CHAPITRE 6

UN ALGORITHME ÉVOLUTIONNAIRE POUR LE PROBLÈME INDUSTRIEL D'ORDONNANCEMENT DE VOITURES

6.1 Introduction

Dans les deux chapitres précédents, nous avons présenté des mécanismes permettant d'utiliser un algorithme génétique pour résoudre le problème théorique d'ordonnancement de voitures. Ces mécanismes, en particulier les nouveaux opérateurs de croisement proposés, ont permis d'obtenir une approche de résolution particulièrement efficace dans le cadre du POV. Comme nous l'avons vu précédemment, le POV est un problème uni-objectif qui ne tient compte que des contraintes de l'atelier de montage. Pour sa part, la formulation industrielle du POV (POVI) subdivise les contraintes de capacité de l'atelier de montage en deux types, les contraintes de capacité liées aux options prioritaires et les contraintes de capacité liées aux options non-prioritaires. De plus, la réalité industrielle ne tient pas uniquement compte des contraintes de l'atelier de montage. En effet, la formulation du problème industriel proposée par le constructeur automobile Renault dans le cadre du Challenge ROADEF 2005 considère également les contraintes de l'atelier de peinture. Dans celui-ci, la minimisation de la consommation de solvant utilisé pour purger les pistolets de peinture à chaque changement de couleur est un élément important à considérer. Toutefois, une trop longue séquence de voitures de couleur identique rend difficile le contrôle visuel de la qualité. Pour assurer ce contrôle de la qualité, le nombre de voitures consécutives de même couleur ne doit pas dépasser un nombre maximal donné.

Le problème industriel d'ordonnancement de voitures est ainsi de nature multi-objectifs avec trois objectifs conflictuels à optimiser. Pour l'atelier de montage, on cherche à minimiser le nombre de violations des contraintes de capacité liées aux options prioritaires (HPO) et aux options non-prioritaires (LPO). Dans l'atelier de peinture, on cherche à

minimiser le nombre de changements de couleur (COLOR). Dans la description du Challenge ROADEF 2005, le constructeur automobile Renault propose également de traiter les trois objectifs du problème selon un ordre lexicographique. Pour y parvenir, les organisateurs proposent d'attribuer des poids de 10^6, 10^3 et 1 respectivement sur le premier, le second et le troisième objectif [Nguyen et Cung 2005] comme indiqué à la Section 2.5.3 du Chapitre 2.

En recensant la littérature sur le POVI au Chapitre 2, nous avons noté que, comme pour le POV, très peu d'auteurs ont proposé des algorithmes génétiques pour résoudre ce problème à l'exception des travaux de Jaszkiewicz *et al.* [2004] durant le Challenge ROADEF et ceux de Siala [2005] à la suite du challenge. Toutefois, les résultats obtenus par Jaszkiewicz *et al.* n'ont pas permis aux auteurs de faire partie des 12 finalistes du Challenge ROADEF 2005 qui comptait au départ 55 équipes. De son côté, les résultats de l'AG proposé par Siala demeurent largement inférieurs aux pires résultats obtenus par les 18 équipes qualifiées pour la seconde phase du challenge [Siala 2005]. Dans un effort d'amélioration, l'auteur propose, par la suite, d'hybrider son AG avec l'algorithme de recherche avec tabous proposé par Cordeau *et al.* [2007]. Comme pour le problème théorique, ces résultats laissent croire que cette métaheuristique est peut-être mal adaptée aux spécificités de ce problème.

L'objectif de ce chapitre est donc de montrer que les AG peuvent être des méthodes de résolution efficaces pour le problème industriel d'ordonnancement de voitures lorsque les différents mécanismes de l'algorithme sont adaptés aux spécificités du problème. Nous présentons les différents choix faits au niveau des opérateurs génétiques de l'algorithme

pour y parvenir. En particulier, nous proposons deux nouveaux opérateurs de croisement spécifiquement dédiés à la nature multi-objectifs du problème. La performance des approches proposées est établie expérimentalement en utilisant les différentes instances du Challenge ROADEF 2005 et comparée avec les meilleurs résultats obtenus durant le challenge.

Le reste du chapitre est organisé de la façon suivante. La Section 6.2 décrit les nouveaux opérateurs de croisement proposés pour le POV industriel. La Section 6.3, de son côté, présente le fonctionnement général de l'AG utilisé dans cette partie de la thèse. Par la suite, nous comparons, à la Section 6.4, les résultats expérimentaux de notre approche avec ceux d'autres algorithmes de la littérature.

6.2 Exploitation de l'information liées aux caractéristiques du problème

Comme nous l'avons constaté au Chapitre 4, les opérateurs de croisement classiques ne sont pas bien adaptés aux spécificités du POV ce qui pénalise grandement les performances des AG lors de la résolution du problème. Toutefois, nous avons obtenu des résultats très intéressants en proposant des opérateurs de croisement hautement spécialisés pour ce même problème.

Pour le problème multi-objectifs du challenge, Jaszkiewicz *et al.* [2004] ont utilisé un opérateur de croisement spécialisé (*common sequence preserving crossover*). L'objectif de cet opérateur est de trouver la plus longue sous-séquence commune entre deux solutions (parents) afin de la transférer à l'enfant en cours de création. Toutefois, même si les résultats de cette approche sont encourageants, ils n'ont pas permis à ces auteurs de se

qualifier parmi les 12 équipes finalistes du challenge ROADEF 2005. De son côté, Siala a utilisé deux opérateurs de croisement classiques pour résoudre le POVI. Les résultats obtenus en utilisant ces opérateurs sont inférieurs aux pires résultats obtenus par les 18 équipes de la deuxième phase du Challenge.

Les deux opérateurs proposés au Chapitre 4 pour le POV, appelés non-conflict position crossover (NCPX) et interest based crossover (IBX), exploitent des informations liées aux caractéristiques du problème pour réaliser le croisement. La notion utilisée par le NCPX et l'IBX pour exploiter l'information liée au problème est appelée *intérêt*. Dans les sections suivantes, nous allons montrer comment adapter les deux opérateurs de croisement NCPX et IBX à la problématique multi-objectifs du POVI.

6.2.1 Adaptation du calcul de l'intérêt pour le POVI

Avant de présenter les adaptations apportées aux deux opérateurs de croisement, il importe de redéfinir le concept de l'intérêt pour tenir compte de l'aspect multi-objectifs du POVI. On utilise le concept de l'*intérêt total pondéré* (TWI) pour établir s'il est intéressant de placer une voiture de la classe v, de couleur *color* à une position i donnée de la séquence. Ce calcul s'effectue selon l'équation suivante :

$$TWI_{v,color,i} = I_{v,i,HPO} * w_{HPO} + I_{v,i,COLOR} * w_{COLOR} + I_{v,i,LPO} * w_{LPO} \qquad (6.1)$$

où w_{HPO}, w_{COLOR} et w_{LPO} correspondent respectivement à la pondération accordée à chacun des objectifs (1000000, 1000 ou 1 selon la priorité des objectifs) et $I_{v,i,HPO}$, $I_{v,i,COLOR}$ et $I_{v,i,LPO}$ correspondent à l'intérêt de placer la classe de voitures v en position i pour chacun des objectifs. Définissons maintenant la notion d'intérêt selon chacun des objectifs.

L'intérêt $I_{v,i,COLOR}$ de placer une voiture de classe v à une position i donnée en cherchant à minimiser l'objectif COLOR est établi, selon l'Équation 6.2, à 1 s'il est possible de poursuivre la sous-séquence de couleur en cours avec une voiture de classe v. Dans le cas contraire, l'intérêt prend la valeur -1.

$$I_{v,i,COLOR} = \begin{cases} 1 & \text{si } nb(v_{color(i-1)}) > 0 \ \& \ run\_length < rl_{m} \\ -1 & \text{sinon} \end{cases} \quad (6.2)$$

$nb(v_{color(i-1)})$ indique le nombre de voitures de la classe v ayant la même couleur que la voiture placée à la position i-1, *run_length* indique la longueur de la sous-séquence de voitures consécutives possédant la même couleur que la voiture placée à la position i-1 et rl_{max} indique la longueur maximale d'une sous-séquence de même couleur. Par cette notion, on cherche à favoriser les classes de voitures comportant des voitures de même couleur que la voiture précédente de manière à allonger au maximum la longueur de la sous-séquence de couleur. À l'inverse, on pénalise les classes de voitures dont l'ajout entraîne un changement de couleur.

De leur côté, $I_{v,i,HPO}$ et $I_{v,i,LPO}$ représentent l'intérêt de placer une voiture de la classe v à une position i donnée de la séquence en cherchant à minimiser les objectifs HPO et LPO. Cet intérêt, selon l'Équation 6.3, corresponds à la difficulté de la classe v si cette classe n'engendre aucun nouveau conflit pour les options prioritaires (*obj* = HPO) ou pour les options non-prioritaires (*obj* = LPO). Dans le cas contraire, on définit un coût correspondant au nombre de nouveaux conflits générés sur les options (prioritaires ou non-prioritaires) pour décourager l'insertion de cette classe en position i.

$$I_{v,i,k} = \begin{cases} D_{v,k} & \text{if } NbNewConflicts_{v,i,k} = 0 \\ -NbNewConflicts_{v,i,k} & \text{otherwise} \end{cases} \quad (6.3)$$

$NbNewConflicts_{v,i,obj}$ correspond au nombre de nouveaux conflits engendrés pour les options prioritaires (*obj* = HPO) ou pour les options non-prioritaires (*obj* = LPO) par l'addition d'une voiture de la classe v à la position i et $D_{v,obj}$ indique la difficulté de la classe de voitures v pour les options prioritaires (*obj* = HPO) ou pour les options non-prioritaires (*obj* = LPO). L'idée derrière cette notion consiste simplement à pénaliser les classes de voitures dont l'ajout entraîne des conflits additionnels pour les options (prioritaires ou non-prioritaires) en considérant ce nombre de nouveaux conflits comme un coût. À l'inverse, lorsque l'ajout d'une classe n'entraîne pas de nouveaux conflits sur les options, on évalue alors le profit de placer cette classe en fonction de sa difficulté.

6.2.2 L'opérateur de croisement IBX multi-objectifs (IBX^{MO})

Le fonctionnement du croisement IBX^{MO} pour le problème industriel d'ordonnancement de voitures s'inspire du fonctionnement du croisement *IBX* présenté au chapitre 4 de cette thèse et se déroule en trois grandes étapes. La première étape consiste à déterminer aléatoirement deux points de coupure pour chaque parent P_1 et P_2. Une fois ces points de coupure temporaires déterminés, les couleurs des voitures précédentes au 1er point de coupure ainsi que celles des voitures situées immédiatement après le 2^{ième} point de coupure dans P_1 sont vérifiées afin de ne pas interrompre une sous-séquence de couleur en cours. Tant que la couleur des voitures situées avant le 1er point de coupure est la même que celle de la voiture située au point de coupure, on déplace le point de coupure vers la gauche. À l'inverse, tant que la couleur de la voiture située au niveau du deuxième point de coupure est identique à celle située après ce point de coupure, on déplace le deuxième point de coupure vers la droite.

À la Figure 6.1, une fois que les points de coupure sont fixés pour les deux parents P_1 = {22351446/46222622} et P_2 = {32421465/24662222}, la sous-séquence de gènes {351/222} comprise entre les deux points de coupure du premier parent ($a_1 \in P_1$) est directement recopiée dans l'enfant. Ensuite, deux listes non ordonnées (L_1 et L_2) sont respectivement créées à partir des sous-séquences b_3 = {32/24} et b_4 = {465/222} de P_2 et serviront à compléter le début et la fin de l'enfant E_1. Cependant, lors de cette opération, il se peut qu'une partie de l'information soit perdue par l'introduction de doublons. Dans l'exemple de la Figure 6.1, on remarque ainsi que les contraintes de production pour les classes de voitures 2, 3, 4 et 5 ne sont plus respectées. De manière à restaurer l'intégralité des gènes et qu'exactement c_v voitures de classe v soient produites, un remplacement des gènes 3/2 et 5/2 (obtenu à partir de a_1-a_2) dont le nombre dépasse les contraintes de production est fait à partir des gènes 4/6 et 2/6 (obtenu à partir de a_2-a_1) dont le nombre est maintenant inférieur aux contraintes de production. Ce remplacement est effectué aléatoirement à la deuxième étape pour ajuster les listes L_1 et L_2.

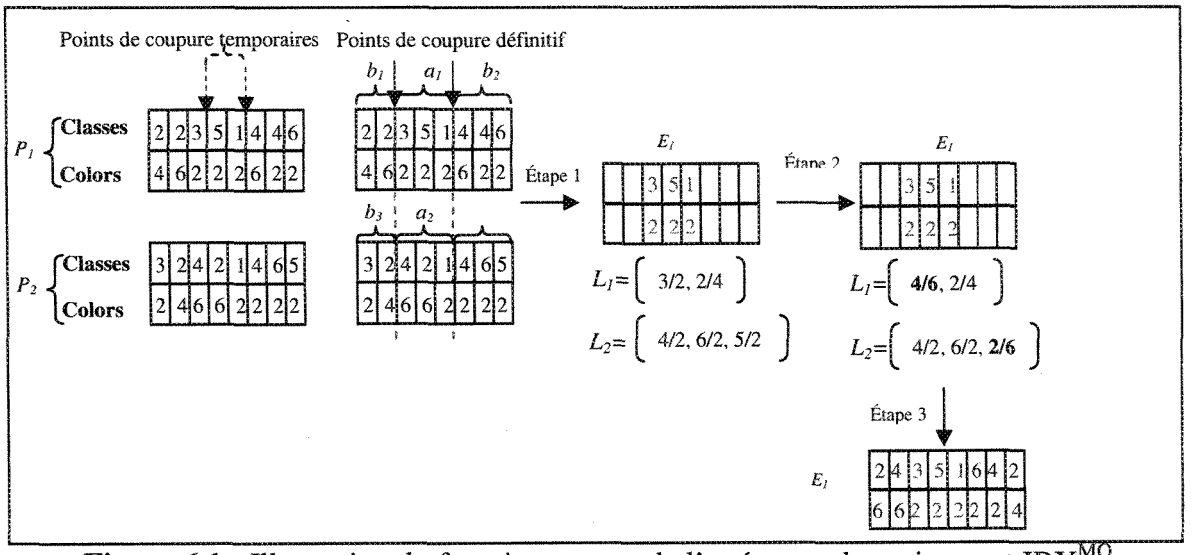

Figure 6.1 : Illustration du fonctionnement de l'opérateur de croisement IBX[MO]

Finalement, la dernière étape consiste à reconstruire le début et la fin de l'enfant à partir des deux listes L_1 et L_2 corrigées en utilisant *TWI* tel que défini à l'Équation 6.1. Dans les deux cas, la reconstruction se fait à partir du point de coupure vers le début ou la fin de l'enfant, selon le cas. Par exemple, pour reconstruire le début de l'enfant, on calcule *TWI* pour chacune des voitures $\in L_1$. La classe de voitures v à placer est choisie dans 95% des cas de manière gourmande et dans 5% des cas de manière probabiliste en utilisant le principe de sélection par roulette [Goldberg 1989]. La couleur associée à cette classe complète alors le deuxième vecteur de la solution à cette position. On retire alors cette classe de la liste L_1 et on recommence le calcul pour la position suivante. On fait de même pour reconstruire la fin de l'enfant à partir de la liste L_2.

Un deuxième enfant est créé suivant le même processus, mais en partant cette fois-ci du parent 2.

6.2.3 L'opérateur de croisement NCPX multi-objectifs ($NCPX^{MO}$)

Le fonctionnement du croisement $NCPX^{MO}$ pour le problème industriel d'ordonnancement se déroule en deux grandes étapes. À l'étape 1, il faut tout d'abord sélectionner un parent P_1 et établir dans ce chromosome le nombre de positions ne faisant pas partie d'un conflit pour les objectifs HPO ($nbpos_{HPO}$) et LPO ($nbpos_{LPO}$) ainsi que le nombre de positions où il n'y a pas de changement de couleur ($nbpos_{COLOR}$). Par la suite, on tire aléatoirement, pour chaque objectif *obj* (*obj* = *HPO, LPO, COLOR*), un nombre nbg_{obj} compris entre 0 et $nbpos_{obj}$. Ces trois nombres servent à déterminer, pour chaque objectif *obj*, le nombre de « bons » gènes qui conserveront dans l'enfant E_1, la même position qu'ils occupaient dans P_1. De manière à tenir compte de l'ordre de priorité des objectifs, on doit

s'assurer que le nombre de bons gènes conservés pour l'objectif principal soit supérieur ou égal à celui de l'objectif secondaire et ainsi de suite. Une fois ces nombres déterminés, une position de départ (*sPos*) est sélectionnée aléatoirement entre 1 et *nc* dans l'enfant à créer. Le processus de copie des bons gènes de P_1 vers l'enfant en cours de création commence à partir de *sPos* en considérant tout d'abord l'objectif principal. Si on atteint la fin du chromosome et que le nombre de gènes copiés pour l'objectif *obj* est inférieur à son nbg_{obj} correspondant, le processus de copie recommence en partant du début de l'enfant jusqu'à *sPos-1*. On reprend le même processus avec les autres objectifs en considérant toutefois les gènes déjà copiés. Les gènes de P_1 sont alors utilisés pour constituer une liste L non ordonnée des voitures restant à placer. On détermine ensuite aléatoirement une position (*Pos*) à partir de laquelle le reste du chromosome E_1 va être complété. À la deuxième étape, les voitures contenues dans L sont ordonnées en fonction de leur TWI. En cas d'égalité, si une des voitures se retrouve dans P_2 à la position à compléter, elle est alors privilégiée. Dans le cas contraire, on tire aléatoirement une voiture parmi celles impliquées dans l'égalité.

Le fonctionnement de cet opérateur de croisement est illustré à la Figure 6.2 pour deux parents P_1 = {21352446/62224622} et P_2 = {32621454/26242622} avec l'ordre de priorité des objectifs HPO-LPO-COLOR. Supposons que l'évaluation de P_1 donne 5 positions ne faisant pas partie d'un conflit pour l'objectif HPO ainsi que pour l'objectif LPO (indiquées par des 0 dans les vecteurs « conflits sur HPO et LPO » en-dessous du chromosome P_1), 4 positions où il n'y a pas de changement de couleur (indiquées par des 0 dans le vecteur « changements de couleur » en-dessous du chromosome P_1) et qu'on a déterminé

aléatoirement que $nbg_{HPO} = 4$, $nbg_{LPO} = 2$, $nbg_{COLOR} = 1$ et $sPos = 3$. À partir de $sPos$ et en considérant l'objectif HPO, on peut copier les gènes 5/2, 4/6, 4/2 et 2/6 dans l'enfant. En reprenant la même procédure avec l'objectif LPO, on note que trois bons gènes (5/2, 4/2 et 2/6) ont déjà été transférés dans l'enfant, ce qui respecte le nombre de bons gènes à transférer pour cet objectif. De même, deux bons gènes (5/2 et 2/6) sont déjà présents dans l'enfant pour l'objectif COLOR, ce qui respecte le nombre de bons gènes à transférer pour cet objectif. Les gènes 1/2, 3/2, 2/4 et 6/2 de P_1 servent alors à constituer la liste non ordonnée L. À la deuxième étape, en supposant que $Pos = 7$ et que le calcul de TWI place les gènes dans l'ordre 3/2, 2/4, 6/2, 1/2 avec une égalité sur les gènes 2/4 et 6/2. Alors, on place le gène 3/2 en position 8, on privilégie de placer le gène 6/2 en position 3 car il occupe cette position dans $P2$ et les gènes 2/4 et 1/2 sont placés respectivement dans les positions 2 et 5. Dans cet exemple, les gènes 1/2 et 6/2 sont donc directement hérités de P_2 puisqu'ils se retrouvent à la même position dans le deuxième parent. L'enfant généré à partir de P_1 et P_2 est alors E_1= {22651443/64222622}.

Un deuxième enfant est créé de manière similaire en partant cette fois-ci de P_2.

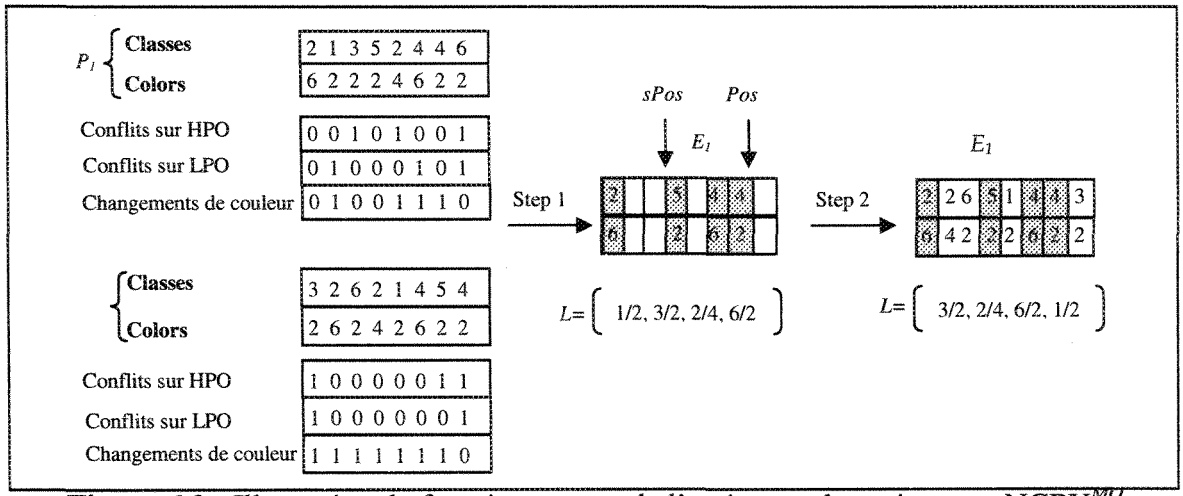

Figure 6.2 : Illustration du fonctionnement de l'opérateur de croisement NCPX^{MO}

6.3 Description de l'algorithme génétique pour le POVI

Dans cette section, nous présentons la description complète de l'algorithme génétique utilisé pour résoudre le POVI multi-objectifs.

6.3.1 Représentation

Tel que présenté à la Figure 2.3 (b) du Chapitre 2, au lieu d'opter pour une représentation classique sous forme de chaîne de bits qui apparaît peu adaptée pour ce type de problème, une solution du POVI est représentée à l'aide de deux vecteurs de longueur *nc* correspondant respectivement à la classe et à la couleur de la voiture.

6.3.2 Création de la population initiale

Dans l'implémentation proposée, les individus de la population initiale sont générés de deux façons : à 70 % de manière aléatoire et à 30 % en utilisant une heuristique gourmande basée sur la notion d'intérêt. Deux heuristiques gourmandes sont utilisées selon l'objectif principal. Dans le cas où l'objectif principal est la minimisation du nombre de changements de couleur (COLOR), l'heuristique gourmande utilisée est appelée *greedy_color*. Dans le cas où l'objectif principal est la minimisation du nombre de conflits sur les options prioritaires (HPO), l'heuristique gourmande utilisée est appelée *greedy_ratio*. Les algorithmes 6.1 et 6.2 résument respectivement le fonctionnement des deux heuristiques. Notons que, dans tous les cas, on s'assure que les individus produits sont des solutions réalisables.

Algorithme 6.1 : *greedy_color*

1: Commencer avec un individu x constitué des voitures de la journée *J-1*
2 : i=1 ; *run_length* =1
3 : *previous_color* = Colors(-1)
4: **Tant qu'**il y a des classes de voitures à placer
5: **Si** *run_length* < rl_{max} er qu'il reste des classes avec *previous_color* **alors**
6: *color* = *previous_color*
7: *run_lenght* ++
8: **Sinon**
9: Choisir aléatoirement *previous_color* != *color*
10: *run_length* =1
11: **Fin Si**
12: Limiter la sélection aux m classes ayant la couleur choisie
13: **Pour** chacune de ces m classes
14: Évaluer l'intérêt $I_{v,i,COLOR}$ d'ajouter une voiture de classe v à la position i
15: **Fin Pour**
16: Choisir aléatoirement un nombre *rnd* entre *0* et *1*
17: **Si** *rnd* < 0.95 **alors**
18: Choisir la classe v selon Arg Max $\{I_{v,i,COLOR}\}$
19: En cas d'égalité, choisir la classe v aléatoirement
20: **Sinon**
21: Choisir v selon le principe de la sélection par roulette
22: **Fin Si**
23: x(i) = v / *color*
24 : i=i+1
25: **Fin Tant que**

Algorithme 6.2 : *greedy_ratio*

1: Commencer avec un individu x constitué des voitures de la journée *J*-1
2 : i=1; *run_length* =1
3 : *previous_color* = Colors(-1)
4: **Tant qu'**il y a des classes de voitures à placer
5: **Si** *run_length* = rl_{max} **alors**
6: Exclure les voitures pour lesquelles *colo r*= *previous_color* de la liste de voitures candidates.
7: **Fin Si**
8: **Pour** chaque classe candidate v
9: Évaluer l'intérêt $I_{v,i,HPO}$ d'ajouter v à la position i
10: **Fin pour**
11: Choisir aléatoirement un nombre *rnd* entre *0* et *1*
12: **Si** *rnd* < 0.95 **alors**
13: Choisir la classe v selon Arg Max $\{I_{v,i,HPO}\}$
14: En cas d'égalité, briser l'égalité de manière lexicographique en utilisant l'intérêt du second puis du troisième objectif ($I_{v,i,LPO}$ ou $I_{v,i,COLOR}$). En cas d'égalité sur les 3 objectifs, choisir une classe aléatoirement.
15: **Sinon**
16: Choisir v selon le principe de la sélection par roulette
17: **Fin Si**
18 : Pour la classe v choisie, choisir *color* avec Arg Max $\{I_{v,i,COLOR}\}$. En cas d'égalité, choisir *color* aléatoirement
19: $x(i)$ = v / *color*
20: **Si** *run_length* = rl_{max} **ou** *color* <> *previous_color* **alors**
21: *run_length*= 1
22 : **Sinon**
23 : *run_length*= *run_length* +1
24 : **Fin Si**
25 : *previous_color*=*color*
26 : i=i+1
27: **Fin Tant que**

L'heuristique *greedy_color* commence avec une solution initiale formée des voitures ordonnancées à la journée précédente. En fait, pour faire le lien avec la journée précédente, il suffit de connaître la valeur maximale de s_o pour l'ensemble des options et cette valeur détermine la longueur de la séquence nécessaire à la fin de la journée précédente pour réaliser l'évaluation de la solution. Par la suite, on initialise le compteur de positions i, la longueur de la sous-séquence courante de même couleur (*run_length*) et la couleur de la dernière voiture produite à la journée précédente (*previous_color*) (lignes 2-3). Le processus itératif de sélection de la prochaine voiture à placer dans la séquence en construction (lignes 4-25) commence par le choix de la couleur *color* à sélectionner en

fonction de rl_{max} et de *previous_color* (lignes 5-11). Une fois la couleur de la prochaine voiture à placer déterminée, on limite le processus de sélection aux m classes de voitures possédant cette couleur. À cette étape, pour chacune de ces classes de voitures, on calcule l'intérêt $I_{v,i,COLOR}$ de placer une voiture de classe v à la position courante i. Dans 95 % des cas, la classe sélectionnée est celle avec le plus grand $I_{v,i,COLOR}$ ($Arg\ Max\ \{\ I_{v,i,COLOR}\ \}$). À l'opposé, dans 5 % des cas, la classe de voitures à placer est sélectionnée selon le principe de la roulette. Une fois la classe de voitures et la couleur sélectionnées, on ajoute la classe de voitures sélectionnée v et la couleur choisie *color* à la position i de la séquence x en construction (ligne 23). Ce processus est ainsi répété jusqu'à ce qu'une séquence complète de voitures soit construite. L'objectif principal de l'heuristique *greedy_color* est donc de minimiser, de manière gourmande, le nombre de changements de couleurs.

La seconde heuristique de construction proposée, appelée *greedy_ratio*, utilise aussi une approche gourmande pour construire un individu x. Toutefois, dans cette heuristique, c'est l'intérêt $I_{v,i,HPO}$ qui est utilisé comme critère gourmand principal pour déterminer la prochaine voiture à ajouter à la fin de la séquence x en cours de construction. Comme pour l'heuristique *greedy_color*, la procédure *greedy_ratio* commence avec une solution initiale formée des voitures déjà ordonnancées de la journée précédente. On initialise les différents compteurs ainsi que la couleur de la voiture précédente de manière similaire à l'heuristique *greedy_color*. La boucle principale de l'algorithme (lignes 4-27) vérifie tout d'abord que la longueur maximale pour une sous-séquence de couleur identique, rl_{max}, n'a pas été atteinte. Si rl_{max} a été atteint, on retire toutes les classes de voitures de la même couleur que *previous_color* de la liste des classes pouvant être placées à la position courante i (liste des

classes de voitures candidates). Cette étape permet d'assurer que la solution générée est une solution valide. Par la suite, pour chaque classe candidate v, on calcule l'intérêt $I_{v,i,HPO}$ de placer une voiture de la classe v à la position courante i selon l'objectif HPO. Par la suite, le choix de la prochaine classe de voitures à positionner dans la séquence se fait dans 95 % des cas en choisissant la classe ayant le plus grand $I_{v,i,HPO}$. On note ici, qu'en cas d'égalité sur les $I_{v,i,HPO}$, l'égalité est brisée en choisissant la classe ayant le plus grand intérêt respectivement sur le second, puis sur le troisième objectif selon la hiérarchisation des objectifs. Dans 5 % des cas, la classe de voitures à placer est sélectionnée selon le principe de la roulette. Une fois la classe de voitures choisie, on sélectionne la couleur de la voiture à ajouter parmi les couleurs disponibles pour cette classe en fonction $I_{v,i,COLOR}$. Si toutes les couleurs ont le même intérêt pour cette classe de voiture, on choisit la couleur aléatoirement. Par la suite, on ajoute la classe de voiture v sélectionnée et la couleur *color* choisie à la position i de la séquence x en construction. Finalement, on met à jour les différents compteurs (*run_length et i*) ainsi que *previous_color*. Ce processus est ainsi répété jusqu'à ce qu'une séquence complète de voitures soit construite.

6.3.3 Sélection

Plusieurs schémas de sélection auraient pu être envisagés pour l'AG permettant de résoudre POVI multi-objectifs. Toutefois, comme elle est peu coûteuse à mettre en œuvre et à exécuter et qu'elle a fait ses preuves sur le POV au Chapitre 4, la procédure de sélection retenue pour résoudre le POVI est une sélection par tournoi binaire.

6.3.4 Opérateur de mutation

Selon la hiérarchisation des objectifs du problème à résoudre, quatre opérateurs de mutation sont utilisés par notre AG : l'*inversion_simple*, l'*échange_aléatoire*, le *group_échange* et l'*inversion_block*. Notons que ces quatre opérateurs ont souvent été utilisés dans la littérature pour le problème industriel d'ordonnancement de voitures pour explorer le voisinage à l'intérieur d'une méthode de recherche locale [Solnon *et al.* 2007]. Comme nous l'avons mentionnée dans la partie problématique de ce document, trois hiérarchisations des objectifs sont possibles : *HPO-COLOR-LPO*, *HPO-LPO-COLOR* et *COLOR-HPO-LPO*. Dans le cas de problèmes avec une hiérarchisation des objectifs selon la priorité HPO-COLOR-LPO et HPO-LPO-COLOR, les opérateurs de mutation utilisés sont l'inversion_simple et l'échange_aléatoire. Une inversion_simple consiste à sélectionner aléatoirement deux positions et à inverser la sous-séquence comprise entre ces deux positions. Un échange_aléatoire consiste simplement à échanger aléatoirement la position de deux voitures appartenant à des classes différentes. Pour la hiérarchisation des objectifs selon l'ordre COLOR-HPO-LPO, les opérateurs de mutation utilisés sont le group_échange et l'inversion_block. La mutation par group_échange consiste à échanger aléatoirement la position de deux groupes de voitures de même couleur. De son côté, l'inversion_block consiste à sélectionner une sous-séquence de voitures consécutives de même couleur et à inverser la position des voitures comprises dans cette sous-séquence.

6.3.5 Stratégie de remplacement

L'AG proposé est un algorithme élitiste. Afin de garantir cet élitisme, la stratégie de remplacement utilisée est un remplacement déterministe de type (λ+μ). Dans ce schéma de

remplacement, les populations parent et enfant sont jointes et triées en ne conservant que les μ meilleurs individus pour former la prochaine génération.

L'Algorithme 6.3 décrit le fonctionnement général de l'algorithme génétique proposé pour résoudre le POVI. L'AG commence par la génération de la population initiale POP_0 dans laquelle tous les individus sont évalués. Par la suite, le processus itératif de l'AG commence. À chaque génération t, un nombre limité d'individus est sélectionné, en fonction d'une probabilité de croisement (p_c), pour être recombiné. Les enfants générés après croisement sont mutés selon une probabilité de mutation (p_m). Finalement, la population courante est mise à jour en sélectionnant les meilleurs individus des populations Parent (POP_t) et Enfant (Q_t). Ce processus est répété jusqu'à ce qu'un critère d'arrêt soit rencontré.

Algorithme 6.3 : AG proposé pour le POVI

1: Générer la population POP_0 aléatoirement ou en utilisant les deux heuristiques gourmandes
2: Évaluer chaque individu $x \in POP_0$ et trier POP_0
3: **Tant qu'**aucun critère d'arrêt n'est atteint
4: **Tant que** $|Q_t| < N$
5: Choisir aléatoirement un nombre *rnd* entre *0* et *1*
6: **Si** $rnd < p_c$ **alors**
7: Sélectionner deux parents P_1 et P_2
8: Créer deux enfants E_1 et E_2 en utilisant le croisement $NCPX^{MO}$ ou IBX^{MO}
9: Évaluer les enfants créés
10: Choisir aléatoirement un nombre *rnd* entre *0* et *1*
11: **Si** $rnd < p_m$ **alors**
12: Muter et évaluer les enfants
13: **Fin Si**
14: Ajouter E_1 et E_2 à Q_t
15: **Fin Si**
16: **Fin Tant que**
17: Trier $Q_t \cup POP_t$
18: Choisir les N premiers individus de $Q_t \cup POP_t$ pour former la prochaine génération POP_{t+1}
19: $t = t + 1$
20: **Fin Tant que**
21: **Retourner** le meilleur individu trouvé

6.4 Expérimentations numériques

L'AG présenté dans ce chapitre a été implémenté en C++ avec Visual Studio .Net 2005. L'ordinateur utilisé pour les expérimentations numériques est un Dell équipé d'un processeur Pentium Xeon 3.6 Ghz avec 1 Go de mémoire vive sous Windows XP. Dans les expérimentations numériques réalisées, les paramètres N, p_c, p_m, T_{max} représentant respectivement la taille de la population, la probabilité de croisement, la probabilité de mutation et le temps maximum alloué à l'algorithme sont fixés à 5, 0.8, 0.35 et 350 secondes. La faible taille de la population ainsi que les probabilités de croisement et de mutation ont été déterminées en se basant sur les résultats théoriques de Goldberg [Goldberg 1989] ainsi que sur les travaux de Coello Coello et Pulido [2001]. Selon ces auteurs, une taille de population très faible suffit à obtenir une convergence indépendamment de la longueur du chromosome. Ainsi, l'utilisation d'une population restreinte avec un fort taux de croisement permet, d'une part, d'augmenter l'efficience de l'AG pour le POV industriel en limitant le temps de calculs pour l'évaluation de la fitness de chaque individu. En effet, l'évaluation de la fitness d'une solution pour le POV industriel représente un temps de calcul non négligeable. D'autre part, une forte probabilité de croisement permet généralement une meilleure exploration de l'espace de recherche [Grefenstette 1986]. Hormis les difficultés liées à la nature multi-objectifs du POV industriel, une limite de temps de 600 secondes sur un PC Pentium4/1.6 Ghz/win2000/1 Go RAM était fixée pour le Challenge ROADEF 2005 et celle-ci a été respectée afin de reproduire le plus fidèlement possible les conditions expérimentales utilisées lors du

challenge. Compte tenu de l'ordinateur utilisé, le temps maximum alloué à l'algorithme a donc été fixé à 350 secondes.

Trois versions de l'AG serviront à réaliser les essais numériques. La première version intègre le croisement $NCPX^{MO}$, la seconde utilise le croisement IBX^{MO} et la troisième intègre le croisement $NCPX^{MO}$ avec une procédure de recherche locale (LS).

6.4.1 Jeux d'essais

La performance de l'AG multi-objectifs proposé est évaluée à l'aide des trois jeux d'essais fournis par le constructeur Renault et disponibles sur Internet (http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2005/). Le premier ensemble (Ensemble A) contient 16 jeux de données pour l'ordonnancement de 334 à 1314 voitures comportant de 6 à 22 options formant de 36 à 287 classes de voitures avec de 11 à 24 couleurs différentes. Cet ensemble a permis d'évaluer les équipes lors de la phase de qualification et ainsi établir les 18 équipes passant à la ronde suivante. Le deuxième ensemble (Ensemble B) est constitué, pour sa part, d'un large éventail de 45 instances composées chacune de 65 à 1270 voitures avec de 4 à 25 options, entre 11 à 339 classes de voitures avec de 4 à 20 couleurs. Cet ensemble a permis d'établir les 12 équipes finalistes du Challenge ROADEF 2005. Finalement, le dernier ensemble (Ensemble X) contient 19 instances avec de 65 à 1319 voitures à ordonnancer, de 5 à 26 options, entre 10 à 328 classes de voitures et de 5 à 20 couleurs différentes. Cet ensemble a servi à l'évaluation finale des 12 équipes pour établir l'équipe gagnante.

En comparaison au problème théorique d'ordonnancement de voitures dont les plus grandes instances comportent 400 voitures, 5 options et entre 18 et 24 classes de voitures, la résolution du POVI multi-objectifs représente donc un défi de taille.

6.4.2 Comparaison expérimentale

Pour analyser la performance des algorithmes proposés dans ce chapitre, une comparaison est réalisée avec les meilleurs résultats obtenus lors du Challenge ROADEF 2005 pour les 61 instances des ensembles *A* et *B*. Tous ces résultats proviennent du site du challenge à l'adresse http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2005/. Ainsi, les Tableaux 6.1 à 6.3 présentent les résultats comparatifs de l'*AG-NCPX*^{MO}, de l'*AG-IBX*^{MO} et de l'*AG-NCPX*^{MO}*+LS* avec les résultats obtenus par l'équipe gagnante du challenge et avec le GLS de Jaszkiewicz [2004] qui est le meilleur algorithme évolutionnaire proposé dans le challenge. En se basant sur les résultats obtenus par les 18 équipes finalistes et les trois algorithmes proposés dans ce chapitre, on retrouve également, dans les Tableaux 6.1 à 6.3, le rang obtenu par chacune des solutions trouvées pour une même instance.

Dans ces tableaux, les instances sont regroupées en trois séries :

- celles où l'objectif principal est la minimisation du nombre de conflits sur les options prioritaires (HPO) et où les contraintes sur les options prioritaires sont, selon Renault, « faciles » à satisfaire (Tableau 6.1) ;
- celles où l'objectif principal est la minimisation du nombre de conflits sur les options prioritaires (HPO) et où les contraintes sur les options prioritaires sont, selon Renault, « difficiles » à satisfaire (Tableau 6.2) ; et

- celles pour lesquels l'objectif principal est la minimisation du nombre de changements de couleur (COLOR) (Tableau 6.3).

Chaque ligne des Tableaux 6.1 à 6.3 donne respectivement le nom de l'instance, la valeur et le rang de la solution obtenue par l'équipe gagnante du challenge, par le GLS et par chacune des versions de l'AG. En ce qui concerne les meilleurs résultats obtenus pour chaque instance, ils sont indiqués en caractères gras dans les différents tableaux. Il est important de mentionner ici que, comme pour les résultats du challenge, les algorithmes génétiques proposés dans cette partie ont été exécutés à une seule reprise. Les résultats présentés dans les différents tableaux représentent la somme pondérée des objectifs ($F(X)$) pour la solution obtenue tel que présentée à l'Équation 2.7 du Chapitre 2.

Le Tableau 6.1 présente les résultats pour les instances, selon Renault, ayant des contraintes « faciles » à satisfaire pour les options prioritaires. Ces instances ont l'ordre de priorité des objectifs HPO-LPO-COLOR ou HPO-COLOR-LPO. En examinant les résultats de ce tableau, on note que l'*AG-NCPX*^{MO} donne de meilleurs résultats que l'*AG-IBX*^{MO} pour toutes les instances de l'ensemble *A* et *B*, à l'exception d'une, l'instance 028_ch2_S23_J3 selon l'ordre des objectifs HPO_COLOR_LPO pour laquelle les des algorithmes obtiennent des résultats identiques. Ces résultats mettent en évidence la supériorité du croisement $NCPX^{MO}$ par rapport au croisement IBX^{MO} dans le contexte de l'ordonnancement industriel de voitures. La différence de performance entre les deux algorithmes s'explique par l'exploitation de l'information sur les positions non conflictuelles effectuée par le croisement $NCPX^{MO}$ qui lui permet de mieux intensifier la recherche dans la limite du temps alloué.

	Équipe gagnante	*GLS (Rang)*	*AG-IBX^{MO} (Rang)*	*AG-NCPX^{MO} (Rang)*	*AG-NCPX^{MO} +LS (Rang)*
Ensemble *A*					
HPO_COLOR_LPO					
022_3_4	**31001 (1)**	37000 (14)	32022 (11)	32001 (8)	**31001 (1)**
025_38_1	231452 (4)	262460 (15)	262460 (15)	231772 (6)	**229295 (1)**
064_38_2_ch1	**112759 (1)**	139757 (15)	184775 (17)	164760 (16)	**112759 (1)**
064_38_2_ch2	**34051 (1)**	36056 (15)	37156 (16)	34052 (8)	**34051 (1)**
HPO_ LPO_COLOR					
025_38_1	99720 (2)	200711 (10)	270686 (14)	150767 (6)	**97076 (1)**
Ensemble *B*					
HPO_COLOR_LPO					
022_S22-J1	**19144 (1)**	23144 (13)	21174 (12)	20176 (9)	**19144 (1)**
025_S22-J3	**172180 (1)**	281877 (20)	264156 (19)	222711 (13)	179378 (3)
028_ch_S22_J2	**54049124 (1)**	54059164 (13)	54072436 (19)	54063113 (14)	**54049124 (1)**
028_ch2_S23_J3	**4071 (1)**	**4071 (1)**	5078 (17)	**4071 (1)**	**4071 (1)**
039_ch1_S22_J4	**78089 (1)**	92308 (17)	82731 (14)	79128 (7)	78220 (3)
039_ch3_S22_J4	189146 (4)	199223 (17)	195718 (15)	192160 (12)	189122 (2)
048_ch1_S22_J3	**161378 (1)**	186438 (18)	180440 (16)	170377 (12)	161401 (2)
064_ch1_S22_J3	**130187 (1)**	158222 (14)	183149 (19)	177376 (17)	134368 (7)
064_ch2_S22_J4	**130069 (1)**	**130069 (1)**	130088 (14)	**130069 (1)**	**130069 (1)**
HPO_LPO_COLOR					
022_S22_J1	**3109 (1)**	3138 (12)	3191 (14)	3186 (13)	**3109 (1)**
025_S22_J3	**3912479 (1)**	3926649 (11)	4041787 (19)	3928619 (12)	**3912479 (1)**
028_ch1_S22_J2	54003079 (4)	54021114 (12)	54065112 (14)	54017116 (9)	54003077 (2)
028_ch2_S23_J3	**70006 (1)**	**70006 (1)**	**70006 (1)**	**70006 (1)**	**70006 (1)**
039_ch1_ S22_J4	**29117 (1)**	29385 (16)	29375 (15)	29272 (11)	**29117 (1)**
039_ch3_ S22_J4	**197 (1)**	276 (12)	315 (17)	290 (13)	201 (2)
048_ch1_S22_J3	**200 (1)**	298 (15)	367 (19)	298 (15)	203 (3)
064_ch1_S22_J3	**182 (1)**	1359 (18)	421 (15)	240 (10)	**182 (1)**
064_ch2_S22_J4	**69130 (1)**	69131 (9)	69161 (19)	69132 (11)	**69130 (1)**

Tableau 6.1 : Résultats comparatifs de l'*Équipe gagnante*, du *GLS*, de l'*AG-IBX^{MO}*, de l'*AG-NCPX^{MO}* et de l'*AG-NCPX^{MO}+LS* pour les instances « faciles » en options prioritaires avec l'objectif HPO comme objectif prioritaire

À l'exception de l'instance 028_ch2_S23_J3 selon l'ordre HPO_LPO_COLOR qui s'avère facile à résoudre pour tous les algorithmes, l'*AG-IBX^{MO}* obtient un classement qui s'étend entre le 11^{ième} et le 19^{ième} rang tandis que l'*AG-NCPX^{MO}* se classe entre le 1er et le 17^{ième} rang selon les instances. Il faut toutefois noter que, contrairement à la plupart des algorithmes du challenge, l'*AG-IBX^{MO}* et l'*AG-NCPX^{MO}* n'utilisent pas de processus de recherche locale.

Lorsque l'on compare maintenant les résultats de l'*AG-NCPX^{MO}* et de l'*AG-IBX^{MO}* à ceux du *GLS* de Jaszkiewicz *et al.* [2004], on note, pour l'ensemble *A*, que le GLS obtient

généralement de meilleurs résultats que l'*AG-IBX*^{MO} mais que l'*AG-NCPX*^{MO} surpasse facilement le *GLS*. En effet, le *GLS* domine l'*AG-IBX*^{MO} pour seulement 3 instances de l'ensemble *A*, est inférieur sur une instance tout en obtenant des résultats identiques sur l'autre instance. À l'opposé, le *GLS* obtient de moins bons résultats que l'*AG-NCPX*^{MO} sur 4 des 5 instances de l'ensemble *A* présentées au Tableau 6.1. Ces résultats se confirment avec quelques nuances sur les instances de l'ensemble *B*. Ainsi, le *GLS* obtient de meilleurs résultats que l'*AG-IBX*^{MO} pour 10 instances, est inférieur pour 7 instances et obtient un résultat identique sur l'instance restante. Pour sa part, le *GLS* obtient de meilleurs résultats que l'*AG-NCPX*^{MO} pour 6 instances, est inférieur pour 8 instances et obtient des résultats identiques pour les 4 autres instances. On remarque donc, pour l'ensemble *B*, un léger avantage pour l'*AG-NCPX*^{MO}. Ces résultats sont très encourageants compte tenu du fait que *GLS* est un algorithme mimétique, c'est-à-dire un algorithme hybridant un algorithme génétique avec une recherche locale.

En comparant maintenant les résultats de l'*AG-NCPX*^{MO} et de l'*AG-IBX*^{MO} aux résultats de l'équipe gagnante du Challenge ROADEF 2005, on constate que les résultats des deux algorithmes génétiques proposés sont bien inférieurs en terme de qualité. Cet écart s'explique par le manque d'intensification de la recherche dans ce type d'approche. En combinant l'*AG-NCPX*^{MO} à une procédure de recherche locale similaire à celle proposée par Estellon *et al.* [Estellon *et al.* 2007] et en utilisant les opérateurs de mutation présentés à la Section 4.4 pour explorer le voisinage, on obtient les résultats présentés à la dernière colonne du Tableau 6.1. On rappelle ici que l'*AG-NCPX*^{MO}*+LS* est soumis aux mêmes limites de temps que celles imposées aux différents algorithmes testés dans cette partie. On

note que l'*AG-NCPX*^{MO}*+LS* est nettement supérieur à l'*AG-NCPX*^{MO} et parvient à rivaliser avec les résultats de la meilleure équipe du challenge pour toutes les instances de l'ensemble *A*. L'*AG-NCPX*^{MO}*+LS* se classe au 1er rang pour toutes ces instances et trouve même, sur les instances 022_3_4 selon l'ordre des objectifs HPO_COLOR_LPO et 025_38_1 selon l'ordre des objectifs HPO_LPO_COLOR, de nouveaux minimums. Pour les instances de l'ensemble *B*, l'*AG-NCPX*^{MO}*+LS* obtient des résultats identiques à ceux de l'équipe gagnante et aux meilleurs résultats de l'ensemble des équipes pour 10 des 16 instances. Pour les autres instances, on observe un faible écart qui provient des résultats obtenus sur les objectifs secondaires et tertiaires compte tenu de la pondération de 1000000 accordée à l'objectif principal dans la somme pondérée des objectifs. En effet, l'*AG-NCPX*^{MO}*+LS* obtient toujours un classement entre le 1er et le 3^{ième} rang, à l'exception de l'instance 064_ch1_S22_J3 selon l'ordre des objectifs HPO_COLOR_LPO où il obtient le 7^{ième} rang.

Le Tableau 6.2 présente les résultats pour les instances, selon Renault, ayant des contraintes « difficiles » à satisfaire pour les options prioritaires. Ces instances ont l'ordre de priorité des objectifs HPO-LPO-COLOR ou HPO-COLOR-LPO. On note, encore une fois, que l'*AG-NCPX*^{MO} domine nettement l'*AG-IBX*^{MO}. Ainsi, pour les instances de l'ensemble *A*, l'*AG-NCPX*^{MO} obtient de meilleurs résultats que l'*AG-IBX*^{MO} pour 6 des 7 instances tandis que l'*AG-IBX*^{MO} obtient un meilleur résultat sur l'autre instance. Ces résultats se confirment également pour les instances de l'ensemble *B* où, cette fois-ci, l'*AG-NCPX*^{MO} obtient toujours de meilleurs résultats que l'*AG-IBX*^{MO}. L'*AG-IBX*^{MO} obtient un classement qui s'étend entre la 12^{ième} et la 20^{ième} position tandis que l'*AG-NCPX*^{MO} se

classe entre la 1ère et la 19ième position selon les instances. Ces algorithmes, malgré le fait qu'ils n'utilisent aucun processus de recherche locale, rivalisent donc assez bien avec les finalistes du challenge. Par contre, comme pour les instances avec des contraintes « faciles » pour les options prioritaires, on note que les résultats des deux algorithmes génétiques proposés ont de la difficulté à rivaliser avec les résultats de la meilleure équipe du challenge.

Si on s'intéresse maintenant à comparer les performances de l'*AG-IBX*^{MO} et de l'*AG-NCPX*^{MO} à celle du *GLS*, on note cette fois-ci, une nette domination du *GLS* par rapport à l'*AG-IBX*^{MO}, aussi bien sur les instances de l'ensemble *A* que celles de l'ensemble *B*. Ainsi, le *GLS* obtient de meilleurs résultats que l'*AG-IBX*^{MO} sur 6 des 7 instances de l'ensemble *A* et sur 11 des 12 instances de l'ensemble *B*. Cette situation s'explique probablement par la difficulté des instances qui, combinée à la limite de temps, fait encore plus ressortir les lacunes en terme d'intensification de l'opérateur de croisement. Toutefois, quand on compare les résultats du *GLS* à ceux de l'*AG-NCPX*^{MO}, on observe sensiblement les mêmes résultats que ceux obtenus au Tableau 6.1 sur les instances de l'ensemble *A*. Ainsi, l'*AG-NCPX*^{MO} obtient de meilleurs résultats que le *GLS* pour 6 des 7 instances de l'ensemble *A*. Par contre, pour les instances de l'ensemble *B*, les résultats sont un peu plus partagés. Ainsi, l'*AG-NCPX*^{MO} est supérieur au *GLS* pour 4 instances, inférieur pour 5 instances et obtient des résultats identiques sur les 3 autres instances.

Ces résultats confirment les observations faites précédemment et mettent une fois de plus en évidence la nécessité d'incorporer des mécanismes d'intensification plus explicites à nos AG. En analysant les résultats obtenus par l'ajout d'une procédure de recherche

locale à l'*AG-NCPX*^{MO} (dernière colonne du Tableau 6.2), on note ainsi une amélioration significative des résultats par rapport à l'algorithme de base pour toutes les instances. En fait, l'*AG-NCPX*^{MO}*+LS* parvient à être compétitif avec les résultats de la meilleure équipe du challenge en obtenant des résultats identiques ou supérieurs pour 9 des 19 instances des deux ensembles *A* et *B* et en se rapprochant significativement pour les autres instances. L'*AG-NCPX*^{MO}*+LS* obtient toujours un classement entre le 1er et le 6^{ième} rang à l'exception de l'instance 024_38_5 selon l'ordre des objectifs HPO_COLOR_LPO où il est au 12^{ième} rang. Comparativement au *GLS*, l'*AG-NCPX*^{MO}*+LS* obtient toujours de meilleurs résultats à l'exception de deux instances pour lesquelles les deux algorithmes ont une performance identique.

	Équipe gagnante	*GLS (Rank)*	*AG-IBX*^{MO} *(Rank)*	*AG-NCPX*^{MO} *(Rank)*	*AG-NCPX*^{MO} *+LS (Rank)*
Ensemble *A*					
HPO_COLOR_LPO					
024_38_3	**4249083 (1)**	4327229 (12)	4471615 (15)	4304266 (9)	4256186 (3)
024_38_5	**4280079 (1)**	7347154 (17)	26015122 (18)	6501289 (15)	4392151 (12)
039_38_4_ch1	**13129000 (1)**	15179000 (11)	17201000 (14)	14122000 (8)	**13129000 (1)**
048_39_1	175615 (4)	202740 (13)	3286796 (18)	191750 (11)	174690 (2)
HPO_LPO_COLOR					
024_38_3	**4000306 (1)**	4041506 (8)	5035482 (13)	6015504 (14)	4033403 (6)
024_38_5	**4034309 (1)**	6080457 (18)	58072610 (17)	5068407 (15)	4045349 (6)
048_39_1	**61290 (1)**	83403 (11)	246439 (17)	81406 (10)	63323 (4)
Ensemble *B*					
HPO_COLOR_LPO					
023_S23_J3	**48310008 (1)**	48349006 (10)	48465018 (18)	48429000 (13)	48313000 (4)
024_V2_S22_J1	**1074299068 (1)**	1100352464 (8)	1124857475 (16)	1106420563 (10)	1078310188 (4)
029_HPO_ S21_J6	**35167170 (1)**	35192150 (14)	35187151 (12)	35173150 (6)	35168171 (3)
035_ch1_ S22_J3	67036064 (7)	67037063 (12)	67044083 (20)	67037063 (12)	**67036061 (1)**
035_ch2_ S22_J3	**385187351 (1)**	**385187351 (1)**	385187353 (17)	**385187351 (1)**	**385187351 (1)**
048_ch2_ S22_J3	**3094029 (1)**	3126017 (15)	3131944 (17)	3124086 (12)	3094030 (2)
HPO_LPO_COLOR					
023_S23_J3	48000317 (3)	48000406 (10)	65000453 (19)	48000496 (15)	**48000316 (1)**
024_V2_S22_J1	**1074850430 (1)**	1097921524 (9)	1179022413 (19)	1113997557 (13)	1075884555 (4)
029_HPO _S21_J6	**37150167 (1)**	37150194 (12)	37150402 (18)	37150182 (9)	**37150167 (1)**
035_ch1 _S22_J3	**67052049 (1)**	67052052 (9)	67059057 (19)	67052052 (9)	**67052049 (1)**
035_ch2 _S22_J3	**385341205 (1)**	**385341205 (1)**	388350188 (20)	385353192 (19)	**385341205 (1)**
048_ch2_S22_J3	**3000337 (1)**	3000375 (14)	3000405 (17)	3000356 (7)	**3000337 (1)**

Tableau 6.2 : Résultats comparatifs de l'*Équipe gagnante*, du *GLS*, de l'*AG-IBX*^{MO}, de l'*AG-NCPX*^{MO} et de l'*AG-NCPX*^{MO}*+LS* pour les instances « difficiles » en options prioritaires avec l'objectif HPO comme objectif prioritaire

Le Tableau 6.3 présente les résultats pour les instances de l'ensemble *A* et *B* avec l'ordre de priorité des objectifs COLOR-HPO-LPO. En comparant l'*AG-IBX*^{MO} et l'*AG-NCPX*^{MO}, on remarque, une fois de plus, que l'*AG-NCPX*^{MO} obtient de meilleurs résultats que l'*AG-IBX*^{MO}. En effet, pour les 19 instances, l'*AG-NCPX*^{MO} obtient de meilleurs résultats sur 18 d'entre elles tout en obtenant un résultat identique sur la dernière instance. Cependant, contrairement aux observations faites précédemment, on note que l'écart entre les deux algorithmes semble moins important pour cette catégorie d'instance. En fait, à l'exception de trois instances, les deux algorithmes obtiennent toujours exactement la même valeur en ce qui concerne l'objectif principal. La différence est donc observée, pour ces instances, sur le second et le troisième objectif. On note toutefois que les résultats des deux algorithmes ne sont pas au niveau des résultats de l'équipe gagnante. À l'exception de l'instance 35_ch2_S22_J4 selon l'ordre des objectifs COLOR_HPO_LPO pour laquelle tous les algorithmes obtiennent le même résultat, l'*AG-IBX*^{MO} obtient un classement entre le 12^{ième} et le 20^{ième} rang tandis que l'*AG-NCPX*^{MO} se classe entre le 1er et le 17^{ième} rang. On remarque toutefois, à l'exception d'une instance pour l'*AG-NCPX*^{MO} et de trois instances pour l'*AG-IBX*^{MO}, que les deux algorithmes obtiennent la même valeur, en ce qui concerne l'objectif principal, que celle obtenue par la meilleure équipe du challenge. On peut tirer cette conclusion en considérant que la pondération accordée à l'objectif principal est de 1000000 et que les écarts sont inférieurs à cette valeur.

En comparant les résultats de nos algorithmes à ceux du *GLS*, on note, une fois de plus, que le *GLS* obtient de meilleurs résultats que l'*AG-IBX*^{MO} pour 2 des 4 instances de l'ensemble *A*, est inférieur pour une instance tout en obtenant un résultat identique pour la

dernière instance. Par contre, pour les instances de l'ensemble *B*, le *GLS* domine encore plus nettement l'*AG-IBX*^{MO} en obtenant de meilleurs résultats pour 11 instances, des résultats inférieurs pour 3 instances et un résultat identique pour l'autre instance. En comparant les résultats de l'*AG-NCPX*^{MO} à ceux du *GLS*, on note que l'*AG-NCPX*^{MO} obtient de meilleurs résultats pour toutes les instances de l'ensemble *A* à l'exception d'une instance où les deux algorithmes obtiennent un résultat identique. Pour les instances de l'ensemble *B*, on constate que l'*AG-NCPX*^{MO} obtient de meilleurs résultats que le *GLS* pour 5 instances, est inférieur pour 6 instances tout en étant identique pour les 4 autres instances. Une fois de plus, on observe une performance très proche entre les deux algorithmes.

	Équipe gagnante	*GLS (Rank)*	*AG-IBX^{MO} (Rank)*	*AG-NCPX^{MO} (Rank)*	*AG-NCPX^{MO}+LS (Rank)*
Ensemble *A*					
COLOR_HPO_LPO					
022_3_4	**11039001 (1)**	11041001 (15)	11039131 (12)	11039098 (11)	**11039001 (1)**
039_38_4 _ch1	68161000 (3)	68265000 (15)	68265000 (15)	68249000 (12)	**68155000 (1)**
064_38_2 _ch1	**63423782 (1)**	63435799 (15)	63443831 (17)	**63423782 (1)**	**63423782 (1)**
064_38_2_ch2	**27367052 (1)**	**27367052 (1)**	27367067 (15)	**27367052 (1)**	**27367052 (1)**
Ensemble *B*					
COLOR_HPO_LPO					
022_S22_J1	**13022148 (1)**	13022154 (11)	13022189 (19)	13022178 (17)	**13022148 (1)**
023_S23_J3	**51327031 (1)**	54349063 (21)	51735264 (20)	51393130 (17)	51343070 (9)
024_V2_S22_J1	**134023158 (1)**	135226676 (20)	134902740 (19)	134230457 (14)	134057341 (4)
025_S22_J3	**126127589 (1)**	133129840 (21)	126300350 (18)	126136839 (12)	**126127589 (1)**
028_ch1_S22_J2	38098201 (4)	38098251 (9)	38099330 (16)	38098334 (12)	**38098188 (1)**
028_ch2_S23_J3	**4000071 (1)**	**4000071 (1)**	5000078 (18)	**4000071 (1)**	**4000071 (1)**
029_ S21_J6	**52711171 (1)**	52755179 (14)	52905570 (20)	52763341 (15)	52717428 (8)
035_ch1_S22_J3	**6156090 (1)**	6156092 (10)	6156109 (18)	6156092 (10)	**6156090 (1)**
035_ch2_S22_J3	**7651671 (1)**	**7651671 (1)**	**7651671 (1)**	**7651671 (1)**	**7651671 (1)**
039_ch1_S22_J4	**55045096 (1)**	55045235 (9)	55046737 (18)	55045235 (9)	**55045096 (1)**
039_ch3_ S22_J4	**59214671 (1)**	59214698 (12)	59214783 (15)	59214681 (9)	**59214671 (1)**
048_ch1_ S22_J3	**64115670 (1)**	64135847 (14)	64153806 (15)	64124687 (12)	**64115670(1)**
048_ch2_ S22_J3	**58283180 (1)**	58288194 (12)	58312194 (19)	58290183 (13)	**58283180 (1)**
064_ch1_ S22_J3	**62095288 (1)**	62108458 (10)	63116379 (19)	62113381 (12)	62097307 (3)
064_ch2_ S22_J4	**31052178 (1)**	31052184 (9)	32052158 (16)	31053188 (13)	**31052178 (1)**

Tableau 6.3 : Résultats comparatifs de l'*Équipe gagnante*, *GLS*, de l'*AG-IBX*^{MO}, de l'*AG-NCPX*^{MO} et de l'*AG-NCPX*^{MO}*+LS* pour les instances avec l'objectif COLOR comme objectif prioritaire

En observant maintenant les résultats des deux algorithmes par rapport aux résultats de la meilleure équipe du challenge, on note que l'*AG-NCPX*^{MO} obtient toujours la même valeur que celle obtenue par l'équipe gagnante sur l'objectif principal. Pour sa part, le *GLS* n'y arrive qu'à 3 reprises. Le *GLS* obtient même la pire solution pour les instances 023_S23_J3 et 025_S22_J3 selon l'ordre des objectifs COLOR_HPO_LPO.

En analysant les résultats de l'*AG-NCPX*^{MO}*+LS*, on note une nette amélioration des performances de l'algorithme. Ainsi, pour les instances de l'ensemble *A*, l'*AG-NCPX*^{MO}*+LS* obtient toujours des résultats au moins identiques ou meilleurs à ceux de l'équipe gagnante du challenge. Pour les instances de l'ensemble *B*, l'*AG-NCPX*^{MO}*+LS* obtient des résultats identiques ou supérieurs à ceux de l'équipe gagnante du challenge pour 11 instances. L'*AG-NCPX*^{MO}*+LS* obtient toujours un classement entre le 1er et le 4^{ième} rang à l'exception des instances 023_S23_J3 et 029_S21_J6 selon l'ordre des objectifs COLOR_HPO_LPO où il est respectivement au 9^{ième} rang et au 8^{ième} rang. Par rapport au *GLS*, l'*AG-NCPX*^{MO}*+LS* obtient de meilleurs résultats pour 16 des 19 instances tout en obtenant des résultats identiques pour les trois autres instances.

Finalement, le Tableau 6.5 présente les résultats des différents algorithmes pour les 19 instances de l'ensemble *X* qui ont été utilisées lors de la phase finale du Challenge ROADEF 2005 pour établir le classement final des équipes. Contrairement aux résultats précédents, tous les algorithmes ont été exécutés à cinq reprises comme cela a été fait pour les différentes équipes participant à cette phase de la compétition. Les résultats présentés dans ce tableau correspondent donc aux résultats moyens obtenus pour 5 exécutions.

Lorsque l'on compare les résultats moyens de l'*AG-IBX*^{MO} et de l'*AG-NCPX*^{MO}, on note également que, sur cet ensemble, l'*AG-NCPX*^{MO} domine nettement l'*AG-IBX*^{MO} en obtenant toujours de meilleurs résultats à l'exception de 4 instances pour lesquelles les deux algorithmes obtiennent exactement les mêmes résultats moyens. On note aussi que, pour ces 4 instances, les deux algorithmes obtiennent la même solution à chacune des cinq exécutions. De plus, les résultats obtenus par les deux algorithmes pour ces 4 instances sont identiques à ceux de la meilleure équipe du challenge. En examinant plus en détail les caractéristiques de ces 4 instances, on constate qu'il s'agit d'instances de petite taille dont le nombre de voitures à ordonnancer varie entre 65 et 376. Ceci explique sans doute le fait que les deux algorithmes génétiques proposés les résolvent aisément. La taille de ces 4 instances explique aussi qu'il n'y ait pas d'écart entre les résultats des deux algorithmes. Comme les résultats obtenus précédemment l'ont montré, l'écart entre les deux algorithmes semble être en relation avec la taille des instances. En effet, l'*AG-IBX*^{MO} a plus de difficulté à converger vers une bonne solution pour des instances de très grande taille. Cette situation se confirme, encore une fois, à l'aide de l'instance 024_S49_J2 selon l'ordre des objectifs HPO_COLOR_LPO contenant cette fois-ci 1319 voitures à ordonnancer. Pour cette instance, l'écart entre les résultats moyens des deux algorithmes sur l'objectif principal est de plus de 26 conflits. À l'exception de 4 instances où il semble facile de trouver la solution, l'*AG-IBX*^{MO} obtient un classement entre le 9^{ième} et le 20^{ième} rang tandis que l'*AG-NCPX*^{MO} se classe entre le 7^{ième} et le 15^{ième} rang.

Si on compare maintenant les résultats de nos deux algorithmes à ceux du *GLS*, on observe des résultats similaires à ceux obtenus pour les ensembles *A* et *B*. En effet, l'*AG-*

IBX^{MO} est moins bon pour 13 instances, est meilleur pour 3 instances et obtient des résultats identiques pour les 3 autres instances. On note aussi que, parmi les 3 instances pour lesquelles l'*AG-IBX*^{MO} obtient une meilleure moyenne que le *GLS*, il y en a une (035_CH1_S50_J4 selon l'ordre des objectifs COLOR_HPO_LPO) pour laquelle le *GLS* n'a pas pu fournir de solution valide au cours de cette phase du challenge. Lorsque l'on compare maintenant le *GLS* à l'*AG-NCPX*^{MO}, on observe cette fois-ci que l'*AG-NCPX*^{MO} fournit de meilleurs résultats moyens que le *GLS* pour 8 instances, est inférieur pour 7 instances tout en produisant des résultats identiques pour les 4 autres instances.

On note aussi que les résultats de l'*AG-IBX*^{MO} et de l'*AG-NCPX*^{MO} ne sont nullement au niveau des résultats moyens enregistrés par l'équipe gagnante. Toutefois, en ajoutant une procédure de recherche locale à l'*AG-NCPX*^{MO}, on parvient à combler en grande partie cet écart en obtenant à 10 reprises les meilleurs résultats moyens tout en étant très près pour les autres instances. L'*AG-NCPX*^{MO}*+LS* obtient un classement entre le 1er et 5^{ième} rang pour l'ensemble des instances de l'ensemble *X*.

	Équipe gagnante	*GLS (Rank)*	*AG-IBX^{MO} (Rank)*	*AG-NCPX^{MO} (Rank)*	*AG-NCPX^{MO} +LS (Rank)*
Ensemble *X*					
HPO_COLOR_LPO					
023_S49_J2	**192466 (1)**	246268.20 (17)	246268.40 (18)	211879 (12)	193077 (3)
024_S49_J2	**337006 (1)**	421425 (8)	27046420.20 (18)	506015 (11)	346202.20 (2)
029_S49_J5	**110442.60 (2)**	120855 (11)	150969.20 (17)	123029.20 (12)	111093.20 (3)
034_VP_S51_J1_J2_J3	**56386.80 (1)**	76217.60 (17)	74354.20 (15)	66750 (12)	57577.40 (5)
034_VU_S51_J1_J2_J3	8087037 (4)	8091450.20 (10)	8112049 (16)	8103064 (15)	**8087035.80 (1)**
039_CH1_S49_J1	**69239 (1)**	69455.60 (6)	69705 (9)	69479.60 (7)	69355.20 (2)
039_CH3_S49_J1	**231030.20 (2)**	239593.20 (16)	250670 (17)	235475.40 (13)	231030.40 (3)
048_CH1_S50_J4	**197044.80 (3)**	206509.60 (16)	207634 (17)	204182 (14)	197045.40 (4)
048_CH2_S49_J5	**31077916.20 (1)**	31104598.80 (12)	31128931 (18)	31106266.2 (13)	31078317.20 (2)
064_CH1_S49_J1	**61187229.80 (1)**	61229518.80 (12)	61309246.20 (20)	61223429 (10)	61190429 (2)
064_CH2_S49_J4	**37000 (1)**	40400 (14)	42000 (15)	39000 (12)	**37000 (1)**
655_CH1_S51_J2_J3_J4	**30000 (1)**	**30000 (1)**	**30000 (1)**	**30000 (1)**	**30000 (1)**
655_CH2_S52_J1_J2_S01_J1	**153034000 (1)**	153035200 (8)	153047000 (12)	153041000 (11)	**153034000 (1)**
COLOR_HPO_LPO					
022_S49_J2	**12002003 (1)**	**12002003 (1)**	12002008 (16)	**12002003 (1)**	**12002003 (1)**
035_CH1_S50_J4	**5010000 (1)**	-	**5010000 (1)**	**5010000 (1)**	**5010000 (1)**
035_CH2_S50_J4	**6056000 (1)**	**6056000 (1)**	**6056000 (1)**	**6056000 (1)**	**6056000 (1)**
HPO_LPO_COLOR					
025_S49_J1	160407.60 (2)	189390.20 (15)	188118.20 (13)	176454.60 (10)	**160407.20 (1)**
028_CH1_S50_J4	36370094 (4)	36377907.20 (5)	49863125.80 (20)	39634315.20 (12)	**36360092.40 (2)**
028_CH2_S51_J1	**3 (1)**	**3 (1)**	**3 (1)**	**3 (1)**	**3 (1)**

Tableau 6.5 : Résultats comparatifs de l'*Équipe gagnante*, du *GLS*, de l'*AG-IBX^{MO}*, de l'*AG-NCPX^{MO}* et de l'*AG-NCPX^{MO}+LS* pour les instances de l'ensemble X

Pour comparer la performance des algorithmes génétiques proposés aux résultats des équipes du challenge, nous avons utilisé la procédure du challenge qui consiste à calculer un *score* pour chacune des instances de l'ensemble *X* selon l'Équation 6.4. Le *score* de chaque algorithme est calculé en fonction de la meilleure et de la pire solution obtenue par les 18 équipes finalistes du challenge et les 3 algorithmes proposés.

$$score(A\lg o) = \frac{résultat_{A\lg o} - Best\_result}{Best\_result - worst\_result} \qquad (6.4)$$

Dans l'Équation 6.4, *Best_result* et *Worst_result* indiquent respectivement le meilleur et le pire résultat obtenu pour une instance donnée tandis que $résultat_{Algo}$ correspond au résultat obtenu par l'algorithme dont on veut établir le score sur cette même instance. Ainsi, on retrouve, à chaque ligne du Tableau 6.6, le score obtenu par nos différents algorithmes pour chaque instance de l'ensemble *X*. La dernière ligne du tableau indique le score total

des approches proposées pour cet ensemble. On remarque, en analysant ces résultats, qu'ils confirment la hiérarchie établie à l'aide des tableaux précédents, à savoir la supériorité de l'*AG-NCPX*^{MO} par rapport à l'*AG-IBX*^{MO} et la supériorité de l'*AG-NCPX*^{MO}*+LS* par rapport aux deux autres algorithmes. Notons aussi que, selon le classement final du challenge publié par les organisateurs et disponible sur le site internet du challenge, le *GLS* s'est classé au 13^{ième} rang et obtient un score total de 16.8937 tandis que la meilleure équipe du challenge obtient un score total de 18.9935. À partir de ces résultats, on peut convenir que la différence entre les résultats de la meilleure équipe du challenge et ceux de notre algorithme génétique avec recherche locale est très faible (0.0345 point). On note aussi que le score de l'*AG-NCPX*^{MO} avec et sans recherche locale est supérieur à celui du *GLS*. On peut donc conclure que les approches proposées dans ce chapitre permettent d'obtenir des résultats compétitifs pour le problème multi-objectifs d'ordonnancement de voitures. On démontre ainsi que les algorithmes génétiques sont tout à fait appropriés pour traiter ce type de problème lorsque les opérateurs génétiques sont définis en tenant compte des caractéristiques du problème.

Ensemble *X*	Score *Équipe gagnante*	Score *AG-IBX^{MO}*	Score *AG-NCX^{MO}*	Score *AG-NCX^{MO}+LS*
HPO_COLOR_LPO				
023_S49_J2	1	0.5575	0.8403	0.9950
024_S49_J2	1	0.4605	0.9966	0.9998
029_S49_J5	0.9980	0.4249	0.8200	0.9888
034_VP_S51_J1_J2_J3	0.9956	0.7949	0.8799	0.9823
034_VU_S51_J1_J2_J3	1	0.9998	0.9999	1
039_CH1_S49_J1	1	0.9755	0.9873	0.9939
039_CH3_S49_J1	0.9999	0.6368	0.9178	1
048_CH1_S50_J4	0.9999	0.9952	0.9968	1
048_CH2_S49_J5	1	0.9868	0.9927	0.9999
064_CH1_S49_J1	1	0.9799	0.9940	0.9995
064_CH2_S49_J4	1	0.8588	0.9435	1
655_CH1_S51_J2_J3_J4	1	1	1	1
655_CH2_S52_J1_J2_S01_J1	1	0.9999	0.9999	1
COLOR_ HPO_LPO				
022_S49_J2	1	0.9999	1	1
035_CH1_S50_J4	1	1	1	1
035_CH2_S50_J4	1	1	1	1
HPO_LPO_COLOR				
025_S49_J1	1	0.9983	0.9990	1
028_CH1_S50_J4	0.9999	0.9553	0.9891	0.9999
028_CH2_S51_J1	1	1	1	1
Score total	**18.9935**	**16.6241**	**18.3569**	**18.9590**

Tableau 6.6 : Score obtenu par l'Équipe gagnante, l'AG-IBX^{MO}, l'AG-NCPX^{MO} et l'AG-NCPX^{MO}+LS pour les instances de l'ensemble X.

6.5 Conclusion

Dans ce chapitre, nous avons proposé un AG utilisant deux opérateurs de croisement spécifiquement dédiés à la nature multi-objectifs du problème industriel d'ordonnancement de voitures proposé par Renault lors du Challenge ROADEF 2005. Si les algorithmes évolutionnaires sont maintenant considérés comme des techniques bien adaptées à la résolution de problèmes multi-objectifs [Barichard 2003; Basseur 2004; Zinflou *et al.* 2006], peu de chercheurs et d'industriels ont cru en cette méthode pour résoudre efficacement cette problématique industrielle. On note ainsi que, parmi les 18 équipes qualifiées pour la seconde phase du challenge, une seule équipe a proposé une approche

basée sur un AG. Cette situation s'explique sans doute par la difficulté de définir des opérateurs de croisement qui tiennent compte des spécificités du problème. L'approche proposée dans ce chapitre repose essentiellement sur l'adaptation d'opérateurs de croisement spécialisés pour la résolution du problème d'ordonnancement de voitures uni-objectif aux spécificités de la version industrielle du problème qui, elle, comporte trois objectifs conflictuels à optimiser. Les expérimentations numériques effectuées ont permis de démontrer l'efficacité de l'approche proposée pour ce type de problème. Une conclusion naturelle à ces résultats expérimentaux est que les AG demeurent des alternatives robustes et efficaces pour résoudre le problème d'ordonnancement de voitures multi-objectifs. Ces résultats mettent une fois de plus en évidence la nécessité d'incorporer des connaissances spécifiques aux problèmes à résoudre lors de la conception d'un algorithme génétique, et ce, même si l'utilisation d'opérateurs classiques est possible. Nous sommes également conscients que le fait de connaître les solutions trouvées par les finalistes du challenge a facilité le travail de calibration de nos algorithmes. Toutefois, l'objectif de ce chapitre visait à démontrer que les algorithmes génétiques peuvent être performants pour résoudre ce type de problème industriel.

Le traitement lexicographique des objectifs tel que proposé par Renault fait en sorte que plusieurs solutions « intéressantes » pour l'entreprise sont ignorées. En effet, le fait de relâcher l'importance accordée à l'objectif principal peut mettre en évidence différentes solutions alternatives moins coûteuses pour l'entreprise. Nous pensons donc que le problème posé par Renault aurait avantage à être traité pour l'obtention de solutions dites de compromis. Dans ce contexte, les AG proposés représentent des alternatives

intéressantes pour la recherche de compromis comme le démontre la revue de la littérature effectuée au Chapitre 2. En effet, la structure des AG se prête facilement à l'optimisation multi-objectifs au sens de Pareto et ces approches ont démontré leur capacité à générer des solutions de compromis en une seule étape d'optimisation.