à l'aide du Croisement Hybride *A ou B*
          Évaluer les enfants créés
      Fin Si
      Si p_m **alors**
        Effectuer une mutation sur l'individu sélectionné et évaluer l'individu muté
      Fin Si
      Effectuer une sélection inverse avec l'enfant y s'il existe
      Ajouter x à Q_t
  Fin Tant que
  Trier $Q_t \cup P_t$
  Conserver les μ meilleurs individus de $Q_t \cup P_t$ pour former P_{t+1}
Fin Tant que
Retourner le meilleur individu trouvé

Figure 4. Fonctionnement de l'algorithme génétique hybride $ILPGA^{NCPX}$ et $ILPGA^{IBX}$

Dans la prochaine section, nous présentons les résultats numériques obtenus avec ces deux approches hybrides pour les problèmes retrouvés dans CSPLib. Nous comparons ceux-ci avec les versions de l'AG sans hybridation (GA^{NCPX} et GA^{IBX}).

5. EXPÉRIMENTATIONS ET RÉSULTATS

Trois ensembles de problèmes sont disponibles dans la librairie CSPLib. Nous allons toutefois restreindre notre comparaison de performance aux problèmes de plus grande taille qui sont également les problèmes les plus difficiles à solutionner. Ces 30 problèmes ont été proposés par Gravel et al. (2005) et comporte 5 options et de 19 à 26 classes.

Chaque instance a été résolue à 10 reprises par les approches hybrides $ILPGA^{NCPX}$ et $ILPGA^{IBX}$ et à 100 reprises par les algorithmes génétiques GA^{NCPX} et GA^{IBX}. La partie de gauche du Tableau 2 présente les résultats comparatifs entre le GA^{NCPX} et le $ILPGA^{NCPX}$. On retrouve respectivement dans le tableau, le nom de l'instance, la valeur de la meilleure solution connue et pour chaque algorithme, le nombre moyen de conflits (*moyenne*) et le temps moyen de calcul exprimé en secondes (*temps moyen*). Il est important de noter que les temps moyens de calcul rapportés ici correspondent au temps pris par chaque algorithme pour effectuer toutes les générations sauf dans le cas où une solution sans conflit est trouvée.

Ces résultats démontrent que le croisement hybride *NCPX* utilisé dans l'algorithme $ILPGA^{NCPX}$ améliore la qualité des solutions en moyenne de 26,39 % par rapport au GA^{NCPX}. Il permet même d'obtenir la meilleure solution connue à chacune des exécutions pour 14 des 30 problèmes, soit 10 de plus que GA^{NCPX}. On note cependant que l'amélioration de la qualité n'est pas fonction de la taille de l'instance et reste relativement stable pour les différentes tailles d'instances avec une amélioration moyenne de 28,55 % pour les instances de 200 voitures, 25,36 % pour les instances de 300 voitures et 25,27 % pour les instances contenant 400 voitures. En observant maintenant les temps d'exécution, on remarque que l'approche hybride nécessite beaucoup plus de temps que le GA^{NCPX} avec, en moyenne, 25 minutes pour le $ILPGA^{NCPX}$ contre 94 secondes pour le GA^{NCPX}. Il faut également mentionner que les temps d'exécution du GA^{NCPX} proviennent d'un ordinateur Xeon 3.6 Ghz sous Windows XP Pro avec 1024 Mo de mémoire vive.

Une analyse détaillée des résultats montre également que la Phase 2 (phase de diversification) de l'approche hybride trouve rarement la meilleure solution. En effet, sur l'ensemble des problèmes, la meilleure solution est trouvée lors de la Phase 2 du $ILPGA^{NCPX}$ dans moins de 10 % des cas. À l'opposé, on note que pour le groupe d'instances contenant 200 voitures, la meilleure solution est trouvée dans 56 % des cas lors de la Phase 1 (phase mixte) contre 36 % des cas lors de la Phase 3 (phase d'intensification) où seul le croisement hybride A est

utilisé. Pour les groupes d'instances contenant 300 et 400 voitures, la meilleure solution est trouvée durant la Phase 3 dans respectivement 54 % et 56 % des cas contre 42 % et 44 % durant la Phase 1.

La partie de droite du Tableau 2 présente les résultats comparatifs entre le GA^{IBX} et le $ILPGA^{IBX}$. Lorsque l'on compare la performance de l'approche hybride par rapport au GA^{IBX}, on remarque que la combinaison de l'AG et du PLNE permet un gain moyen de 37 % en terme de qualité de solutions sur l'ensemble des problèmes. Le $ILPGA^{IBX}$ permet d'obtenir la meilleure solution connue à chaque exécution de l'algorithme pour 10 des 30 problèmes contre seulement deux pour le GA^{IBX}. Par ailleurs, on note, en observant les résultats plus attentivement, que contrairement au $ILPGA^{NCPX}$, les gains moyens obtenus par le $ILPGA^{IBX}$ par rapport au GA^{IBX} augmentent en fonction de la taille de l'instance. En effet, le gain moyen passe respectivement de 33 % pour le groupe d'instances avec 200 voitures à 34 % pour les instances à 300 voitures et à plus de 44 % pour les instances avec 400 voitures. Cette situation s'explique sans doute en partie par les performances du GA^{IBX} qui sont inférieures à celles du GA^{NCPX}, particulièrement pour les instances de grande taille. Lorsque l'on observe maintenant les temps d'exécution, on constate que l'approche hybride requiert un temps d'exécution nettement plus important que le GA^{IBX}.

Lorsque l'on s'attarde à vérifier à quelle étape de l'algorithme la meilleure solution est trouvée, on remarque que les résultats sont biens différents de ceux obtenus avec le $ILPGA^{NCPX}$. En effet, pour l'ensemble des instances, on observe que la meilleure solution est trouvée, dans 82 % des cas, lors de la Phase 3 de l'algorithme et seulement dans 6% des cas lors de la Phase 1.

Globalement, on observe que le $ILPGA^{NCPX}$ permet généralement d'obtenir une meilleure performance que le $ILPGA^{IBX}$. Les meilleurs résultats sont montrés en caractères gras au Tableau 2.

		GA^{NCPX}		$ILPGA^{NCPX}$		GA^{IBX}		$ILPGA^{IBX}$	
Instances	**Meilleure solution connue**	**# Conflit moyen**	**Temps Moyen (s)**	**# Conflit moyen**	**Temps Moyen (s)**	**# Conflit moyen**	**Temps Moyen (s)**	**# Conflit moyen**	**Temps Moyen (s)**
200_01	0	1.23	67.73	**0.00**	1147.00	3.13	77.28	0.90	1992.8
200_02	2	2.94	69.73	2.20	1287.00	3.93	76.05	**2.00**	2273.3
200_03	3	7.41	79.44	5.30	1841.40	11.47	81.10	**4.60**	2240.0
200_04	7	7.39	77.68	**7.00**	1343.70	7.47	80.43	7.10	2301.4
200_05	6	6.69	71.40	**6.00**	1091.70	7.71	74.71	6.10	1842.6
200_06	6	6.00	71.55	**6.00**	1773.60	6.00	75.92	**6.00**	2031.7
200_07	0	0.15	21.92	**0.00**	413.90	3.44	76.35	**0.00**	1583.9
200_08	8	8.00	62.29	**8.00**	1604.80	8.00	70.32	**8.00**	1669.0
200_09	10	10.53	75.41	**10.00**	1567.80	11.40	78.25	**10.00**	2068.3
200_10	19	21.40	67.40	**19.00**	1422.30	20.72	69.08	19.10	2040.4
300_01	0	2.79	103.98	**0.80**	1325.60	5.55	115.81	3.00	2240.9
300_02	12	12.02	112.73	**12.00**	1461.10	14.78	116.52	12.10	2250.8
300_03	13	13.11	120.63	**13.00**	1554.80	15.92	123.90	13.00	2349.6
300_04	7	7.71	115.98	**7.50**	1620.50	10.98	117.49	7.80	2659.2
300_05	27	42.83	115.90	33.00	2601.90	39.39	118.54	**31.90**	2855.7
300_06	2	5.30	106.61	**3.60**	1597.70	8.24	115.14	4.70	2855.2
300_07	0	0.08	23.17	**0.00**	426.40	3.78	116.77	**0.00**	2217.4
300_08	8	8.00	105.03	**8.00**	1680.10	9.11	113.27	**8.00**	2298.7
300_09	7	7.36	105.31	**7.20**	1649.70	9.40	108.79	7.30	2433.5
300_10	21	28.48	100.37	**22.4**	2338.90	32.23	106.13	22.5	2926.1
400_01	1	1.81	137.89	**1.90**	1782.00	2.98	151.98	2.20	3313.6
400_02	15	19.31	147.51	**18.30**	2081.10	23.41	148.10	18.70	2502.6
400_03	9	10.79	128.73	**9.90**	2338.70	12.21	143.06	10.40	3146.2
400_04	19	19.12	164.57	**19.00**	2354.90	20.37	161.75	**19.00**	3287.1
400_05	0	0.00	2.45	**0.00**	19.80	5.06	139.81	**0.00**	2860.3
400_06	0	0.16	38.69	**0.00**	604.80	4.44	146.06	**0.00**	2461.0
400_07	4	4.72	130.98	**4.70**	1479.50	5.59	141.81	4.90	2785.8
400_08	4	4.73	131.45	**4.20**	1203.30	7.22	144.15	5.80	2789.0
400_09	5	10.58	149.18	**7.20**	2136.50	17.38	155.08	8.70	3312.4
400_10	0	0.71	102.27	**0.00**	1890.90	4.79	147.75	0.20	2613.2

Tableau 2. Résultats comparatifs GA^{NCPX} vs $ILPGA^{NCPX}$ et GA^{IBX} vs $ILPGA^{IBX}$

Les deux approches hybrides présentées précédemment ont un fonctionnement différent l'une de l'autre sans pour autant être complètement opposées. On peut donc les combiner très facilement en supposant que leur utilisation simultanée puisse apporter une plus grande diversité dans le processus et ainsi mener à une meilleure

qualité de solution. Dans l'algorithme résultant (*ILPGA*$^{NCPX/IBX}$), les opérateurs de croisement sont appliqués de la manière suivante en fonction de la phase de création de la population enfant :

- Phase 1 : créer, selon une probabilité $Prob_{ILP}$, des enfants en utilisant aléatoirement le *Croisement Hybride A ou B* pour le premier 10 % de la population et en utilisant aléatoirement le croisement *NCPX* ou *IBX* pour le reste de la population;
- Phase 2 : utilisation du croisement NCPX dans 65 % des cas et IBX dans 35% des cas;
- Phase 3 : utilisation aléatoire du *Croisement Hybride A ou B*.

La meilleure performance du croisement *NCPX* par rapport à *IBX* sur l'ensemble des problèmes testés explique les pourcentages utilisés à la Phase 2. Les résultats obtenus par le *ILPGA*$^{NCPX/IBX}$ sont présentés dans la partie gauche du Tableau 3.

	ILPGA$^{NCPX/IBX}$		*GA*$^{NCPX/IBX}$ *avec RL*	
Instances	**# Conflit moyen**	**Temps Moyen (s)**	**# Conflit moyen**	**Temps Moyen (s)**
200_01	**0.00**	2155.10	0.14	47.50
200_02	**2.00**	1620.10	**2.00**	89.99
200_03	**4.10**	1456.50	5.13	101.21
200_04	**7.00**	2344.90	**7.00**	98.01
200_05	**6.00**	1637.00	**6.00**	90.39
200_06	**6.00**	2031.70	**6.00**	93.71
200_07	**0.00**	1583.90	**0.00**	10.11
200_08	**8.00**	1669.00	**8.00**	80.44
200_09	**10.00**	1771.00	**10.00**	97.26
200_10	**19.00**	1739.70	**19.00**	83.91
300_01	0.30	1606.90	**0.08**	135.29
300_02	**12.00**	1882.90	**12.00**	145.60
300_03	**13.00**	2222.60	**13.00**	157.54
300_04	**7.00**	1357.00	7.26	140.05
300_05	**30.50**	2045.60	32.08	147.88
300_06	2.60	1454.90	**2.55**	137.86
300_07	**0.00**	879.20	**0.00**	11.60
300_08	**8.00**	2064.60	**8.00**	148.46
300_09	7.30	2197.70	**7.03**	139.73
300_10	**21.20**	2725.70	21.30	134.04
400_01	**1.10**	2594.90	1.29	171.78
400_02	**16.70**	2101.40	16.75	175.24
400_03	**9.60**	2761.20	10.07	160.11
400_04	**19.00**	3028.20	**19.00**	211.72
400_05	**0.00**	64.60	**0.00**	5.26
400_06	**0.00**	899.10	**0.00**	7.57
400_07	**4.00**	2051.10	4.09	163.79
400_08	4.30	2185.50	**4.00**	165.66
400_09	**5.80**	1598.80	6.69	185.74
400_10	**0.00**	2487.90	**0.00**	33.15

Tableau 3. Résultats de l'algorithme *ILPGA*$^{NCPX/IBX}$ et du *GA*$^{NCPX/IBX}$ avec recherche locale

L'approche mixte *ILPGA*$^{NCPX/IBX}$ permet globalement d'obtenir de meilleurs résultats que les deux approches hybrides présentées précédemment au Tableau 2. En effet, cet algorithme obtient des résultats identiques ou meilleurs à ceux du *ILPGA*^{NCPX} et du *ILPGA*^{IBX} sur 29 des 30 instances. On peut alors supposer que l'utilisation des deux opérateurs de croisement dans le *ILPGA*$^{NCPX/IBX}$ permet à l'algorithme de mieux diversifier sa recherche tout en conservant un bon équilibre entre intensification et diversification.

Le Tableau 3 permet également de comparer la performance du *ILPGA*$^{NCPX/IBX}$ aux résultats du *GA*$^{\boldsymbol{NCPX/IBX}}$ utilisant une recherche locale. Dans Zinflou et al. (2007b), il a été montré que l'approche mixte *GA*$^{\boldsymbol{NCPX/IBX}}$ intégrant une procédure de recherche locale s'est avérée l'algorithme le plus performant pour résoudre l'ensemble des problèmes de la CSPLib. Au Tableau 3, on note que les performances des deux algorithmes sont très proches l'une de l'autre. Toutefois, le *ILPGA*$^{NCPX/IBX}$ obtient des résultats identiques ou meilleurs pour 26 des 30 instances. Sa performance est particulièrement intéressante sur les instances 200_3 et 300_05 où, en moyenne, il existe une différence de plus d'un conflit entre les solutions des deux algorithmes. Les meilleurs résultats sont montrés en caractères gras au Tableau 3.

Il existe toutefois un écart important entre les temps moyens d'exécution des deux algorithmes. En effet, le *GA*$^{\boldsymbol{NCPX/IBX}}$ utilisant une recherche locale prend en moyenne 112 secondes par problème comparativement à 1874 secondes pour l'ILPGA$^{NCPX/IBX}$. Les écarts observés doivent toutefois être nuancés par l'utilisation d'ordinateurs différents lors de l'exécution des deux algorithmes.

6. CONCLUSIONS

Dans cet article, nous avons présenté trois approches hybrides intégratives permettant de résoudre efficacement le problème théorique d'ordonnancement de voitures. Ces approches reposent principalement sur un algorithme génétique intégrant deux croisements qui utilisent la programmation linéaire en nombres entiers. La nature différente des deux croisements hybrides proposés a permis de les combiner efficacement et de montrer que l'algorithme résultant permettait d'obtenir des résultats compétitifs, en terme de qualité de solution, par rapport aux résultats du *GA*$^{\boldsymbol{NCPX/IBX}}$ utilisant une recherche locale et qui représentent les meilleurs résultats connus à ce jour. L'ensemble des travaux réalisés démontre bien l'intérêt de concevoir des approches hybrides exploitant les forces de différentes méthodes.

Les expérimentations numériques effectuées ont toutefois mis en évidence l'écart considérable entre les temps d'exécution des approches hybrides et ceux des AG. Ces différences, même si elles doivent être nuancées du fait de différentes configurations expérimentales, laissent entrevoir des améliorations possibles au niveau de l'implémentation des méthodes hybrides pour les rendre encore plus compétitives. Ces améliorations peuvent, par exemple, passer par le développement de versions parallèles des algorithmes proposés ou encore par l'utilisation de technologies

permettant de développer des puissances de calculs importantes comme les grilles de calculs ou encore le calcul pair à pair.

D'autres perspectives peuvent aussi être envisagées comme le remplacement de l'opérateur de mutation par une méthode exacte en s'inspirant du fonctionnement des algorithmes mémétiques. De plus, il serait intéressant d'évaluer les coopérations de type collaboratives ainsi qu'une éventuelle combinaison avec les approches proposées dans cet article. Finalement, les stratégies d'hybridation proposés ne se limitent pas uniquement à leur application au POV. Il serait donc intéressant de les généraliser à d'autres problématiques ainsi qu'à des versions industrielles.

7. RÉFÉRENCES

Barichard, V., 2003. *Approches hybrides pour les problèmes multiobjectifs*. Thèse de doctorat, Université d'Angers, France.

Basseur, M., 2004. *Conception d'algorithmes coopératifs pour l'optimisation multi-objectifs : Application aux problèmes d'ordonnancement de type flow-shop*. Thèse de doctorat. Université de Lille, France.

Blum, C., Roli, A. and E. Alba, 2005. An Introduction to Metaheuristics Techniques, in *Parallel Metaheuristics*, E. Alba (Ed.), Wiley.

Chew, T.-L., David J.-M., Nguyen A. and Y. Tourbier, 1991. Solving constraint satisfaction problem with simulated annealing: the Car Sequencing Problem revisited, *12th International Conference on Artificial Intelligence*, Expert Systems and Natural Language, 405-416.

Dumitrescu, I. and T. Stützle, 2003. Combinations of local search and exact algorithms. *Lecture Notes in Computer Science 2611*, 211-223.

Drexl A. and A. Kimms, 2001. Sequencing JIT mixed-model assembly lines under station load and part-usage constraints, *Management Science*. **47**(3): 480-291.

Davenport, A. and E. Tsang, 1999. Solving constraint satisfaction sequencing problems by iterative repair. In *Proceeding of the First International Conference on the Practical applications of Constraint Technologies and Logic Programming*. London, England. Practical Applications Company, 345–357.

Estellon, B., Gardi, F. and K. Nouioua, 2007. Large neighborhood improvements for solving car-sequencing problems. *RAIRO - Operations Research*. to appear.

Gagné, C., Gravel, M. and W.L. Price, 2006. Solving real car sequencing problems with Ant Colony *Optimization. European Journal of Operational Research*. 174 (3), 1427-1448.

Gagné C., Gravel M., Morin S. and W.L. Price, 2007. Impact of the pheromone trail on the performance of ACO algorithms for solving the car-sequencing problem. *Journal of the Operational Research Society*, to appear.

Gottlieb J., Puchta M. and C.Solnon, 2003. A study of greedy, local search and ant colony optimization approaches for car sequencing problems. *Computers Science*, 246-257.

Gravel M., Gagné C. and W.L. Price, 2005. Review and comparison of three methods for the solution of the car sequencing problem. *Journal of the Operational Research Society,* 56: 1287-1295.

Jussien, N. and F. Laburthe, 2004. Preface : Hybrid optimization techniques. *Annals of Operations Research*. 130 (1-4), 17-18.

Kis, T., 2004. On the complexity of the car sequencing problem. *Operation Research Letters*. 32 (4), 331-336.

Lopez, P. and F. Roubellat, 2001. Ordonnancement de la production. *Hermès Science Ppublications*, Paris.

Morin S., Gagné C. and M. Gravel, 2007. Ant colony optimization with a specialized pheromone trail for the car-sequencing problem. *European Journal of Operational Research*, to appear.

Parello, B. D., Kabat, W. B. and L. Wos, 1986. Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem. *Journal of Automated Reasoning*. 2, 1-42.

Parello, B. D., 1988. CAR WARS: The (almost) birth of an expert system. *AI Expert*. 3, 60-64.

Puchinger, J. and G.R., Raidl, 2005. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. In: *First International Work-Conference on the Interplay Between Natural and Artificial Computation*, LNCS 3562, Las Palmas, Spain, 41-53 (2005).

Puchta, M. and J. Gottlieb, 2002. Solving Car Sequencing Problems by Local Optimization. In: *Proceedings EvoWorkshops*, Lecture Notes in Computer Science 2279, 132-142.

Smith B., 1997. Succeed-first or Fail-first: A Case Study in Variable and Value Ordering Heuristics. In: *Proceedings of the third international Conference on the Practical Applications of Constraint Technology*. London, UK, 321-330.

Smith, K., Palaniswami, M. and M. Krishnamoorthy, 1996. Traditional heuristic versus Hopfield neural network approaches to a car sequencing problem. *European Journal of Operational Research*. 93 (2), 300-317.

Talbi, E.-G., 2002. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*. 8, 541-564.

Terada J., Vo H. and D. Joslin, 2006. Combining Genetic Algorithms with Squeaky-Wheel Optimization. In: *Proceedings of the 8th annual conference on Genetic and Evolutionary Computation*. Seattle, Washington, 1329-1336.

Warwick T. and E. Tsang, 1995. Tackling car sequencing problem using a generic genetic algorithm. *Evolutionary Computation*. **3**(3): 267-298.

Zinflou A., Gagné C. and M. Gravel, 2007a. Crossover operators for the car-sequencing problem, *Seventh European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2007)*, LNCS 4446, C. Cotta and J. van Hemert (Eds.), Springer-Verlag Berlin Heidelberg, 229-239.

Zinflou A., Gagné C. and M. Gravel, 2007b. A new evolutionary approach for the car-sequencing problem. *Soumis pour publication en août 2007*.