UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN INGÉNIERIE

PAR
Arnaud Zinflou

ALGORITHMES ÉVOLUTIONNAIRES POUR L'ORDONNANCEMENT INDUSTRIEL : APPLICATION À L'INDUSTRIE AUTOMOBILE

DÉCEMBRE 2008

Bibliothèque
Paul-Émile-Boulet

UQAC
Université du Québec
à Chicoutimi

Mise en garde/Advice

Afin de rendre accessible au plus grand nombre le résultat des travaux de recherche menés par ses étudiants gradués et dans l'esprit des règles qui régissent le dépôt et la diffusion des mémoires et thèses produits dans cette Institution, **l'Université du Québec à Chicoutimi (UQAC)** est fière de rendre accessible une version complète et gratuite de cette œuvre.

Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptation and diffusion of dissertations and theses in this Institution, the **Université du Québec à Chicoutimi (UQAC)** is proud to make a complete version of this work available at no cost to the reader.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

RÉSUMÉ

Dans plusieurs secteurs d'activité comme l'aéronautique, l'informatique, les télécommunications, l'environnement, le transport et autres, les décideurs sont confrontés à des problèmes de complexité grandissante. Il peut s'agir d'optimiser le trajet d'un véhicule, de minimiser des coûts de production, de supporter la prise de décision, d'améliorer les performances d'un circuit électronique ou encore d'ordonnancer les processus dans un système informatique. Dans de nombreux cas, le problème à résoudre peut s'exprimer comme un problème d'optimisation combinatoire qui est rarement uni-objectif. En effet, la plupart des problèmes d'optimisation combinatoire rencontrés dans la pratique nécessitent l'optimisation simultanée de plusieurs objectifs souvent contradictoires. C'est par exemple le cas pour le problème d'ordonnancement de voitures. Toutefois, malgré l'intérêt indéniable d'aborder les problèmes industriels d'un point de vue multi-objectifs, plusieurs auteurs ont noté, en recensant la littérature, que les chercheurs s'attardaient principalement à des contextes théoriques de base. En contexte multi-objectifs, l'utilisation de métaheuristiques s'avère souvent d'une grande utilité.

Parmi les métaheuristiques, les algorithmes évolutionnaires, et plus particulièrement les algorithmes génétiques, se sont distingués comme étant des techniques bien adaptées à la résolution de problèmes multi-objectifs notamment à cause de leur faculté à exploiter de vastes espaces de recherche et à générer des compromis multiples en une seule étape d'optimisation. Les algorithmes génétiques tentent de simuler le processus de sélection naturelle dans un environnement hostile lié au problème à résoudre en s'inspirant des théories de l'évolution proposées par Darwin et des méthodes de combinaison de gènes introduites par Mendel.

Dans cette thèse, nous proposons des approches de résolution efficaces basées sur des algorithmes évolutionnaires permettant de supporter la prise de décision pour des problèmes d'ordonnancement industriel multi-objectifs comme le problème d'ordonnancement de voitures. En particulier, nous présentons dans un premier temps deux nouveaux opérateurs de croisement pour le problème théorique d'ordonnancement de voitures. Nous mettons ainsi en évidence l'importance d'utiliser des opérateurs génétiques dédiés à la problématique étudiée et ce, même lorsque l'emploi d'opérateurs naturels est possible. Une fois cette étape réalisée, nous proposons deux schémas de coopération entre algorithmes évolutionnaires et méthodes exactes pour résoudre le problème théorique d'ordonnancement de voitures. Par la suite, nous abordons la résolution du problème industriel d'ordonnancement de voitures en proposant un algorithme génétique efficace qui permet de résoudre le problème de manière lexicographique. Finalement, nous abordons la problématique industrielle d'ordonnancement de voitures d'un point de vue intégralement multi-objectifs en développant un algorithme Pareto générique qui hybride des concepts issus des algorithmes génétiques avec des concepts issus de la métaphore immune. C'est, à notre connaissance, le premier travail de recherche qui considère la problématique industrielle d'ordonnancement de voitures en traitant les objectifs simultanément sans leur

attribuer d'ordre ou de poids. En plus d'avoir produit d'excellents résultats sur le problème industriel d'ordonnancement de voitures, l'approche proposée s'est aussi montrée particulièrement efficace sur un benchmark classique en optimisation multi-objectifs (le problème de sac à dos multi-objectifs). Ces résultats mettent en évidence l'intérêt pratique de ce genre d'approches en contexte industriel.

ABSTRACT

Decision makers have to face increasingly complex problems in a wide array of activity sectors namely aeronautics, IT, telecommunications, environment and transportation among others. These problems could involve optimizing the route taken by a vehicle, minimizing production costs, supporting decision-making, improving the performance of an electronic circuit or scheduling processes in an operating system. In a wide number of cases, these problems can be expressed as single or multi-objective combinatorial optimization problems. In fact, most of the combinatorial optimization problems encountered in real life situations involve simultaneous optimization of multiple incommensurable and often competing objective functions. For instance, this applies to the industrial situation analyzed in this thesis, the car sequencing problem. However, despite the real interest to tackle industrial problems in a multi-objective context, a survey of past publications carried out by several authors has concluded that researchers have mainly limited their works to the treatment of basic problem situations. In multi-objective optimization, metaheuristics can be extremely useful tools for the engineer and the decision maker.

Among metaheuristics, evolutionary algorithms such as genetic algorithms seem to be particularly well suited for multi-objective optimization because of their ability to handle complex problems and to find multiple compromise solutions in one single simulation run. Genetic algorithms are stochastic algorithms based upon Darwin's theory of natural selection and the genetic inheritance laws of Mendel.

In this thesis, we suggest efficient approaches based on evolutionary algorithms which can be used to support the decision process for industrial multi-objective optimization problems such as the car sequencing problem. In particular, we first introduce two new crossover operators to solve the standard car sequencing problem. We thus highlight the importance of incorporating specific problem knowledge into genetic operators, even if classical genetic operators could be used. In the next step, we suggest two hybrid strategies combining evolutionary algorithms and exact methods for solving the standard car sequencing problem. Thereafter, we tackle the industrial car sequencing problem by suggesting an efficient genetic algorithm which solves lexicographically the problem. Finally, we tackle the industrial problem in order to obtain so-called "compromise solutions" by developing a new generic Pareto algorithm which combines concepts from genetic algorithms and from the immune metaphor. In our knowledge, this is the first study which considers the different objectives of the industrial car sequencing problem simultaneously, without assigning priority or weight to each of them. In addition to its good performance compared to popular and modern multi-objective evolutionary algorithms for the real-world industrial car sequencing problem, the suggested approach has also obtained competitive results on a classical multi-objective benchmark (the multi-objective knapsack problem). These results highlight the practicality of using these types of approaches in a real-world industrial context.

REMERCIEMENTS

Je tiens tout d'abord à remercier Mme Caroline Gagné, ma directrice de recherche pour son support et son implication dans ce travail de recherche ainsi que pour son encadrement sans faille. En particulier, je la remercie pour m'avoir laissé une grande liberté dans mes recherches, pour avoir été attentive à tous mes commentaires ou idées et pour avoir su répondre à mes nombreuses interrogations. Ses encouragements et ses conseils ont été une aide précieuse sans laquelle ce travail n'aurait sans doute pas vu le jour.

Je remercie également M. Marc Gravel, M. Wilson Price et M. Jacques Teghem pour avoir accepté de faire partie de mon jury d'évaluation.

Je remercie M. Djamal Rébaïne pour l'intérêt qu'il a porté à mon travail ainsi que pour ses commentaires et conseils qui m'ont permis d'en améliorer le fond et la forme.

Mes remerciements vont aussi à tous les membres du Groupe de Recherche en Informatique de l'UQAC (GRI) pour la bonne ambiance qu'ils ont su mettre dans les locaux du GRI. En particulier, je remercie Pierre Delisle avec qui j'ai partagé un bureau pendant une grande partie de ma thèse. Les différents échanges que nous avons eus ainsi que les différentes parties de badminton ont rendu le séjour au laboratoire plus agréable. Merci également à Aymen Sioud et Eunice Lemamou, ce fut un plaisir de vous côtoyer quotidiennement.

Un travail de recherche ne pourrait être réalisé sans support financier. À ce titre, je voudrais remercier le Conseil de Recherche en Sciences Naturelles et en Génie du Canada (CRSNG) pour son appui sous forme de bourse doctorale.

Je tiens aussi à remercier Malla et Michel pour avoir contribué à améliorer la qualité du français de ce document.

Ces remerciements ne seraient pas complets sans y associer ma famille. Je remercie sincèrement mes parents, mes deux frères, ma sœur et toute ma famille pour leur soutien et leur confiance. Sans vous, ce travail n'aurait jamais pu être réalisé. Je tiens, en particulier, à remercier ma mère Mme Adélaïde Zinflou qui malheureusement est décédée pendant que je réalisais ce travail.

Merci pour ton amour, ton soutien et ta clairvoyance. Merci pour tous les sacrifices que toi et papa avez consentis pour notre bien-être. Merci de continuer à m'inspirer, à guider mes pas et à veiller sur moi de là-haut. Tu me manques tellement et tu resteras toujours présente dans mon cœur et mon esprit. Ce travail t'est dédié.

Merci enfin à tous ceux qui de près ou de loin m'ont accompagné tout au long de cette aventure.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES PUBLICATIONS

Les travaux réalisés dans le cadre de ce travail de recherche ont mené aux publications suivantes :

Chapitre de livre

Design of an efficient genetic algorithm to solve the industrial car sequencing problem, in: Evolutionary Algorithms, I-Tech Education and Publishing, Vienna, Austria, Chapter 19, pages 377-400, November 2008, ISBN 978-953-7619-11-4.

Journaux et revues scientifiques

Genetic Algorithm with Hybrid Integer Linear Programming Crossover Operators for the Car-Sequencing Problem, Journal of the Operational Research Society (en révision)

Combining metaheuristics and ILP for solving the car-sequencing problem, Revue d'Intelligence Artificielle (RIA), Hermès, Vol. 22, No.2, 2008.

Actes de conférences avec comité de lecture

Tackling the industrial car sequencing problem using genetic algorithm, 2nd International Conference on Metaheuristics and Nature Inspired Computing, (META'08), Hammamet, Tunisie, 29-31th october 2008.

Designing hybrid integrative evolutionary approaches to the car sequencing problem, 11th International Workshop on Nature Inspired Distributed Computing (NIDISC'08), Miami, Floride, USA, 14-18 avril, 2008.

Algorithme génétique avec croisements hybrides utilisant la programmation linéaire en nombres entiers, Actes de la 7e Conférence Internationale de MOdélisation et SIMulation (MOSIM'08), S. Lamouri, A. Thomas, A. Artiba et F. Vernadat Editeurs, 2008, p. 967-975.

Genetic algorithm for the car sequencing problem, 7th Metaheuristics International Conference (MIC'2007), Montréal, Canada, june 25-29, 2007

Crossover operators for the car sequencing problem, 7th European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP'07), LNCS 4446, C. Cotta and J. Van Hemert (Eds.), Springer-Verlag Berlin Heidelberg, 2007, p. 229-239. (***Scholarship Award***)

Opérateurs de croisement génétique pour l'ordonnancement de voitures, First Workshop on Metaheuristics (META'06), Hammamet, Tunisie, 2-4 novembre 2006.

CHAPITRE 1

INTRODUCTION

Dans de nombreux secteurs industriels, les décideurs sont confrontés à des problèmes de complexité grandissante. Il peut s'agir d'optimiser le trajet des autobus dans une compagnie de transport publique, de minimiser des coûts de production, de supporter la prise de décision, d'améliorer les performances d'un circuit électronique ou encore d'ordonnancer les processus dans un système informatique. Dans de nombreux cas, le problème à résoudre peut s'exprimer comme un problème d'optimisation combinatoire.

Résoudre un problème d'optimisation consiste à déterminer la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise au moins une mesure de performance permettant d'effectuer une comparaison. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenue la meilleure évaluation par rapport à l'objectif défini. Lorsqu'un seul objectif est spécifié, par exemple un objectif de minimisation de la distance totale, la solution optimale est clairement définie : c'est celle qui a la plus petite distance. Cependant, dans de nombreuses situations, il y a souvent plusieurs objectifs contradictoires à satisfaire simultanément. En fait, les problèmes d'optimisation rencontrés en pratique sont rarement uni-objectif. C'est par exemple le cas pour le problème d'ordonnancement de voitures sur une chaîne de montage automobile (POV). Ce problème consiste à déterminer l'ordre dans lequel des automobiles doivent être produites, en considérant les options de chacune, en respectant les différentes contraintes de la chaîne de montage et en tenant compte des différents objectifs de l'environnement de production.

Pour les problèmes multi-objectifs, le concept de solution optimale devient alors plus difficile à définir. Dans ce cas, la solution recherchée n'est plus un point unique mais un ensemble de solutions dites de compromis encore appelé ensemble Pareto. Résoudre un problème comprenant plusieurs objectifs, appelé communément problème multi-objectifs, consiste donc à déterminer le meilleur ensemble de solutions de compromis.

Qu'ils comportent un seul ou plusieurs objectifs, les problèmes d'optimisation sont en général difficiles à résoudre. Nombre d'entre eux sont dits NP-Difficiles et ne peuvent être résolus de façon optimale par des algorithmes exacts en raison, entre autres, du temps de calcul qui est souvent prohibitif. La nécessité de trouver rapidement des solutions acceptables à plusieurs de ces problèmes a entraîné le développement d'algorithmes d'approximation dont font partie les métaheuristiques.

Parmi les métaheuristiques, les algorithmes évolutionnaires sont des approches particulièrement intéressantes pour résoudre des problèmes d'optimisation multi-objectifs. Cet intérêt s'explique notamment par leur faculté à exploiter de vastes espaces de recherche et à générer plusieurs solutions de compromis en une seule étape d'optimisation. Une revue de la littérature [Jaszkiewicz *et al.* 2004; Gagné *et al.* 2006; Terada *et al.* 2006; Estellon *et al.* 2007; Solnon *et al.* 2007] montre toutefois que peu d'efforts ont été faits jusqu'à présent pour la résolution du problème d'ordonnancement de voitures par des algorithmes évolutionnaires. L'utilisation de ce type d'algorithmes pour traiter le problème est donc une voie prometteuse encore peu explorée.

À partir de deux méthodes de résolution différentes, il est possible de concevoir des méthodes *hybrides* dans le but de fournir des résultats supérieurs à ceux des deux méthodes

qui les composent. Une grande proportion des approches proposées dans la littérature hybride deux métaheuristiques. Récemment, des approches hybridant des métaheuristiques et des méthodes exactes ont été proposées et représentent des alternatives intéressantes [Basseur 2004]. Même si les performances générales de tels algorithmes hybrides suscitent un intérêt croissant de la part des chercheurs, beaucoup de travail reste encore à faire dans ce domaine qui est relativement jeune et particulièrement dans la définition des schémas de coopération.

Plusieurs auteurs [Allahverdi *et al.* 1999; McKay et Wiers 1999; Portougal et Robb 2000] ont noté que la majorité des travaux en ordonnancement s'attardaient principalement à des contextes de base. Malgré la complexité de ces contextes, ce sont néanmoins des versions simplifiées des situations pratiques. Il en résulte ainsi un écart entre les approches de résolution proposées pour les problèmes d'ordonnancement théoriques et les situations pratiques [McKay et Wiers 1999; Portougal et Robb 2000; Gao 2004]. Les problèmes d'ordonnancement industriel constituent donc une problématique de recherche actuelle où de nombreuses avenues de recherche sont encore à explorer pour tenter de rapprocher les travaux de recherches aux besoins des praticiens.

1.2 Objectifs de la recherche

Cette thèse a pour objectif général d'enrichir les modèles existants et de proposer des approches de résolution efficaces basées sur des AG permettant de supporter la prise de décision pour des problèmes d'ordonnancement industriel multi-objectifs comme le

problème d'ordonnancement de voitures. Plus spécifiquement, ce travail devra rencontrer les objectifs spécifiques suivants :

- comprendre et isoler les mécanismes liés à la métaphore évolutionnaire, en particulier les AG, afin de proposer une stratégie de résolution efficace pour la résolution d'un problème uni-objectif;
- proposer des schémas de coopération originaux entre AG et différentes autres approches de résolution;
- proposer un AG efficace permettant de résoudre de manière lexicographique le POV industriel en adaptant la stratégie de résolution développée pour un contexte d'application uni-objectif; et
- développer un nouvel algorithme Pareto générique permettant de résoudre des problèmes d'optimisation combinatoire multi-objectifs. Cet algorithme devra permettre d'approximer l'ensemble PO en une seule exécution.

1.3 Approche méthodologique

L'objectif général de la thèse consiste à proposer des approches de résolution efficaces basées sur les AG pour résoudre une problématique d'ordonnancement industriel. Dans cette optique, le contexte applicatif utilisé est le POV tel que décrit dans le Challenge ROADEF 2005 [Nguyen et Cung 2005]. L'approche méthodologique préconisée pour atteindre cet objectif est divisée en trois étapes :

1) développer un AG en utilisant des représentations adaptées au problème à résoudre;

2) développer des schémas de coopération entre AE et différents algorithmes; et

3) Aborder la problématique industrielle d'un point de vue multi-objectifs.

1.3.1 Développement d'un AG uni-objectif

Les AE, notamment les AG, constituent une famille de techniques d'optimisation inspirée de la nature qui ont fait leur preuve pour résoudre de nombreux problèmes d'optimisation [Ben Hamida 2001; Berro 2001; Barichard 2003; Zinflou *et al.* 2006]. Cependant, une revue de la littérature montre que ce type d'algorithmes demeure très peu appliqué à la résolution de notre contexte d'application aussi bien dans sa version théorique qu'industrielle. On peut supposer que cette situation s'explique, en partie, par la difficulté qu'il y a à définir des opérateurs génétiques efficaces et adaptés au POV. La réalisation de cette étape mènera à la conception et au développement d'un algorithme génétique uni-objectif pour la résolution du POV théorique afin de démontrer son efficacité et de justifier le choix de cette méthode de résolution. Dans le but de mieux comprendre l'influence des opérateurs génétiques sur la performance de l'approche, des représentations et des opérateurs génétiques spécifiques au POV seront définis lors de la conception de l'algorithme. Les performances de l'algorithme proposé seront évaluées à partir de deux ensembles tests tirés de CSPLib [Gent et Walsh 1999] et un proposé par Gravel *et al.* [2005]. Les résultats de l'approche pourront ainsi être comparés aux meilleurs résultats obtenus dans la littérature à ce jour.

1.3.2 Schémas de coopérations entre AG et autres méthodes de résolution

Une manière de combler les lacunes d'une approche de résolution consiste à la combiner avec une autre méthode de résolution. De cette manière, il est possible d'obtenir des approches hybrides combinant les forces des méthodes qui les composent. Lors de

l'optimisation d'un problème, deux buts contradictoires sont généralement poursuivis : l'amélioration des solutions existantes (*intensification*) et l'exploration de nouvelles régions (*diversification*). Les AG effectuent naturellement des phases de diversification pour peu que le processus de sélection soit choisi de manière adéquate [Barichard 2003]. Toutefois, ils sont moins performants lorsqu'il s'agit d'exploiter les solutions trouvées [Basseur 2004].

Une grande proportion des approches hybrides proposées dans la littérature combine des métaheuristiques. Récemment, des approches combinant métaheuristiques et méthodes exactes ont été proposées et représentent des avenues de recherche prometteuses [Basseur 2004; Puchinger et Raidl 2005]. L'hybridation n'est cependant pas une tâche aisée [Jussien et Laburthe 2004], en particulier lorsqu'il s'agit de combiner des métaheuristiques avec des méthodes exactes. D'une part, l'utilisation d'outils d'optimisation commerciaux lors du développement de méthodes exactes pose de nombreux problèmes techniques lors de leurs combinaisons avec des métaheuristiques développées dans d'autres environnements de programmation. D'autre part, le concepteur de l'approche hybride peut avoir une expertise plus limitée dans la pratique d'une des méthodes qui composent l'algorithme hybride.

L'idée des schémas de coopération proposés à cette étape est de combiner un AG à des méthodes spécialisées dans l'intensification de la recherche. En se basant sur l'approche de résolution initiale à l'aide d'un AG développé à l'étape précédente, les approches coopératives exploitées dans ce travail hybrideront l'AG avec des méthodes exactes.

En effet, la complexité des problèmes d'optimisation combinatoire fait que leurs résolutions par des méthodes exactes est souvent une tâche ardue et est généralement inapplicable pour les problèmes les plus difficiles. Toutefois au fil des années, la puissance

de calcul disponible n'a cessé de croître ce qui a permis à plusieurs auteurs de proposer de nouvelles stratégies de plus en plus performantes. Dans le contexte d'application du POV, on peut citer les méthodes exactes de Drexl et Kimms [2001], Prandstetter et Raidl [2007] ou encore Gravel *et al.* [2005]. Comme méthode exacte, nous reprendrons le modèle de programmation en nombres entiers proposé par Gravel *et al.* [2005] comme modèle de base. Ce modèle a été choisi car il a fourni de bons résultats sur de nombreuses instances. Au niveau technologique, l'environnement de développement nécessaire à ce type d'hybridation est disponible notamment grâce à la présence d'un serveur pour les calculs et du logiciel d'optimisation CPLEX de la compagnie ILOG qui permet l'accès aux différentes librairies d'optimisation par des codes C^{++} ou Java. À ce niveau, les schémas d'hybridation proposés seront regroupés en deux catégories selon la classification de Puchinger et Raidl [2005] : des schémas d'*hybridation collaboratifs* et des schémas d'*hybridation intégratifs*.

Les schémas de coopération seront appliqués à la résolution du POV théorique. Ainsi, leurs performances seront évaluées les unes par rapport aux autres et par rapport à celles de l'algorithme génétique uni-objectif.

1.3.3 Traitement multi-objectifs

Le POV industriel est un problème où trois objectifs sont à minimiser. Ces objectifs correspondent au nombre de purges, au nombre de conflits sur les options prioritaires et au nombre de conflits sur les options non prioritaires. D'après la définition du problème dans le challenge ROADEF 2005 [Nguyen et Cung 2005], on constate toutefois que les trois objectifs sont traités selon un ordre de priorité sans compensation possible entre les

objectifs. Gagné *et al.*[2006] mentionnent dans leur article, l'intérêt qu'il y aurait à considérer simultanément les objectifs afin de générer des solutions de compromis potentiellement intéressantes dans la pratique. Cette étape a pour but, d'une part, de proposer un nouvel algorithme hybride pour l'optimisation multi-objectifs et, d'autre part, d'aborder la problématique industrielle d'ordonnancement de voitures en traitant les différents objectifs simultanément sans leur attribuer d'ordre ou de poids. À notre connaissance, c'est le premier travail de recherche à considérer le problème d'un point de vue intégralement multi-objectifs. À cette étape, nous développerons tout d'abord un algorithme génétique multi-objectif pour résoudre le POV industriel de manière lexicographique en adaptant les concepts proposés pour le POV théorique. Les performances de l'approche seront évaluées à partir des instances disponibles du Challenge ROADEF en relation avec les résultats publiés de la littérature [Jaszkiewicz *et al.* 2004; Benoit 2007; Briant *et al.* 2007; Estellon *et al.* 2007; Solnon *et al.* 2007]. Par la suite, nous proposerons une nouvelle stratégie hybride pour résoudre les problèmes multi-objectifs. Cette approche, appelée GISSMOO, est un algorithme Pareto qui combine des concepts issus des algorithmes génétiques avec des paradigmes issus des systèmes immunitaires artificiels. Les performances de GISSMOO seront évaluées, dans un premier temps, en les comparant avec celles d'autres algorithmes multi-objectifs de la littérature sur le problème multi-objectifs de sac à dos en 0/1 [Zitzler *et al.* 1999]. Dans un deuxième temps, nous appliquerons GISSMOO à la résolution du POV industriel en considérant les objectifs simultanément.

1.4 Organisation du document

L'objectif général de ce travail de recherche consiste à enrichir les modèles existants et à proposer des approches de résolution efficaces basées sur des AG permettant de supporter la prise de décision pour des problèmes d'ordonnancement industriel multi-objectifs. L'atteinte de cet objectif passe par une connaissance adéquate du contexte industriel étudié, des problèmes d'optimisation multi-objectifs et des algorithmes évolutionnaires. Le contenu de cette thèse est organisé en 8 chapitres.

Le Chapitre 2 décrit la problématique d'ordonnancement de voitures dans une chaîne de montage automobile. En particulier, nous présentons la version simplifiée du problème que l'on retrouve fréquemment dans la littérature ainsi que la version industrielle du problème qui, elle, comporte plusieurs objectifs. Pour ces deux problèmes, un état de l'art des différentes approches de résolution utilisées dans la littérature est aussi présenté.

Le Chapitre 3 porte sur les algorithmes évolutionnaires et l'optimisation multi-objectifs. Tout d'abord, les concepts de base de l'optimisation combinatoire multi-objectifs sont définis. Par la suite, les deux types de classification des méthodes de résolution de problèmes d'optimisation multi-objectifs sont présentés. Les algorithmes évolutionnaires sont ensuite décrits comme une technique récente d'optimisation qui possède des caractéristiques intéressantes pour résoudre des problèmes multi-objectifs. Outre les algorithmes évolutionnaires classiques, ce chapitre présente une autre classe d'algorithmes moins connus mais aussi inspirés de la biologie, les systèmes immunitaires artificiels. Finalement, un certain nombre d'algorithmes évolutionnaires multi-objectifs sont présentés en essayant d'apporter un regard critique sur chacun d'eux.

Au Chapitre 4, nous définissons deux nouveaux opérateurs de croisement spécialisés pour le problème théorique d'ordonnancement de voitures. Ces deux nouveaux opérateurs génétiques exploitent des informations liées aux caractéristiques du problème pour réaliser le croisement. Nous comparons les performances des différentes approches proposées avec celles des meilleurs algorithmes de la littérature pour ce problème.

Le Chapitre 5 est consacré au développement de mécanismes d'hybridation entre algorithmes génétiques et méthodes exactes. En particulier, nous proposons deux schémas d'hybridation qui intègrent un programme linéaire en nombres entiers à un algorithme génétique lors du processus de croisement.

Au Chapitre 6, nous proposons un nouvel algorithme génétique permettant de résoudre de manière lexicographique le problème industriel d'ordonnancement de voitures. En particulier, nous adaptons les concepts introduits au chapitre 4 afin de tenir compte de l'aspect multi-objectifs du problème.

Le Chapitre 7 présente GISSMOO, un nouvel algorithme Pareto générique permettant de traiter les problèmes d'optimisation multi-objectifs. GISSMOO est un algorithme hybride qui combine la métaphore immune avec un algorithme génétique afin d'identifier et mettre l'emphase sur les régions de l'espace de solution peu exploitées au cours des itérations de l'algorithme. Afin d'évaluer les performances de notre approche, nous comparons, dans un premier temps, ses résultats à ceux de trois algorithmes représentatifs de l'état de l'art en optimisation multi-objectifs évolutionnaire en utilisant un « benchmark » classique : le problème de sac à dos multi-objectifs.

Dans un deuxième temps, le Chapitre 8 présente l'adaptation de GISSMOO à la résolution du problème industriel d'ordonnancement de voitures. C'est, à notre connaissance, la première fois que le problème est abordé en considérant les différents objectifs simultanément.

Enfin, nous terminons ce travail de recherche par une conclusion générale qui récapitule les différentes contributions apportées dans cette thèse et donne quelques perspectives de recherche.

CHAPITRE 2

PROBLÉMATIQUE D'ORDONNANCEMENT DE VOITURES

2.1 Introduction

L'industrie automobile a aujourd'hui plus de cent ans. D'une production artisanale, elle est rapidement passée à une production de masse dès les années 20, lorsque la demande est devenue forte. À l'époque, les consommateurs étaient largement limités dans le choix de leurs voitures par l'offre disponible : comme l'illustre la célèbre déclaration d'Henri Ford : « *le client peut tout choisir, même la couleur, à condition qu'elle soit noire.* ». Depuis cette époque, l'environnement économique a beaucoup évolué. Pour survivre à la compétitivité mondiale, les entreprises doivent sans cesse innover et répondre le plus rapidement possible et sans erreurs aux exigences et attentes de ses clients en termes de qualité, de coût et de délais de mise à disposition. Ainsi, l'industrie est passée de la production en *flux poussés*, à la production en *flux tirés* : c'est la commande qui déclenche la production de la voiture, et non plus la production de la voiture qui déclenche la recherche d'un acheteur. Le but est de pouvoir fournir au client la voiture prévue, le jour prévu, au coût minimum et sans négliger la qualité. Les efforts sont faits à tous les niveaux de la fabrication : aussi bien pour la gestion des commandes que pour la production de la voiture ou la livraison de celle-ci. Les usines d'assemblage n'échappent pas à cet effort. En effet, bien que les modèles de voitures fabriqués par les constructeurs soient toujours limités, la présence d'options choisies par le client augmente considérablement le nombre de types de voitures produits. Cependant, étant donné le coût des installations, ces différents modèles sont tous assemblés sur la même chaîne de montage. De nouveaux problèmes d'ordonnancement sur ces chaînes sont alors apparus comme le problème d'ordonnancement de voitures (*car sequencing*).

L'ordonnancement de la production, dans l'industrie automobile comme dans la plupart des industries, est une problématique complexe. Elle consiste à déterminer une séquence d'exécution de travaux de façon à satisfaire un ou plusieurs objectifs pouvant être conflictuels, en tenant compte d'un certain nombre de contraintes temporelles (délais, contraintes d'enchaînement) et de contraintes portant sur la disponibilité des ressources requises [Lopez et Roubellat 2001]. Si la formulation de tels problèmes peut paraître simple, leurs résolutions s'avèrent souvent complexes. En effet, les problèmes d'ordonnancement sont souvent des problèmes d'optimisation combinatoire dont la complexité est maintenant établie. Pour la majorité de ceux-ci, il n'existe pas d'algorithmes connus pour les résoudre de façon optimale dans un temps polynomial. La plupart des problèmes d'ordonnancement sont *NP-difficiles* [Garey et Johnson 1979; Chapman 1987]. Il s'ensuit que la plupart des problèmes d'ordonnancement industriels s'avèrent encore plus complexes en raison de leurs tailles et des différentes contraintes supplémentaires relatives à l'environnement de production. Apporter des solutions efficaces et performantes à ce type de problèmes constitue assurément un enjeu économique important pour une entreprise.

Ce chapitre décrit la problématique d'ordonnancement de voitures dans une chaîne de montage automobile qui est utilisée dans la suite de cette thèse. La section 2.2 dresse un rapide aperçu de l'industrie automobile mondiale et canadienne. Par la suite, nous décrivons brièvement, à la section 2.3, le fonctionnement d'une usine d'assemblage de voitures, cette présentation ne se veut pas exhaustive, mais a uniquement pour but d'introduire un minimum d'éléments nécessaires permettant au lecteur de mieux cerner la problématique de ce travail de recherche.

2.2 Présentation de l'industrie automobile mondiale

L'industrie automobile est un des secteurs industriels suscitant le plus d'intérêt au niveau de la communauté scientifique. Cet intérêt s'explique par le fait que cette industrie représente des enjeux économiques considérables pour les pays producteurs. Ce secteur constitue aussi un immense champ d'innovation tant au niveau technologique qu'au niveau organisationnel ou conceptuel. Afin de mieux comprendre les enjeux de ce secteur, les prochaines sous-sections présentent un rapide portrait de l'industrie automobile et de son évolution.

2.2.1 Historique

Avant 1914, la construction automobile répond aux caractéristiques des produits de luxe et aux exigences d'une clientèle restreinte. Les voitures sont construites quasiment sur mesure et ce, souvent par des artisans. Rapidement, la demande de la clientèle va imposer la transformation de cet artisanat en une industrie centrée sur un découpage des tâches et sur l'utilisation à chaque étape de machines spécialisées.

Ce type d'organisation de la production industrielle est formalisé pour la première fois par Frederick W. Taylor qui commence sa carrière comme ouvrier avant de devenir ingénieur. En 1903, il développe sa théorie de l'organisation scientifique du travail dans son ouvrage *Shop Management* [Taylor 1911]. L'idée principale est de décomposer les tâches, de minuter les gestes des ouvriers pour améliorer la qualité, diminuer les coûts et les délais. C'est le début du travail à la chaîne et de la production de masse. Au-delà de la production, le taylorisme touche également le management de l'entreprise au sens large.

Reprenant les théories libérales, Taylor soutient que chaque individu en donnant le meilleur de lui-même concourt au bien général de toute l'entreprise.

Dès 1908, Henry Ford voit tout le bénéfice que l'industrie automobile peut tirer de l'application de ces théories. La Ford T naît ainsi d'un concept industriel : la fabrication en grande série. Voiture rustique et sans variantes, elle est accessible à une large classe moyenne. Associée à une solide réflexion commerciale (Henry Ford développe les premières formules de crédit), la logique industrielle est un succès populaire. Jusqu'en 1927, quinze millions de Ford T seront produites. Le taylorisme connaît dès lors, avec l'essor du Fordisme, un développement fulgurant.

Dans ce contexte de profonde mutation industrielle caractérisant l'entre-deux-guerres, émergent des mouvements sociaux de contestation que les patrons vont rapidement prendre en compte. En effet, si le taylorisme permet d'accroître la productivité et favorise l'emploi peu qualifié, il apparaît vite que l'accroissement de la production ne peut être obtenu que par le surmenage. L'ouvrier réduit au rang de manœuvre voit sa situation intellectuelle et sociale amoindrie, la monotonie du travail et l'absence d'effort intellectuel sont décourageants.

Après 1945, l'industrialisation redevient la priorité des états. En France, les nationalisations et le Plan encouragent la reconstruction et la modernisation de l'outil de production afin de pérenniser la croissance économique que risque parallèlement d'entraver la pénurie de main d'œuvre. La production automobile est alors partagée entre les principaux constructeurs de l'époque. Cependant, les années 60 et 70 voient la population ouvrière décliner. Les industries, notamment automobile, n'apparaissent alors plus comme

le fer de lance de l'économie, desservie par l'image du taylorisme. Les préoccupations grandissent en outre quant à l'impact environnemental de telles activités. Dans ce contexte, se développent des thèses fondées sur la motivation du travailleur et sur l'enrichissement des tâches par l'accomplissement d'une œuvre utile et personnalisée. D'autres pays, comme l'Allemagne ou le Japon ne connaissent pas la même pénurie de main d'œuvre et ne vont pas suivre la même évolution. Au début des années 60, l'économie japonaise est dominée par des groupes industriels dont les activités automobiles se partagent un marché étroit. Le taylorisme y apparaît inapplicable notamment en raison du manque de place pour entreposer les stocks importants liés à une telle organisation de la production. Pour supprimer ces stocks, Taiichi Ohno, ingénieur chez Toyota, invente le moyen de supprimer ces stocks en modifiant l'organisation de la sous-traitance et de la production. Afin de supprimer tout gaspillage qui augmente les coûts, il ne faut produire que ce qui est commandé, ne commander que ce qui est nécessaire et ne livrer les commandes qu'au moment où elles sont utiles. Ainsi naissait le système de production Toyota basé sur ces principes de *juste à temps* et de *flux tirés* [Soulard 2002].

Les crises des années soixante-dix plongent tous les secteurs industriels dans une période de récession. Dans l'industrie automobile, comme ailleurs, le modèle fordien atteint ses limites. À l'image des constructeurs japonais, les occidentaux doivent envisager d'autres façons de produire, car d'autres manières de consommer sont apparues. La priorité aujourd'hui est de mieux prendre en compte la demande et les évolutions du marché. La qualité, le respect des coûts et des délais, la flexibilité nécessaire pour produire des modèles

de plus en plus nombreux et diversifiés deviennent les maîtres mots de l'organisation et de la stratégie industrielle de la majorité des constructeurs.

En changeant d'objectif, l'industrie automobile change aussi de visage. La section suivante présente les principaux acteurs actuels de ce secteur industriel.

2.2.2 Les principaux acteurs

En une quinzaine d'années, la production automobile mondiale a progressé de plus de 40 % avec 69.3 millions de véhicules, dont 50 millions de véhicules légers. En 2006, la production a progressé de 4.2 % soit un ralentissement de 1.6 % par rapport à 2004. Depuis le début du XXIe siècle, l'Europe n'est plus la première zone de production. Elle est désormais devancée par la zone Asie-Océanie dont la production a progressé de 60 % depuis 1990 grâce essentiellement à la Chine qui fabrique actuellement plus de 7.2 millions de véhicules.

La production automobile est toujours dominée par les dix premiers groupes qui produisent 73 % des véhicules contre 80 % il y a 7 ans. Après avoir dépassé Ford pour devenir le second constructeur mondial derrière General Motors en 2004, le groupe Toyota est devenu, en 2007, le premier constructeur mondial tous véhicules confondus. Le Tableau 2.2 présente le classement des principaux constructeurs automobiles ainsi que leur production pour l'année 2006. De son côté, le Tableau 2.3 présente le classement *Automotive news 2008* pour les ventes de voitures des 10 premiers groupes pour l'année 2007.

Rang	Constructeur	Production totale (en milliers d'unités)	Production de véhicules légers
1	General Motors	8 926	5 708
2	Toyota	8 036	6 800
3	Ford	6 268	3 801
4	Alliance Renault-Nissan	5 716	4 598
5	Groupe VW	5 685	5 430
6	Hyundai-Kia	3 844	3 413
7	Honda	3 670	3 550
8	PSA Peugeot-Citroën	3 357	2 961
9	Chrysler	2 545	710
10	Fiat-Iveco	2 318	1 754

Tableau 2.2 : La production des principaux constructeurs automobile en 2006 [source CCFA-OCIA]

Rang	Constructeur	Nombre d'unités vendues
1	Toyota	9 366 000
2	General Motors	8 902 252
3	Groupe VW	6 191 618
4	Ford	5 964 000
5	Hyundai-Kia	3 961 629
6	Honda	3 831 000
7	Nissan	3 675 574
8	PSA Peugeot-Citroën	3 428 400
9	Chrysler	2 676 268
10	Fiat-Iveco	2 620 864

Tableau 2.3 : Classement *Automotive New'2008 Global Data Book* pour les ventes de véhicules en 2007

En 2006, les immatriculations de véhicules ont augmenté de 3.6 % par rapport à 2005 pour atteindre 67,8 millions dans le monde. La croissance se situe principalement en Europe de l'est et en Asie, notamment en Chine. Les marchés nord-américains et ouest européen présentent, pour leur part, des croissances régulières mais faibles. Ainsi, l'Europe occidentale et l'Amérique du Nord, marchés de tailles comparables, ne concentrent plus que 55 % des immatriculations mondiales, contre 65 % en 2000. Le Tableau 2.4 présente la

production de véhicules et le nombre d'immatriculations par zone d'activité pour l'année 2006.

	Production (en millier de véhicules)	**Immatriculation** (en millier de véhicules)
Asie-Océanie	28 192	21 196
Europe	21 406	21 851
ALENA	15 882	19 895
Amérique du sud	3 212	3 529
Afrique	566	1 312

Tableau 2.4 : Production et immatriculation dans le monde en 2006 [source CCFA-OCIA]

2.2.3 La place de l'industrie automobile au Canada

Au Canada, l'industrie automobile est le premier secteur de fabrication en importance. Elle représente environ 12 % du PIB de la fabrication et près de 25 % des expéditions manufacturières. La valeur de la production de ce secteur a atteint près de 90 milliards de dollars canadien en 2006. Avec une production de plus de 2.5 millions d'unités, en 2007, le Canada est le 9ème producteur mondial de véhicules et le premier producteur mondial par habitant. Toutefois, l'industrie automobile canadienne subit, sur le continent nord américain, une pression, sans cesse grandissante, du Mexique qui devrait prendre la place du Canada en 2008 comme 9ème producteur mondial de véhicules. Malgré ce recul, le rapport 2007 du consultant Harbour classe le Canada comme le pays le plus productif en Amérique du Nord en ce qui concerne la fabrication de véhicules. En effet, le montage d'un véhicule prend en moyenne 21 heures au Canada, alors que cela prend presque 8% de plus

au États-Unis et plus de 82% au Mexique. Toujours selon ce rapport, le Canada possède les sites industriels les plus performants (1ère, 2ème et 4ème usines de montage les plus productives d'Amérique du nord).

Le marché domestique canadien représente un volume de ventes variant entre 1.4 et 1.7 millions de véhicules par an, réparties entre près de 3.500 concessionnaires. En 2006, le parc automobile canadien compte 19.2 millions de véhicules, dont l'âge moyen est de 8.4 ans; environ 60.5% des véhicules canadiens sont immatriculés en Ontario ou au Québec.

Toutefois, les restructurations et fermetures de sites lancées par GM, Ford et Chrysler à la fin de l'année 2005 (3500 emplois seront éliminés au Canada d'ici la fin de l'année 2008) montrent que le pays n'est pas à l'abri des difficultés globales du secteur automobile. La situation s'est encore dégradée ces derniers mois sous l'effet de plusieurs facteurs, notamment le renchérissement important du dollar canadien qui, allié au ralentissement de la croissance économique américaine, a un impact dévastateur sur l'industrie automobile et plus généralement l'industrie manufacturière canadienne.

2.3 Description d'une usine d'assemblage de voitures

Après avoir brossé un rapide historique de l'industrie automobile et présenté la situation actuelle du secteur, il est maintenant important de préciser le mode de fonctionnement d'une usine d'assemblage de voitures afin de mieux comprendre la problématique de ce travail de recherche. La description d'une usine d'assemblage de voitures présentée dans cette partie de la thèse est basée sur la description proposée dans la donnée du Challenge ROADEF 2005 [Nguyen et Cung 2005].

À ce stade, il faut de mentionner qu'une usine d'assemblage de voitures est généralement constituée de chaînes de montage. Chaque chaîne de production est composée de trois ateliers successifs par lesquels les voitures passent successivement à vitesse constante sur la courroie d'un convoyeur : *la tôlerie*, *la peinture* et *l'assemblage*. De cette manière, les voitures passent dans le même ordre par les différents postes de travail de chacun des trois ateliers l'un à la suite de l'autre. Autrement dit, la séquence de production déterminée à l'entrée de la chaîne est la même pour les trois ateliers. Comme nous pourrons le constater, chacun de ces ateliers ne poursuit toutefois pas les mêmes objectifs. Les trois ateliers conventionnels d'une chaîne de montage automobile sont illustrés à la Figure 2.1.

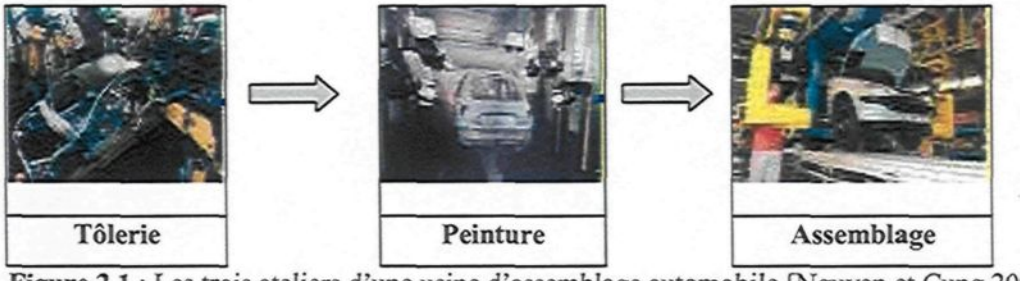


Figure 2.1 : Les trois ateliers d'une usine d'assemblage automobile [Nguyen et Cung 2005]

Chaque atelier possède ses propres contraintes. Dans l'atelier de tôlerie, les différentes armatures métalliques de la voiture sont assemblées et soudées afin d'obtenir la carrosserie de la future voiture avec ses ouvrants. En général, c'est l'atelier le plus automatisé car la plupart des manipulations et des soudures sont faites par des robots. Dans cet atelier, la durée de changements des outils de presses incite à la fabrication par lot [Lopez et Roubellat 2001].

Après être passées par l'atelier de tôlerie, les caisses à nues passent à l'atelier de peinture où elles sont traitées contre les agressions extérieures qu'elles auront à subir. La

caisse est ensuite rendue étanche au bruit et à l'eau avant d'être peinte en utilisant des pistolets à peinture. À chaque changement de teintes dans la séquence de production, les pistolets sont purgés à l'aide d'un solvant spécial. Les coûts d'utilisation du solvant pour chacune des purges étant non négligeables, il importe de minimiser la consommation dudit solvant et, par conséquent, le nombre de purges. Dans cet atelier, on cherche donc à regrouper les voitures de même teinte, aussi appelées *rafale de teinte,* afin de minimiser le nombre de purges en générant des rafales de teintes les plus longues possible.

Une fois peintes, les caisses passent à l'atelier d'assemblage. C'est dans cet atelier que les différentes pièces mécaniques et électriques, les garnitures et les options de la voiture sont ajoutées. Ces pièces représentent près de 70 % de la valeur de la voiture finale [Lopez et Roubellat 2001]. En général, l'atelier d'assemblage est l'atelier qui nécessite le plus de personnel car une grande partie des travaux y est effectuée manuellement. La diversité de voitures à assembler dans cet atelier entraîne indubitablement une variabilité sur les différents postes en bord de chaîne qu'il est important de *lisser* afin d'éviter un blocage complet de la chaîne de montage. Pour cela, il faut espacer le plus possible les voitures nécessitant des opérations lourdes afin de ne pas surcharger les postes de travail concernés par l'assemblage de ces voitures.

Les différents ateliers de la chaîne de montage possèdent donc des contraintes spécifiques, très souvent conflictuelles, qu'il est impératif de respecter pour que chaque atelier fonctionne correctement. Dans cet optique et afin d'éviter des coûts supplémentaires, il est important de déterminer au préalable dans quel ordre les voitures doivent arriver sur la chaîne. L'évolution des besoins de la clientèle a toutefois entraîné une diversité importante

au niveau de la production de voitures et a complexifié la tâche du planificateur. On est alors confronté à un problème qualifié *d'ordonnancement sur ligne de produits hétérogènes* qui cherche à éviter les phénomènes de *désamorçage* ou de *saturation* conduisant à des arrêts de la ligne. C'est cette problématique que nous abordons dans ce travail de recherche. La section suivante présente le problème d'ordonnancement de voitures (*car sequencing*) qui représente un enjeu économique important pour les constructeurs.

2.4 Problématique d'ordonnancement de voitures

La problématique d'ordonnancement de voitures est apparue dans le secteur de production lorsque la personnalisation des voitures a remplacé la standardisation de masse. Dans la littérature, on retrouve plusieurs formulations de problème d'ordonnancement de voitures en fonction des différents objectifs visés, les sections suivantes en présentent quelques-unes.

2.4.1 Ordonnancement et juste à temps

Une des préoccupations des centres de production automobile est de rechercher la meilleure synchronisation possible entre le flux principal des voitures et tous les flux d'approvisionnement de composants. Ceci doit permettre de réduire les coûts de stockage de ces constituants. Dans ce contexte, le problème d'ordonnancement consiste à déterminer l'ordre dans lequel les voitures seront produites afin de garder constant le taux d'utilisation des différentes pièces. Dans l'industrie automobile, ce genre d'approche a été initialement préconisé par la firme japonaise Toyota.

Bautista *et al.* [1996] démontrent que le problème de maintien d'un niveau constant d'utilisation des pièces équivaut à un problème de recherche de chemin minimum dans un graphe. Par conséquent, n'importe quel algorithme de recherche de chemin minimum pourrait être utilisé pour obtenir une solution optimale au problème. Cependant, les auteurs mettent en évidence l'inaptitude de ce genre d'algorithme à résoudre des problèmes de tailles réelles à cause du nombre considérable d'arcs contenu dans des problèmes réels.

Parmi les approches heuristiques utilisées pour résoudre ce problème, on retrouve l'algorithme de Monden connu sous le nom de *Goal chasing method* (GCM) [Monden 1998]. Il s'agit d'une heuristique de construction progressive de la séquence qui sélectionne, à chaque itération, le modèle permettant de minimiser la distance entre la consommation effective et la consommation idéale des composants. L'estimation de l'éloignement des taux de consommation est calculée à partir du principe de la distance euclidienne. De leur côté, Leu *et al.* [1996] développent un algorithme génétique pour minimiser la variabilité de la consommation de pièces. Les auteurs montrent la supériorité de leur algorithme par rapport à l'heuristique de Monden dans la plupart des cas examinés.

2.4.1 Ordonnancement par lissage de la charge de travail

L'approche par lissage de la charge consiste à uniformiser la demande en main d'œuvre tout au long de la chaîne. Une de ces formulations est connue sous le nom de *problème de variation du taux de produit*. Soit $qant_m$ la quantité de voitures d'un modèle m donné à ordonnancer, avec $m = 1 \ldots M$ et N voitures à produire au total.

On a alors : $$N = \sum_{m=1}^{M} qant_m \qquad (2.1)$$

Soit nr_{mp} le nombre d'unité du modèle m ordonnancée entre les positions 1 et p. Le taux idéal τ_m d'unités du modèle m dans la séquence de voitures est donné par :

$$\tau_m = {qant_m}/{N} \qquad (2.2)$$

Entre les positions 1 et p le nombre idéal de voitures du modèle m est égal à ni_m, son nombre réel est nr_{mp}. L'objectif est donc de minimiser la distance entre le nombre idéal et le nombre réel de voitures d'un modèle donné. Ceci s'écrit :

$$\sum_{p=1}^{N}\sum_{m=1}^{M}\left(nr_{mp} - ni_m\right)^2 \qquad (2.3)$$

Plusieurs auteurs ont proposé des méthodes de résolution pour cette formulation du problème. Xiaobo et Ohno [1997] développent un algorithme de séparations et d'évaluations progressives permettant de trouver la solution optimale pour de petites instances de problèmes. Thomopoulos [1967] cherche à minimiser plusieurs indicateurs de la perte d'efficacité des opérateurs sur la ligne comme, par exemple, la surcharge de travail. L'auteur propose une méthode de construction progressive de la séquence des voitures, en choisissant à chaque itération la voiture qui minimise les pénalités associées aux différentes pertes d'efficacité. Macaskill [1973] fournit une amélioration de cette méthode. Sumichrast *et al.* [1992] développent, de leur côté, une heuristique reposant sur le même principe mais où l'objectif consiste à minimiser la différence entre le temps (cumulé) idéal d'assemblage et le temps effectif (itérations passées).

2.4.2 Problème théorique d'ordonnancement de voitures (POV)

2.4.2.1 Définition du problème

La version *théorique* du POV que l'on retrouve dans la littérature est une version simplifiée de la problématique industrielle qui se préoccupe uniquement des contraintes de l'atelier d'assemblage. Dans cet atelier, chaque voiture est caractérisée par un ensemble d'options O (toits ouvrant, système ABS, etc.). Chaque option est montée sur différents postes conçus pour traiter au plus un certain pourcentage de voitures passant dans l'atelier. Pour éviter que la capacité des postes ne soit jamais dépassée, les voitures nécessitant les mêmes options doivent par conséquent être espacées afin de réguler le rythme de production.

Cette formulation du problème a été décrite pour la première fois par Parello *et al.* [1986]. Le POV consiste donc à déterminer l'ordre dans lequel nc voitures doivent être produites dans l'atelier d'assemblage de manière à installer différentes options sur chacune d'elles. Afin de réguler le rythme de production, la formulation de *contraintes de capacité* sur les options est nécessaire. Ces contraintes s'expriment sous la forme d'un ratio r_k/s_k signifiant que pour toute sous-séquence consécutive de s voitures, au plus r voitures peuvent posséder l'option k. Lorsqu'une contrainte de capacité n'est pas respectée à un endroit de la sous-séquence, on est alors en présence d'un *conflit*. Par exemple, le poste des régulateurs de vitesse peut présenter une contrainte de la forme 3/5, ce qui signifie que sur n'importe quelle sous-séquence de 5 voitures consécutives sur la ligne, au plus 3 voitures peuvent demander un régulateur de vitesse. Lorsque que plus de 3 voitures demandent l'option dans un bloc de 5 voitures, il y a alors violation de la contrainte de capacité et donc

on comptabilise un conflit. Kis a démontré que ce problème est NP-difficile au sens fort [Kis 2004]. Le POV intéresse la communauté scientifique depuis le milieu des années 80. Une revue détaillée du problème et des méthodes de résolutions utilisées dans la littérature pour le résoudre est proposée dans Delaval [1997].

Afin de simplifier la résolution du problème, les voitures identiques, autrement dit celles qui nécessitent une même configuration d'options sont regroupées en *V classes* de voitures pour lesquelles on connaît le nombre de voitures à produire c_v. Ces quantités représentent les *contraintes de production du problème.*

Le Tableau 2.5, tiré de Smith [1997], présente un exemple d'instance du POV. Pour cette instance, on a 5 options (*Opt*), 12 classes de voitures (*V*) et $\sum_{v=1}^{V} c_v = nc = 25$.

			Classes de voitures											
Options	***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
c_v			3	1	2	4	3	3	2	1	1	2	2	1

Tableau 2.5 : Un exemple d'instance du POV

Une solution d'une instance d'un POV est représentée par un vecteur de longueur *nc* contenant une et une seule classe de voiture dans chacune de ses cases. On peut construire une matrice *Mat* de valeurs binaires de taille *Opt* **nc* à partir des informations contenues dans le vecteur de la solution dans le but d'évaluer cette dernière. On a $Mat_{ki} = 1$ si la classe

de voiture assignée à la position *i* du vecteur de la solution demande l'option *k*, 0 sinon. La Figure 2.2 présente une solution partielle pour l'instance du Tableau 2.5. 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 *i+1* à *i+3* car la contrainte d'espacement 2/3 n'est pas respectée. Il en va de même pour l'option 4 aux positions *i* à *i+4* et pour l'option 5 aux positions *i+1*à *i+5*. On peut conclure qu'il existe 3 conflits dans cette partie de la solution.

Solution partielle pour l'instance du Tableau 2.5 :

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

Matrice *Mat* 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 2.2 : Exemple d'évaluation d'une solution du POV

À titre indicatif, le Tableau 2.6 résume les principales notations utilisées dans ce travail pour décrire le POV.

Notations	Signification
nc	Nombre total de voitures à ordonnancer
r_k/s_k	Contrainte de capacité pour l'option *k*
Opt	Nombre total d'options
V	Nombre total de classes
Mat	Matrice binaire de taille (*Opt* **nc*)
c_v	Nombre de voitures de la classe *v*

Tableau 2.6 : Résumé de la notation du POV

2.4.2.2 Taux d'utilisation et difficulté d'une classe de voitures

L'espace de recherche d'une instance du POV théorique est constitué par l'ensemble de toutes les permutations possibles de *nc* voitures requérant différentes configurations d'options. On peut donc facilement en déduire que, pour *V* classes données, l'espace de recherche d'une instance de taille *nc* correspondra a :

$$\frac{nc!}{c_1!*c_2!...*c_v!} \tag{2.4}$$

Contrairement à d'autres problèmes d'ordonnancement, la difficulté d'une instance du POV ne dépend pas uniquement de la taille de son espace de recherche, mais aussi du *taux d'utilisation* des différentes options (tu_k), comme l'a indiqué Smith [Smith 1997]. Le taux d'utilisation d'une option *k* correspond à la proportion de voitures qui requièrent l'option *k* par rapport à la capacité maximale pour cette option. Formellement, le taux d'utilisation d'une option *k* s'exprime comme suit :

$$tu_k = nb_k \Big/ \lceil nc * r_k / s_k \rceil \tag{2.5}$$

où nb_k correspond au nombre de voitures nécessitant l'option *o*. Un taux d'utilisation proche de 1 indique que la demande est proche de la capacité maximale du poste, et par conséquent que l'option est très contrainte [Gottlieb *et al.* 2003].

La difficulté D_v d'une classe *v* est par la suite obtenue en additionnant simplement les taux d'utilisation des options appartenant aux voitures de cette classe [Gottlieb *et al.* 2003] :

$$D_v = \sum_{k=1}^{Opt} O_{vk} tu_k \qquad (2.6)$$

2.4.2.3 Résolution du POV à l'aide de méthodes exactes

Le POV a souvent été formulé comme un problème de satisfaction de contraintes (CSP). Il peut donc être résolu à l'aide d'algorithme issu de la programmation par contrainte (backtracking, backjumping, foward checking ...) [Régin et Puget 1997; Smith 1997; Boivin 2005]. Cependant, il a été observé que la complexité des contraintes inhérentes au POV fait que ce type d'algorithmes ne représente pas généralement des approches adéquates pour la résolution de certaines instances de ce problème [Smith 1997].

Une autre manière d'approcher ce problème, à l'aide de méthodes exactes, consiste à le formuler sous forme de programme linéaire. Drexl et Kimms [2001], Estellon *et al.* [2007], Prandtstetter et Raidl [2007] et Gravel *et al.* [2005] proposent chacun une formulation du problème à l'aide d'un modèle linéaire en nombres entiers. Les auteurs ont testé les performances de leurs approches à l'aide d'un jeu d'instances générées pour Drexl et Kimm et d'ensembles tests de la littérature pour Estellon *et al.*, Prandtstetter et Raidl et Gravel *et al.* Si les résultats se sont montrés concluants pour certains problèmes, ils montrent aussi les limites de ce type d'approche pour les problèmes les plus difficiles. Il est important de préciser que, contrairement aux autres approches, la stratégie proposée par Estellon *et al.* [2007] n'utilise pas la fonction d'évaluation classique du POV pour compter le nombre de conflits. En effet, les auteurs ont préféré aborder le problème en utilisant la fonction d'évaluation introduite pour le problème industriel d'ordonnancement de voitures. Cette fonction d'évaluation est présentée en détail à la Section 2.4.3.

2.4.2.4 Résolution du POV à l'aide approches heuristiques

Les méthodes exactes ayant montré leurs limites, plusieurs auteurs se sont naturellement tournés vers des approches heuristiques. Parmi les approches heuristiques proposées, on retrouve principalement des méthodes de recherche dans le voisinage [Davenport et Tsang 1999; Estellon *et al.* 2007; Gottlieb *et al.* 2003; Neveu *et al.* 2004; Perron et Shaw 2004]. Les différentes approches de recherche dans le voisinage proposées pour résoudre le POV ont généralement été testées à l'aide d'ensembles test de la littérature et diffèrent principalement sur trois points : la manière de construire la solution initiale, le voisinage exploré à chaque mouvement et la manière de se déplacer dans le voisinage.

Vers le milieu des années 80, Parello *et al.* décident d'appliquer des techniques d'intelligence artificielle (IA) classique pour résoudre le POV [Parello *et al.* 1986]. Dans leur article, les auteurs présentent un modèle du problème à base de prédicats et de règles de production. Cependant, les résultats de l'approche ne sont pas concluants et l'utilisation de cette technique d'IA sera plus tard remise en cause [Parello 1988] en raison de la complexité des prédicats à mettre en place pour résoudre un POV de taille intéressante.

Dincbas *et al.*[1988] abordent le POV en utilisant un modèle de programmation logique sous contrainte (PLC). Les auteurs présentent leurs résultats pour des instances de problèmes générées aléatoirement. Les instances les plus complexes sont composées de 200 voitures et ont un pourcentage d'utilisation moyen de 90 %. Même si dans tous les cas, le modèle est parvenu à trouver des solutions sans conflits, l'utilisation de la PLC entraîne des temps de calculs imprévisibles et quelquefois élevés. Smith *et al.* [1996] proposent d'aborder le problème en utilisant un réseau de neurones basé sur le modèle de Hopfield

[Hopfield et Tank 1985]. Les résultats expérimentaux présentés par les auteurs montrent que leur approche se limite à la résolution de très petites instances du problème avec un faible taux d'utilisation.

On retrouve aussi des approches basées sur l'optimisation par colonies de fourmis (ACO). Solnon [2000] a introduit un premier ACO dédié à la résolution des problèmes de satisfaction de contrainte par permutation. L'algorithme sera par la suite bonifié par l'addition d'une heuristique gourmande [Gottlieb *et al.* 2003]. Gravel *et al.* [2005], de leur côté, ont introduit un algorithme d'ACO (ACS-2D) dont l'une des particularités concerne la probabilité de transition, *i.e* la probabilité de choisir une classe de voitures, qui est calculée en fonction de trois éléments : une trace de phéromones à deux dimensions, le nombre de nouveaux conflits engendrés par l'ajout d'une classe de voitures et la difficulté d'une classe D_v. Récemment, une version raffinée de l'ACS-2D, appelée ACS-3D, qui incorpore une nouvelle structure de trace de phéromones à 3 dimensions a été présentée [Gagné *et al.* 2008]. Les différents algorithmes ACO s'avèrent particulièrement performants pour la résolution du POV.

Finalement, on dénombre très peu d'approches basées sur des algorithmes génétiques. Warwick et Tsang [1995] proposent le GAcSP qui est, à notre connaissance, la première tentative de résolution du POV à l'aide d'algorithmes génétiques. À chaque génération, les individus sélectionnés dans la population sont combinés à l'aide d'un opérateur de croisement adaptatif (UAX). Une fois créés, les nouveaux individus peuvent ne plus respecter les contraintes de production, ils sont alors réparés à l'aide d'un algorithme glouton. Une fois le processus de réparation terminé, chaque nouvel individu est amélioré

en utilisant une procédure de recherche locale. Les expérimentations des auteurs indiquent que le GAcSP permet de résoudre les instances « faciles » du POV, celles avec de faibles taux d'utilisation. Cependant, la performance de la méthode est sérieusement diminuée lorsque le taux d'utilisation augmente. Plus récemment, Terada *et al.* [2006] présentèrent un AG classique pour résoudre le POV. Dans leur approche, les auteurs utilisent un croisement standard à un point de coupure sans aucun opérateur de mutation aléatoire. Comme pour le GAcSP, les résultats expérimentaux présentés par les auteurs montrent que leur approche permet de résoudre des instances avec de faibles taux d'utilisation. Cependant, les performances sont très vite dégradées dès que le taux d'utilisation augmente.

2.4.3 Problème industriel d'ordonnancement de voitures

2.4.3.1 Définition du problème

Nous présentons, dans cette section, les principaux éléments définissant le problème industriel d'ordonnancement de voitures sur une chaîne d'assemblage tel que décrit par le constructeur automobile Renault dans le cadre du Challenge ROADEF 2005. Le lecteur peut trouver une description détaillée du problème dans Nguyen et Cung [2005] et Solnon *et al.* [2007].

À chaque journée de production, les commandes des clients sont transmises en temps réel aux usines d'assemblage. La tâche quotidienne des planificateurs de ces usines consiste alors : (1) à affecter une journée de fabrication à chaque voiture commandée en fonction des contraintes de capacité des lignes d'assemblage et des délais de livraison promis aux clients; et (2) à ordonnancer les voitures à l'intérieur de chaque journée de fabrication de

manière à respecter le mieux possible les contraintes des trois ateliers de la chaîne d'assemblage. L'ordonnancement étant réalisé en considérant les trois objectifs selon un ordre lexicographique, la séquence trouvée est ainsi appliquée à l'ensemble de la chaîne d'assemblage. Dans la définition du problème industriel d'ordonnancement de voitures présentée dans le cadre du Challenge ROADEF 2005, le constructeur automobile Renault a établi que les technologies utilisées dans l'usine font en sorte que les contraintes de l'atelier de tôlerie peuvent tout simplement être négligées. Le POV consiste ainsi à déterminer la séquence de production d'un ensemble de voitures (nc) en considérant les contraintes de l'atelier de peinture et de montage.

Au niveau de l'atelier de peinture, on cherche à regrouper les voitures d'une même couleur de manière consécutive dans la séquence pour minimiser le nombre d'opérations de nettoyage des pistolets. En effet, les pistolets utilisés doivent obligatoirement être nettoyés à l'aide de solvant entre deux voitures de couleur différente ainsi qu'après un nombre maximal de voitures (rl_{max}) d'une même teinte pour assurer la qualité. Il faut préciser la particularité à l'effet que chaque purge entraîne obligatoirement un changement de couleur. Ainsi, toute solution comportant un nombre de voitures consécutives de même couleur supérieur à rl_{max} doit donc être considérée comme une solution non réalisable.

Au niveau de l'atelier de montage, de nombreux éléments sont ajoutés à la carrosserie peinturée pour terminer l'assemblage de la voiture. Celle-ci est caractérisée par un ensemble d'options Opt pour lesquelles les postes réalisant ces options sont conçus pour traiter au plus un certain pourcentage des voitures passant dans l'atelier. Comme pour le problème théorique d'ordonnancement de voitures, ces contraintes de capacité sont

exprimées sous la forme d'un ratio r_k/s_k. Cependant, la formulation industrielle du POV subdivise les contraintes de capacité de l'atelier d'assemblage en deux types, les contraintes liées aux *options prioritaires* et celles liées aux *options non-prioritaires*. La distinction entre ces deux groupes d'options réside essentiellement dans le fait que l'on peut contrer les effets d'un conflit sur les contraintes non-prioritaires en ajoutant, par exemple, du personnel supplémentaire pour accroître le rythme de production. À l'opposé, les contraintes prioritaires sont des contraintes pour lesquelles on ne dispose pas de moyen pour contrer l'effet d'une violation de la contrainte et qui entraînent un blocage de la chaîne. Le respect ou non des contraintes non-prioritaires relève de la politique de l'entreprise alors que celui des contraintes prioritaires est une obligation. Dans cet atelier, nous cherchons ainsi à optimiser deux objectifs distincts : le nombre de conflits liés aux contraintes de capacité sur les options prioritaires (*HPO*) et le nombre de conflits liés aux contraintes de capacité sur les options non-prioritaires (*LPO*).

Nous avons choisi de regrouper les voitures identiques, autrement dit celles possédant une même configuration d'options prioritaires et non-prioritaires, à l'intérieur de V classes pour lesquelles on peut ainsi établir le nombre de voitures à produire (c_v). Ces quantités représentent alors les contraintes de production du problème. La Figure 2.3 (a) présente un exemple de problème industriel pour la production de 25 voitures (*nc*) possédant 5 options (*Opt*) formant 6 classes de voitures (V) et avec une possibilité de 4 couleurs différentes réparties dans chacune des classes. Dans cet exemple, on considère que les 3 premières options sont des options prioritaires tandis que les deux options restantes sont non-prioritaires. On définit ainsi une séquence x de production par deux vecteurs représentant

respectivement la classe des voitures (Classes) et les codes de couleur de celles-ci (Couleurs) tel que présenté à la Figure 2.3 (b). Une séquence de production du POVI sera notée x = {Classes/ Couleurs} dans la suite de la thèse et les éléments situés à la position i de la séquence seront définis par $x(i)$ = Classes(i)/ Couleurs (i).

			Classes #					
Options	***r***	***s***	1	2	3	4	5	6
1	1	2	0	1	1	0	0	0
2	2	5	1	0	1	0	1	1
3	1	3	0	1	0	0	0	0
4	3	5	0	0	0	1	0	1
5	2	3	0	1	1	0	1	0
c_v			5	5	4	4	3	4
Couleur #		1	2	1	1	2	1	1
		2	1	1	0	2	1	1
		3	1	3	2	0	0	2
		4	1	0	1	0	1	0

(a)

x	1	2	3	4	5	6	....	21	22	23	24	25
Classes	3	5	5	4	6	4		3	1	4	5	1
Couleurs	4	4	2	2	2	2		3	3	1	1	1

(b)

Figure 2.3 : Exemple d'une instance et d'une solution du POVI

Une autre particularité du problème industriel d'ordonnancement de voitures par rapport à sa version théorique réside dans le fait qu'il faut faire le lien entre les différentes journées de production. Ainsi, il faut évaluer la solution en considérant les voitures déjà planifiées à la journée précédente et en extrapolant le nombre minimum de conflits générés avec la journée suivante. De même, on ajoutera un changement de couleur si la première voiture de la journée courante est de couleur différente de la dernière voiture de la journée précédente. Pour évaluer le nombre de conflits sur les options, on procède, tout d'abord à la construction d'une matrice *Mat* de valeurs binaires de taille *Opt* * *nc* à partir des

informations contenues dans le vecteur de la solution. Ainsi, on a $Mat_{ki} = 1$ si la classe de voiture assignée à la position i du vecteur de la solution demande l'option k, 0 sinon. La décomposition du vecteur de classes de la solution de la Figure 2.3 en ses différentes options pour former la matrice *Mat* est présentée à la Figure 2.4. On a également reporté, à la Figure 2.4 (a), la fin de la solution de la journée précédente (positions -5 à -1) pour permettre l'évaluation du nombre de conflits à la jonction des deux journées. De plus, à la Figure 2.4 (b), on évalue la solution en fonction de la journée suivante en supposant des voitures sans aucune option. Ainsi, dans l'exemple de la Figure 2.4 la position -1 correspond à la dernière position de la solution trouvée pour la journée précédente tandis que la position 26 correspond à la première position de la solution de la journée suivante. Pour la journée courante J, les options 1, 3 et 4 ne causent aucun conflit dans cette partie de solution. En effet, pour chacune de ces options, on ne retrouve jamais de situation où, pour tous blocs de longueur s, il y a plus de r voitures possédant cette option. Par contre, pour l'option 2, il existe un conflit aux positions 1 à 5 puisqu'il y a 3 voitures possédant l'option alors que la contrainte établie le maximum à 2. De plus, il existe un conflit aux positions 2 à 6, un autre conflit entre les positions 20 et 24 ainsi que deux autres conflits aux positions 21 à 25 car la contrainte de capacité 2/5 n'est pas respectée. Pour l'option 5, il existe un conflit puisque la contrainte de capacité 2/3 est violée pour les positions 1 à 3. Pour la jonction avec la journée précédente, on note un conflit aux positions -1 à 1 pour l'option 1 ainsi qu'un autre conflit aux positions -1 à 2 pour l'option 5. Pour la jonction avec la journée suivante, on note un conflit aux positions 22 à 26 pour l'option 2. Si on considère que les trois premières options sont prioritaires et que les deux autres sont non-prioritaires, il existe

par conséquent, dans cette solution x, 8 conflits sur l'objectif HPO et 2 conflits sur l'objectif LPO. Il ne reste plus qu'à faire le décompte du nombre de changements de couleurs (COLOR) pour terminer l'évaluation de la solution x.

		Journée précédente (J-1)					Journée courante (J)						
Positions		-5	-4	-3	-2	-1	1	2	3	4	5	6	.....
Classes		4	1	4	4	2	3	5	5	4	6	4	
Option	1/2	0	0	0	0	**1**	**1**	0	0	0	0	0	
	2/5	0	1	0	0	0	**0**	**1**	**1**	**0**	**1**	**0**	
	1/3	0	0	0	0	1	0	0	0	0	0	0	
	3/5	1	0	1	1	0	0	0	0	1	1	1	
	2/3	0	0	0	0	**1**	**1**	**1**	**1**	0	0	0	

(a)

		Journée courante (J)						Journée suivante (J+1)				
Positions		....	21	22	23	24	25	26	27	28	29	30
Classes			3	1	4	5	1					
Option	1/2		1	0	0	0	0	0	0	0	0	0
	2/5		**1**	**1**	**0**	**1**	**1**	**0**	0	0	0	0
	1/3		0	0	0	0	0	0	0	0	0	0
	3/5		0	0	1	0	0	0	0	0	0	0
	2/3		1	0	0	1	0	0	0	0	0	0

(b)

Figure 2.4 : Évaluation de la solution présentée à la Figure 2.3

Le Tableau 2.7 résume la notation employée dans cette thèse pour présenter le POVI.

Notations	Signification
nc	Nombre total de voitures à ordonnancer
r_k/s_k	Contrainte de capacité pour l'option k
Opt	Nombre total d'options
V	Nombre total de classes
rl_{max}	Nombre maximum de voitures consécutives ayant la même couleur
HPO et *LPO*	Nombre de conflits liés aux options prioritaires et non-prioritaires
COLOR	Nombre de changement de couleur

Tableau 2.7 : Résumé de la notation du POV industriel

Dans la donnée du Challenge ROADEF 2005, les organisateurs proposent d'utiliser une somme pondérée en appliquant des poids w_1, w_2 et w_3 à chacun des objectifs pour établir

l'évaluation de la solution x. Ainsi la qualité d'une solution x est établie par l'équation suivante :

$$F(x)= w_1*obj_1+ w_2*obj_2+ w_3*obj_3 \qquad (2.7)$$

où obj_1, obj_2 et obj_3 correspondent respectivement à la valeur obtenue pour la solution x sur chacun des objectifs selon l'ordre de priorité attribué. Les pondérations w_1, w_2 et w_3 sont respectivement fixées à 1000000, 1000 et 1 [Nguyen et Cung 2005]. Selon la configuration des différentes usines du groupe Renault, les trois hiérarchisations suivantes des objectifs sont possibles : *HPO-COLOR-LPO*, *HPO-LPO-COLOR* et *COLOR-HPO-LPO*.

2.4.3.2 Méthodes de résolution

Le Challenge ROADEF a connu un engouement sans précédent avec un record de 55 équipes inscrites au début du challenge [Solnon *et al.* 2007] dont 27 ont proposé des méthodes de résolution. Parmi les algorithmes proposés par les équipes ayant participées au challenge, on retrouve principalement des méthodes de recherche dans le voisinage comme le recuit simulé, la recherche tabou itérative, la recherche locale itérative et la recherche à voisinage variable (*variable neighborhood search*) [Benoit 2007; Briant *et al.* 2007; Cordeau *et al.* 2007; Estellon *et al.* 2007; Gavranovic 2007; Ribiero *et al.* 2007], un algorithme d'optimisation par colonie de fourmis (ACO) [Gagné *et al.* 2006] et un algorithme génétique (AG) [Jaszkiewicz *et al.* 2004]. Les travaux de toutes les équipes n'ayant pas été publiés, l'énumération précédente n'est donc pas exhaustive. À la suite du challenge, d'autres auteurs ont abordé ce problème en utilisant la programmation linéaire en nombres entiers [Estellon *et al.* 2005; Gagné *et al.* 2006; Prandtstetter et Raidl 2007], un algorithme hybridant recherche à voisinage variable et programmation linéaire en nombres

entiers [Prandtstetter et Raidl 2007], un algorithme de recherche locale itérative [Ribeiro *et al.* 2007] ou encore un algorithme génétique [Siala 2005].

On note ainsi que peu d'auteurs ont proposé des algorithmes évolutionnaires et plus particulièrement des algorithmes génétiques pour résoudre ce problème à l'exception des travaux de Jaszkiewicz *et al.* [2004] et Siala [2005].

2.6 Conclusion

Ce chapitre a permis de présenter les concepts généraux de l'ordonnancement de la production ainsi que la problématique industrielle sur laquelle portera cette thèse : le problème d'ordonnancement de voitures. En résumé, l'ordonnancement peut être vu comme une problématique complexe dans le monde industriel et représente souvent un enjeu important influençant la productivité des entreprises. Dans l'industrie automobile, les nombreux travaux effectués sur le sujet, notamment dans le cadre du Challenge ROADEF 2005, témoignent de l'intérêt que suscite ce problème dans un contexte de production. Le POVI est un problème comportant trois objectifs. D'après la définition du problème [Nguyen et Cung 2005], on constate toutefois que les trois objectifs sont traités selon un ordre de priorité sans compensation possible entre les objectifs. Gagné *et al.*[2006] mentionnent dans leur article l'intérêt qu'il y aurait à considérer simultanément les objectifs afin de générer des solutions de compromis potentiellement intéressantes dans la pratique. Le chapitre suivant présente un domaine utile pour résoudre ce genre de problème, celui de l'optimisation multi-objectifs.

CHAPITRE 3

OPTIMISATION MULTI-OBJECTIFS ET ALGORITHMES ÉVOLUTIONNAIRES

3.1 Introduction

Le chapitre 2 a permis de montrer l'intérêt de résoudre adéquatement le problème d'ordonnancement de voitures dans l'industrie automobile actuelle, mais aussi les difficultés que peuvent entraîner un tel exercice. Même dans sa version théorique, Kis [2004] a prouvé que ce problème appartenait à la classe des problèmes NP-difficiles au sens fort. Le problème industriel s'avère donc davantage complexe en raison des différentes contraintes industrielles qui lui sont spécifiques. Comme la plupart des problèmes d'optimisation rencontrés dans la pratique, le POV industriel est un problème comportant plusieurs objectifs conflictuels. L'optimisation multi-objectifs s'intéresse à la résolution de ce type de problèmes. L'optimisation multi-objectifs n'est pas une science nouvelle, elle est apparue au 19^{ième} siècle avec les travaux en économie de Edgeworth [1881] et de Pareto [1896]. Elle a été appliquée initialement en économie et en gestion avant d'être graduellement utilisée dans d'autres domaines comme l'ingénierie ou l'informatique.

Résoudre un problème d'optimisation consiste souvent à déterminer la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise au moins une mesure de performance permettant d'effectuer une comparaison. Cette mesure de performance correspond souvent à un des objectifs du problème. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenue la meilleure évaluation par rapport à l'objectif défini. Lorsqu'un seul objectif est spécifié, par exemple un objectif de minimisation de la distance totale, la solution optimale est clairement définie : c'est celle qui a la plus petite distance. Pour les Problèmes Multi-

Objectifs (PMO) le concept de solution optimale devient alors plus difficile à définir. Dans ce cas, la solution recherchée n'est plus un point unique mais un ensemble de solutions dites de compromis aussi appelé ensemble des solutions *Pareto Optimales* (PO). La détermination ou l'approximation de l'ensemble PO n'est qu'une première phase possible dans la résolution pratique d'un PMO, la deuxième phase consistant à utiliser l'expérience du décideur afin d'approfondir la recherche dans une zone plus spécifique de l'ensemble PO.

Ce chapitre introduit, dans un premier temps, les concepts de base de l'optimisation combinatoire multi-objectifs (MOCO). Dans un deuxième temps, il présente les deux types de classification de méthodes de résolution de PMO. Les algorithmes évolutionnaires sont ensuite décrits comme une technique récente d'optimisation qui possède des caractéristiques intéressantes pour résoudre un PMO en portant un intérêt plus particulier sur les algorithmes génétiques. Outre les algorithmes évolutionnaires classiques, nous présentons une autre classe d'algorithmes, moins connus mais aussi inspirés de la biologie, les systèmes immunitaires artificiels. Un large éventail d'algorithmes évolutionnaires multi-objectifs de la littérature est ensuite présenté en essayant d'apporter un regard critique sur chacun d'eux. Finalement, la dernière partie de ce chapitre présente quelques mesures permettant d'évaluer les performances d'un algorithme multi-objectifs

3.2 Optimisation multi-objectifs

La difficulté principale d'un PMO est qu'il n'existe pas de définition de la solution optimale. Le décideur peut simplement exprimer le fait qu'une solution soit préférable à une

autre mais il n'existe généralement pas une unique solution meilleure que toutes les autres. Dès lors, résoudre un PMO ne consiste pas à rechercher la solution optimale mais l'ensemble des solutions satisfaisantes pour lesquelles on ne pourra pas effectuer une opération de classement. Les méthodes de résolution de PMO sont donc des méthodes d'aide à la décision car le choix final est laissé au décideur.

Un PMO peut être défini de la manière suivante dans un contexte de minimisation :

$$PMO \begin{cases} \min \; F(x)=\left(f_1(x),f_2(x),\ldots,f_{nobj}(x)\right) \\ s.c. \; x \in E \end{cases} \qquad (3.1)$$

où $nobj \geq 2$ est le nombre de fonctions objectifs, $x = (x_1,\ldots, x_n)$ est le vecteur représentant les variables de décision, E représente l'ensemble des solutions réalisables associées à des contraintes d'égalité, d'inégalité et des bornes explicites et $F(x) =(f_1(x), f_2(x),\ldots, f_{nobj}(x))$ est le vecteur des objectifs à minimiser.

3.2.1 La notion de dominance et optimum Pareto

Au XIX^{ième} siècle, Vilfredo Pareto, un mathématicien italien, formule le concept suivant : dans un PMO, il existe un équilibre tel que l'on ne peut pas améliorer un objectif sans détériorer au moins un des autres [Pareto 1896].

Cet équilibre a été appelé optimum de Pareto. Un point $x^* \in E$ (ensemble de solutions) est dit *Pareto optimal* s'il n'est dominé par aucun autre point appartenant à E. Autrement dit, un point $x^* \in E$ est Pareto optimal si et seulement s'il n'existe aucun autre point $x \in E$ tel que $f_{obj}(x^*) \leq f_{obj}(x)$ pour tout $obj = 1,\ldots, nobj$ et qu'il existe au moins un $obj2$ tel que

$f_{obj2}(x^*) < f_{obj2}(x)$. Ces points sont également appelés solutions *non inférieures* ou *non dominées*. L'ensemble de tous les points Pareto optimaux constitue la *frontière Pareto*.

Plusieurs autres relations de dominance ont été présentées comme la *a-dominance* [Othomani 1998], la *dominance au sens de Geoffrion* [Ehrgott 2000] ou encore la *cône-dominance* [Collette et Siarry 2002]. Toutefois, dans cette thèse, nous utilisons uniquement la dominance au sens Pareto à partir du concept d'optimum Pareto.

À la Figure 3.1, les solutions représentées par des ronds gris foncés représentent les solutions Pareto optimales. L'ensemble de ces solutions reliées par la ligne en pointillés constitue la frontière Pareto. Les solutions représentées par des ronds blancs représentent l'ensemble des solutions dominées.

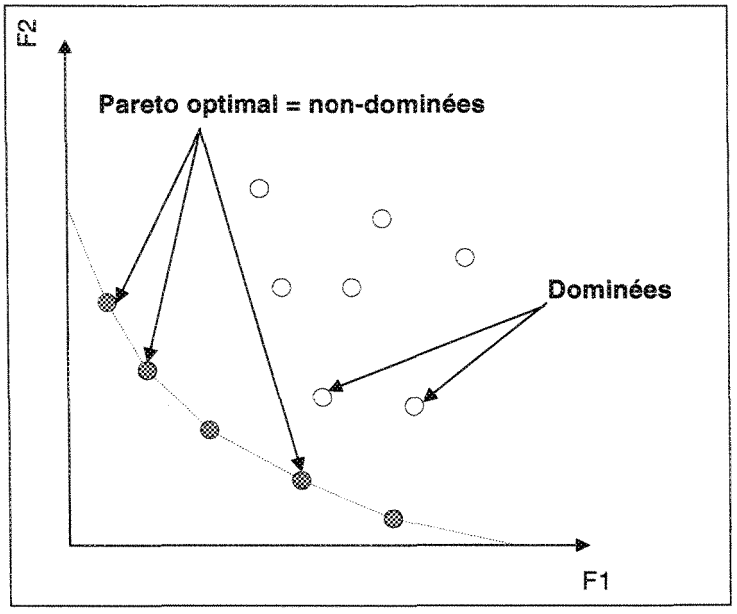


Figure 3.1 : Illustration du concept d'optimum Pareto [Zitzler 2001]

3.2.2 Classification

Dans les différentes publications, nous rencontrons deux classifications différentes des méthodes de résolution de PMO. Le premier classement adopte le point de vue du décideur en classant les méthodes en fonction du moment où l'information sur les préférences du décideur est demandée. Le deuxième classement, plus théorique, trie les méthodes en fonction de leur façon de traiter les fonctions objectifs [Berro 2001].

Dans la *classification décideur*, en fonction du moment où l'information sur les préférences du décideur est demandée, les méthodes de résolution de PMO peuvent être regroupées en trois catégories [Hwang et Masud 1980] : *a priori*, *a posteriori* et *interactive*.

Dans les méthodes *a priori*, l'information sur les préférences du décideur est requise avant la résolution du problème. Dans cette catégorie, les méthodes utilisées pour résoudre les PMO consistent souvent à combiner les différentes fonctions objectif en une fonction d'utilité suivant les préférences du décideur. Dans ce cas, le décideur est supposé connaître *a priori* le poids ou l'importance de chaque objectif afin de les combiner dans une fonction unique. Cela revient à résoudre un problème à objectif unique. Cependant, dans la plupart des cas, le décideur ne peut pas exprimer clairement le poids à attribuer aux différents objectifs, soit par manque d'expérience ou d'informations, soit parce que les différents objectifs sont non commensurables. Des objectifs sont non commensurables si leurs valeurs sont exprimées dans des unités différentes et difficilement comparables.

Dans les méthodes *a posteriori*, la recherche de solutions est effectuée sans qu'aucune information sur les préférences ne soit fournie. Le résultat du processus de recherche est l'ensemble PO ou son approximation à partir duquel le décideur choisira la solution la plus

satisfaisante selon ses préférences. Autrement dit, les compromis sont faits par le décideur après la génération de l'ensemble des solutions non dominées. Si cette approche évite certains inconvénients de l'approche *a priori*, elle exclut l'articulation des préférences par le décideur qui permet de réduire la complexité de l'espace de recherche. Ce type d'approche n'est donc utilisable que dans le cas où la cardinalité de l'ensemble PO est réduite.

Dans les méthodes *interactives*, il y a coopération progressive entre le décideur et le système. Les processus de décision et d'optimisation sont alternés. Par moment, le décideur intervient de manière à spécifier davantage d'informations sur ses préférences. Ces dernières sont alors prises en compte par le système pour la résolution du problème. Le décideur modifie ainsi le compromis entre ses préférences et les résultats. Cette technique permet de faire une exploration guidée de l'ensemble PO. Cependant, il n'est pas garanti que la solution finale soit obtenue après un nombre fini d'itérations.

La *classification concepteur* qui est utilisée dans le reste de la thèse adopte un point de vue plus théorique articulé autour des notions d'agrégation et d'optimum Pareto. Selon cette classification, les approches utilisées pour la résolution de PMO peuvent être regroupées dans les trois catégories suivantes [Talbi 1999] : *les approches basées sur la transformation du problème en un problème uni-objectif*, *les approches non Pareto* et *les approches Pareto*.

L'ensemble des méthodes de la première catégorie repose sur l'axiome suivant : tout décideur essaie inconsciemment de maximiser une fonction d'utilité U [Berro 2001] :

$$U = U(f_1, f_2, \ldots, f_{nobj}). \qquad (3.2)$$

Dans cette catégorie d'approches, les modèles les plus utilisés sont le modèle additif :

$$U = \sum_{obj=1}^{nobj} U_{obj}(f_{obj}) \qquad (3.3)$$

et le modèle multiplicatif :

$$U = \prod_{obj=1}^{nobj} U_{obj}(f_{obj}) \qquad (3.4)$$

Cependant, l'utilisation de ces modèles impose généralement que les objectifs soient commensurables. Il est donc très difficile d'utiliser ces techniques lorsque l'ensemble des objectifs est composé à la fois d'objectifs qualitatifs et quantitatifs. Parmi les techniques représentatives de cette classe, il y a la *moyenne pondérée* et la *programmation par but.*

La technique de la moyenne pondérée consiste à additionner tous les objectifs en affectant à chacun d'eux un coefficient de poids. Ce coefficient représente l'importance relative que le décideur attribue à l'objectif. Cela modifie un PMO en un problème uni-objectif de la forme :

$$\min \sum_{obj=1}^{nobj} w_{obj} f_{obj}(x) \qquad (3.5)$$

où w_{obj} représente le poids affecté à l'objectif *obj*, $w_{obj} \geq 0$ et $\sum_{obj=1}^{nobj} w_{obj} = 1$. Si ce genre de méthode est facile à mettre en œuvre, la détermination des poids s'avère être une question délicate qui détermine l'efficacité de la méthode. De plus, cette méthode est généralement sensible à la forme de la frontière Pareto [Francisci 2002].

Dans la programmation par but, le décideur fixe un but G_{obj} à atteindre pour chaque objectif *obj* [Charnes et Cooper 1961]. Ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires. La nouvelle fonction objectif est alors modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre :

$$\min \sum_{obj=1}^{nobj} | f_{obj}(x) - G_{obj} | \qquad (3.6)$$

Différentes variantes et applications de cette technique ont été proposées [Ignizio 1981; Van Veldhuizen 1999]. Comme pour la somme pondérée, cette méthode est facile à mettre en œuvre. De plus, elle fournit un résultat même si un mauvais choix initial a conduit le décideur à donner un ou plusieurs buts T_i non réalisables. Cependant, l'efficacité de la méthode dépend fortement de la définition des buts à atteindre [Berro 2001].

Les *approches non Pareto*, de leur côté, ne transforment pas le PMO en un problème uni-objectif et n'utilisent pas les concepts de dominance ou d'optimum Pareto. En général, cette classe de méthodes possède un processus de recherche qui traite séparément les différents objectifs. Dans cette catégorie, une des techniques les plus répandues est la *sélection lexicographique*.

La sélection lexicographique a été introduite par Fourman [1985]. Dans cette technique, les objectifs sont préalablement rangés par ordre d'importance par le décideur. Ensuite, l'optimum est obtenu en minimisant tout d'abord la fonction objectif la plus importante puis la deuxième, et ainsi de suite sans jamais détériorer les objectifs précédents. Le risque essentiel de cette méthode est la grande importance attribuée aux objectifs classés en premier. La meilleure solution f_1* trouvée pour l'objectif le plus important risque de faire

converger l'algorithme vers une zone restreinte de l'espace des solutions et enfermer les points dans cette région [Berro 2001].

La dernière catégorie, les *approches Pareto,* utilise directement la notion d'optimum Pareto dans leur processus de recherche. Le processus de sélection des solutions générées est basé sur la notion de non-dominance. Le principal avantage de ces méthodes est qu'elles peuvent générer des solutions PO dans toutes les régions de la frontière Pareto. Certaines techniques de résolution appartenant à cette catégorie sont présentées plus en détail dans la Section 3.6 du présent chapitre.

Classiquement, les PMO sont résolus à l'aide de techniques comme la moyenne pondérée ou la programmation par but car ces techniques permettent souvent d'utiliser des algorithmes éprouvés pour des problèmes uni-objectif [Francisci 2002]. Cependant, Deb [1999] a montré que certaines de ces techniques avaient un champ d'applications limité. De plus, les méthodes classiques ont toutes en commun qu'elles requièrent généralement plusieurs étapes d'optimisation pour obtenir une approximation de l'ensemble PO [Francisci 2002].

Récemment, les algorithmes évolutionnaires se sont révélés être une alternative intéressante aux méthodes classiques grâce à leur faculté à exploiter de vastes espaces de recherche et à générer des compromis multiples en une seule étape d'optimisation [Francisci 2002]. De plus, les algorithmes évolutionnaires sont peu sensibles à la forme de la frontière Pareto. Les algorithmes évolutionnaires font partie des méthodes métaheuristiques. Ce sont des techniques d'approximation conçues dans le but de s'attaquer à des problèmes d'optimisation complexes qui n'ont pu être résolus de façon efficace par

les heuristiques et les méthodes d'optimisation classique [Osman et Laporte 1996]. Parmi les métaheuristiques les plus utilisées, citons le recuit simulé [Kirkpatrick *et al.* 1983], la recherche avec tabous [Glover 1986], l'optimisation par colonie de fourmis [Dorigo 1992] et bien entendu les algorithmes génétiques [Holland 1975] qui font partie de la famille des algorithmes évolutionnaires dont les principes de base sont brièvement décrits à la section suivante.

3.3 Les Algorithmes Évolutionnaires (AE)

Les algorithmes évolutionnaires sont des méthodes qui tentent de simuler le processus de l'évolution naturelle dans un environnement hostile lié au problème à résoudre. Schématiquement, on peut les assimiler à un processus d'optimisation où des individus évoluent dans le temps, afin de devenir de plus en plus adapté à un environnement donné : le problème à résoudre. Les AE se sont avérés être des alternatives efficaces aux méthodes classiques pour la résolution de nombreux problèmes d'optimisation uni-objectifs et multi-objectifs. En général, les algorithmes évolutionnaires sont divisés en quatre grandes classes :

- les algorithmes génétiques [Holland 1975; Goldberg 1989];
- les stratégies d'évolution [Rechenberg 1965; Schewfel 1981];
- la programmation évolutive [Fogel *et al.* 1966]; et
- la programmation génétique [Koza 1992].

Cette section présente brièvement les quatre classes d'algorithmes évolutionnaires en insistant plus particulièrement sur les algorithmes génétiques.

3.3.1 Les Algorithmes Génétiques (AG)

Les algorithmes génétiques tentent de simuler le processus de sélection naturelle dans un environnement hostile lié au problème à résoudre en s'inspirant des théories de l'évolution proposées par Charles Darwin. Les AG sont à la base des algorithmes d'optimisation stochastiques, mais ils peuvent également servir pour l'apprentissage automatique. Les premiers travaux dans ce domaine ont commencé dans les années cinquante lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Entre 1960 et 1970, John Holland, sur la base des travaux précédents, développe les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique [Holland 1975]. Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation des AG sur des problèmes réels de grande taille. La parution en 1989 de l'ouvrage de référence écrit par Goldberg qui décrit l'utilisation de ces algorithmes dans le cadre de résolution de problèmes concrets a permis de mieux faire connaître ces derniers dans la communauté scientifique et a marqué le début d'un nouvel intérêt pour cette technique d'optimisation qui demeure néanmoins très récente. La Figure 3.2 illustre le fonctionnement général d'un algorithme génétique standard.

Générer aléatoirement une population *Pop* de |*Pop*| individus
TANT QUE Nombre de générations ≤ Nombre de générations Maximum **FAIRE**
Évaluation : Calculer la fitness de chaque individu x
Sélection : Établir de façon probabiliste les paires d'individus qui vont se reproduire en accordant une plus grande chance aux meilleurs individus
Croisement : Recombiner les paires sélectionnées en fonction de p_c .
Mutation : Effectuer une mutation en fonction de p_m
Remplacement : Les nouveaux individus produits représentent la nouvelle population

Figure 3.2 : Un algorithme génétique standard

Le but principal d'un AG est de chercher, dans un espace de solutions E, un élément de cet espace ayant la meilleure adaptation à l'environnement posé. Chaque élément de E est un individu, noté x. Une population *Pop* est donc un ensemble de $|Pop|$ éléments (individus) de E : $Pop = (x_1, x_2, \ldots, x_{|Pop|})$. La mesure du degré d'adaptation de chaque individu constitue son indice de qualité f (*fitness*). L'objectif d'un AG est de faire évoluer cette population *Pop* afin de trouver le meilleur individu x^*. Dans ce but, à chaque génération t, les individus de la population $Pop(t)$ sont sélectionnés, croisés et mutés selon des probabilités prédéfinies respectivement p_c (probabilité de croisement) et p_m (probabilité de mutation).

Bien que les principes sous-jacents soient simples, ces algorithmes s'avèrent être des mécanismes de recherche généraux, puissants et robustes [Francisci 2002]. De plus, les AG semblent être spécialement utiles en optimisation multi-objectifs car ils sont capables de déterminer plusieurs solutions en une seule exécution. Certains auteurs suggèrent même que l'optimisation multi-objectifs peut être un domaine où les AG font mieux que d'autres méthodes de recherche heuristique [Fonseca et Fleming 1995; Valenzuela-Rendon et Uresti-Charre 1997].

La suite de cette section présente sommairement le fonctionnement d'un AG. Cette présentation permettra d'introduire certaines notions qui seront utiles pour la compréhension du reste de ce document.

3.3.1.1 Représentation des individus

Dans la nature, les structures géniques sont codées en base 4, dont les « chiffres » sont les quatre bases azotées : l'*adénine*, la *thymine*, la *cytosine* et la *guanine*. Dans le cadre des

AG, ce type de codage est bien difficile à utiliser et n'a donc pas été retenu. Nous présentons ici deux des représentations utilisées pour le codage des individus dans un AG à savoir la représentation *binaire naturelle* et la représentation *par chemin*.

Historiquement, la représentation binaire est la première représentation utilisée par les AG. Chaque individu x est représenté sous forme de chaînes de bits où chaque élément prend la valeur 0 ou 1 :

$$x = (e_1, e_2, \ldots, e_L) \in \{0,1\}^L, \qquad (3.7)$$

Où L est la taille du vecteur (nombre de bits). L'espace de recherche est alors $\{0,1\}^L$. Ce type de représentation possède plusieurs avantages : alphabet minimum, facilité de mise en place d'opérateurs génétiques et existence de résultats théoriques [Goldberg 1989]. Néanmoins, cette représentation présente deux inconvénients majeurs :

- les performances de l'algorithme sont diminuées lorsque la longueur de la chaîne augmente; et
- deux éléments voisins en terme de distance de Hamming (représente le nombre de bits dont diffèrent deux nombres binaires) ne codent pas nécessairement deux éléments proches dans l'espace de recherche.

La représentation par chemin est une manière naturelle de coder une solution d'un Problème de Voyageur de Commerce (PVC) [Potvin 1996]. Chaque individu x est représenté sous forme de chaînes de symboles. Deux symboles successifs d'un individu x représentent deux numéros de villes adjacentes dans le chemin proposé. Pour être valide, un individu doit comporter chaque symbole (chaque ville à visiter dans le cas d'un PVC) et ce, une et une seule fois.

3.3.1.2 La sélection

La sélection est un des principaux opérateurs utilisés par les AG. Son rôle est de sélectionner les individus relativement performants afin qu'ils se reproduisent. Plusieurs stratégies de sélection ont été mises en place et les paragraphes suivants décrivent deux des plus répandues : *la roulette* et *le tournoi.*

La première sélection mise en place pour les AG est le *tirage à la roulette* (*Roulette Wheel Selection* (RWS)) introduite par Goldberg [1989]. C'est une méthode stochastique qui exploite la métaphore d'une roulette de casino. Elle consiste à associer à chaque individu un segment (ou case de la roue) dont la longueur est proportionnelle à sa fitness comme l'illustre la Figure 3.3. La roue étant lancée, l'individu sélectionné est celui sur lequel la roue s'est arrêtée. Cette méthode favorise les meilleurs individus, mais tous les individus conservent néanmoins des chances d'être sélectionnés.

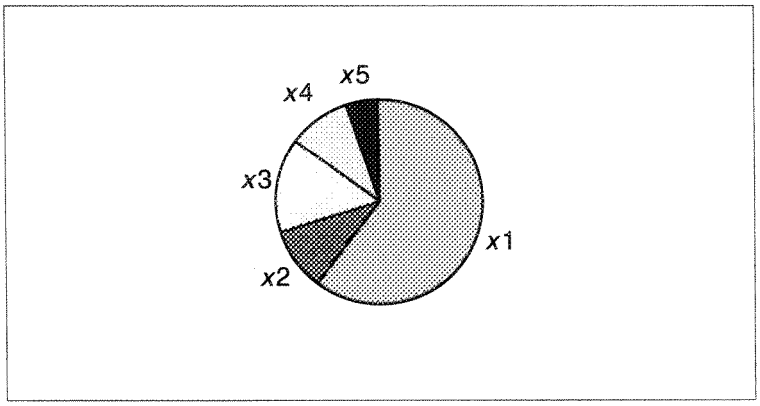


Figure 3.3 : Roulette pour une population de 5 individus avec $f(x_i) = \{60,10, 15,10, 5\}$.

De son côté, *la sélection par tournoi* fait intervenir le concept de comparaison entre les individus. Son principe est de choisir un groupe de q individus aléatoirement dans la

population et de sélectionner d'une manière déterministe le meilleur dans ce groupe. Au cours d'une génération, il y a autant de tournois que d'individus à sélectionner. La pression de sélection est ajustée par le nombre de participants q à un tournoi. Un q élevé a une forte pression de sélection et inversement. L'avantage de cette technique de sélection est qu'elle n'est pas coûteuse à mettre en œuvre et à exécuter.

3.3.1.3 Le croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants mais il est tout à fait possible d'imaginer des croisements avec α parents pour produire β enfants. Un croisement consiste à échanger les gènes des parents afin de donner des enfants qui comportent des propriétés combinées. Bien qu'il soit souvent aléatoire, cet échange d'informations offre aux algorithmes génétiques une part de leur puissance : quelquefois, de *bons* gènes d'un parent viennent remplacer les *mauvais* gènes d'un autre et créent des enfants mieux adaptés que les parents. Dans le paragraphe suivant, nous allons présenter le *croisement à un point* qui est une des premières stratégies de croisement utilisées par les AG [Holland 1962].

Le but de cette stratégie de croisement est d'échanger un fragment de gènes entre deux individus. Le croisement à un point consiste à choisir au hasard un point de croisement pour chaque couple d'individus, et ensuite à échanger un fragment de gènes entre les deux individus afin de créer deux enfants. À la Figure 3.4, le point de coupure entre les deux parents se situe à la position 5. À partir du premier parent, le premier enfant est obtenu en copiant les gènes du deuxième parent à droite du point de coupure. Le deuxième enfant est

obtenu à partir du deuxième parent en copiant les gènes du premier parent à droite du point de coupure.

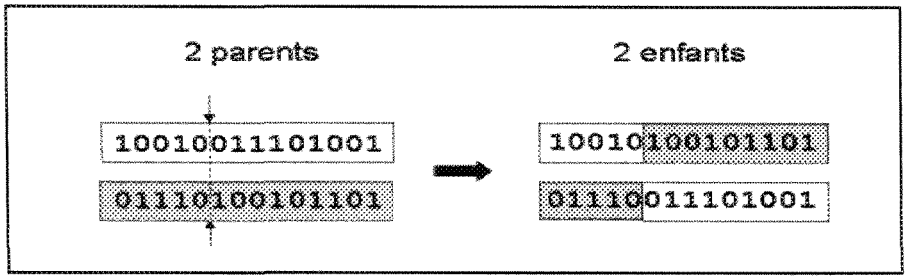


Figure 3.4 : Illustration du croisement à un point

3.3.1.4 La mutation

L'opérateur de mutation joue le rôle de bruit et tente d'empêcher une convergence trop hâtive. Plusieurs opérateurs de mutation ont été mis en place, parmi lesquels l'inversion et l'échange. La mutation par inversion est souvent utilisée avec la représentation binaire [Ben Hamida 2001]. Elle consiste simplement à inverser de manière aléatoire un gène d'un individu. À la Figure 3.5, la position 9 est sélectionnée et le gène est inversé, il passe de 1 à 0.

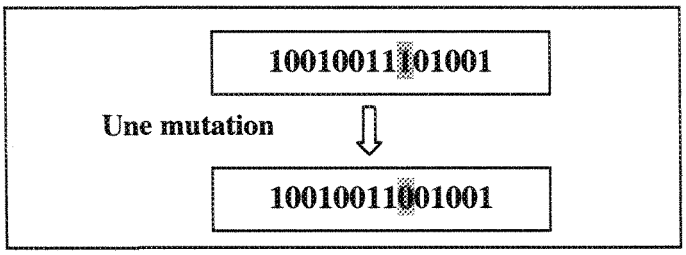


Figure 3.5 : Illustration de la mutation par inversion

La mutation par échange consiste à sélectionner de manière aléatoire deux gènes d'un individu et d'échanger les positions respectives des deux éléments choisis. À la Figure 3.6, les gènes aux positions 4 et 6 sont échangés.

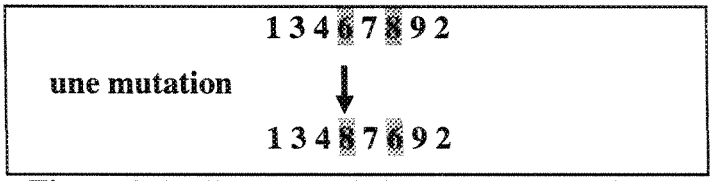


Figure 3.6 : Illustration de la mutation par échange

3.3.2 Les stratégies d'évolution

Les stratégies d'évolution constituent une technique d'AE développée par Rechenberg et Schwefel au milieu des années 1960. Ces techniques représentent chaque individu comme un ensemble de caractéristiques de la solution potentielle. En général, cet ensemble prend la forme d'un vecteur de nombres réels de dimension fixe. Les stratégies d'évolution utilisent une population de parents, de taille μ, à partir de laquelle des individus sont sélectionnés aléatoirement afin de générer une population d'enfants, de taille $\lambda \geq \mu$. Ceux-ci sont ensuite modifiés à l'aide des mutations qui consistent simplement à ajouter une valeur générée aléatoirement selon une fonction de densité de probabilité. Les opérateurs de mutation utilisé dans les stratégies d'évolution sont souvent les mêmes que ceux définis pour les algorithmes génétiques comme par exemple la mutation par échange ou la mutation par inversion présentées à la section précédente. Les paramètres de la fonction de densité de probabilité, nommés les *paramètres de la stratégie*, évoluent eux aussi dans le temps selon les mêmes principes que les paramètres caractérisant les individus. Pour former la nouvelle population parents, deux stratégies peuvent être employées :

- la première stratégie, l'approche (μ, λ), consiste à choisir les μ meilleurs individus parmi les λ enfants;

- la deuxième technique, l'approche (μ + λ), consiste à retenir les μ meilleurs individus dans les populations parents et enfants combinées.

3.3.3 La programmation évolutive

La programmation évolutive a été initialement conçue pour des machines à états finis et a été par la suite étendue aux problèmes d'optimisation [Fogel *et al.* 1966]. Cette approche met l'emphase sur la relation entre les parents et leurs descendants plutôt que de simuler des opérateurs génétiques d'inspiration naturelle. Contrairement aux autres AE classiques, la programmation évolutive n'utilise pas une représentation spécifique mais plutôt un *modèle évolutionnaire* jumelé à une représentation des solutions et à un opérateur de mutation choisi en fonction du problème à résoudre.

Pour effectuer la programmation évolutive, une population de μ solutions potentielles au problème est d'abord générée de façon aléatoire. Chaque individu x de la population ainsi produite génère par la suite λ enfants par mutation. Une opération de sélection naturelle est alors appliquée afin de former une nouvelle population de μ individus. Le processus de mutation/sélection est répété de manière itérative jusqu'à ce qu'une solution acceptable soit trouvée.

3.3.4 La programmation génétique

La programmation génétique [Koza 1992] est un paradigme permettant la programmation automatique d'ordinateurs par des heuristiques basées sur les mêmes principes d'évolution que les AG à savoir : sélection, croisement et mutation. La différence principale entre ces deux AE se situe au niveau de la représentation des individus. En effet,

dans la programmation génétique on fait évoluer des individus dont la structure est similaire à celle de programmes informatiques.

Initialement, la programmation génétique représente les individus sous forme d'arbres, c'est-à-dire des graphes orientés et sans cycle, dans lesquels chaque nœud est associé à une opération élémentaire relative au domaine du problème. Plusieurs autres représentations comme des programmes linéaires [Banzhaf *et al.* 1998] et des graphes cycliques [Teller et Veloso 1996], ont été par la suite utilisées. La programmation génétique s'avère particulièrement adaptée pour faire évoluer des structures complexes de dimension variable.

3.3.5 AE et optimisation multi-objectifs

C'est dans le milieu des années 80 [Schaffer 1985] que, pour la première fois, les AE ont été utilisés pour l'optimisation multi-objectifs. Depuis, plusieurs implémentations différentes des AE ont été proposées et appliquées avec succès à divers PMO [Hajela et Lin 1992; Fonseca et Fleming 1993; Horn *et al.* 1994; Srivinas et Deb 1994; Ishibuchi et Murata 1996; Valenzuela-Rendon et Uresti-Charre 1997]. Récemment, certains chercheurs ont étudié quelques aspects des AE pour l'optimisation multi-objectifs comme la convergence vers la frontière Pareto [Rudolph 1998] ou l'*élitisme* [Obayashi *et al.* 1998; Parks et Miller 1998]. Parallèlement, de nouvelles techniques qui exploitent la *notion de dominance* au sens Pareto ont été développées [Zitzler et Thiele 1998; Knowles et Corne 2000; Coello Coello et Pulido 2001].

3.3.5.1 Maintenir la diversité : les techniques de nichage

Une des principales difficultés à surmonter lorsque l'on veut utiliser un AE et en particulier un AG, pour l'optimisation multi-objectifs est le maintien de la diversité. En effet, un algorithme génétique simple a, en général, tendance à converger vers une solution unique en négligeant souvent les autres solutions [Mahfoud 1995]. Dans un contexte multi-objectifs, on ne cherche pas une mais un ensemble de solutions. Il est donc essentiel de préserver une certaine diversité entre les différentes solutions trouvées à chaque itération par un AG multi-objectifs. Les techniques de nichage ont été développées en ce sens. Ces techniques ont pour objectif la formation et le maintien de sous-populations stables appelées *niches* dans une même population. Les paragraphes suivants décrivent deux des principales techniques de nichage : le *partage de fitness* et la *méthode de remplacement.*

Le *partage de fitness* est une technique introduite par Goldberg et Richardson [1987]. Dans cette méthode, la fitness d'un individu est dégradée en fonction du nombre d'individus qui lui sont proches dans la population. Le but principal de cette méthode est de pénaliser les individus qui sont trop proches les uns des autres en terme de distance. Ainsi, la fitness partagée f' d'un individu x est donnée par :

$$f'(x) = \frac{f(x)}{\sum_{\forall\, y \in Pop} sh\,(dist\ \ (x, y))} \qquad (3.8)$$

où *sh* désigne la fonction de partage qui a comme paramètre d'entrée la distance *(dist)* entre deux solutions. Elle retourne 1 si les deux solutions sont identiques, 0 si la distance entre les deux dépasse un certain seuil (σ_{sh}). La somme des valeurs des fonctions de partage est

appelée *compte de niche*. Une des fonctions de partage les plus utilisées est celle proposée par Goldberg et Richardson [1987] :

$$sh(dis)=\begin{cases} 1-\left(\dfrac{dis}{\sigma_{sh}}\right)^{\alpha} & si\ dis(x,y)<\sigma_{sh} \\ 0 & \text{sinon} \end{cases} \qquad (3.9)$$

où α est une constante de contrôle de la fonction de partage et σ_{sh} représente le rayon de la niche. Le calcul de la distance *(dist)* peut se faire à l'aide d'une distance de *Hamming* ou d'une distance *Euclidienne*.

La *méthode de remplacement* a été introduite par De Jong [1975]. Le principe originel consiste à remplacer des éléments d'une population par d'autres éléments similaires et meilleurs qu'eux. Cette méthode est une analogie des systèmes vivants en compétition pour une ressource. Dans les problèmes d'optimisation, la ressource est l'optimum à atteindre. Des individus situés dans des zones différentes de l'espace des solutions ne sont pas en concurrence pour le même optimum, alors que des individus proches sont en compétition. En remplaçant les individus semblables, la méthode de remplacement permet de conserver les différentes niches de la population tout en introduisant de la diversité dans la population. Cependant, Mafhoud [1995] a prouvé que bien qu'une telle méthode permette d'obtenir une certaine diversité dans la population, elle s'avère peu efficace pour bien approximer la frontière Pareto. Ceci explique, en partie, pourquoi cette technique est moins utilisée dans les AE que le partage de fitness [Blickle et Thiele 1996].

3.4 Les Systèmes Immunitaires Artificiels (SIA)

Le terme système immunitaire artificiel (*Artificial Immune System*) s'applique, en général, aux algorithmes inspirés du fonctionnement du système immunitaire des organismes vertébrés. Même si cette classe d'algorithmes est moins connus et répandus que les AE, un nombre croissant de systèmes immunitaires artificiels ont été proposés ces dernières années pour être appliqués dans différents domaines, tels que la robotique, la classification, la détection d'anomalies ou encore l'optimisation [De Castro et Von Zuben 2000].

D'un point de vue biologique, le système immunitaire est responsable de la protection de l'organisme contre les agressions extérieures. Pour cela, les cellules de l'organisme possèdent sur leur membrane un certain nombre de molécules spécifiques appelées *antigènes*. L'ensemble des antigènes présents sur les cellules constituent un identifiant unique pour chaque organisme. Dans le système immunitaire des organismes vertébrés, les *lymphocytes* sont des cellules qui possèdent des récepteurs capables de se lier spécifiquement à un antigène donné, permettant ainsi de reconnaître une cellule étrangère à l'organisme. Lorsqu'un lymphocyte reconnaît une cellule étrangère, il va être stimulé puis proliférer avant de se différencier en cellules permettant de reconnaître l'antigène ciblé, ou en cellule permettant de combattre les agressions. Dans le premier cas, l'organisme sera capable de réagir plus rapidement à une nouvelle exposition à l'antigène, c'est le principe de des vaccins. Dans le second cas, l'organisme combattra les agressions en secrétant des *anticorps*. Ces derniers sont en fait des molécules réceptrices permettant de reconnaître l'antigène et de le bloquer. Le processus de prolifération des lymphocytes se fait par la

production de *clones*, tandis que la différentiation, quant à elle, se fait par un mécanisme *hyper-mutation* des cellules clonées. C'est cette métaphore qui est utilisée dans les SIA, en particulier le processus de sélection par clonage et de rétroaction permettant la multiplication et la mémoire du système. Certains auteurs considèrent l'approche utilisée dans les SIA très proche de celle d'un AE ou d'un réseau de neurones [Dréo et Siarry 2003; Coello Coello et Cortés 2005]. L'algorithme 3.1 résume le fonctionnement général d'un SIA simple.

Algorithme 3.1 : un SIA standard [Dréo et Siarry 2003]

1: Générer un ensemble de solutions POP, composé d'un ensemble de cellules mémoires POP_M ajoutées à la population courante POP_t : $POP = POP_M + POP_t$
2 : Déterminer les n meilleures cellules de POP en utilisant un critère d'affinité
3 : Cloner les n individus sélectionnés pour former une population de clones C_{pop}. Le nombre de clones produits pour chaque individu est fonction de l'affinité;
4 : Effectuer une hyper mutation des clones produit afin d'obtenir une population de clones mutés C^*_{POP}. La mutation est proportionnelle à l'affinité ;
5 : Sélectionner les individus de C^*_{POP} pour former la nouvelle population mémoire POP_M;
6 : Éliminer les plus mauvais individus de POP pour former POP_{t+1} ;
7 : Si aucun critère d'arrêt n'est pas atteint, retourner en 1

En se basant sur l'Algorithme 3.1, on constate qu'on peut, dans le cadre de l'optimisation combinatoire, considérer les SIA comme une forme d'algorithme évolutionnaire utilisant des opérateurs particuliers. Par exemple, pour effectuer la sélection, on se fonde principalement sur une mesure d'affinité entre le récepteur d'un anticorps et d'un antigène; la mutation s'opère pour sa part via un opérateur d'hyper mutation issu de la métaphore immune [Dréo et Siarry 2003]. Une description détaillée des fondements théoriques des SIA peut être trouvée dans [Dasgupta et Attoh-Okine 1997; De Castro et Von Zuben 1999; De Castro et Von Zuben 2000]. Les SIA ont été surtout appliqués dans

les domaines de l'extraction de connaissances [Timmis et Knight 2001], de l'extraction de règles [Carvalho et Freitas 2001], du clustering [De Castro et Von Zuben 2000; Nasraoui *et al.* 2002], de détection de fraudes [Lee *et al.* 2000; Overill 2002] ou encore de classification [Carter 2000; Twycross et Cayzer 2002]. On retrouve aussi des SIA pour l'optimisation multimodale [Forrest et Perelson 1991; Smith *et al.* 1992; Smith *et al.* 1993; de Castro et Von Zuben 2002] et l'optimisation uni-objectif [de Castro et Von Zuben 2002]. Toutefois, malgré certaines similitudes entre les AE et les SIA, on note que contrairement aux AE, les SIA ont très peu été appliqués pour l'optimisation multi-objectifs [de Castro et Von Zuben 2002; Coello Coello et Cortés 2005].

La section suivante présente un éventail des méthodes de résolution de PMO basées sur les AE en les regroupant selon la *classification concepteur* décrite à la Section 3.2.2.

3.5 AE basés sur la transformation du problème en un problème uni-objectif

Les implémentations des AG de cette catégorie d'approche sont basées sur les techniques classiques pour générer des compromis sur l'ensemble PO. Certaines approches comme le *Weight-Based Genetic Algorithm* (*WBGA*) [Hajela et Lin 1992] ou le *Multi-Objective Genetic Local Search* (*MOGLS*) [Ishibuchi et Murata 1996] utilisent la technique de la moyenne pondérée. Comme chaque individu utilise une combinaison particulière de poids qui est soit choisie aléatoirement, soit encodée dans l'individu, tous les membres de la population sont évalués par différentes fonctions objectifs. En conséquence, l'optimisation est effectuée dans plusieurs directions simultanément [Francisci 2002].

Cependant, les inconvénients potentiels des techniques classiques, comme la sensibilité à la forme de la frontière Pareto pour certaines, peuvent restreindre l'efficacité de ce type d'AG [Van Veldhuizen 1999].

3.6 AE non Pareto

En général, les méthodes dites non Pareto possèdent un processus de recherche qui traite séparément les objectifs. Cette section présente un algorithme génétique représentatif de cette classe d'approche : le *Vector Evaluated Genetic Algorithm* (VEGA).

En 1985, Schaffer propose une extension d'un algorithme génétique simple pour la résolution d'un PMO [Schaffer 1985]. Cette méthode est appelée *Vector Evaluated Genetic Algorithm*. La seule différence avec un algorithme génétique simple est la manière dont s'effectue la sélection. Comme le montre la Figure 3.7, l'idée est simple. Pour un problème comportant *nobj* objectifs et une population de $|POP|$ individus, une sélection de $|POP|/nobj$ individus est effectuée pour chaque objectif. Ainsi, *nobj* sous-populations vont être créées, chacune d'entre elles contenant les $|POP|/nobj$ meilleurs individus pour un objectif particulier. Les *nobj* sous-populations sont ensuite mélangées afin d'obtenir une nouvelle population de taille $|POP|$. Le processus de sélection se termine par l'application des opérateurs génétiques de croisement et de mutation.

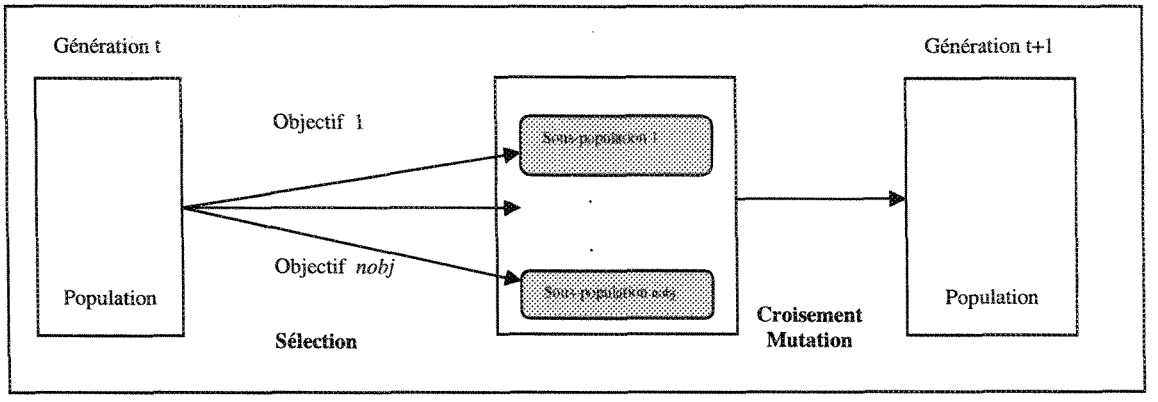


Figure 3.7 : Sélection parallèle dans l'algorithme VEGA [Talbi 1999]

Le VEGA est l'un des premiers AG multi-objectifs. Cette technique est très facilement implémentable dans un AG classique et ceci explique sans doute pourquoi elle est encore très souvent utilisée [Berro 2001]. Cependant, avec une telle sélection, les points se répartissent sur les extrêmes de la frontière Pareto au détriment des solutions de *compromis* qui ont une fitness générale acceptable mais ne possèdent aucun objectif fort [Talbi 1999].

3.7 AE Pareto

L'idée d'utiliser la dominance au sens de Pareto dans un AE a été proposée par Goldberg [1989] pour résoudre les problèmes proposés par Schaffer [1985]. L'utilisation d'une sélection basée sur la notion de dominance de Pareto va faire converger la population vers un ensemble de solutions efficaces. Ce concept ne permet pas de choisir une alternative plutôt qu'une autre, mais il apporte une aide précieuse au décideur.

Les paragraphes suivants présentent des techniques inspirées des algorithmes évolutionnaires qui utilisent cette notion. Cette présentation est divisée en deux parties : dans un premier temps, nous allons nous intéresser aux techniques non élitistes et, dans un deuxième temps, nous aborderons les techniques élitistes.

3.7.2 Les techniques non élitistes

Une méthode de résolution est dite non élitiste lorsqu'elle ne comporte aucun mécanisme explicite permettant la conservation des meilleures solutions tout au long de son exécution.

3.7.2.1 Multiple Objective Genetic Algorithm (MOGA)

Proposé par Fonseca et Fleming [1993], le MOGA est une méthode dans laquelle chaque individu de la population est rangé en fonction du nombre d'individus qui le domine. Ensuite, l'algorithme utilise une fonction de calcul de la fitness permettant de prendre en compte le rang de l'individu et le nombre d'individus ayant le même rang.

Soit un individu x à la génération t, dominé par $\alpha(t)$ individus. Le rang de cet individu est :

$$\text{rang}(x, t) = 1 + \alpha(t) \qquad (3.10)$$

La procédure de sélection utilise ensuite ces rangs pour sélectionner ou éliminer des blocs d'individus. De plus, les auteurs calculent la fitness des individus de la façon suivante :

1. calcul du rang de chaque individu; et
2. affectation de la fitness de chaque individu par application d'une fonction de changement d'échelle sur la valeur de son rang. Cette fonction est en général linéaire. Suivant le problème, d'autres types de fonction pourront cependant être

envisagés afin d'augmenter ou de diminuer l'importance des meilleurs rangs ou d'atténuer la largeur de l'espace entre les individus de plus fort rang et ceux de plus bas rang.

Cette méthode pose comme inconvénient un risque de convergence prématurée à cause de la grande pression exercée par la sélection. Pour éviter ce problème, les auteurs ont introduit une fonction de partage de fitness afin de mieux répartir les solutions le long de la frontière Pareto. Les performances de l'algorithme demeurent toutefois dépendantes de la valeur du paramètre σ_{sh} utilisé dans la fonction de partage.

3.7.2.2 Niched Pareto Genetic Algorithm (NPGA)

Cette méthode proposée par Horn et Napfliotis [1994] utilise une sélection par tournoi basée sur la notion de dominance de Pareto. Elle compare deux individus pris au hasard avec une sous-population de taille t_{dom} également choisie au hasard. Si un de ces deux individus est non dominé par le sous-groupe et l'autre non, il est alors sélectionné. Lorsque les deux individus sont soit non dominés soit dominés par le sous-groupe, une fonction de partage de fitness est appliquée pour déterminer celui qui sera sélectionné. Le paramètre t_{dom} permet d'exercer une pression variable sur la population et ainsi d'augmenter ou de diminuer la convergence de l'algorithme.

Le principal avantage de cette technique est qu'elle ne nécessite pas d'assigner de manière explicite une fitness à un individu. Néanmoins, comme pour le MOGA, la performance de l'algorithme dépend du paramètre σ_{sh} de la fonction de partage et de la valeur de t_{dom}.

3.7.2.3 Non dominated Sorting Genetic Algorithm (NSGA)

Cette méthode a été proposée par Srinivas et Deb [1994]. Comme le MOGA, l'approche NSGA est basée sur le classement non dominé. Elle affecte à chaque individu une fitness factice selon le front auquel il appartient. Chaque front correspond à un groupe d'individus ayant le même degré de dominance au sens Pareto. Les individus du Front 1 auront une meilleure fitness que ceux du Front 2 qui eux, auront une meilleure fitness que ceux du Front 3 et ainsi de suite. Pour la sélection, elle applique ensuite le partage de fitness au niveau de chaque front pour maintenir la diversité. La Figure 3.8 montre le classement d'une population par front dans un contexte de minimisation. L'ensemble des solutions non dominées est représenté par les solutions du front 1.

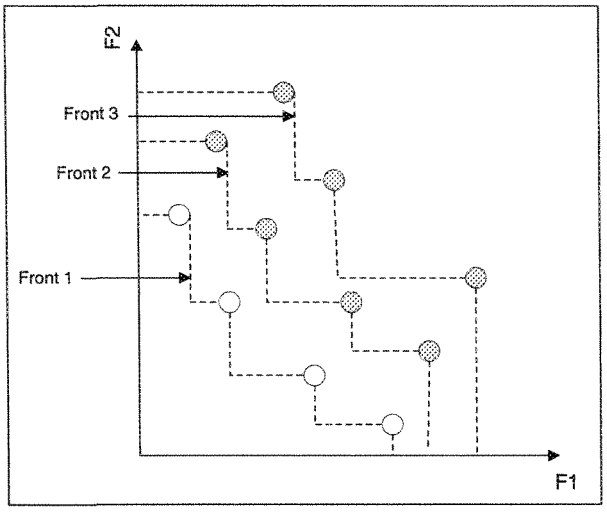


Figure 3.8 : Illustration du classement par front dans le NSGA [Zitzler *et al.* 1999]

Cette méthode paraît moins efficace en temps de calcul que la méthode MOGA car le temps de calcul nécessaire à la construction des fronts et au partage de la fitness est important ($O(nobj|POP|^3)$ avec $nobj$ = le nombre d'objectifs et $|POP|$ = la taille de la

population). L'utilisation d'une fonction de partage sur l'espace des solutions et le tri des solutions en différentes frontières semblent toutefois plus appropriés pour maintenir une grande diversité de la population et pour répartir plus efficacement les solutions sur la frontière Pareto [Berro 2001]. Cependant, comme pour le NPGA et le MOGA, l'efficacité de la méthode dépend de la spécification des paramètres de la fonction de partage.

3.7.3 Les techniques élitistes

À l'opposé des méthodes non élitistes, une technique élitiste comporte une ou plusieurs stratégies permettant de conserver les meilleures solutions trouvées au cours de l'exécution de l'algorithme.

3.7.3.1 NSGAII

Dans cette deuxième version du NSGA [Deb 2000], l'auteur tente de résoudre les critiques faites à la première version de l'algorithme : complexité, utilisation du partage de fitness et non-élitisme.

La complexité de l'algorithme NSGA est notamment due à la procédure de création des différents fronts. Pour diminuer la complexité de calcul du NSGA, Deb propose une modification de la procédure de tri de la population en plusieurs fronts. Au total, cette nouvelle procédure a une complexité de $O(nobj|POP|^2)$.

L'autre critique sur le NSGA concerne l'utilisation de la fonction de partage, méthode qui exige le réglage d'un ou plusieurs paramètre(s) et qui est également forte consommatrice de calculs. Dans le NSGAII, Deb remplace la fonction de partage de fitness par une fonction de remplacement. Pour cela, il attribue deux caractéristiques à chaque individu :

- i_{rank} qui représente le rang de non-dominance de l'individu. Cette caractéristique dépend de la frontière à laquelle appartient l'individu. Les individus du front 1 auront un i_{rank} de 0 car ils sont non dominés. Les individus du front 2 un i_{rank} de 1 car ils ne sont dominés que par des individus du front 1 et ainsi de suite; et
- $i_{distance}$ qui représente la distance de remplacement de l'individu et permet d'estimer la densité de la population autour de lui.

Comme illustré dans la Figure 3.9, pour estimer la densité au voisinage d'une solution i, on calcule la distance moyenne sur chaque objectif, entre les deux points les plus proches situés de part et d'autre de la solution. Cette quantité, appelée $i_{distance}$, sert d'estimateur de taille du plus large hyper-cube incluant le point x sans inclure un autre point de la population. L'algorithme de calcul est de complexité $O(nobj|POP|\log(|POP|))$. Cette distance de remplacement va être utilisée pour guider le processus de sélection.

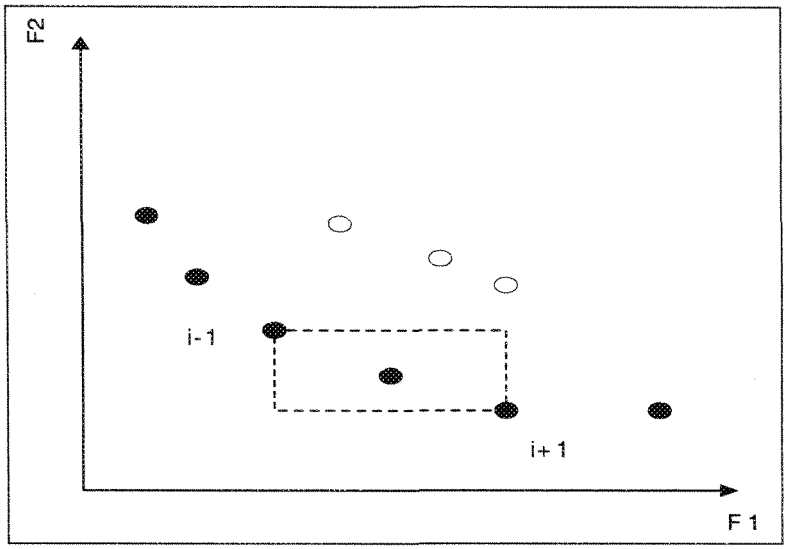


Figure 3.9 : Illustration du calcul de la $i_{distance}$ du NSGAII [Deb 2000]

Pour répondre à la critique de non-élitisme, l'auteur utilise dans cette version une sélection par tournoi et modifie la procédure de passage entre deux générations. Si deux solutions sont sélectionnées pour participer au tournoi, la solution de plus bas rang i_{rang} sera retenue. Si les deux rangs sont identiques, il est préférable de choisir la solution située dans une région dépeuplée, c'est-à-dire avec une valeur $i_{distance}$ importante.

La Figure 3.10 illustre le fonctionnement du NSGAII. À chaque itération t de l'algorithme, une population d'enfant Q_t de taille $|POP|$ est générée à partir d'une population POP_t de parents de taille identique. Les deux populations sont alors combinées dans une population $POP_t \cup Q_t$ de taille $2*|POP|$. La nouvelle population est ensuite triée en plusieurs fronts selon le degré de non-dominance. Lorsque la procédure de tri est complétée, une nouvelle population POP_{t+1} de taille $|Pop|$ est créée à partir des solutions des différents fronts de $POP_t \cup Q_t$. On commence avec les solutions du premier front et ainsi de suite. Comme la taille de $POP_t \cup Q_t$ est de $2*|Pop|$ et que la taille de la nouvelle population est juste de $|POP|$, tous les fronts ne pourront pas être contenus dans POP_{t+1}. Les fronts qui n'entreront pas dans la nouvelle population seront tout simplement supprimés. Cependant, il se peut qu'il y ait dans un front plus de solutions que de places disponibles. Dans ce cas, les individus du front situés dans les régions les plus isolées ($i_{distance}$ importante) sont choisis et les autres sont éliminés.

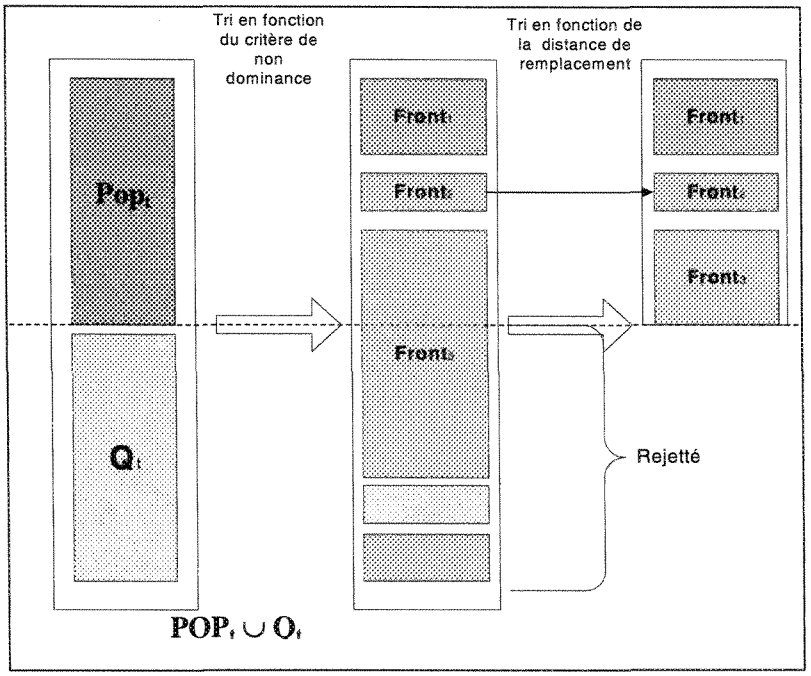


Figure 3.10 : Illustration du fonctionnement du NSGAII [Deb 2001]

Cette nouvelle version de NSGA a permis de réduire la complexité de l'algorithme à $O(nobj|POP|^2)$, de créer une méthode plus élitiste et de supprimer les paramètres de la fonction de partage de fitness. Cependant, il y a un risque de perte de solutions lorsque le nombre de solutions PO dépasse la taille de la population.

3.7.3.2 Strength Pareto Evolutionary Algorithm (SPEA) et SPEA2

Proposé par Zitler *et al.* [1999], le SPEA est basé simultanément sur les concepts de non-dominance et d'élitisme. En plus d'une population initiale de taille $|POP|$, le SPEA utilise une population externe, appelée archive, pour maintenir les solutions PO. Dans cette méthode, le passage d'une génération à une autre commence par la mise à jour de l'archive. Tous les individus non dominés sont copiés dans l'archive et les individus dominés déjà présents sont supprimés. Si le nombre d'individus dans l'archive excède un nombre donné,

on applique une technique de regroupement (*clustering*) pour réduire l'archive. La fitness de chaque individu est ensuite mise à jour avant d'effectuer la sélection en utilisant les deux ensembles. Pour terminer, on applique les opérateurs génétiques de croisement et de mutation.

Une version révisée du SPEA a été récemment proposée pour corriger les lacunes de la version précédente : le SPEA 2 [Zitzler *et al.* 2001]. L'algorithme du SPEA 2 comporte trois principales différences avec son prédécesseur : changement au niveau du calcul de la fitness, introduction d'une nouvelle technique d'estimation de la densité et d'une nouvelle technique pour la réduction de l'archive.

Pour éviter que les individus dominés par les mêmes membres de l'archive aient des fitness identiques, le mécanisme permettant le calcul de la fitness du SPEA 2 tient compte à la fois du nombre d'individus dominés par une solution et du nombre d'individus qui domine la solution. Autrement dit, pour chaque individu x de l'archive $\overline{POP_t}$ de taille $|\overline{POP_t}|$ et de la population POP_t de taille $|POP|$, on calcule la force $S(x)$ représentant le nombre de solutions qu'il domine. La fitness brute $R(x)$ d'un individu x correspondant à la somme des forces des individus qui dominent x.

Bien que la fitness brute soit une technique de nichage basée sur la notion de dominance, elle se révèle inefficace lorsque la plupart des individus sont des solutions non dominées [Zitzler *et al.* 2001]. C'est pourquoi les auteurs ont introduit une technique d'estimation de la densité basée sur le concept du $k^{\text{ième}}$ plus proche voisin [Silverman 1986]. Pour cela, la distance entre un individu x et tous les individus y de l'archive et de la population est calculée et stockée dans une liste. Après avoir trié la liste en ordre croissant, le $k^{\text{ième}}$

élément donne la distance σ_x^k recherchée. Dans l'algorithme, les auteurs utilisent $k=\sqrt{|Pop|+|\overline{Pop}|}$. La densité *Dens(x)* correspondant à *x* est donc :

$$Dens(x) = \frac{1}{\sigma_x^k + 2} \qquad (3.11)$$

Au dénominateur, on ajoute deux pour s'assurer que celui-ci sera supérieur à zéro et que $Dens(x) < 1$. Finalement, en ajoutant *Dens(x)* à la fitness brute *R(x)* on obtient la fitness *f(x)* de l'individu *x*.

Pour réduire les pertes de solutions dues à l'ancienne méthode de regroupement, les auteurs ont introduit une nouvelle technique de réduction de l'archive. La Figure 3.11 représente le principe de réduction de la taille de l'archive du SPEA 2. Sur la gauche, un ensemble de solutions non dominées est présenté. Sur la droite, on montre quelles solutions seront enlevées de l'archive et dans quel ordre par la fonction de réduction. Si la taille de l'archive doit contenir seulement cinq éléments, la fonction de réduction supprimera en premier la solution1 puis la solution 2 et enfin la solution 3.

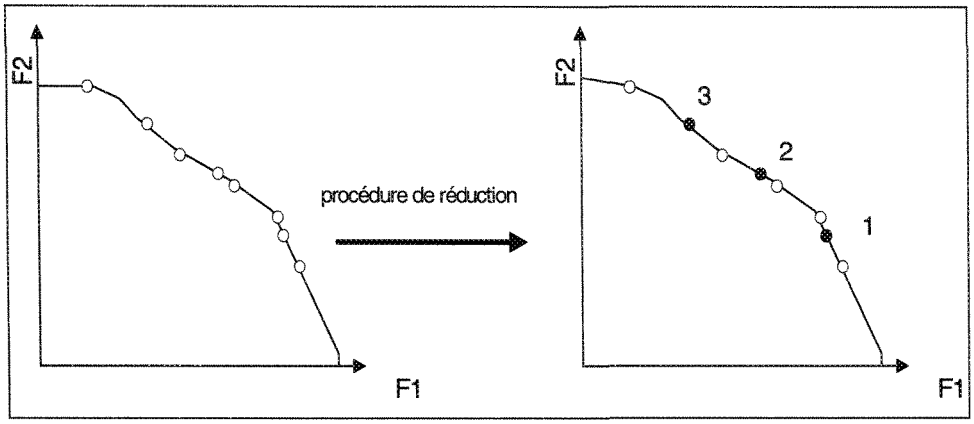


Figure 3.11 : Illustration de la méthode de réduction de l'archive utilisée par le SPEA 2 [Zitzler *et al.* 2001].

Cette nouvelle version du SPEA a permis de combler certaines lacunes de son prédécesseur. Cependant, les performances de l'algorithme dépendent généralement de la taille de l'archive.

3.7.3.3 Pareto Archived Evolution Strategy (PAES) et Memetic- PAES (M-PAES)

Le PAES est une méthode initialement développée comme une méthode de recherche locale pour un problème de routage d'information off-line. Les premiers travaux de Knowles et Corne [1999] ont montré que cette méthode uni-objectif fournissait des résultats supérieurs aux méthodes de recherche basées sur une population. Ces performances ont poussé les auteurs à adapter cette technique à la résolution de problèmes multi-objectifs. Les particularités de cette méthode sont les suivantes :

- elle n'est pas basée sur une population. Elle n'utilise qu'un seul individu à la fois pour la recherche des solutions;
- elle utilise une population annexe de taille déterminée permettant de stocker les solutions temporairement PO;

- l'algorithme utilisé est très simple et inspiré d'une *stratégie d'évolution* (*1+1*) [Rechenberg 1973]; et
- elle utilise une technique de *crowding* basée sur un découpage en hyper-cubes de l'espace des objectifs.

Le M-PAES est une extension de la recherche locale (*1+1*)-PAES combinée avec l'utilisation d'une population *POP* [Knowles et Corne 2000]. Dans cette approche, les optima locaux trouvés par la procédure (*1+1*)-PAES sont périodiquement combinés par croisement. La mise à jour de l'archive s'avère cependant plus complexe que celle utilisée par le PAES. En effet, dans l'algorithme de PAES l'archive permet de mémoriser les solutions non dominées trouvées jusque là et sert aussi comme ensemble de comparaison permettant d'évaluer la dominance des nouveaux individus. Pour effectuer ces deux rôles, le M-PAES a besoin de deux archives : une archive globale pour mémoriser un ensemble de solutions non dominées et une archive locale utilisée comme échantillon de comparaison lors des recherches locales.

Les tests effectués par les auteurs montrent que M-PAES est supérieure dans tous les cas à (*1+1*)-PAES. Mais la comparaison avec SPEA a produit des résultats équivalents bien qu'il soit difficile de comparer ces deux algorithmes.

3.7.3.4 Pareto Envelope Based Selection Algorithm (PESA) et PESAII (Region Based Selection)

Également proposé par Knowles et Corne [2000], le PESA reprend approximativement le principe de crowding développé dans PAES et définit un paramètre appelé *squeeze_factor* qui représente la mesure d'encombrement d'une zone de l'espace. Alors que

PAES est basé sur une stratégie d'évolution, PESA est une méthode basée sur les algorithmes génétiques qui définit deux paramètres concernant la taille des populations d'individus : POP_I (taille de la population interne) et POP_E (taille de la population externe ou archive).

Au cours de la phase de recherche, une solution courante de Pop_I peut entrer dans l'archive POP_E si elle est non dominée dans POP_I et si elle est non dominée dans POP_E. Une fois la solution insérée dans l'archive, on supprime tous les membres de l'archive qu'elle domine. Si l'ajout crée un dépassement de capacité de POP_E alors le membre de l'archive ayant le *squeeze_factor* le plus élevé est supprimé. Le paramètre *squeeze_factor* est égal au nombre d'individus qui appartiennent au même hyper-cube. Il est utilisé comme mesure de fitness des individus qui appartiennent à cette zone. Contrairement au PAES où la mesure d'encombrement n'est utilisée que pour la mise à jour de l'archive, le *squeeze_factor* est aussi utilisé lors du processus de sélection dans l'algorithme de PESA. Lors de la sélection par tournoi, le candidat choisi sera celui qui a le plus petit *squeeze_factor*. Ainsi, les recherches seront orientées vers des zones de la frontière de Pareto qui sont le moins représentées dans la population courante.

Récemment, Corne [2001] propose une nouvelle version de l'algorithme PESA basée sur l'utilisation d'hyper-cubes dans l'espace des objectifs. Au lieu d'effectuer une sélection en fonction de la fitness des individus comme dans PESA, cette méthode effectue une sélection par rapport aux hyper-cubes occupés par au moins un individu. Après avoir sélectionné l'hyper-cube, on choisit aléatoirement l'individu dans l'hyper-cube. Cette méthode se montre plus efficace que son prédécesseur à répartir les solutions sur la

frontière de Pareto. Cela est dû à sa capacité de choisir avec une plus grande probabilité que le tournoi classique, des individus situés dans des zones peu explorées.

3.7.3.5 Micro-genetic Algorithm (Micro-Ga)

Coello trouve que les recherches actuelles n'accordent pas assez d'importance à l'efficience des méthodes d'optimisation multi-objectifs. L'auteur propose alors une méthode basée sur une population avec un nombre très faible d'individus [Coello Coello et Pulido 2001] afin de limiter les temps de calculs. Cette technique se base sur les résultats théoriques obtenus par Goldberg [1989], selon lesquels une taille de population très faible suffit à obtenir une convergence indépendamment de la longueur du chromosome. Le mécanisme suggéré par Goldberg est d'utiliser une population très petite et d'appliquer les opérateurs génétiques jusqu'à obtention d'une convergence (identifiée par une perte de diversité génétique). Ensuite, on crée une nouvelle population en copiant les meilleurs individus de la population précédente, le reste de la population étant générée aléatoirement.

Coello applique ce mécanisme aux problèmes d'optimisation multi-objectifs en utilisant un algorithme génétique avec une petite taille de population associée à une archive et une heuristique de distribution géographique. Il définit une population extérieure divisée en deux parties : une partie remplaçable et une partie non remplaçable. La portion non remplaçable ne change pas durant le processus de modification et sert à maintenir la diversité. Elle ne sera mise à jour que lorsque le Micro-GA aura convergé. La portion remplaçable est totalement modifiée à intervalle régulier. Ce dernier est défini en nombre de cycles du Micro-GA.

Au début de l'algorithme du Micro-GA, la population est constituée à l'aide d'individus sélectionnés aléatoirement dans la population externe. Ensuite, l'algorithme se déroule de manière classique. En fin de cycle, lorsque la population du Micro-GA a perdu sa diversité, deux individus non dominés sont sélectionnés pour mettre à jour l'archive. L'approche utilisée est similaire à celle de PAES. Ensuite, ces deux mêmes individus sont comparés à deux individus de la partie non remplaçable. Si l'un des deux premiers domine l'un des deux autres alors, il le remplace dans la partie non remplaçable.

3.7.3.6 Pareto Memetic Strategy for Multiple Objective optimization (PMS^{MO})

Proposé récemment par Zinflou *et al.* [2006], le PMS^{MO} est un algorithme mimétique élitiste combinant les concepts de dominance Pareto, de niche et d'élitisme avec des opérateurs de recherche locale.

En plus d'une population initiale *POP*, le PMS^{MO} utilise deux archives : une *archive locale,* notée *A* et une *archive globale*, notée Ă. L'archive locale consiste en une population externe de taille déterminée et fixée à $|A|$ permettant de stocker les solutions temporairement PO qui participent au processus de sélection. Lorsque le nombre de solutions non dominées dans l'archive locale dépasse $|A|$, une procédure de réduction de l'archive est appliquée afin de ne conserver que les individus situés dans les régions les plus isolées. Les autres individus sont, quant à eux, transférés dans l'archive globale et ne participent plus au processus de sélection.

Le PMS^{MO} utilise deux paramètres pour calculer la fitness à un individu : le *facteur de dominance* R^+ et le *facteur d'isolement D*. Le *facteur de dominance* est déterminé dans l'algorithme à l'aide de l'équation suivante :

$$R^{+}(x)=\begin{cases} \dfrac{S(x)}{1+2*S(x)} & si \quad \sum_{y\in Pop_t\cup A_t,\ y\succ x} S(y)=0 \\ \sum_{y\in Pop_t\cup A_t,\ y\succ x} S(y) & sinon \end{cases} \qquad (3.12)$$

Pour les solutions non dominées, cette méthode de calcul du *facteur de dominance* permet de mieux tenir compte de la distribution des solutions dominées dans l'espace de recherche en favorisant les solutions non dominées situées dans des régions sous exploitées. Le *facteur d'isolement* quant à lui est calculé selon le concept du $k^{\text{ième}}$ plus proche voisin [Silverman 1986] de façon similaire au SPEA 2. À la seule différence que dans le PMS^{MO}, la distance d'un individu x de l'archive locale est calculée seulement par rapport aux autres individus y de la même archive.

Après avoir été généré par croisement, chaque nouvel individu est amélioré à l'aide d'une procédure de recherche locale. Au cours de cette procédure, un mouvement d'une solution x vers une solution voisine y est effectué dans deux cas. Tout d'abord, si la solution voisine y domine la solution de départ x alors un mouvement est effectué. Si, au contraire, aucune relation de dominance ne peut être identifiée entre x et y alors, à chaque génération, un poids w_{obj} est déterminé aléatoirement pour chacun des objectifs *obj* du problème à résoudre. La somme des poids w_{obj} étant égale à 1. La valeur normalisée *vn* est ensuite calculée selon la formule :

$$vn = \prod_{obj=1}^{nobj} \left(\frac{f_{obj}(y)}{f_{obj}(x)} \right)^{w_{obj}} \qquad (3.13)$$

où *nobj* représente le nombre total d'objectifs. Si $vn<1$, pour un contexte de minimisation, on effectue un mouvement. Dans le cas contraire, aucun mouvement n'est effectué.
Les tests effectués par les auteurs montrent que le PMS^{MO} est une approche compétitive par rapport au NSGAII et au SPEA2 pour le problème de sac à dos multidimensionnel ainsi que pour un problème industriel de planification de coulée de l'aluminium.

3.8 La mesure des performances des algorithmes multi-objectifs

Comment comparer deux ensembles de solutions pour un problème multi-objectifs ? Intuitivement, la première réponse qui vient à l'esprit serait de considérer chaque ensemble un à un. Toutefois, en optimisation multi-objectifs, cette tâche n'est pas aisée. En effet, plusieurs aspects entrent en ligne de compte comme, par exemple, la qualité (la proximité par rapport à la frontière Pareto théorique), la distribution (est-ce que toutes les parties de la frontière Pareto réelles sont découvertes), la répartition (est-ce que les points sont répartis de manière homogène tout au long de la frontière) et autres. En fait, il est très difficile, voire même impossible, de prendre en compte tous ces paramètres au travers d'une seule valeur numérique [Barichard 2003]. C'est pourquoi il est courant d'utiliser plusieurs mesures (ou métriques) pour tester tel ou tel aspect de l'ensemble obtenu.

De plus, il faut distinguer deux types de métriques : les métriques relatives, qui comparent deux ensembles, et les métriques absolues, qui évaluent un ensemble sans avoir besoin d'autres points ou ensemble de référence. Dans la suite de cette section, nous allons présenter quelques mesures couramment utilisées. Nous mettrons en avant leurs points forts

et leurs points faibles, de manière à aiguiller le choix du développeur quant à la métrique qu'il peut utiliser.

3.8.1 La métrique d'espacement (*Sp*)

La première métrique présentée dans cette partie du document est la mesure d'espacement (spacing metric) introduite par Schott [1995]. Cette métrique permet de mesurer l'uniformité de la répartition des points composant la surface de compromis. Cette métrique peut se définir de la manière suivante. Soit *A* un ensemble de *n* éléments de dimensions *nobj,* alors *Sp(A)* est définie par :

$$Sp(A) = \left[\frac{1}{n-1} * \sum_{i=1}^{n} \left(\overline{dist} - dist_i \right)^2 \right]^{1/2} \qquad (3.14)$$

avec $dist_i = \min_j \left(\left| f_1^i(\vec{x}) - f_1^j(\vec{x}) \right| + \ldots + \left| f_{nobj}^i(\vec{x}) - f_{nobj}^j(\vec{x}) \right| \right)$ $i,j \in [1,\ldots, n]$ *et* $i \neq j$, $\overline{dist}$: moyenne de tous les $dist_i$; et n : nombre d'éléments de l'ensemble de solutions. Pour illustrer la métrique d'espacement, nous reprenons ici un des exemples donnés dans Collette et Siarry [2002]. Cet exemple considère une surface de compromis composée de trois points. Le tableau des distances entre les points de la surface de compromis est représenté au Tableau 3.1.

Point *i*	**Point *j***	**Distance**
(2.5, 9)	(3, 6)	3.5
(2.5, 9)	(5,4)	7.5
(3, 6)	(2.5,9)	3.5
(3, 6)	(5,4)	4
(5,4)	(2.5,9)	7.5
(5,4)	(3, 6)	4

Tableau 3.1 : Exemple de calcul de la métrique d'espacement [Collette et Siarry 2002]

On obtient donc les valeurs $dist_i$ suivante :

- pour le point de coordonnées (2.5, 9), $dist_1 = 3.5$;
- pour le point de coordonnées (3, 6), $dist_2 = 3.5$; et
- pour le point de coordonnées (5, 4), $dist_3 = 4$.

On peut ainsi calculer $\overline{dist} = \frac{3.5+3.5+4}{3} = 3.67$. Pour cet exemple, on a donc

$$Sp = \left[\frac{1}{2} * \left((3.67-3.5)^2 + (3.67-3.5)^2 + (3.67-4)^2\right)\right]^{\frac{1}{2}} = 0.288$$

Plus le résultat de la mesure est proche de 0, meilleure est la répartition des points sur la surface de compromis. Cette métrique est un moyen de mettre en avant l'aspect « diversité » d'un algorithme, mais, dans certains cas, l'évaluation peut être biaisée. En effet, si le front Pareto est discontinu, on aura alors un *Sp* élevé à cause des discontinuités introduites par la forme de la surface de compromis. Ainsi, les trous présents dans la surface de compromis influent sur le calcul de la métrique d'espacement et rendent souvent difficile l'interprétation des résultats [Collette et Siarry 2002].

3.8.2 La métrique (C)

La couverture de deux ensembles ou métrique ***C*** a été introduite en 1999 par Ziztler [1999]. C'est une métrique relative qui permet de comparer deux surfaces de compromis *A* et *B*. La valeur ***C*** (*A*, *B*) correspond au pourcentage d'éléments de *B* dominés par au moins un des éléments de *A*. Le calcul de ***C*** (*A*, *B*) s'effectue selon la formule suivante [Zitzler 1999] :

$$C(A,B) = \frac{\left|\left(b \in B \middle| \exists\ a \in A : a \succ b\right)\right|}{|B|} \qquad (3.15)$$

où | | représente la cardinalité de l'ensemble et le symbole $\succ$ indique une relation de dominance de a par rapport à b. Une mesure $\boldsymbol{C}$ (A, B) = 1 signifie que tous les vecteurs de décision de B sont dominés par ceux de A. À l'inverse, $\boldsymbol{C}$ (A, B) = 0, signifie qu'aucun des points de B n'est faiblement dominé par un point de A. Ainsi, plus la valeur $\boldsymbol{C}$ (A, B) se rapproche de 1, meilleure la surface de compromis A est considérée par rapport à B. La métrique C n'est pas symétrique : $\boldsymbol{C}$ (A, B) $\neq$ 1 - $\boldsymbol{C}$ (A, B), il est par conséquent nécessaire de considérer les deux valeurs $\boldsymbol{C}$ (A, B) et $\boldsymbol{C}$ (B, A) pour obtenir une mesure plus fiable des deux surfaces de compromis A et B à comparer.

La métrique $\boldsymbol{C}$ est un outil intéressant de mesure relative. Elle permet de différencier nettement deux surfaces de compromis lorsque d'autres mesures donnent des évaluations trop proches. Cependant, cette métrique doit être utilisée conjointement à d'autres métriques pour ne pas fausser le jugement [Collette et Siarry 2002; Barichard 2003].

3.8.3 L'hyper-volume (H)

Comme la métrique $\boldsymbol{C}$, l'hyper-volume $\boldsymbol{H}$ a été proposé par Zitzler dans sa thèse de doctorat [1999]. Schématiquement, cette métrique calcule une approximation du volume compris sous la courbe formée par les points de l'ensemble à évaluer. Ainsi, lorsque le problème comporte deux critères, ce calcul correspond au calcul d'une aire. Par contre, lorsque le problème comporte trois critères, la valeur calculée est un volume. Formellement, la métrique $\boldsymbol{H}$ se définit de la manière suivante [Zitzler 1999] : soit A un

sous ensemble de n éléments. La fonction $\boldsymbol{H}$ calcule le volume borné par l'union de tous les polytopes $p_1, p_2, \ldots, p_n$, où chaque p_i est formé par l'intersection des hyperplans des x_i par rapport aux axes du repère, pour chaque axe de l'espace des objectifs, il existe un hyperplan perpendiculaire à l'axe et passant par le point $(f_1(x_i), f_2(x_i), \ldots, f_n(x_i))$. Pour le cas à deux dimensions, chaque p_i représente un rectangle défini par les points de coordonnées (0, 0) et $(f_1(x_i), f_2(x_i))$.

Ainsi, la métrique $\boldsymbol{H}$ permet de donner une valeur numérique à une surface de compromis composée a priori de plusieurs points. Il est clair que la valeur calculée ne permet de mesurer qu'un aspect de la surface de compromis et que, dans certains cas, comme les problèmes non convexes [Van Veldhuizen 1999], la valeur calculée peut être erronée. Un des principaux avantages de cette métrique réside dans le fait qu'elle ne nécessite aucun autre point ou ensemble de référence [Zitzler 1999]. Toutefois, son principal inconvénient est qu'elle nécessite souvent des temps de calcul important [Barichard 2003]. En effet, pour calculer l'union des hyper-volumes induits par chaque point de la surface de compromis, il est nécessaire de calculer un grand nombre d'intersections de volumes. Cette opération, très rapide dans le cas d'un problème à deux objectifs, est beaucoup plus longue lorsque le nombre d'objectifs dépasse trois.

3.8.4 La différence de couverture (*Diff*)

La dernière métrique présentée dans ce chapitre est la différence de couverture entre deux ensembles ou métrique ***Diff***. Cette métrique, aussi introduite par Zitzler [1999], permet de surmonter certains inconvénients de la métrique $\boldsymbol{C}$. Formellement, la métrique ***Diff*** se définit comme suit [Zitzler 1999] : soit deux ensembles A et B, la fonction ***Diff***

définie par ***Diff***$(A,B) = \boldsymbol{H}(A+B) - \boldsymbol{H}(B)$ indique la taille de l'espace dominé par A et pas par B dans l'espace des solutions. Comme pour la métrique **C**, pour comparer deux ensembles *A* et *B* entre eux, il est nécessaire de considérer ***Diff***(A,B) et ***Diff***(A,B). Toutefois, la métrique ***Diff*** étant basée sur l'hyper-volume, elle nécessite souvent des temps de calculs plus importants que ceux de la métrique ***C***.

CHAPITRE 4

ALGORITHME ÉVOLUTIONNAIRE POUR LE PROBLÈME THÉORIQUE D'ORDONNANCEMENT DE VOITURES

4.1 Introduction

Le chapitre précédent a permis de présenter les algorithmes évolutionnaires et plus particulièrement les algorithmes génétiques comme des mécanismes de recherche généraux, puissants et robustes ayant été appliqués avec succès à la résolution de nombreux problèmes d'optimisation combinatoire. Toutefois, dans le Chapitre 2, nous avons remarqué que même si le POV intéresse la communauté scientifique depuis le milieu des années 80 et constitue maintenant un benchmark classique en optimisation combinatoire [Dincbas *et al.* 1988; Gent et Walsh 1999; Solnon 2000; Gottlieb *et al.* 2003; Gravel *et al.* 2005; Morin *et al.* 2006; Solnon 2006], très peu de travaux portent sur la conception d'algorithmes génétiques aussi bien pour la version théorique qu'industrielle du problème. À notre connaissance, Warwick et Tsang [1995] furent les premiers à appliquer les AG à la résolution du POV théorique. Dans leur approche, à chaque itération les individus sélectionnés sont croisés à l'aide d'un opérateur de croisement uniforme adaptatif (UAX). Après croisement, chaque enfant est réparé à l'aide d'une fonction de restauration des gènes puis amélioré à l'aide d'une procédure de recherche locale (hill-climbing). Plus récemment, Terada *et al.* [2006] proposèrent un algorithme génétique classique pour la résolution du POV théorique dans le but de le combiner par la suite avec une variante de recherche locale appelée : Squeaky-Wheel Optimization (SWO) [Joslin et Clements 1998]. Les expérimentations effectuées par les auteurs montrent que les approches proposées à base d'AG sont capables de résoudre des instances du problème de car sequencing avec un faible taux d'utilisation. Toutefois, l'efficacité est sérieusement diminuée pour les instances avec un taux d'utilisation plus élevé [Warwick et Tsang 1995; Terada *et al.* 2006]. On peut

supposer que cette situation s'explique en partie par la difficulté qu'il y a à définir des opérateurs génétiques efficaces et adaptés à la résolution du problème. En effet, les opérateurs de croisement génétiques sont généralement présentés dans la littérature pour résoudre des problèmes de voyageur de commerce (TSP), des problèmes utilisant une codification binaire [Michalewicz 1994; Potvin 1996] ou ceux utilisant une représentation réelle [Ben Hamida 2001] et ne sont donc pas adaptés aux spécificités du POV.

L'objectif principal de ce chapitre consiste donc à résoudre efficacement le POV théorique à l'aide d'un algorithme génétique en proposant deux nouveaux opérateurs de croisement spécifiques pour ce type de problème. Le reste du chapitre est organisé de la façon suivante. La Section 4.2 introduit les deux nouveaux opérateurs proposés. La Section 4.3, de son côté, présente les différents opérateurs de mutation utilisés. Le fonctionnement général de l'algorithme génétique est par la suite présenté à la Section 4.4. Finalement, la Section 4.5 présente les résultats expérimentaux des approches proposées et les compare avec ceux d'autres approches de résolution tirées de la littérature.

4.2 Nouveaux opérateurs de croisement pour le POV

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. Cette situation explique sans doute le fait que cette métaheuristique a très peu été appliquée à la résolution du POV. Dans cette partie, nous proposons de combler cette lacune en introduisant deux nouveaux opérateurs de croisement spécifiquement conçus

pour le POV. On note ici que chacun des opérateurs de croisement présenté dans cette section génèrent deux enfants.

4.2.1 Notion d'intérêt

Afin de présenter les différents opérateurs de croisement proprement dit, il importe de définir au préalable une notion importante qui est utilisée par ces opérateurs : la notion d'intérêt. L'intérêt de placer une voiture de la classe v à une position i donnée de la séquence en fonction des classes de voitures déjà placées est calculé selon l'équation suivante :

$$I_{vi} = \begin{cases} D_v & \text{si } NbNouveauxConflits_{vi} = 0 \\ -NbNouveauxConflits_{vi} & \text{sinon} \end{cases} \quad (4.1)$$

où $NbNouveauxConflits_{vi}$ correspond au nombre de nouveaux conflits engendrés par l'addition d'une voiture de la classe v à la position i et D_v correspond à la difficulté de la classe comme calculée à l'Équation 2.3 du chapitre 2.

4.2.2 Non Conflict Position Based Crossover (NCPX)

L'idée principale derrière tout opérateur de croisement consiste à générer différents individus potentiellement prometteurs. Pour y arriver, un bon opérateur de croisement pour le POV devrait tenir compte, autant que possible, des spécificités des contraintes d'espacement.

Le premier opérateur de croisement présenté cherche à produire un enfant à partir de gènes situés à des positions non conflictuelles dans les parents. Pour ce faire, la première opération consiste à tirer de manière aléatoire un nombre nb_g entre 0 et *nbpossconflit*.

nbpossconflit correspond au nombre de position sans conflit retrouvé dans le parent 1 (P_1). Ce nombre nb_g sert à déterminer le nombre de « bons » gènes qui conserveront la même position dans l'enfant et dans le Parent 1. Par la suite, une position de départ (Pos_d) est déterminée aléatoirement entre 1 et $|P_1|$ dans l'enfant à créer, où $|P_1|$ indique la longueur du chromosome parent. Les gènes situés à des positions non conflictuelles sont copiés, dans un premier temps, de P_1 vers l'enfant en partant de Pos_d jusqu'à la fin du chromosome. Dans un deuxième temps, si le nombre de gènes recopiés est inférieur à nb_g, le processus de copie recommence en partant du début de l'enfant jusqu'à Pos_{d-1}. Le reste des gènes de P_1 est utilisé pour constituer une liste de classe de voitures à placer L. Par la suite, on détermine aléatoirement une position (Pos) à partir de laquelle le reste du chromosome enfant va être complété. Pour finir, les classes de voitures contenues dans L sont placées en fonction de leur intérêt selon l'équation suivante :

$$v = \begin{cases} \arg\max\{I_{vi}\} & \text{si } p \leq 0.95 \\ V & \text{sinon} \end{cases} \qquad (4.2)$$

où p est un nombre aléatoire entre 0 et 1 et V déterminé de manière probabiliste parmi les classes de voitures introduisant le moins de conflits. On note cependant qu'en cas d'égalité sur $\arg\max\{I_{vi}\}$, si une des classes de voitures impliquée dans l'égalité se retrouve dans P_2 à la position i sans conflit alors on retient cette classe de voitures. Dans le cas contraire, on tire aléatoirement une classe de voitures parmi celles qui sont impliquées dans l'égalité. Le fonctionnement de l'opérateur de croisement *NCPX* est illustré à la Figure 4.1 pour deux individus P_1 = 21352446 et P_2 = 32621454. Si on suppose que l'évaluation de P_1 a retourné 5 positions sans conflits et qu'on a tiré $nb_g = 4$ et $Pos_d = 3$. À partir de Pos_d, on peut copier les gènes 5, 4, 4 et 2 dans l'enfant. Les gènes 2, 3, 1 et 6 de P_1 servent à constituer la liste

initiale L. Finalement, en supposant qu'on ait tiré Pos = 7, on commence à placer respectivement les gènes 3, 2, 6 et 1 de L dans l'enfant à partir de Pos. Ainsi, les gènes 1,2 et 6 sont directement hérités de P_2. L'enfant généré à partir de P_1 et P_2 avec cette technique est E_1= 22651443.

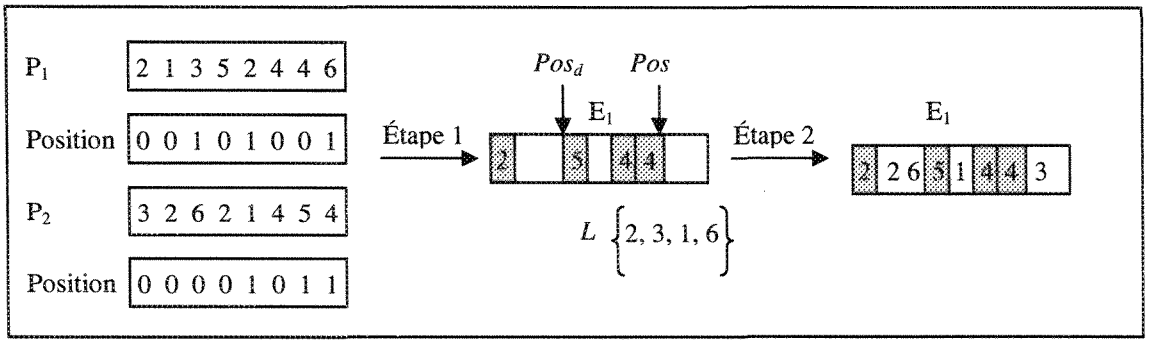


Figure 4.1 : Illustration du fonctionnement du croisement *NCPX*

L'objectif de cette technique de croisement consiste donc, au fil des générations, à favoriser le plus possible les « bons » gènes, i.e ceux situés à des positions non conflictuelles, provenant des parents de manière à minimiser le nombre de classes de voitures à repositionner.

4.2.3 Interest Based Crossover (IBX)

Le deuxième opérateur de croisement proposé s'inspire du PMX [Goldberg et Lingle 1985]. La Figure 4.2 illustre le fonctionnement de cet opérateur afin de tenir compte des contraintes de production et d'espacement du POV. Dans un premier temps, deux points de coupure sont déterminés aléatoirement sur chaque parent P_1 et P_2. Afin de créer un enfant, la sous-chaîne 351 comprise entre les deux points de coupure du premier parent ($a_1 \in P_1$) est directement recopiée dans l'enfant. Ensuite, deux listes (L_1 et L_2) non ordonnées de

classes de voitures sont respectivement créées à partir des sous-chaînes $b_3 = \{3, 2\}$ et $b_4 = \{4, 5, 6\}$ de P_2. Cependant, lors de cette opération, il se peut qu'une partie de l'information soit perdue par l'introduction de doublons supplémentaires. Dans notre exemple, on remarque à l'aide de la Figure 4.2 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 la classe v soient produites, un remplacement des gènes 3 et 5 (obtenu à partir de a_1-a_2) dont le nombre dépasse les contraintes de productions par les gènes 4 et 2 (obtenu à partir de a_2-a_1), dont le nombre est maintenant inférieur aux contraintes de production, est aléatoirement effectué dans les listes L_1 et L_2 à la deuxième étape. Finalement, la dernière étape consiste à reconstruire le début et la fin de l'enfant à partir des deux listes. Pour réaliser cette étape du croisement, les classes de voitures $\in L_1$ sont ordonnées en fonction de leur *Intérêt* en partant du premier point de coupure vers le début de la séquence. Pour cela, la classe v à placer est donnée en utilisant l'Équation 4.2.

Pour déterminer V, on utilise le même principe que la sélection par roulette présentée au chapitre précédent en limitant toutefois le tirage aux classes de voitures introduisant le moins de conflits. Pour cela, on associe à chaque classe candidate un segment de la roue proportionnelle à son intérêt I_{vi}. Par la suite, on concatène ces segments sur un axe gradué entre 0 et 1. Finalement, on tire aléatoirement un nombre *rnd* entre 0 et 1 et la classe V choisie est celle dont le segment correspond à *rnd*. Notons aussi que dans l'approche gourmande en cas d'égalité sur arg max$\{I_{vi}\}$, l'égalité est brisée à 35% de manière aléatoire et le reste du temps en choisissant la classe de voitures possédant l'option la plus difficile (i.e celle ayant le ratio r_o/s_o le plus petit et s'appliquant sur le plus de voitures).

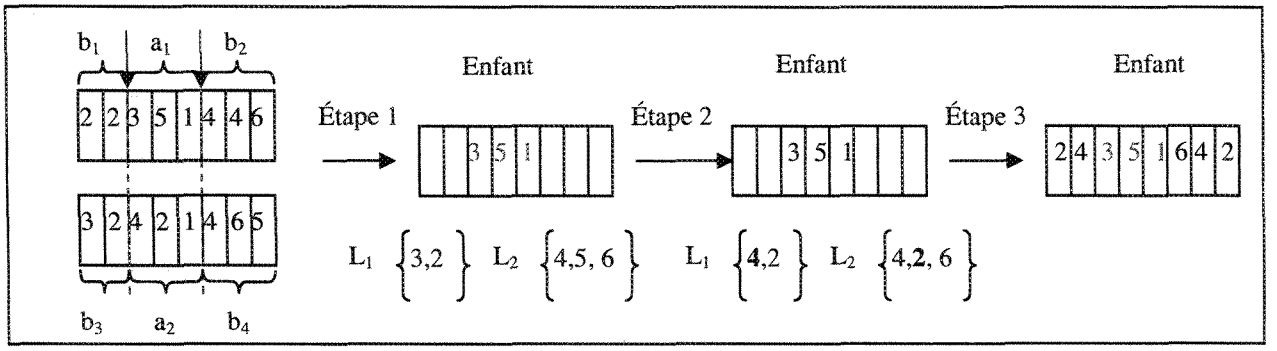


Figure 4.2 : Illustration du fonctionnement du croisement *IBX*.

Les voitures sont ordonnées de manière similaire à partir du deuxième point de coupure vers la fin de la séquence à l'aide de la liste L_2. Un deuxième enfant est créé de manière similaire en partant cette fois ci du parent 2.

Cette technique de croisement contrairement au PMX n'essaie pas de préserver la position absolue des gènes mais cherche plutôt à garder les gènes dans la même portion du chromosome que celles qu'ils occupaient dans l'un des deux parents. Dans les faits, le nombre de gènes qui ne seront pas situés dans la même zone qu'ils occupaient dans l'un des deux parents sera au plus égal à la longueur de la sous-chaîne comprise entre les deux points de coupure. Dans l'exemple précédent, seul les gènes 4 et 2 se retrouvent dans des portions du chromosome différentes de celles occupées à l'origine dans l'un des deux parents.

4.3 Opérateurs de mutation

Quatre opérateurs de mutation de base ont été utilisés : l'*échange*, l'*inversion*, le *mélange aléatoire* et le *déplacement*. Les trois premiers opérateurs ont souvent été utilisés

dans la littérature à l'intérieur d'algorithmes de recherche locale pour le POV. Un échange consiste simplement à échanger la position de deux voitures appartenant à des classes différentes dont au moins une participe à un conflit. Une inversion entre deux classes de voitures v_i et v_l consiste à inverser la sous-séquence comprise entre les positions i et l en s'assurant que l'une des deux classes de voitures participe à un conflit. Le mélange aléatoire entre deux classes de voitures v_i et v_l revient à mélanger aléatoirement les positions des voitures à l'intérieur de la sous-séquence bornée par les classes v_i et v_l. Finalement, l'opérateur de mutation par déplacement sélectionne une sous-séquence (un bloc) de voitures et l'insère à une autre position dans la séquence. Deux types de déplacement sont utilisés ici, le déplacement d'un bloc de voitures dont au moins une participe à un conflit ($depl_{cflt}$), ou le déplacement d'un bloc de voitures choisi aléatoirement ($depl_{aléa}$).

4.4 Algorithme génétique pour le POV

Dans cette section, nous présentons un nouvel AG pour résoudre le POV. Les différents éléments de son implémentation y sont présentés plus en détails.

4.4.1 Représentation des individus

Au lieu d'opter pour une représentation classique sous forme de chaîne de bits, qui apparaît peu adaptée pour ce type de problème, chaque individu de l'algorithme génétique proposé est représenté à l'aide d'un vecteur d'entiers de longueur *nc*, où chaque valeur correspond à une classe de voitures du problème à résoudre.

4.4.2 Création de la population initiale

Généralement, les individus de la population de départ sont initialisés de manière aléatoire et parfois à l'aide d'un algorithme glouton et stochastique [Basseur 2004]. Dans l'implémentation proposée, les individus de la population initiale sont générés de 2 façons : à 70 % de manière aléatoire en veillant à ce que les individus produits soient des solutions valides, et à 30 % en utilisant une heuristique gourmande basée sur l'Équation 4.1.

4.4.3 Sélection des individus

À chaque itération, lors de la phase de génération de la population d'enfants, la première étape effectuée par l'AG consiste à sélectionner des individus pour le croisement. La procédure de sélection utilisée est une sélection par tournoi, car cette technique de sélection n'est pas coûteuse à mettre en œuvre et à exécuter. Comme notre algorithme est élitiste, une faible taille de tournoi permet de limiter la pression de la sélection et par conséquence une convergence trop hâtive. C'est dans cette optique que, dans notre implémentation, la taille du tournoi est limitée à deux individus.

4.4.4 Application des opérateurs génétiques

Une fois sélectionnés, les individus sont croisés selon une certaine probabilité (p_c). Les opérateurs de croisement utilisés sont choisis parmi ceux présentés à la Section 4. On note que si le critère p_c n'est pas rencontré, alors un individu α est introduit dans la population. Cet individu, que l'on appelle *migrant,* est constitué à l'aide de l'heuristique gourmande utilisée lors de la création de la population initiale. Après avoir été générés par croisement, les enfants subissent une mutation selon une certaine probabilité (p_m). Les opérateurs de mutation utilisés sont sélectionnés par paire (*déplacement-inversion* ou *échange-mélange*)

parmi les quatre opérateurs présentés à la section précédente, chaque paire ayant une chance sur deux d'être tirée. Si la paire échange-mélange est sélectionnée, la probabilité de choisir un des deux opérateurs est respectivement de 0.99 et 0.01. Dans le cas où la paire déplacement-inversion est tirée, les probabilités de choisir l'un ou l'autre passe respectivement à 0.75 et 0.25. Finalement, on note que lorsqu'une mutation par déplacement est appliquée, on effectue à 90% du temps un $depl_{aléa}$. Mentionnons que les probabilités utilisées pour les différents opérateurs de croisement ont été déterminées de manière empirique.

4.4.5 Procédure de remplacement

Deux procédures de remplacement élitiste de la population sont utilisées dans l'algorithme génétique proposé. Un remplacement déterministe de type $(\mu+\lambda)$ et un de type *sélection inverse*. Le remplacement $\mu+\lambda$ consiste simplement à réunir les populations parent et enfant et à conserver les μ meilleurs individus. La stratégie de remplacement par sélection inverse consiste quant à elle, lorsque un enfant x donné est produit, à remplacer le pire individu de la population parent par l'individu x créé. Toutefois, une particularité de notre implémentation consiste à permettre le remplacement du pire individu de la population par un enfant ayant une moins bonne fitness dans 1% des cas afin d'éviter une convergence prématurée de l'algorithme. Comme chacun des opérateurs de croisement génèrent deux enfants, un remplacement de type $\mu+\lambda$ est appliqué en utilisant tous les premiers des deux enfants créés par croisement plus les éventuels migrants afin de former une population enfant de taille λ. Le deuxième des deux enfants sera introduit ou non dans

la population en utilisant le remplacement par sélection inverse. Il est important de préciser que dans notre algorithme $\mu = \lambda = N$.

4.5 Expérimentations numériques

Les différents algorithmes génétiques présentés dans cet article ont tous été implémentés 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 et tournant sous Windows XP.

Dans les expérimentations numériques réalisées pour évaluer la performance des différentes versions de l'algorithme génétique proposé, les paramètres N, p_c, p_m, $NbGen$ qui représentent respectivement la taille de la population, la probabilité de croisement, la probabilité de mutation et le nombre maximum de générations sont fixés à 250, 0.8, 0.09 et 700. Ces paramètres ont été déterminés de manière empirique afin d'obtenir une base de comparaison équitable avec d'autres algorithmes de la littérature.

4.5.1 Jeux d'essai

La performance des opérateurs de croisement est évaluée à l'aide de trois ensembles d'instances de POV disponibles sur Internet. Ces instances de problèmes présentent entre 100 et 400 voitures. Il s'agit des mêmes instances ayant servi à Gravel *et al.* [2005] et à Gagné *et al.* [2006] pour démontrer la performance de leurs algorithmes. Les trois ensembles sont tirés de CSPLib (http://www.csplib.org/). Le premier ensemble (ET1) contient 70 instances contenant chacune 200 voitures, 5 options et entre 17 et 30 classes de voitures. Ces 70 instances sont divisées en 7 groupes selon le taux d'utilisation. Pour

chacune de ces instances, il existe une solution sans conflit. Le deuxième ensemble (ET2) [Gent et Walsh 1999] est composé de 9 instances plus difficiles dont 5 pour lesquelles aucune solution sans conflits n'est connue. Chacune des instances de l'ET2 est composée de 100 voitures ayant chacune 5 options et entre 18 et 24 classes de voitures. Finalement, le dernier ensemble (ET3) proposé par Gravel *et al.*[2005] contient 30 instances difficiles de taille variant entre 200 et 400 voitures avec 5 options et entre 18 et 24 classes de voitures. Parmi les 30 instances de l'ET3, il y en a 23 pour lesquels aucune solution sans conflits n'est connue.

4.5.2 Comparaison expérimentale des différents opérateurs de croisement

L'objectif de la première série d'expérimentation est de comparer différentes versions de l'AG proposé avec d'autres AG de la littérature (GAcSP [Warwick et Tsang 1995] et le GA [Terada *et al.* 2006]) ainsi qu'avec un algorithme d'optimisation par colonie de fourmis qui s'est avéré particulièrement performant sur les différents jeux d'essai, l'ACS-3D [Gagné *et al.* 2008]. Les différentes versions de notre algorithme se divisent en trois : une version avec croisement NCPX (AG-NCPX), une version utilisant le croisement IBX (AG-IBX) et une version utilisant les deux opérateurs de croisement (AG-NCPX/IBX). On veut ainsi déterminer, d'une part, lequel des opérateurs proposés est le plus performant sur les différents ensembles tests et, d'autre part, démontrer la performance des AG proposés comparativement aux algorithmes de la littérature. On note que les résultats des différents algorithmes de la littérature ont été directement extraits des articles correspondants. De plus, les AG de la littérature n'ont été testés que sur les instances de l'ET1, la comparaison

entre ces approches et les méthodes proposées dans cette thèse article se limite donc à ces instances.

Les résultats de la comparaison du GAcSP, du GA, de l'ACS-3D, de l'AG-IBX, de l'AG-NCPX et de l'AG-NCPX/IBX sont présentés au Tableau 4.1. Chaque ligne du tableau donne le nom du groupe d'instances, suivi pour chaque algorithme du pourcentage d'exécutions ayant trouvé une solution sans conflit. Chaque groupe d'instances a été résolu à 100 reprises par le GAcSP, l'ACS-3D, l'AG-IBX, l'AG-NCPX et l'AG-NCPX/IBX et à 24 reprises par le GA. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. On constate une nette différence entre les résultats de l'AG-IBX, l'AG-NCPX, l'AG-NCPX/IBX et de l'ACS-3D et ceux du GAcSP et du GA. En effet, l'AG-IBX, l'AG-NCPX et l'AG-NCPX/IBX et l'ACS-3D obtiennent un taux de succès de 100 % pour toutes les instances de l'ET1. À l'opposé, les performances du GAcSP et du GA sont sérieusement dégradées pour les groupes d'instances 80, 85 et 90 et ce, même lorsque les auteurs augmentent le nombre de générations en le faisant passer de 1000 à 5000 générations [Terada *et al.* 2006].

Par ailleurs, l'écart entre les performances du GAcSP et du GA avec celles de l'AG-IBX, de l'AG-NCPX et de l'AG-NCPX/IBX mettent en évidence l'importance d'intégrer des opérateurs propres au problème considéré pour améliorer la performance globale des algorithmes. En effet, le GA proposé par Terada *et al.* est un algorithme classique utilisant un croisement à un point de coupure qui gagnerait probablement en performance s'il intégrait des opérateurs génétiques ou des heuristiques dédiés au POV. Les auteurs mentionnent, dans ce sens, que l'ajout d'une heuristique de réparation après croisement

améliore les performances de l'algorithme [Terada *et al.* 2006] pour les groupes d'instances 80-*, 85-* et 90-*. Ces résultats doivent cependant être nuancés, par le fait que le GAcSP n'a pas été testé sur toutes les instances de l'ET1 par ses auteurs et que le monde des métaheuristiques ainsi que l'informatique ont beaucoup évolué depuis la publication des résultats de Warwick et Tsang en 1995.

Cet ensemble test ne peut toutefois pas être utilisé pour différencier les performances des trois approches proposées entre elles ni par rapport à l'ACS-3D, mais il permet de mettre en valeur l'efficacité de l'AG-IBX, de l'AG-NCPX et de l'AG-NCPX/IBX sur des instances dites « faciles ».

Instance	GAcSP	GA	ACS-3D	AG-IBX	AG-NCPX	AG-NCPX/IBX
	%	%	%	%	%	%
60-*	19	100	100	100	100	100
65-*	-	100	100	100	100	100
70-*	23	100	100	100	100	100
75-*	-	80	100	100	100	100
80-*	9	16	100	100	100	100
85-*	-	2	100	100	100	100
90-*	-	1	100	100	100	100

Tableau 4.1 :Résultats expérimentaux du GAcSP [Warwick et Tsang 1995], du GA [Terada *et al.* 2006], de l'ACS-3D [Gagné *et al.* 2008], de l'AG-IBX, de l'AG-NCPX et de l'AG-NCPX/IBX sur l'ET1

Le Tableau 4.2 présente les résultats de l'AG-IBX, de l'AG-NCPX, de l'AG-NCPX/IBX et de l'ACS-3D pour les neuf instances de l'ET2. Chaque instance a été résolue à 100 reprises par chaque algorithme et 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 nombre de fois où chaque algorithme parvient à obtenir la meilleure solution connue (*# fois meilleure solution*). En ce qui concerne les

meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. En comparant, dans un premier temps, les nombres moyens de conflits obtenus par les trois algorithmes génétiques, on note que l'AG-NCPX et l'AG-NCPX/IBX obtiennent de meilleurs résultats que l'AG-IBX sur six des neuf instances tout en obtenant des résultats identiques sur les trois instances restantes. On remarque aussi que l'AG-NCPX permet d'obtenir plus régulièrement la meilleure solution connue que l'AG-IBX pour toutes les instances de l'ET2. En particulier, l'AG-NCPX obtient la meilleure solution connue à toutes les exécutions pour sept des neuf instances. Si on compare l'AG-NCPX et l'AG-NCPX/IBX, on remarque que l'AG-NCPX/IBX surclasse l'AG-NCPX sur deux instances tout en obtenant des résultats identiques sur les instances restantes. On note aussi que l'AG-NCPX/IBX obtient la meilleure solution connue à toutes les exécutions pour huit des neuf instances. Dans un deuxième temps, en comparant les résultats obtenus par les trois approches proposé à ceux de l'ACS-3D, on remarque que l'ACS-3D surclasse l'AG-IBX sur cinq instances, obtient de moins bon résultats sur deux instances tout en obtenant des performances identiques sur les deux instances restantes. Par contre, lorsque l'on compare maintenant les performances de l'AG-NCPX et l'AG-NCPX/IBX à celle de l'ACS-3D, on note, cette fois ci, que l'AG-NCPX et l'AG-NCPX/IBX obtiennent un nombre moyen de conflits inférieur à celui de l'ACS-3D sur six des neuf instances tout en obtenant des résultats identiques sur les trois autres instances. Si on s'intéresse maintenant au nombre de fois où chaque algorithme arrive à obtenir la meilleure solution connue, on constate qu'à ce niveau aussi, l'AG-NCPX et l'AG-NCPX/IBX dominent l'ACS-3D en obtenant plus régulièrement la meilleure solution connue, l'ACS-3D obtenant, de son coté plus

fréquemment la meilleure solution que l'AG-IBX. Cela se remarque en particulier pour le problème 10_93 où l'AG-NCPX et l'AG-NCPX/IBX parviennent à obtenir la meilleure solution connue respectivement à 45 et 52 reprises comparativement à 18 reprises pour l'ACS-3D et à aucune reprise pour l'AG-IBX.

Instance	Meilleure Solution Connue	AG-IBX		AG-NCPX		AG-NCPX/IBX		ACS-3D	
		Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution
10_93	3	4.00	0	3.55	45	3.48	52	4.03	18
16_81	0	0.65	35	0.03	97	0.00	100	0.58	47
19_71	2	2.26	74	2.00	100	2.00	100	2.04	96
21_90	2	2.25	75	2.00	100	2.00	100	2.02	98
26_82	0	0.01	99	0.00	100	0.00	100	0.00	100
36_92	2	2.41	59	2.00	100	2.00	100	2.03	97
4_72	0	0.00	100	0.00	100	0.00	100	0.01	99
41_66	0	0.00	100	0.00	100	0.00	100	0.00	100
6_76	6	6.00	100	6.00	100	6.00	100	6.00	100

Tableau 4.2 : Résultats obtenus par l'AG-IBX, l'AG-NCPX et de l'AG-NCPX/IBX et l'ACS-3D [Gagné *et al.* 2008] pour l'ET2

En complément, nous avons comparé la convergence des trois algorithmes génétiques sur les différentes instances de l'ET2 et la Figure 4.3 montre l'évolution du nombre moyen de générations pris par l'AG-NCPX, l'AG-IBX et l'AG-NCPX/IBX pour converger vers la meilleure solution en fonction de l'instance. On peut remarquer à l'aide du graphique que la convergence vers la meilleure solution se fait relativement rapidement pour les trois algorithmes. On note toutefois un net avantage pour l'AG-NCPX qui converge beaucoup plus rapidement que l'AG-IBX et l'AG-NCPX/IBX. En effet, l'AG-NCPX converge vers la meilleure solution en moyenne en moins de 90 générations alors que l'AG-NCPX/IBX, de son côté prend 110 générations et l'AG-IBX utilise en moyenne jusqu'à 303 générations. Cette situation peut s'expliquer par l'exploitation de l'information sur les positions ne

générant aucun conflit effectué par l'AG-NCPX qui permet à cet AG de mieux intensifier la recherche que son vis-à-vis l'AG-IBX.

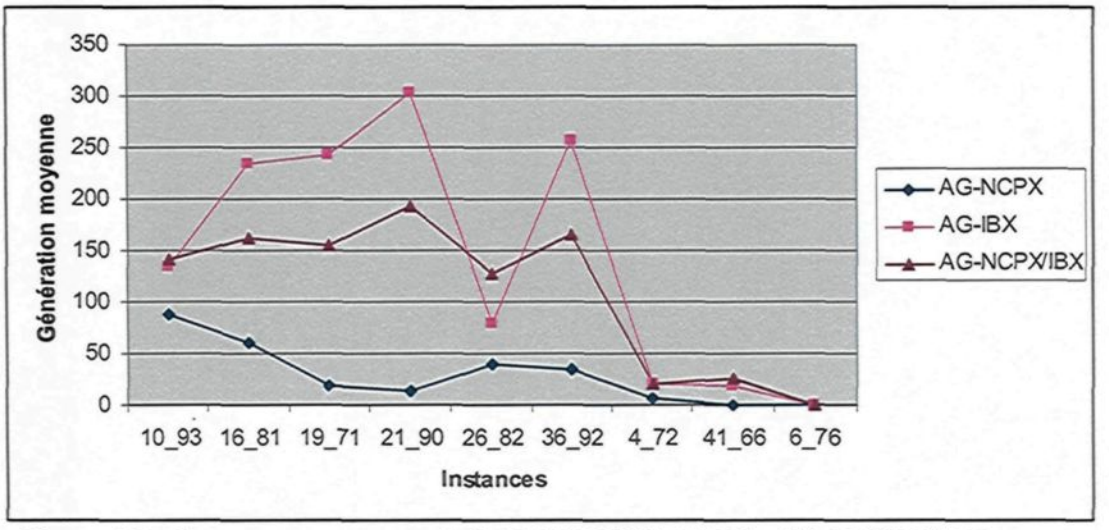


Figure 4.3 : Convergence moyenne de l'AG-NCPX, l'AG-IBX et l'AG-NCPX/IBX sur l'ET2

Le Tableau 4.3 présente, de manière similaire au Tableau 4.2, les résultats des quatre algorithmes pour les 30 instances de l'ET3. On note, en comparant les résultats de l'AG-IBX à ceux de l'AG-NCPX, que ce dernier domine l'AG-IBX sur 26 des 29 instances, obtient des résultats identiques sur 2 (200_06 et 200_08) et inférieurs sur les instances 200_10 et 300_05. La baisse du nombre moyen de conflits obtenu par l'AG-NCPX par rapport à l'AG-IBX varie entre 1 et 100 % pour la totalité des instances de l'ET3. Il est important de préciser ici que, même si ce groupe d'instances est plus difficile, l'AG-NCPX arrive à trouver les meilleures solutions connues pour 22 problèmes sur 30 alors que l'AG-IBX ne l'obtient que pour 11 instances. En comparant maintenant l'AG-NCPX à l'AG-NCPX/IBX, on remarque que l'AG-NCPX/IBX surclasse l'AG-NCPX sur toutes les

instances exceptées cinq où les deux algorithmes obtiennent des performances similaires. Si on compare maintenant les performances des trois approches proposés à celles de l'ACS-3D, on remarque, une fois de plus, que l'AG-NCPX et l'AG-NCPX/IBX obtiennent de meilleurs résultats que l'ACS-3D respectivement pour 26 et 27 instances et ont des performances similaires sur une instance (200_08). En fait, l'ACS-3D obtient de meilleurs résultats que l'AG-NCPX uniquement pour trois instances. Par rapport à l'AG-NCPX/IBX, l'ACS-3D obtient une meilleure moyenne pour seulement deux instances sur les trente. On note, par ailleurs, que l'écart entre les deux algorithmes génétiques et l'ACS-3D semble plus important pour les instances de plus grande taille. Lorsqu'on compare l'ACS-3D à l'AG-IBX, on note, cette fois ci, que l'ACS-3D surclasse l'AG-IBX sur 20 instances, obtient des résultats identiques sur une instance et inférieurs sur les dix instances restantes. En observant maintenant le nombre de fois où chaque algorithme arrive à atteindre la meilleure solution connue, on observe que l'AG-NCPX et l'AG-NCPX/IBX obtiennent, encore une fois, plus régulièrement la meilleure solution connue avec 22 instances sur 30 pour l'AG-NCPX comparativement à 14 instances pour l'ACS-3D et 11 pour l'AG-IBX. On note toutefois que pour le problème 200_02, l'ACS-3D obtient plus régulièrement la meilleure solution connue.

Instance	Meilleure Solution Connue	AG-IBX		AG-NCPX		AG-NCPX/IBX		ACS-3D	
		Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution
200_01	0	3.13	0	1.23	14	1.12	17	2.00	0
200_02	2	3.93	0	2.94	8	2.80	21	2.38	62
200_03	3	11.47	0	7.41	0	7.07	0	7.45	0
200_04	7	7.47	59	7.39	63	7.29	72	7.87	13
200_05	6	7.71	3	6.69	33	6.66	36	7.29	0
200_06	6	6.00	100	6.00	100	6.00	100	6.03	97
200_07	0	3.44	1	0.15	85	0.15	85	0.67	36
200_08	8	8.00	100	8.00	100	8.00	100	8.00	100
200_09	10	11.40	5	10.53	48	10.46	55	10.97	3
200_10	19	20.72	0	21.40	0	21.29	0	20.19	11
300_01	0	5.55	0	2.79	0	2.69	0	3.89	0
300_02	12	14.78	0	12.02	98	12.00	100	12.57	43
300_03	13	15.92	1	13.11	89	13.09	91	13.85	15
300_04	7	10.98	0	7.71	40	7.61	46	8.69	2
300_05	28	39.39	0	42.83	0	41.56	0	42.54	0
300_06	2	8.24	0	5.30	0	5.05	0	5.79	0
300_07	0	3.78	0	0.08	92	0.02	98	0.97	14
300_08	8	9.11	17	8.00	100	8.00	100	8.95	5
300_09	7	9.40	0	7.36	66	7.27	74	8.00	2
300_10	21	32.23	0	28.48	0	27.94	0	32.56	0
400_01	1	2.98	1	1.81	26	1.74	33	3.50	0
400_02	15	23.41	0	19.31	0	19.30	0	23.82	0
400_03	9	12.21	0	10.79	1	10.67	3	13.64	0
400_04	19	20.37	14	19.12	88	19.09	91	20.38	1
400_05	0	5.06	0	0.00	100	0.00	100	2.68	0
400_06	0	4.44	0	0.16	84	0.10	90	1.53	3
400_07	4	5.59	5	4.72	39	4.56	50	8.68	0
400_08	4	7.22	0	4.73	43	4.66	47	12.67	0
400_09	5	17.38	0	10.58	0	10.12	0	16.01	0
400_10	0	4.79	0	0.71	47	0.50	56	2.66	0

Tableau 4.3 : Résultats obtenus par l'AG-IBX, l'AG-NCPX et de l'AG-NCPX/IBX et l'ACS-3D [Gagné *et al.* 2008] pour l'ET3

En analysant maintenant la convergence des trois AG proposés sur l'ET3, on observe une fois de plus, à l'aide de la Figure 4.4, qu'aucun des algorithmes n'utilise les 700 générations disponibles et que l'AG-NCPX converge plus rapidement, en moyenne moins de 200 générations, que l'AG-IBX et l'AG-NCPX/IBX qui eux prennent en moyenne près de 300 générations pour converger.

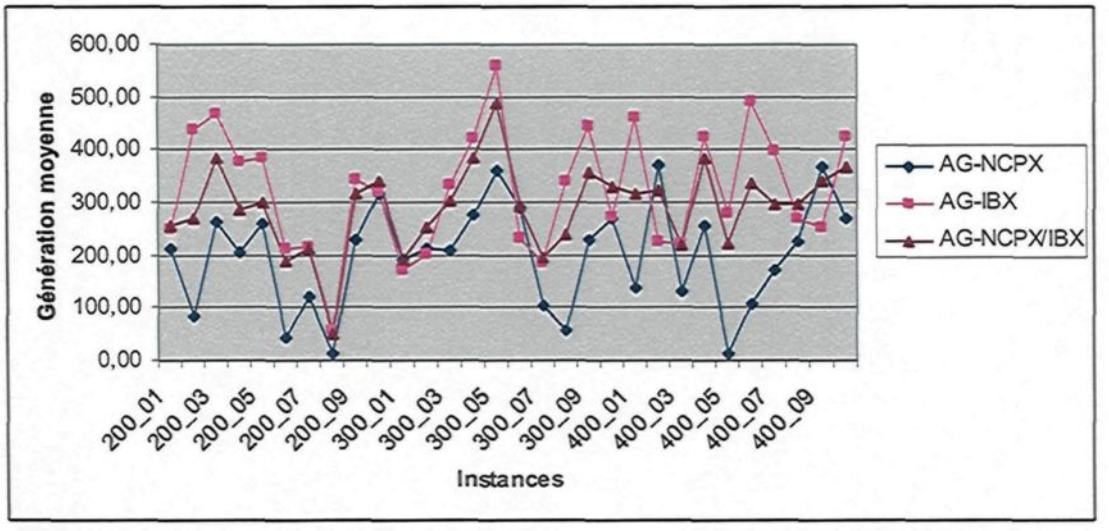


Figure 4.4 : Convergence moyenne de l'AG-IBX, l'AG-NCPX et l'AG-NCPX/IBX sur l'ET3

Les résultats obtenus dans cette première série d'expérimentation permettent de conclure, d'une part, que l'AG-NCPX/IBX obtient les meilleures performances sur l'ensemble de problèmes testés suivi par l'AG-NCPX et l'AG-IBX. Les bonnes performances du croisement NCPX par rapport au croisement IBX s'explique sans doute par l'exploitation de l'information sur les positions non conflictuelles opérées par le croisement NCPX. En effet, cette particularité du croisement NCPX semble lui permettre « d'apprendre » au fil des générations afin de l'aider à utiliser l'expérience passée pour intensifier plus rapidement la recherche par rapport au croisement IBX. Ces résultats montrent aussi que la combinaison des deux opérateurs de croisement dans un seul algorithme permet un gain au niveau des performances. En effet, l'utilisation simultanée des deux opérateurs de croisement semble apporter une diversité supplémentaire dans la population et ainsi évite une convergence trop rapide de l'algorithme. D'autre part, la

comparaison des performances des approches proposées par rapport à d'autres approches de la littérature permet de conclure que les AG sont des techniques d'optimisation efficaces pour résoudre le POV lorsque les spécificités du problème sont bien prises en compte. En effet, les performances de l'AG-NCPX/IBX et de l'AG-NCPX se comparent avantageusement à ceux des meilleurs résultats publiés à ce jour dans la littérature.

4.5.3 Ajout d'une procédure de recherche locale

Dans les expérimentations numériques suivantes, une procédure de recherche locale est combinée à l'AG-NCPX/IBX afin de comparer la performance de l'AG-NCPX/IBX avec recherche locale (AG-NCPX/IBX + LS) à celle de l'ACS-3D, lui aussi avec recherche locale. Dans l'ACS-3D, une procédure de recherche locale est appliquée sur la meilleure solution trouvée à chaque cycle ainsi que sur la meilleure solution globale à la fin de l'algorithme [Gagné *et al.* 2008]. Dans l'AG-NCPX/IBX, la procédure de recherche locale est appliquée avec une probabilité de 20% pendant les 250 premières générations de l'algorithme à la meilleure solution courante trouvée lorsque cette solution est améliorée ainsi que sur la meilleure solution obtenue par l'algorithme à la fin des 700 générations. Les Tableaux 4.4 et 4.5 présentent respectivement les résultats obtenus par les deux algorithmes avec recherche locale pour l'ET2 et l'ET3. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. Les instances de l'ET1 ne sont pas utilisées pour comparer les deux algorithmes avec recherche locale car les versions sans recherche locale des deux algorithmes solutionnent presque instantanément toutes ces instances. On note, par ailleurs, qu'une colonne supplémentaire est ajoutée dans les deux tableaux afin de montrer les résultats que l'on pourrait obtenir si on laissait l'AG-

NCPX/IBX + LS s'exécuter pendant dix minutes au lieu de limiter le nombre maximum de générations alloué à 700 générations comme dans les expérimentations précédentes. En effet, les paramètres précédemment utilisés avaient été déterminés de manière à obtenir une comparaison la plus équitable possible avec l'ACS-3D. En examinant les résultats des deux tableaux, on remarque que l'addition des procédures de recherche locale améliore nettement les performances de l'AG-NCPX/IBX et de l'ACS-3D sur les deux ensembles tests. En comparant maintenant la performance des deux algorithmes sur l'ET2, on remarque que l'AG-NCPX/IBX + LS obtient la meilleure solution connue à chaque exécution sur tous les problèmes sauf l'instance 10_93, l'écart pour ce problème est toutefois minime. L'ACS-3D n'obtenant, pour sa part, pas la meilleure solution connue pour les problèmes 10_93 et 16_81, l'écart pour le problème 16_81 est toutefois négligeable. L'AG-NCPX/IBX surclasse l'ACS-3D sur le problème 10_93 en obtenant la meilleure solution connue 97 fois sur 100 comparativement à 63 fois pour l'ACS-3D. Lorsqu'on exécute l'AG-NCPX/IBX + LS pendant dix minutes on obtient la meilleure solution connue à toutes les exécutions pour toutes les instances de l'ET2.

Instance	Meilleure Solution Connue	ACS-3D + LS		AG-NCPX/IBX+ LS		AG-NCPX/IBX + LS (10min)	
		Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution
10_93	3	3.37	63	3.03	97	3.00	100
16_81	0	0.03	97	0.00	100	0.00	100
19_71	2	2.00	100	2.00	100	2.00	100
21_90	2	2.00	100	2.00	100	2.00	100
26_82	0	0.00	100	0.00	100	0.00	100
36_92	2	2.00	100	2.00	100	2.00	100
4_72	0	0.00	100	0.00	100	0.00	100
41_66	0	0.00	100	0.00	100	0.00	100
6_76	6	6.00	100	6.00	100	6.00	100

Tableau 4.4 : Résultats comparatifs de l'ACS-3D + LS [Gagné *et al.* 2008] et l'AG-NCPX/IBX + LS sur l'ET2

Quand on compare maintenant les résultats sur l'ET3 à l'aide du Tableau 4.5, on constate, une fois de plus, que l'AG-NCPX/IBX surclasse l'ACS-3D sur la majorité des instances, à savoir 23 instances au total. Pour les sept instances restantes, l'ACS-3D et l'AG-NCPX/IBX obtiennent des résultats identiques. On note que lorsqu'on ne limite pas le nombre de générations à 700 l'AG-NCPX/IBX + LS parvient à trouver la meilleure solution connue à chaque exécution pour 28 des 30 instances de l'ET3.

Instance	Meilleure Solution Connue	ACS-3D + LS		AG-NCPX/IBX+ LS		AG-NCPX/IBX+ LS (10min)	
		Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution	Moyenne	# fois meilleure solution
200_01	0	0.41	61	0.14	86	0.00	100
200_02	2	2.00	100	2.00	100	2.00	75
200_03	3	5.76	0	5.13	0	3.25	100
200_04	7	7.00	100	7.00	100	7.00	100
200_05	6	6.16	84	6.00	100	6.00	100
200_06	6	6.00	100	6.00	100	6.00	100
200_07	0	0.00	100	0.00	100	0.00	100
200_08	8	8.00	100	8.00	100	8.00	100
200_09	10	10.00	100	10.00	100	10.00	100
200_10	19	19.02	98	19.00	96	19.00	100
300_01	0	1.87	1	0.08	92	0.00	100
300_02	12	12.01	99	12.00	100	12.00	100
300_03	13	13.02	98	13.00	100	13.00	100
300_04	7	7.70	42	7.26	74	7.00	100
300_05	28	32.53	0	32.08	0	28.90	10
300_06	2	3.59	10	2.55	50	2.00	100
300_07	0	0.08	92	0.00	100	0.00	100
300_08	8	8.00	100	8.00	100	8.00	100
300_09	7	7.18	82	7.03	97	7.00	100
300_10	21	22.25	16	21.30	69	21.00	100
400_01	1	2.55	1	1.29	69	1.00	100
400_02	15	17.46	2	16.75	4	15.00	100
400_03	9	10.08	4	10.07	6	9.00	100
400_04	19	19.01	99	19.00	100	19.00	100
400_05	0	0.02	98	0.00	100	0.00	100
400_06	0	0.10	90	0.00	100	0.00	100
400_07	4	5.58	5	4.09	91	4.00	100
400_08	4	5.24	21	4.00	100	4.00	100
400_09	5	7.31	2	6.69	9	5.00	100
400_10	0	0.89	28	0.00	100	0.00	100

Tableau 4.5 : Résultats comparatifs de l'ACS-3D + LS [Gagné *et al.* 2008] et l'AG-NCPX/IBX + LS sur l'ET3

4.6 Conclusion

Dans ce chapitre, nous avons proposé un algorithme génétique permettant de résoudre efficacement le problème théorique d'ordonnancement de voitures. En particulier, deux nouveaux opérateurs de croisement spécifiques pour la résolution du problème ont été proposés. Les performances des différents opérateurs de croisement ont été validées à l'aide de 3 ensembles test contenant au total 109 instances publiées. Les résultats expérimentaux ont montré, dans un premier temps, que le l'AG avec croisement NCPX obtenait les meilleurs résultats sur les instances testées par rapport au second opérateur de croisement proposé dans ce chapitre. Dans un deuxième temps, Nous avons aussi montré l'intérêt de combiner ces deux opérateurs de croisement. En effet, la combinaison des deux opérateurs de croisement dans un même algorithme permet d'obtenir des résultats se comparant avantageusement par rapport à ceux d'autres algorithmes réputés de la littérature pour ce problème. Chacun des travaux réalisés montre que nous avons su prendre en compte des connaissances du domaine afin de développer des opérateurs de croisement les mieux adaptés. Les résultats des différentes approches ont aussi permis de remarquer l'importance d'utiliser des opérateurs génétiques dédiés à la problématique et ce, même lorsque l'emploi d'opérateurs naturels est possible.

Finalement, les dernières expérimentations ont consisté à ajouter une procédure de recherche locale à l'algorithme génétique afin, d'une part, de confirmer les performances de l'approche proposée par rapport à une autre approche intégrant elle aussi des procédures de recherche locale. D'autre part, cet ajout a aussi permis de mettre en évidence l'importance des mécanismes d'intensification sur la qualité globale des solutions obtenues. Une fois de

plus, les résultats obtenus sont probants. En effet, l'hybridation d'une procédure de recherche locale à l'AG-NCPX/IBX améliore de façon notable ses performances qui restent supérieures à celles de l'autre algorithme en compétition. Ces derniers résultats sont encourageants car les mécanismes d'intégration utilisés dans ce chapitre pour incorporer la procédure de recherche locale sont très simplistes et laisse beaucoup de place à l'amélioration. Cette étape représente donc un premier pas vers l'intégration d'algorithmes d'intensification plus évolués de type heuristiques ou méthodes exactes. Le chapitre suivant approfondit cette avenue de recherche en proposant des approches de résolutions coopératives combinant un algorithme génétique avec d'autres méthodes d'intensification de la recherche.

CHAPITRE 5

ALGORITHMES COOPÉRATIFS POUR LE PROBLÈME THÉORIQUE D'ORDONNANCEMENT DE VOITURES

5.1 Introduction

Le chapitre précédent a permis de présenter une nouvelle approche permettant de résoudre efficacement le problème théorique d'ordonnancement de voitures en utilisant un AG. Les expérimentations numériques ont montré que l'algorithme proposé offre un bon compromis entre intensification et diversification du processus recherche notamment par l'intermédiaire de nouveaux opérateurs de croisement, de différentes méthodes de remplacement ainsi que par l'application de plusieurs opérateurs de mutation. Toutefois, l'ajout d'une procédure de recherche locale à l'AG proposé a permis d'améliorer de façon notable ses performances et de montrer l'intérêt de combiner cette approche avec d'autres méthodes d'intensification de la recherche afin de le rendre plus performant et robuste. Ce genre de combinaison communément appelée hybridation représente une avenue de recherche prometteuse sur laquelle de nombreux chercheurs se penchent depuis quelques années [Talbi 2002; Barichard 2003; Basseur 2004]. L'hybridation consiste à combiner les avantages de différentes méthodes d'optimisation en un seul algorithme. Selon Talbi [2002], ce type d'approche a permis d'obtenir de très bons résultats dans une grande variété de problèmes théoriques d'optimisation combinatoire 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 ce chapitre, nous proposons d'explorer certains mécanismes d'hybridation entre une métaheuristique à base de population et une méthode exacte pour résoudre le problème théorique d'ordonnancement de voitures sur une chaîne d'assemblage (*car-sequencing problem*). Le chapitre est organisé de la manière suivante : à la Section 5.2, nous présentons les principales classifications des stratégies d'hybridation métaheuristiques/méthode exactes. La Section 5.3 décrit la forme d'hybridation proposée entre un algorithme génétique et la programmation linéaire en nombres entiers. À la Section 5.4, nous présentons les résultats numériques obtenus sur les instances du problème théorique d'ordonnancement de voitures utilisées au chapitre précédent. Finalement, quelques conclusions et remarques sont présentées à la dernière section.

5.2 Classification des stratégies d'hybridation métaheuristiques/méthodes exactes

Les études sur l'hybridation entre métaheuristiques et méthodes exactes sont moins courantes que celle portant sur l'hybridation de deux métaheuristiques. Ce qui fait qu'il existe peu de classification pour ce type d'approche [Dimistrescu et Stützle 2003; Basseur 2004]. Toutefois, l'intérêt grandissant des chercheurs pour ce type d'approche et le nombre croissant de méthodes proposées ont récemment amené certains auteurs à proposer des classifications pour ce type de coopération. Parmi ces classifications on peut citer, sans être exhaustif, la classification de Dimitrescu et Stützle [2003], de Basseur [2004] ou encore de Puchinger et Raidl [2005].

Dans leur classification, Dimitrescu et Stützle [2003] distinguent cinq types d'approches de coopération métaheuristiques/méthodes exactes et détaillent une application trouvée dans la littérature pour chaque type d'approche. La classification proposée par les auteurs sépare les approches selon la méthode utilisée pour la coopération. Les cinq classes proposées par Dimitrescu et Stützle regroupent :

- les approches utilisant un algorithme exact pour explorer de larges voisinages dans un algorithme de recherche locale;
- les approches qui utilisent une méthode heuristique afin de réduire l'espace de recherche de la méthode exacte;
- les approches qui exploitent 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.

On note toutefois que même si l'état de l'art de Dimitrescu et Stützle survole les principales stratégies de coopération entre méthodes exactes et métaheuristiques, la plupart des exemples proposés par les auteurs utilisent des méthodes de recherches locales 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 dite de *bas niveau* lorsque les éléments fonctionnels qui constituent une 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 : un *mode relais* et un *mode simultané*. En mode relais, les 2 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 2 approches explorent en parallèle l'espace de recherche.

La dernière classification présentée, qui sera utilisée dans la suite de ce travail, est celle proposée par Puchinger et Raidl [2005]. Dans cette classification, 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 ou les méthodes hybridées échangent de l'information de manière séquentielle, parallèle ou entrelacée tout en conservant chacune leur intégrité. Au niveau des stratégies d'hybridations intégratives, une des deux approches est modifiée afin d'incorporer l'autre sous un modèle maître/esclave.

5.3 Algorithme génétique hybride

Dans ce chapitre, on propose d'hybrider un algorithme génétique avec un programme linéaire en nombre entiers (PLNE). Un PLNE consiste à formuler de façon mathématique un problème d'optimisation. Pour cela, on définit une fonction objectif linéaire à maximiser ou minimiser en fonction de contraintes, qui sont, elles aussi, exprimées de manière linéaire. Les variables de décisions doivent être entières et les solutions du programme doivent respecter les différentes contraintes du problème [Nedzela 1990]. La Figure 5.1 présente le programme linéaire proposé par Gravel *et al.* [2005] pour résoudre le POV.

Minimiser

$$F = \sum_{k=1}^{opt} \sum_{j=1}^{nc-sk+1} Y_{kj} \qquad (1)$$

Sous Contrainte

$$\sum_{v=1}^{V} C_{vj} = 1 \qquad pour\ j = 1..\ nc \qquad (2)$$

$$\sum_{j=1}^{nc} C_{vj} = c_v \qquad pour\ v = 1..\ V \qquad (3)$$

$$\sum_{v=1}^{V} \sum_{l=j}^{j+sk-1} O_{vk} * C_{vl} \leq r_k + M * Y_{kj} \qquad pour\ k = 1..\ opt\ et\ \ j = 1..\ (nc - s_k + 1) \qquad (4)$$

C_{vj} booléen $\quad pour\ v = 1..\ V\ et\ j = 1..\ nc$

Y_{kj} booléen $\quad pour\ k = 1..\ opt\ et\ j = 1..\ nc - s_k + 1$

Figure 5.1 : Un PLNE pour le POV [Gravel *et al,* 2005]

Dans la fonction objectif (1), les auteurs cherchent à minimiser le nombre de fois où une contrainte de capacité, exprimée à l'aide d'un ratio r_k/s_k pour une option k, sera dépassée en considérant chaque sous séquence de longueur s_k. Par la suite, la Contrainte (2) empêche de placer plus d'une classe de voitures à chaque position de la séquence. La Contrainte (3) permet de s'assurer que toutes les contraintes de production du problème sont respectées. Finalement, la dernière contrainte formalise les contraintes de capacité du problème tout en permettant de les enfreindre lorsqu'aucune solution sans conflit ne peut être trouvée.

Compte tenu de la difficulté du POV, le PLNE proposé par Gravel *et al.* [2005] n'a été testé que sur les instances de l'ET1 et l'ET2. Si les résultats de l'approche ont permis de confirmer les meilleures solutions connues pour certains problèmes, ils mettent aussi en évidence les limites de ce type d'approche lors de la résolution d'instances plus difficiles (plus de 16h d'exécution pour trouver la meilleure solution connue pour l'instance 10_93). Ce PLNE obtient donc des résultats intéressants sur le POV et il utilise la fonction

d'évaluation classique de la littérature. C'est pour cette raison que nous avons choisi ce modèle comme méthode de résolution exacte dans cette partie de la thèse.

La stratégie d'hybridation introduite dans ce chapitre vise donc à intégrer le PLNE de Gravel *et al.* [2005] lors du processus de croisement pour ainsi créer un *croisement hybride*. Deux croisements hybrides sont ainsi proposés respectivement à partir des opérateurs *NCPX* et *IBX* présentés au chapitre précédent.

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 5.2). 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 du programme linéaire en nombres entiers. Les mêmes étapes sont réalisées avec le deuxième parent de manière à créer un autre enfant.

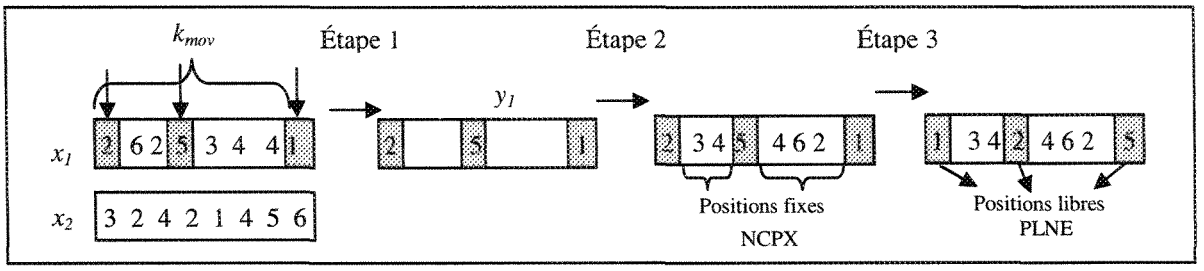


Figure 5.2 : Illustration du fonctionnement du croisement hybride A

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 comme le suggère Estellon *et al.* [2007] dans leur article. 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 fait 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 5.3). 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.

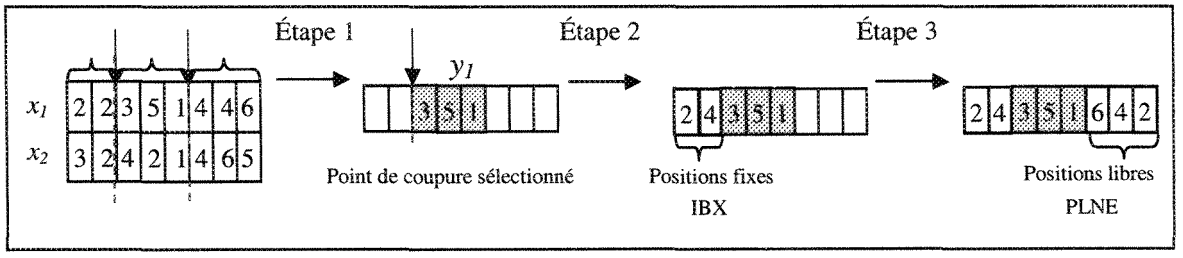


Figure 5.3 : Illustration du fonctionnement du croisement hybride B

L'algorithme 5.1 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 (Pop_{t-1}) : une *phase mixte* (300^{ières} générations), une *phase de diversification* (de la 301^{ième} à la 650^{ième} génération) et une *phase d'intensification* de la recherche (pendant les 50 dernières générations). La durée de chacune de ces trois phases a été fixée en relation avec les convergences moyennes obtenues par les AG proposés au chapitre précédent pour le même problème. 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. On note que, comme pour l'AG proposé au chapitre précédent, le critère d'arrêt utilisé dans ce chapitre correspond au nombre maximum de générations allouées à l'AG hybride.

Algorithme 5.1 : Algorithme génétique hybride pour le POV

Création aléatoire de la population initiale POP_0
Évaluer chaque individu $x \in POP_0$ et trier POP_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
      Tirer aléatoirement un nombre *rnd* entre *0* et *1*
      Si $rnd < p_m$ **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 $Q_t \cup POP_t$
  Conserver les μ meilleurs individus de $Q_t \cup POP_t$ pour former POP_{t+1}
Fin Tant que
Retourner le meilleur individu trouvé

5.4 Résultats et discussion

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 et *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.

Le Tableau 5.1 présente les expérimentations effectuées afin de déterminer le temps maximum alloué à CPLEX pendant la phase 1 et la phase 3. Ces résultats correspondent à une moyenne de 5 exécutions et sont obtenus en utilisant le problème 300_05 qui est, selon Estellon *et al.*[2007], un des plus difficiles à résoudre de l'ET3. À partir de ce tableau, on note que les meilleurs résultats en termes de qualité de solution et de temps d'exécution sont obtenus en accordant trois secondes à CPLEX dans la Phase 1 et dix secondes dans la Phase 3. En ce qui concerne la taille de k_{mov}, on note que pour chaque cas présenté au Tableau 5.3 elle varie en moyenne entre 57 et 71.5 positions. On rappelle que pour cette série d'expérimentation la taille initiale de k_{mov} est fixée à 15 positions (5% * 300 voitures) et qu'elle est par la suite incrémentée d'une position jusqu'à ce par le PLNE résolution échoue.

Limite de temps allouée à CPLEX		Moyenne	Temps (s)	k_{mov} maximum
Phase 1	Phase 3			
3	10	33.0	2601.9	70
5	10	33.6	2702.4	70
10	10	33.0	3346.8	71.5
3	3	38.5	1864.0	57
3	5	37.6	2380.6	66

Tableau 5.1 : Résultats obtenus en faisant varier le temps alloué à CPLEX dans la Phase 1 et la Phase 3 pour le problème 300_05

5.4.1 Jeux d'essai

La performance des AG hybrides est évaluée à l'aide des deux ensembles de problèmes de la librairie CSPLib (http://www.csplib.org/) non résolus de manière triviale au chapitre précédent, l'ET2 et l'ET3.

5.4.2 Comparaison expérimentale

Dans la série d'expérimentations suivantes, nous avons utilisé trois versions de notre algorithme hybride : une version avec croisement IBX ($ILPGA^{IBX}$), une version utilisant le croisement NCPX ($ILPGA^{NCPX}$) et une version utilisant les deux opérateurs de croisement ($ILPGA^{NCPX/IBX}$). Les performances des approches hybrides proposées dans cette thèse sont comparées aux résultats des deux algorithmes hybrides proposés Gravel *et al.* [2008] (*ACS-IH* et *GA-IH*), ainsi qu'à ceux du *VLNS* (very large neighbourhood search) de Estellon *et al.* [2007] et de l'approche hybride (*VNS* et *PLNE*) de Prandtstetter et Raidl [2007]. Le Tableau 5.2 résume les résultats obtenus par les différents algorithmes pour les neuf instances de l'ET2. On retrouve respectivement dans ce tableau, le nom de l'instance, la valeur de la meilleure solution connue et pour chaque algorithme le nombre moyen de conflits (*moyenne*) ainsi que le temps moyen d'exécution en secondes (*temps*) des six premiers algorithmes. Il est important de noter que les temps moyens d'exécution 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. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués en caractères gras.

Les expérimentations numériques de l'AG^{NCPX} et de l'AG^{IBX} ayant été effectuées en utilisant un ordinateur différent, nous avons donc utilisé les références suivantes : http://www.spec.org/cpu/results/res2004q2/cpu2000-20040329-02938.html et http://www.spec.org/cpu/results/res2004q2/cpu2000-20040614-03077.html afin de déterminer le ratio de performances entre nos deux ordinateurs. À l'aide de ces deux sites Internet, on note que le temps d'exécution sur 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 est en moyenne de 16% à 43% plus lent qu'un ordinateur équipé d'un processeur Pentium Xeon 3.6 Ghz avec 1 Go de mémoire vive et tournant sous Windows XP. Nos propres expérimentations numériques montrent que, pour les AG présentés au chapitre précédent, ce ratio est d'environ 33%. Ainsi, les temps d'exécution de l'AG^{NCPX} et de l'AG^{IBX} présentés au Tableau 5.2 doivent être augmentés de 33% pour une comparaison équitable avec l'*ILPGA*^{NCPX} et l'*ILPGA*^{IBX}.

Chaque instance a été résolue 10 fois par l'*ILPGA*^{NCPX}, l'*ILPGA*^{IBX}, le *VLNS* [Estellon *et al.* 2007] et le *VNS/ILP* [Prandtstetter et Raidl 2007], 100 fois par l'AG^{NCPX} et l'AG^{IBX} et 5 fois par l'*ACS-IH* et l'*AG-IH*. Le lecteur peut consulter Gravel *et al.* [2008] pour plus de détails sur l'implémentation de l'*ACS-IH et de l'AG-IH*.

On note que, pour le *VNS/ILP*, les auteurs [Prandtstetter et Raidl 2007] n'indiquent ni le type d'hybridation utilisée ni les temps d'exécution de leur approche. Finalement, il est

important de mentionner qu'il est impossible de comparer les temps d'exécution du *VLNS* avec ceux de nos approches car les auteurs présentent uniquement les temps de calcul pris par leur algorithme pour atteindre le nombre moyen de conflits présenté dans leur article.

L'analyse comparative des résultats de l'AG^{NCPX} et de l'$ILPGA^{NCPX}$ montre que l'approche hybride obtient la meilleure solution connue à chaque exécution pour toutes les instances de l'ET2 tandis que l'AG^{NCPX} trouve la meilleure solution connue pour 7 des 9 instances. Lorsque l'on compare maintenant l'AG^{IBX} avec l'$ILPGA^{IBX}$, on note que l'approche hybride obtient de meilleurs résultats pour 6 instances tout en obtenant la meilleure solution connue à chaque exécution pour 8 des 9 instances. Lorsqu'on compare maintenant les deux algorithmes hybrides, on constate que l'$ILPGA^{NCPX}$ obtient de meilleurs résultats que l'$ILPGA^{IBX}$ pour une instance tout en obtenant des résultats identiques pour les 8 instances restantes. On note aussi que l'$ILPGA^{NCPX}$ s'exécute plus rapidement que l'$ILPGA^{IBX}$ pour toutes les instances de l'ET2. À l'aide du Tableau 5.2, on remarque aussi que l'$ILPGA^{NCPX}$, l'AG-IH, l'ACS-IH et le VLNS obtiennent des résultats identiques (*moyenne*) pour toutes les instances à l'exception de l'instance 10-93 pour laquelle l'ACS-IH est moins bon que les 3 autres algorithmes. Finalement, le VNS/ILP obtient les pires résultats pour toutes les instances à l'exception de l'instance 41_66.

		AG^{NCPX}		$ILPGA^{NCPX}$		AG^{IBX}		$ILPGA^{IBX}$		*ACS-IH*		*GA-IH*		*VLNS*	*VNS/ILP*
Instance	**Meilleure Solution Connue**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Moyenne**
10_93	3	3.55	42.3	**3.00**	585.1	4.00	45.3	**3.20**	626.9	3.40	1757.6	**3.00**	2567.4	**3.00**	16.7
16_81	0	0.03	4.6	**0.00**	28.8	0.65	39.2	**0.00**	71.8	**0.00**	37.4	**0.00**	46.2	**0.00**	19.6
19_71	2	**2.00**	38.4	**2.00**	529.8	2.26	43.3	**2.00**	911.8	**2.00**	2796.4	**2.00**	3286.4	**2.00**	9.5
21_90	2	**2.00**	34.8	**2.00**	618.7	2.25	40.0	**2.00**	707.7	**2.00**	2758.6	**2.00**	3490.8	**2.00**	4.1
26_82	0	**0.00**	2.0	**0.00**	36.1	0.01	5.2	**0.00**	142.6	**0.00**	0.4	**0.00**	3.8	**0.00**	3.4
36_92	2	**2.00**	36.4	**2.00**	405.6	2.41	39.7	**2.00**	623.6	**2.00**	2589.4	**2.00**	3071.6	**2.00**	9.4
4_72	0	**0.00**	0.4	**0.00**	10.6	**0.00**	1.3	**0.00**	19.9	**0.00**	3.8	**0.00**	0.4	**0.00**	7.9
41_66	0	**0.00**	0.1	**0.00**	0.7	**0.00**	1.1	**0.00**	1.0	**0.00**	0.2	**0.00**	0.01	**0.00**	0.0
6_76	6	**6.00**	31.7	**6.00**	621.2	**6.00**	37.5	**6.00**	915.2	**6.00**	4180.2	**6.00**	3687.8	**6.00**	7.3

Tableau 5.2 : Résultats de l'AG^{NCPX} vs $ILPGA^{NCPX}$, AG^{IBX} vs $ILPGA^{IBX}$, ACS-IH, AG-IH, VLNS et VNS/ILP sur les instances l'ET2

Le Tableau 5.3 présente les résultats obtenus par les différents algorithmes pour les instances de l'ET3 de manière similaire au Tableau 5.2. Il est important de préciser que Prandtstetter et Raidl [2007] n'ont pas utilisé leur algorithme pour résoudre cet ensemble d'instances. Les résultats présentés dans la première partie du Tableau 5.3 montrent que le croisement hybride *A* utilisé dans l'algorithme $ILPGA^{NCPX}$ améliore la qualité des solutions en moyenne de 26,39 % par rapport à l'AG^{NCPX}. Il permet même d'obtenir la meilleure solution connue à chacune des exécutions pour 16 des 30 problèmes, soit 12 de plus que l'AG^{NCPX}. Ces résultats sont indiqués en gras dans le tableau. 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 l'AG^{NCPX} avec, en moyenne, 25 minutes pour l'$ILPGA^{NCPX}$ contre 125 secondes pour l'AG^{NCPX} (en ajustant les temps d'exécution en fonction des différents ordinateurs utilisés).

		AG^{NCPX}		$ILPGA^{NCPX}$		AG^{IBX}		$ILPGA^{IBX}$		*ACS-IH*		*GA-IH*		*VLNS*
Instance	**Meilleure Solution Connue**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**
200_01	0	1.23	67.73	**0.00**	1147.00	3.13	77.28	0.90	1992.8	0.00	788.8	0.00	526.2	0.1
200_02	2	2.94	69.73	2.20	1287.00	3.93	76.05	**2.00**	2273.3	2.80	2942.2	2.60	3021.0	2.4
200_03	3	7.41	79.44	5.30	1841.40	11.47	81.10	**4.60**	2240.0	6.00	3086.2	5.00	3215.2	5.8
200_04	7	7.39	77.68	**7.00**	1343.70	7.47	80.43	7.10	2301.4	7.00	2853.2	7.00	2875.2	7.2
200_05	6	6.69	71.40	**6.00**	1091.70	7.71	74.71	6.10	1842.6	6.00	2785.2	6.00	3052.2	6.0
200_06	6	**6.00**	71.55	**6.00**	1773.60	**6.00**	75.92	**6.00**	2031.7	6.00	3360.0	6.00	3508.0	6.0
200_07	0	0.15	21.92	**0.00**	413.90	3.44	76.35	**0.00**	1583.9	0.00	199.8	0.00	192.8	0.0
200_08	8	**8.00**	62.29	**8.00**	1604.80	**8.00**	70.32	**8.00**	1669.0	8.00	3609.2	8.00	3540.8	8.0
200_09	10	10.53	75.41	**10.00**	1567.80	11.40	78.25	**10.00**	2068.3	10.00	3167.6	10.00	3264.0	10.0
200_10	19	21.40	67.40	**19.00**	1422.30	20.72	69.08	19.10	2040.4	19.60	3205.0	19.60	3190.2	19.0
300_01	0	2.79	103.98	0.80	1325.60	5.55	115.81	3.00	2240.9	1.00	3125.0	1.00	3216.8	1.2
300_02	12	12.02	112.73	**12.00**	1461.10	14.78	116.52	12.10	2250.8	12.40	3153.0	12.00	3335.6	12.0
300_03	13	13.11	120.63	**13.00**	1554.80	15.92	123.90	**13.00**	2349.6	13.20	3398.2	13.00	3621.2	13.0
300_04	7	7.71	115.98	7.50	1620.50	10.98	117.49	7.80	2659.2	8.20	3459.6	7.80	3380.6	8.8
300_05	27	42.83	115.90	33.00	2601.90	39.39	118.54	**31.90**	2855.7	33.00	3151.8	32.40	3343.0	33.1
300_06	2	5.30	106.61	3.60	1597.70	8.24	115.14	4.70	2855.2	3.60	3187.4	2.80	3505.0	3.1
300_07	0	0.08	23.17	**0.00**	426.40	3.78	116.77	**0.00**	2217.4	0.00	434.2	0.00	45.6	0.0
300_08	8	**8.00**	105.03	**8.00**	1680.10	9.11	113.27	**8.00**	2298.7	8.00	3497.2	8.00	3618.0	8.0
300_09	7	7.36	105.31	7.20	1649.70	9.40	108.79	7.30	2433.5	7.60	3310.4	7.00	3375.0	8.0
300_10	21	28.48	100.37	22.4	2338.90	32.23	106.13	22.5	2926.1	23.40	3265.0	22.2	3540.6	21.4
400_01	1	1.81	137.89	1.90	1782.00	2.98	151.98	2.20	3313.6	1.60	3383.6	1.00	3713.2	1.6
400_02	15	19.31	147.51	18.30	2081.10	23.41	148.10	18.70	2502.6	17.40	3531.4	17.00	3798.0	16.3
400_03	9	10.79	128.73	9.90	2338.70	12.21	143.06	10.40	3146.2	9.60	3716.2	9.80	4232.2	12.0
400_04	19	19.12	164.57	**19.00**	2354.90	20.37	161.75	**19.00**	3287.1	19.00	3790.4	19.00	4293.4	20.1
400_05	0	**0.00**	2.45	**0.00**	19.80	5.06	139.81	**0.00**	2860.3	0.00	149.6	0.00	4.2	0.0
400_06	0	0.16	38.69	**0.00**	604.80	4.44	146.06	**0.00**	2461.0	0.00	416.8	0.00	64.4	0.0
400_07	4	4.72	130.98	4.70	1479.50	5.59	141.81	4.90	2785.8	4.20	3405.4	4.00	3592.2	4.6
400_08	4	4.73	131.45	4.20	1203.30	7.22	144.15	5.80	2789.0	4.00	3164.0	4.20	3529.6	4.1
400_09	5	10.58	149.18	7.20	2136.50	17.38	155.08	8.70	3312.4	7.20	3421.0	6.00	3870.0	8.3
400_10	0	0.71	102.27	**0.00**	1890.90	4.79	147.75	0.20	2613.2	0.00	98.0	0.00	248.0	0.0

Tableau 5.3 : Résultats de l'AG^{NCPX} vs $ILPGA^{NCPX}$, AG^{IBX} vs $ILPGA^{IBX}$, ACS-IH, AG-IH et VLNS sur les instances de L'ET3

La Figure 5.4 montre le pourcentage de fois où chaque phase trouve la meilleure solution. On remarque, à l'aide de cette figure, 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 dans la Phase 2 (phase de diversification) de l'algorithme 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 par la Phase 1 (phase mixte) contre 36% pour la Phase 3 (phase d'intensification) où seul le croisement hybride est utilisé. Pour les groupes de 300 et 400 voitures la meilleure solution est trouvée généralement par la Phase 3 dans respectivement 54 et 56 % des cas contre 42 et 44% pour la Phase 1.

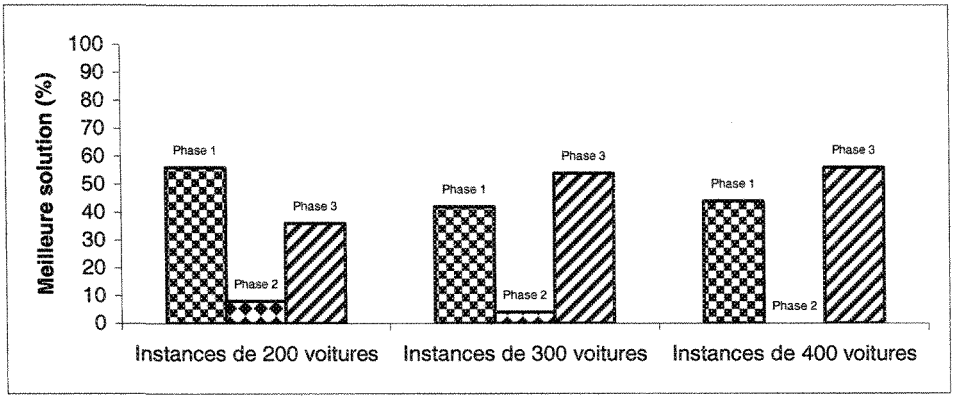


Figure 5.4: Pourcentage de fois où la meilleure solution est trouvée durant la Phase 1, Phase 2 et la Phase 3 de l'ILPGA^{NCPX}

À titre de comparaison, la Figure 5.5 indique, de manière similaire à la Figure 5.4, le pourcentage de fois où chaque étape de l'AG^{NCPX} trouve la meilleure solution. On note ainsi que dans 76% des cas la meilleure solution est trouvée avant la 300^{ième} génération, dans 23% des cas entre la 301^{ième} et la 650^{ième} génération et seulement dans 1% des cas

après la 650^{ième} génération. Il est important de mentionner que ces statistiques ont été utilisées pour définir les différentes phases des algorithmes hybrides.

En comparant ces résultats à ceux de la Figure 5.4, on remarque que la combinaison AG/PLNE permet à l'algorithme hybride de se sortir du piège de l'optimum local. Ainsi, l'*ILPGA*^{NCPX} continue à améliorer la qualité des solutions trouvées après la 301^{ième} génération alors que l'AG ^{NCPX} a, en général, stagné après cette génération. La contribution du PLNE sur la qualité finale des solutions de l'*ILPGA*^{NCPX} est importante puisque c'est en général dans les phases où il est utilisé que la meilleure solution est trouvée.

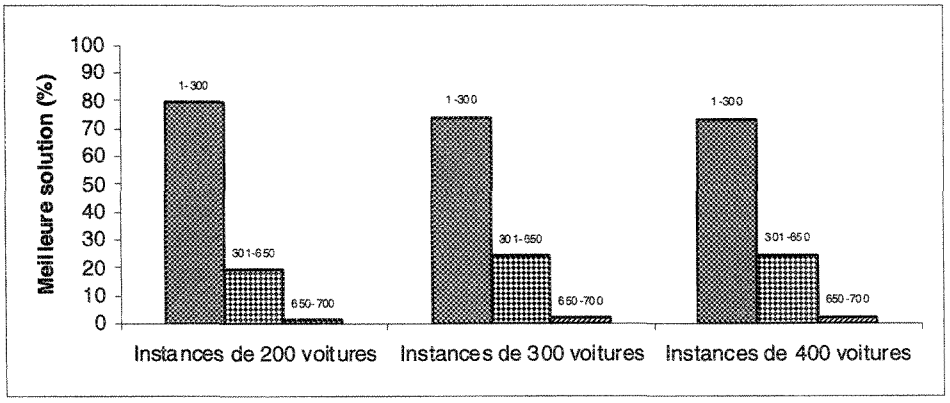


Figure 5.5: Pourcentage de fois où chaque étape de l'AG^{NCPX} trouve la meilleure solution

La seconde partie du Tableau 5.2 présente les résultats comparatifs entre l'AG^{IBX} et l'*ILPGA*^{IBX} sur l'ET3. Lorsque l'on compare la performance de l'approche hybride par rapport au l'*AG*^{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 instances. Ainsi, l'*ILPGA*^{IBX} obtient la meilleure solution connue à chaque exécution pour 11 des 30 problèmes contre seulement deux pour l'AG^{IBX}. Par ailleurs, on note, en observant les résultats plus attentivement, que contrairement à l'*ILPGA*^{NCPX} les gains moyens obtenus par

l'*ILPGA*^{IBX} par rapport à l'A*G*^{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 de l'A*G*^{IBX} qui sont inférieures à celles de l'A*G*^{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 celui de l'A*G*^{IBX}.

Lorsque l'on veut maintenant savoir à quelle étape de l'*ILPGA*^{IBX} la meilleure solution est découverte, on remarque à l'aide de la Figure 5.6 que les résultats sont biens différents de ceux obtenus avec l'*ILPGA*^{NCPX}. En effet, pour les trois groupes d'instances on observe que la meilleure solution est essentiellement trouvée lors de la phase d'intensification de la recherche c'est-à-dire la Phase 3 de l'algorithme, en moyenne dans 82% des cas, contre seulement 6% et 12% des cas respectivement pour la Phase 1 et 2.

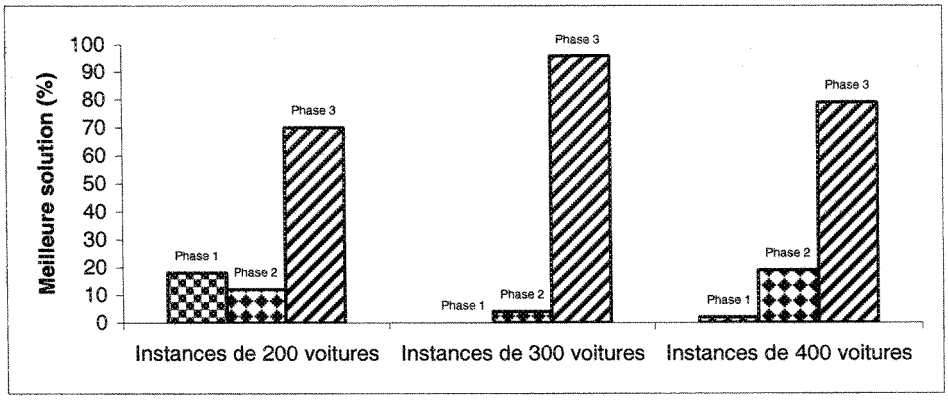


Figure 5.6: Pourcentage de fois où la meilleure solution est trouvée durant la Phase 1, Phase 2 et la Phase 3 de l'ILPGA^{IBX}

La Figure 5.7 présente, en utilisant les mêmes phases que l'ILPGA^{IBX}, le pourcentage de fois où chaque phase trouve la meilleure solution lors de l'exécution de l'*AG*^{IBX}. On note ainsi que l'*AG*^{IBX} converge vers la meilleure solution dans 49% des cas avant la 300^{ième} génération, dans 44% des cas entre la 301^{ième} et la 650^{ième} génération et seulement dans 7% des cas après la 650^{ième} génération. Si on compare ces résultats avec ceux de la Figure 5.6, on peut en conclure, une fois de plus, que la version hybride permet d'éviter une convergence prématurée de l'algorithme et ainsi de sortir du piège de l'optimum local.

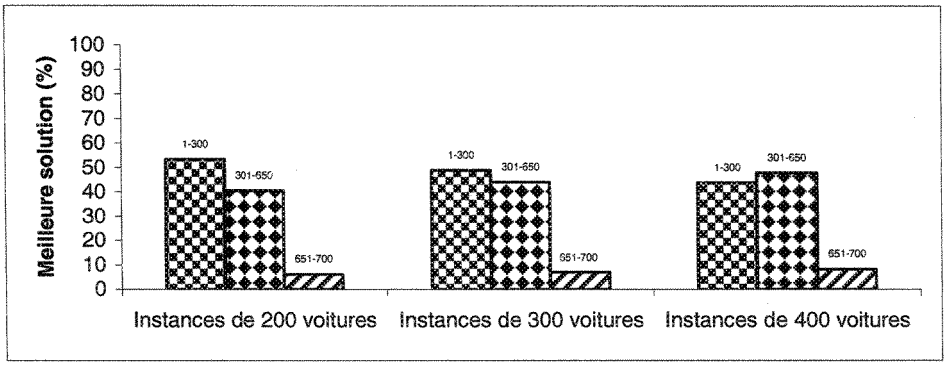


Figure 5.7: Pourcentage de fois où chaque étape de l'AG^{IBX} trouve la meilleure solution

Globalement, on observe que l'*ILPGA*^{NCPX} obtient de meilleurs résultats l'*ILPGA*^{IBX} pour 17 des 30 instances alors que l'*ILPGA*^{IBX} est meilleur pour seulement 3 instances. Ces résultats montrent la supériorité du croisement hybride *NCPX* par rapport au croisement hybride *IBX*.

Lorsqu'on compare maintenant les performances de l'*ILPGA*^{NCPX} et de l'*ILPGA*^{IBX} à celles de l'*ACS-IH*, on remarque que l'*ILPGA*^{NCPX} surclasse l'*ACS-IH* sur 9 instances, obtient de moins bons résultats pour 5 instances tout en obtenant des performances identiques sur les instances restantes. De plus, le temps moyen d'exécution de l'*ACS-IH* (2702 secondes) est plus important que celui de l'*ILPGA*^{NCPX} (1521 secondes). On en

conclut donc qu'en accordant le même temps moyen d'exécution aux deux algorithmes la différence serait encore plus importante. De son côté, l'*ILPGA*^{IBX} obtient de meilleurs résultats que l'*ACS-IH* pour 9 instances, est moins bon sur 12 instances et obtient une performance identique sur les instances restantes. Dans ce cas, les temps d'exécution des deux algorithmes sont comparables. En effet, le temps moyen d'exécution de l'*ACS-IH* est de 2702 secondes contre 2473 secondes pour l'*ILPGA*^{IBX}. Toutefois, on note que l'*ACS-IH* effectue plus d'évaluations que l'*ILPGA*^{NCPX} et l'*ILPGA*^{IBX}.

Si on compare maintenant l'*ILPGA*^{NCPX} et l'*ILPGA*^{IBX} à l'*AG-IH*, on remarque cette fois-ci que l'*ILPGA*^{NCPX} obtient de meilleurs résultats pour 4 instances et est moins bon sur 10 instances. Toutefois, le temps moyen d'exécution de l'*AG-IH* (2825 secondes) est beaucoup plus important que celui de l'*ILPGA*^{NCPX} (1521 secondes). De son côté, l'*ILPGA*^{IBX} surclasse l'*AG-IH* sur 5 instances et est moins bon sur 14 instances. Le temps moyen d'exécution l'*ILPGA*^{IBX} (2473 secondes) est légèrement inférieur à celui de l'*AG-IH* (2825 secondes). Ces résultats ne sont pas réellement surprenants, en effet l'*AG-IH* utilise aussi l'opérateur de croisement *NCPX* proposé au chapitre précédent et effectue plus de générations que les deux approches proposées dans ce chapitre. De plus, nous allons montrer ultérieurement qu'une version de notre algorithme intégrant les deux opérateurs de croisement hybrides permet de surclasser clairement l'*AG-IH*.

Si on compare les performances de l'*ILPGA*^{NCPX} et de l'*ILPGA*^{IBX} avec celles du *VLNS* qui sont présentées dans la dernière colonne du Tableau 5.2, on en conclut que les deux approches hybrides obtiennent de meilleurs résultats. En effet, l'*ILPGA*^{NCPX} surclasse le *VLNS* sur 24 des 30 instances tandis que l'*ILPGA*^{IBX} obtient de résultats au moins

identiques à ceux du VLNS pour 17 des 30 instances. Ainsi, on peut en conclure que les heuristiques à base de populations, comme les AG, contribuent grandement sur les performances des approches hybrides proposées dans cette thèse. En effet, contrairement à l'*ILPGA*^{NCPX} et à l'*ILPGA*^{IBX} le *VLNS* est une heuristique à solution unique qui appelle de manière itérative un PLNE.

Au Tableau 5.4, nous analysons l'influence de la valeur initiale de k_{mov} sur la qualité de la solution finale et sur le temps d'exécution de l'*ILPGA*^{NCPX}. Ces résultats ont été obtenus sur le problème 300_05 en accordant 3 secondes à CPLEX dans la première phase et 10 secondes dans la 3^{ième}. Les résultats présentés dans ce tableau correspondent aux résultats moyens obtenus après 5 exécutions consécutives de l'*ILPGA*^{NCPX}. À partir de ce tableau, on note que la valeur initiale de k_{mov} peut varier entre 5% et 15% de la taille du problème (ce qui correspond à entre 15 et 45 positions pour ce problème) sans que la qualité de la solution finale soit dégradée. Entre cet intervalle, les temps d'exécution de l'algorithme sont équivalents. Dans leur article, Estellon *et al.* [2007] utilisent un k_{mov} compris entre 10 et 50 positions pour tous les problèmes. Cependant, nous pensons qu'il est plus judicieux de commencer avec une plus petite valeur initiale de k_{mov} et de l'augmenter progressivement. Pour des problèmes plus difficiles comme le problème 300_05, cette stratégie permet d'éviter que la résolution du PLNE par CPLEX échoue dans le temps imparti.

*% * nc*	**moyenne**	**Temps (s)**	k_{mov} **maximum**
3%	33.6	2049.8	70
5%	33.0	2702.4	70
10%	33.0	2846.8	71
15%	33.0	2586.4	71
20%	33.2	2456.6	64.6

Tableau 5.4: Résultats obtenus en faisant varier la valeur initiale de k_{mov} pour le problème 300_05

Les croisements hybrides présentés dans ce chapitre ont un fonctionnement différent l'un de l'autre sans pour autant être complètement opposées. Ainsi, on peut donc facilement les combiner dans un même algorithme en supposant que leur utilisation simultanée peut amener un certain gain aussi bien en termes de diversification de la recherche que de qualité finale des solutions. Dans l'algorithme résultant (*ILPGA*$^{NCPX/IBX}$), les différents opérateurs de croisement sont appliqués de la manière suivante en fonction de la phase de création de la population enfant :

- Phase 1: utilisation aléatoire d'un des deux croisements hybrides (croisement hybride *A ou B)* pour créer 10 % de la population Enfant et à parts égales du croisement *NCPX* et *IBX* pour le reste de la population;
- Phase 2: utilisation du croisement NCPX dans 65 % des cas et du croisement IBX pour le reste;
- Phase 3 : utilisation aléatoire d'un des deux croisements hybrides (*A ou B).*

La supériorité du croisement *NCPX* par rapport au croisement *IBX* observée au chapitre précédent explique les pourcentages utilisés pour les deux opérateurs à la Phase 2. Les résultats de l'*ILPGA*$^{NCPX/IBX}$ sont présentés dans la partie gauche du Tableau 5.5.

Lorsque l'on compare les résultats de l'ILPGA$^{NCPX/IBX}$ à ceux de l'*ILPGA*^{NCPX} et de l'*ILPGA*^{IBX}, on note que l'approche mixte fournit globalement de meilleurs résultats que les deux autres algorithmes. En effet, l'ILPGA$^{NCPX/IBX}$ obtient des résultats identiques ou meilleurs à ceux de l'*ILPGA*^{NCPX} et de l'*ILPGA*^{IBX} pour 29 des 30 instances. De plus, si on compare maintenant les résultats de l'ILPGA$^{NCPX/IBX}$ à ceux de l'*ACS-IH* et de l'A*G-IH,* on

note que l'ILPGA$^{NCPX/IBX}$ obtient de meilleurs résultats pour 29 des 30 instances par rapport à l'ACS-IH et pour 28 des 30 instances par rapport à l'AG-IH. On peut donc conclure que l'intégration des différents opérateurs de croisement dans l'ILPGA$^{NCPX/IBX}$ permet à l'algorithme de mieux diversifier sa recherche tout en conservant un bon équilibre entre intensification et diversification.

	ILPGA$^{NCPX/IBX}$		*AG$^{NCPX/IBX}$* + LS		*VLNS*
Instance	**Moyenne**	**Temps (s)**	**Moyenne**	**Temps (s)**	**Moyenne**
200_01	**0.00**	2155.1	0.14	47.50	0.1
200_02	**2.00**	1620.1	**2.00**	89.99	2.4
200_03*	**4.10**	1456.5	5.13	101.21	5.8
200_04	**7.00**	2344.9	**7.00**	98.01	7.2
200_05	**6.00**	1637.0	**6.00**	90.39	6.0
200_06	**6.00**	2031.7	**6.00**	93.71	6.0
200_07	**0.00**	1583.9	**0.00**	10.11	0.0
200_08	**8.00**	1669.0	**8.00**	80.44	8.0
200_09	**10.00**	1771.0	**10.00**	97.26	10.0
200_10	**19.00**	1739.7	**19.00**	83.91	19.0
300_01	0.30	1606.9	**0.08**	135.29	1.2
300_02	**12.00**	1882.9	**12.00**	145.60	12.0
300_03	**13.00**	2222.6	**13.00**	157.54	13.0
300_04*	**7.00**	1357.0	7.26	140.05	8.8
300_05*	**30.50**	2045.6	32.08	147.88	33.1
300_06*	2.60	1454.9	**2.55**	137.86	3.1
300_07	**0.00**	879.2	**0.00**	11.60	0.0
300_08	**8.00**	2064.6	**8.00**	148.46	8.0
300_09*	7.30	2197.7	**7.03**	139.73	8.0
300_10	**21.20**	2725.7	21.30	134.04	21.4
400_01*	**1.10**	2594.9	1.29	171.78	1.6
400_02*	**16.70**	2101.4	16.75	175.24	16.3
400_03*	**9.60**	2761.2	10.07	160.11	12.0
400_04	**19.00**	3028.2	**19.00**	211.72	20.1
400_05	**0.00**	64.6	**0.00**	5.26	0.0
400_06	**0.00**	899.1	**0.00**	7.57	0.0
400_07	**4.00**	2051.1	4.09	163.79	4.6
400_08	4.30	2185.5	**4.00**	165.66	4.1
400_09*	**5.80**	1598.8	**6.69**	185.74	8.3
400_10	**0.00**	2487.9	**0.00**	33.15	0.0

Tableau 5.5: Résultats de l'ILPGA$^{NCPX/IBX}$, de l'AG$^{NCPX/IBX}$ + LS et du VLNS sur les instances de l'ET3

Le Tableau 5.5 permet également de comparer la performance de l'*ILPGA$^{NCPX/IBX}$* aux résultats de l'*AG$^{NCPX/IBX}$* + LS. Dans le chapitre précédent, il a été montré que l'approche

mixte $AG^{NCPX/IBX}$ + LS s'est avérée l'algorithme le plus performant pour résoudre l'ensemble des problèmes de la librairie CSPLib. Au Tableau 5.5, on note que les performances des deux algorithmes sont très proches l'une de l'autre. Toutefois, l'$ILPGA^{NCPX/IBX}$ 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 5.5.

En plus des résultats présentés au Tableau 5.5, nous avons aussi effectué un test statistique pour déterminer si l'$ILPGA^{NCPX/IBX}$ est réellement plus performant que l'$AG^{NCPX/IBX}$ +LS. Comme les deux algorithmes obtiennent souvent des résultats identiques et à cause de la faible variabilité des résultats, il est difficile d'utiliser un test statistique pour différencier les deux approches de résolution. Cependant, les travaux de Conover et Iman [1981] expliquent comment effectuer un test paramétrique *t* (*t*-test) sur la moyenne des rangs dans les cas où l'on a plus de 80% de résultats identiques. Ainsi, uniquement les 9 instances identifiées par un astérisque (*) au Tableau 5.5 ont été considérées dans le test paramétrique présenté au Tableau 5.6. Ces résultats indiquent, avec un niveau de confiance de 99%, que l'$ILPGA^{NCPX/IBX}$ obtient de meilleurs résultats que l'$AG^{NCPX/IBX}$ + LS.

	$ILPGA^{NCPX/IBX}$	$AG^{NCPX/IBX}$ + LS
Moyenne des rangs	56.94	41.14
Variance	698.64	834.05
Nombres de cas	900	90
Valeur *t*	5.357	
Probabilité 2-tail	0,000	

Tableau 5.6: t-test pour la comparaison de la moyenne des rangs entre $ILPGA^{NCPX/IBX}$ et l'$AG^{NCPX/IBX}$ +LS pour les 9 instances considérées (deux échantillons indépendants) avec un niveau de signification de 1%.

Il existe toutefois un écart important entre les temps moyens d'exécution des deux algorithmes. En effet, l'$AG^{NCPX/IBX}$ utilisant une recherche locale prend en moyenne 149 secondes par problème comparativement à 1874 secondes pour l'ILPGA$^{NCPX/IBX}$. Toutefois, un temps d'exécution d'environ 30 minutes ne constitue pas un réel handicap dans l'industrie automobile car les opérations liées à l'ordonnancement de voitures sont effectuées une journée avant le début de la production. Ainsi, si l'utilisation d'une méthode exacte dans un algorithme hybride permet d'améliorer la qualité des solutions obtenues, ce type d'approche sera probablement implémenté en contexte industriel. Il serait donc intéressant de tester les approches proposées dans ce chapitre sur des problèmes de taille industrielle afin de voir si la différence entre les approches hybrides et heuristiques augmente ou non.

Lorsqu'on compare maintenant l'*ILPGA*$^{NCPX/IBX}$ au VLNS, on note que l'*ILPGA*$^{NCPX/IBX}$ obtient généralement de meilleurs résultats que le *VLNS*. En effet, l'*ILPGA*$^{NCPX/IBX}$ obtient des résultats identiques ou meilleurs à ceux du VLNS pour 28 des 30 instances de l'ET3.

5.5 Conclusion

Dans ce chapitre, nous avons présenté deux techniques hybrides permettant de résoudre efficacement le problème théorique d'ordonnancement de voitures. Ces différentes méthodes sont basées sur des métaheuristiques à base de populations, les algorithmes génétiques, et sur un programme linéaire en nombres entiers. En particulier, les approches proposées reposent sur deux nouveaux croisements hybrides. La nature différente des

opérateurs de croisement proposés nous a permis de les combiner et de montrer que l'algorithme résultant permettait d'obtenir des résultats compétitifs, en termes de qualité de solutions, par rapport aux résultats obtenus aux résultats de l'AG-NCPX avec recherche locale du chapitre précédent. Les travaux réalisés dans ce chapitre montrent donc l'intérêt de ce type de méthode lorsque les connaissances du domaine sont bien prises en compte afin de développer des méthodes adaptées qui permettent d'exploiter les forces de chacune des approches hybridées.

Toutefois, les expérimentations numériques effectuées ont mis en évidence l'écart considérable entre le temps d'exécution des approches hybrides et ceux des AG proposés au Chapitre 4. 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 mimétiques. De plus, il serait intéressant d'évaluer les coopérations de type collaboratives ainsi qu'une éventuelle combinaison avec les approches proposées dans cette partie. Finalement, afin de valider les différents mécanismes présentés dans cette partie, il serait intéressant de les tester sur d'autres types de problèmes.

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.

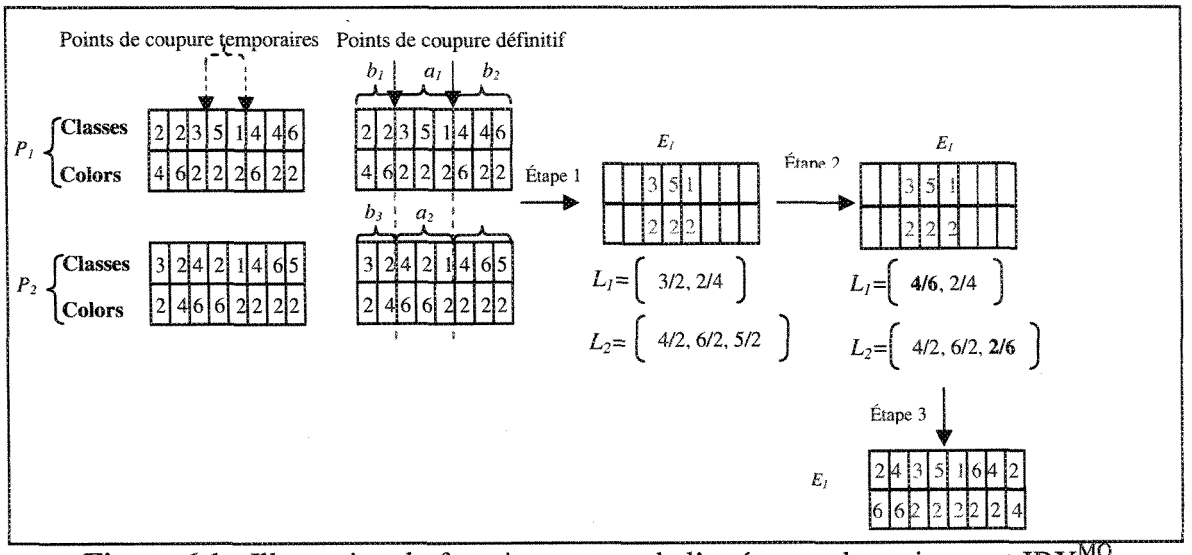


Figure 6.1 : Illustration du fonctionnement de l'opérateur de croisement IBX[MO]

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.

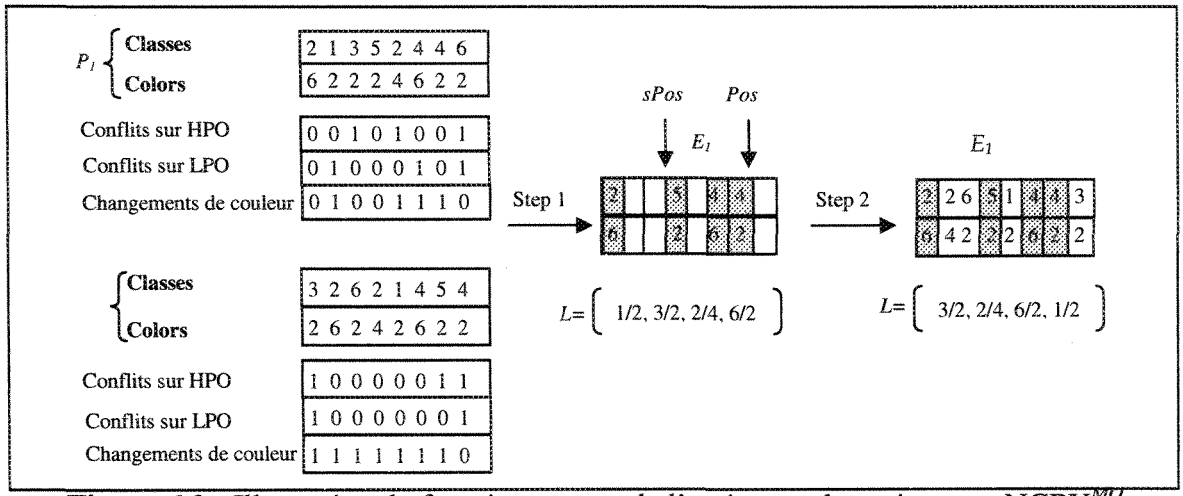


Figure 6.2 : Illustration du fonctionnement de l'opérateur de croisement NCPX^{MO}

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.

CHAPITRE 7

UNE APPROCHE HYBRIDE POUR L'OPTIMISATION MULTI-OBJECTIFS : GENETIC IMMUNE STRATEGY FOR MULTIPLE OBJECTIVE OPTIMIZATION (GISMOO)

7.1 Introduction

Dans la pratique, les situations industrielles rencontrées correspondent rarement à des problèmes nécessitant l'optimisation d'un seul objectif. En effet, la plupart des problèmes d'optimisation combinatoire rencontrés dans l'industrie nécessitent l'optimisation simultanée de plusieurs objectifs souvent conflictuels. Le problème industriel d'ordonnancement de voitures traité dans cette thèse illustre bien cet état de fait. Toutefois, malgré l'intérêt indéniable d'aborder les problèmes industriels d'un point de vue multi-objectifs, plusieurs auteurs ont noté [McKay et Wiers 1999; Yang et Liao 1999], en recensant la littérature, que les chercheurs s'attardaient principalement à des contextes théoriques de base.

Dans ce chapitre ainsi que dans le suivant, nous allons nous intéresser à la résolution de problème d'optimisation combinatoire multi-objectifs au sens Pareto sans aggrégation des objectifs contrairement à l'approche lexicographique présentée au chapitre précédent. En particulier, nous proposons GISMOO, un nouvel algorithme Pareto générique pour traiter les problèmes d'optimisation multi-objectifs. Cet algorithme combine des concepts issus des algorithmes génétiques avec des paradigmes issus des systèmes immunitaires artificiels. L'originalité de cette approche réside, d'une part, dans la manière dont les concepts issus des deux domaines sont combinés et utilisés dans un même algorithme. D'autre part, Coello Coello et Cortés [2005] ont noté que jusqu'à récemment les efforts pour étendre les approches basées sur des systèmes immunitaires à l'optimisation combinatoire multi-objectifs étaient pratiquement inexistants. Dans cette optique, GISMOO représente donc une contribution visant à combler en partie cette lacune. GISMOO est donc un nouvel

algorithme multi-objectifs au sens Pareto. Afin d'évaluer ses performances, nous devons le comparer avec d'autres algorithmes multi-objectifs de la littérature. Le problème industriel d'ordonnancement de voitures présenté au Chapitre 2 est un problème où trois objectifs conflictuels sont à minimiser. Toutefois, à notre connaissance, ce problème a toujours été traité dans la littérature en considérant les trois objectifs de manière lexicographique. Les principaux algorithmes multi-objectifs n'ont donc, dans leur version intégrale, jamais été appliqués à la résolution du POVI. C'est pourquoi nous avons choisi, dans un premier temps, de valider notre algorithme en analysant ses performances à l'aide d'un problème classique largement étudié dans la communauté multi-objectifs, le problème de sac à dos multidimensionnel multi-objectifs [Zitzler et Thiele 1998; Jaszkiewicz 2000; Knowles et Corne 2000; Zitzler *et al.* 2001; Barichard 2003; Zinflou *et al.* 2006]. Dans le chapitre suivant, nous analysons, dans un deuxième temps, les performances de GISMOO sur le POVI. Ainsi, nous validons les performances de notre approche aussi bien sur des problèmes multi-objectifs théoriques que sur des problèmes multi-objectifs réels. De plus, il est important de préciser que les tests effectués dans ces deux chapitres sont réalisés en suivant les standards adoptés par la communauté multi-objectifs [Coello Coello *et al.* 2002].

Le reste du présent chapitre est organisé de la manière suivante. On passe brièvement en revue à la Section 7.2 les principales méthodes basées sur les SIA pour l'optimisation multi-objectifs. La Section 7.3 introduit GISMOO un nouvel algorithme Pareto qui combine plusieurs concepts issus des algorithmes génétiques et des systèmes immunitaires artificiels d'une manière unique. Par la suite, nous présentons brièvement, à la Section 7.4,

le problème d'application qui est utilisé dans ce chapitre : *le problème de sac à dos multi-objectifs*. Finalement, la Section 7.5 présente les résultats expérimentaux de l'approche proposée, dans le contexte du problème multi-objectifs de sac à dos multidimensionnel, en les comparant à ceux d'autres algorithmes de la littérature.

7.2 Résolution de problèmes multi-objectifs à l'aide de SIA

Dans le passé, les systèmes immunitaires artificiels (SIA), que l'on a présenté à la Section 3.4 du Chapitre 3, ont démontré leur potentiel pour maintenir la diversité dans la population d'un algorithme génétique pour l'optimisation multimodale [Forrest et Perelson 1991; Smith *et al.* 1992; Smith *et al.* 1993]. Smith *et al.* [1993] montrent, dans leurs travaux qu'une forme de partage de la performance émerge lorsqu'on combine un algorithme génétique avec un système immunitaire artificiel. Les SIA ont aussi été combinés à des algorithmes évolutionnaires pour résoudre des problèmes de satisfaction de contraintes [Cui *et al.* 2001].

Toutefois, on note que très peu d'auteurs ont proposé des méthodes basées sur des systèmes immunitaires artificiels pour résoudre des problèmes d'optimisation combinatoire multi-objectifs [Coello Coello et Cortés 2005]. À notre connaissance, une des premières approches basées sur les SIA pour résoudre des problèmes d'optimisation multi-objectifs a été proposée par Yoo et Hajela [1999]. Cette approche hybride un algorithme génétique avec un SIA. Dans leur algorithme, les auteurs utilisent une fonction d'agrégation linéaire pour combiner les objectifs et déterminer la performance des solutions générées par l'algorithme. Par la suite, les meilleures solutions sont désignées comme étant des antigènes

et le reste de la population constitue les anticorps du SIA. Finalement, une simulation d'une réponse immune est effectuée pour identifier les anticorps qui sont les plus proches des antigènes sélectionnés en terme de distance de Hamming. Les auteurs ont appliqué leur algorithme à la résolution de problèmes d'optimisation structurelle. Toutefois, les auteurs n'ont comparé leur algorithme à aucune autre technique de résolution dans leur étude.

Il est aussi important de mentionner l'algorithme CLONALG introduit par De Castro et Von Zuben [2002] qui est un SIA basé sur le principe de sélection par clonage [De Castro et Timmis 2002]. L'algorithme CLONALG a été développé pour résoudre des problèmes d'optimisation multimodale ou uni-objectif. De leur côté, Coello Coello et Cortés [2005] ont proposé le Multi-objective Immune System Algorithm (MISA) qui est considéré comme la première tentative réelle de résolution de problèmes d'optimisation combinatoire multi-objectifs générique à l'aide d'un SIA. Contrairement à l'approche de Yoo et Hajela [1999], MISA n'est pas un algorithme hybride et, comme CLONALG, il se base sur le principe de sélection par clonage. Plus récemment, Tavakkoli-Moghaddam *et al.* [2007] proposèrent une autre approche immune pour résoudre un problème de flow-shop bi-objectifs. Comme CLONALG et MISA, l'approche de Tavakkoli-Moghaddam et al. se base aussi sur le principe de sélection par clonage. Les résultats obtenus par ces trois algorithmes suggèrent que le principe de sélection par clonage est une voie prometteuse en optimisation multi-objectifs où beaucoup de travail reste à faire.

7.3 Genetic Immune Strategy for Multiple Objectives Optimization (GISMOO)

Dans cette section, nous présentons GISMOO, un nouvel algorithme combinant un algorithme génétique élitiste utilisant des concepts de dominance Pareto avec les concepts de sélection par clonage et d'hyper mutation issus des systèmes immunitaires artificiels. L'originalité de cette approche réside principalement dans la manière avec laquelle la métaphore immune est utilisée, dans un algorithme génétique Pareto, pour identifier et mettre l'emphase sur les régions de l'espace de solution peu exploitées lors des itérations de l'algorithme. L'ajout d'une phase immune favorise ainsi une plus grande diversité au sein de la population de notre algorithme multi-objectifs. De plus, même si l'hybridation entre AG et SIA n'est pas en soit une nouveauté, GISMOO représente cependant, à notre connaissance, un des premiers algorithmes Pareto générique hybridant algorithmes génétiques et systèmes immunitaires artificiels dans un même algorithme pour résoudre des problèmes d'optimisation combinatoire multi-objectifs.

7.3.1 Création de la population initiale

Dans l'approche 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 liée au problème à résoudre.

7.3.2 Gestion de l'élitisme

GISMOO est un algorithme élitiste c'est-à-dire qu'il possède des mécanismes assurant la conservation des individus non dominés au cours de la recherche. Pour cela, GISSMO utilise le principe d'archive pour conserver les solutions non dominées rencontrées durant

la recherche. Toutefois, dans notre approche, tous les individus de l'archive ne participent pas au processus de sélection. En effet, dans l'algorithme GISSMO, seuls les individus non dominés encore présents dans la population courante participent au processus de sélection. Il est important de noter ici que la taille de l'archive de notre algorithme n'est pas fixe et n'est pas limitée à un nombre maximum d'individus. Finalement, pour s'assurer que les meilleurs individus de la population sont conservés dans la population courante, GISSMO utilise une procédure de remplacement déterministe décrite à la Section 7.3.6.

7.3.3 Assignation de la performance

Un des principes élémentaires de tous algorithmes multi-objectifs est de distribuer la population de manière homogène le long de la frontière Pareto. Dans ce but, l'assignation de la performance d'un individu est un élément fondamental pour le succès d'une approche. GISMOO est un algorithme qui exploite la notion de dominance au sens Pareto proposée par Goldberg [1989]. L'évaluation de la performance des individus se fait donc selon les concepts de *dominance* et d'*isolement*.

7.3.3.1 Le facteur de dominance

À chaque itération t, la première étape du calcul de la performance consiste à assigner à chaque individu x, de la population Parent (POP_t) combinée à la population d'enfants et de clones (Q_t), une force S(x) correspondant au nombre de solutions y dominées par x, tel que :

$$S(x) = \left| \{ y \mid y \in POP_t \cup Q_t, x \succ y \} \right| \tag{7.1}$$

où | | représente la cardinalité de l'ensemble et le symbole $\succ$ indique une relation de dominance de x par rapport à y. À partir de $S(x)$, le *facteur de dominance* d'un individu x, noté $R^+(x)$, est déterminé dans GISMOO à l'aide de l'équation suivante:

$$R^+(x) = \begin{cases} \dfrac{S(x)}{1+2*S(x)} & si \sum_{y \in POP_t \cup Q_t, y \succ x} S(y) = 0 \\ \sum_{y \in POP_t \cup Q_t, y \succ x} S(y) & sinon \end{cases} \tag{7.2}$$

Ainsi, les individus non dominés y pour lesquels $\sum_{y \in POP_t \cup Q_t, y \succ x} S(y) = 0$ n'ont pas tous un facteur de dominance identique et égal à 0, mais un facteur de dominance compris entre 0 et 0.5 en fonction du nombre de solutions qu'elles dominent. Pour les solutions non dominées, cette méthode d'assignation du facteur de dominance permet de mieux tenir compte de la distribution des solutions dominées des populations *POP* et *Q* dans l'espace de recherche. De cette manière, le calcul du facteur de dominance favorise les solutions non dominées situées dans des régions sous-exploitées.

On note des similitudes entre les mécanismes d'assignation de la performance du PMS^{MO} [Zinflou *et al.* 2006] et ceux utilisés par GISMOO. Toutefois, les populations considérées dans les deux algorithmes pour l'assignation de la performance ne sont pas les mêmes. En effet, dans GISMOO le calcul de la performance tient compte des populations Parent et Enfant courantes, alors qu'au niveau du PMS^{MO} le calcul se fait en considérant la population Parent courante et l'archive.

7.3.3.2 Le facteur d'isolement

Le calcul du facteur de dominance tel que présenté à la sous-section précédente est une technique de nichage basée sur la notion de dominance au sens Pareto. Toutefois, lorsque la plupart des individus évalués sont des solutions non dominées, ou ont les mêmes performances brutes, cette technique peut s'avérer inefficace. Afin d'éviter ce genre de situation et d'introduire un peu plus de diversité dans la population, une information additionnelle sur la densité de solutions entourant un individu x est calculée dans notre approche. Cette information supplémentaire correspond au *facteur d'isolement* (*Dist*(x)) de l'algorithme. Ce calcul se fait selon l'équation suivante :

$$Dist(x) = \min_y \left[\sqrt{(f_1(x) - f_1(y))^2 + \ldots + (f_{noj}(x) - f_{nobj}(y))^2} \right] \text{ avec } x \neq y \qquad (7.3)$$

où *nobj* indique le nombre d'objectifs du problème. On remarque que le calcul de la densité (*Dist*) de GISMOO s'inspire du calcul de la métrique d'espacement ***Sp*** [Schott 1995] et a pour objectif d'évaluer la distance séparant un individu x de son plus proche voisin. Il est important de mentionner que le calcul du facteur d'isolement se fait en considérant les individus des populations Parent et Enfant uniquement. Notons aussi que cette information additionnelle n'est pas directement incorporée au calcul de la performance d'un individu. En effet, le facteur d'isolement est utilisé, dans la phase génétique, pour différencier deux individus lors du processus de sélection en cas d'égalité sur le facteur de dominance et lors des procédures de tris des populations. Dans la phase immune, cette information additionnelle est aussi utilisée pour déterminer le nombre de clones à produire pour un individu x donné.

7.3.4 Sélection

Plusieurs schémas de sélection auraient pu être envisagés dans l'approche proposée dans ce chapitre. Toutefois, comme elle est peu coûteuse à mettre en œuvre et à exécuter et qu'elle a fait ses preuves avec les autres algorithmes génétiques proposés dans cette thèse, la procédure de sélection retenue dans la phase génétique de GISMOO est une sélection par tournoi binaire. Précisons toutefois que le tournoi est effectué en considérant le facteur de dominance des deux individus participants au tournoi et en cas d'égalité, l'égalité est brisée en utilisant le facteur d'isolement.

7.3.5 Phase immune

Dans l'approche proposée, la population Enfant (Q) est subdivisée en deux portions de même taille. Une sous-population d'enfants de taille $N/2$ générée en utilisant les opérateurs de sélection, croisement et mutation décrits précédemment et une sous-population de clones, elle aussi de taille $N/2$, générée selon le principe de sélection par clonage introduit par De Castro et Timmis [2002]. Le principe de sélection par clonage permet de modéliser le fait que seuls les meilleurs « *anticorps* » vont proliférer dans la population. Dans notre algorithme, les anticorps correspondent aux individus non dominés de la population courante. Ainsi, pour créer la sous-population de clones, la première étape consiste à trier la population Parent courante (POP_{t+1}) en front selon le même principe que l'algorithme du NSGAII [Deb 2000]. Les individus non dominés de POP_{t+1}, ceux situés dans le premier front, sont alors sélectionnés comme population d'anticorps à cloner. Dans GISMOO, le nombre de clones produit pour chaque anticorps n'est pas constant, mais est proportionnel à son degré d'isolement. Ainsi, plus un individu est éloigné des autres en terme de distance,

plus le nombre de clones qu'il génère est important. Par cette technique, on cherche à identifier et à mettre l'emphase sur les solutions non dominées situées dans des régions isolées. Le calcul du nombre de clones pour chaque anticorps x se fait comme suit :

$$nb\_clones(x) = round\left[\left(Dist(x)*\frac{N}{2}\right)\Big/\sum_{x=1}^{|Front_1|}Dist(x)\right] \tag{7.4}$$

Dans cette équation, *nb_clones(x)* indique le nombre de clones pour l'individu non dominé x, $|Front_1|$ donne le nombre d'individus non dominés de la population et $Dist(x)$ correspond au facteur d'isolement de l'individu x tel que décrit à la Section 7.3.3.2. La fonction *round* arrondit le résultat obtenu à l'entier le plus proche.

Une fois le nombre de clones à retenir pour un individu x déterminé, on produit à chaque itération de la phase d'expansion par clonage deux clones (c_1^{clo}, c_2^{clo}) qui sont des copies de l'individu x. Ces clones sont, par la suite, mutés en utilisant deux opérateurs de mutation différents adaptés au problème à résoudre afin d'obtenir (c_1^{mut}, c_2^{mut}). Après évaluation, les deux clones mutés sont comparés et on conserve le meilleur des deux (celui qui domine l'autre au sens Pareto). En cas d'égalité, on sélectionne un des deux de manière aléatoire. Le clone muté sélectionné est par la suite ajouté à la population Q courante. Ce processus est ainsi répété pour chaque individu x du 1er front jusqu'à ce que le nombre de clones sélectionnés pour x soit égal à nb_clones(x).

Comme dans l'algorithme MISA [Coello Coello et Cortés 2005], GISMOO utilise le principe de sélection par clonage dans sa phase immune. Toutefois, il existe des différences majeures entre les deux algorithmes notamment en ce qui concerne le nombre de gènes à muter, les taux de mutation, le nombre de clones à générer pour chaque individu ainsi que

les stratégies de sélection des individus à cloner. Par exemple, dans MISA le nombre de gènes à muter pour chaque clone varie en fonction du rang du clone sélectionné alors que, dans GISMOO, ce nombre est constant et aucune procédure de classement supplémentaire n'est nécessaire. L'algorithme MISA utilise un taux de mutation variable pour les « moins bons » anticorps ce qui n'est pas le cas de GISMOO. Dans notre approche, le nombre total de clones à produire est fixé à 50% de la taille de la population alors que, dans MISA, ce nombre est fixé à 600% de la taille de la population. Une autre différence majeure entre les deux algorithmes réside dans le fait que seuls les individus non dominés de la population courante sont sélectionnés pour être clonés dans GISMOO alors que, dans MISA, si le nombre de solutions non dominées est inférieur à 5% de la taille de la population, des individus dominés sont sélectionnés pour être clonés. On note aussi que, dans l'approche proposée dans ce chapitre, le nombre de clones pour chaque anticorps est proportionnel à son facteur d'isolement et est calculé à chaque itération selon l'Équation 7.4. À l'opposé, dans MISA, chaque anticorps produit le même nombre de clones au début de l'algorithme. Par la suite, le nombre de clones produits par chaque anticorps augmente ou diminue selon certaines règles en fonction de la taille de la population secondaire de l'algorithme [Coello Coello et Cortés 2005]. On constate aussi que les deux algorithmes utilisent des stratégies différentes de sélection des anticorps et d'ajout des clones à la population. Finalement, une autre différence de taille entre les deux approches est que GISMOO est une approche hybride alors que MISA est un SIA simple.

7.3.6 Remplacement

GISMOO est un algorithme élitiste. Dans cette optique et afin de conserver les meilleurs individus dans la population courante, la stratégie de remplacement utilisée est un remplacement déterministe de type $(\lambda+\mu)$ où λ indique la taille de la population Parent et μ la taille de la population Enfant. Dans ce schéma de remplacement, les populations Parent et Enfant sont jointes et triées et on ne conserve que les λ meilleurs individus afin de former la prochaine génération. Dans l'approche proposée, on a $\lambda=\mu=N$.

7.3.7 Algorithme général

Le pseudo-code présenté à l'Algorithme 7.1 décrit le fonctionnement général de l'approche GISMOO. Après l'initialisation des différentes variables, l'algorithme débute par la génération de la population initiale POP_0 (ligne 2). Chaque individu de cette population est évalué et une performance lui est assignée conformément aux mécanismes d'assignation de la performance décrits à la Section 7.3.3. Une fois la performance assignée, on génère une population initiale d'enfants Q_0 que l'on évalue. Le processus itératif de l'algorithme (lignes 7-41) commence par la phase génétique (lignes 8-24) en combinant les populations Parent et Enfant courantes. Les individus de la nouvelle population combinée sont évalués et leurs facteurs d'isolement calculé (lignes 9-11). À cette étape, on note que, pour des fins de normalisation, on calcule une approximation du *point idéal* et du *point Nadir* à partir des individus de l'archive (ligne 10). Les coordonnées du point idéal correspondent aux meilleures valeurs de chaque objectif des solutions de la frontière Pareto. À l'opposé, les coordonnées du point nadir correspondent aux pires valeurs de chaque objectif des solutions de la frontière Pareto. La population combinée est

ensuite triée et seuls les *N* premiers individus sont conservés pour former la prochaine génération courante. On retrouve, par la suite, les étapes classiques de sélection par tournoi, de croisement et de mutation d'un algorithme génétique classique. Notons ici que lors d'un croisement deux enfants peuvent être générés; dans ce cas, on compare les deux enfants l'un par rapport à l'autre à l'aide de la procédure *Compet* (ligne 22). L'algorithme 7.2 décrit la procédure *Compet* utilisée à la ligne 22 pour comparer deux solutions. Cette procédure reçoit en paramètres deux solutions *x* et *y* et retourne la meilleure des deux. Le meilleur des deux enfants est alors ajouté à la population Enfant (ligne 23). Ce processus est ainsi répété jusqu'à ce que *N/2* enfants soit ajoutés à la population en cours de création *Q*. Une fois la phase génétique terminée, la phase immune de GISMOO (lignes 25-39) débute par le tri de la population *POP* courante en fronts selon le même principe que le NSGAII. On calcule, par la suite, la somme des facteurs d'isolement des individus situés dans le premier front et on commence la phase d'expansion par clonage tel qu'indiquée à la Section 7.3.5 de manière à ajouter *N/2* clones à la population d'enfants. Il est important de mentionner que, à chaque itération de l'algorithme, chaque nouvel individu non dominé trouvé vient s'ajouter à l'archive et les individus de l'archive dominés par le nouvel arrivant sont supprimés. Ainsi, à la fin de l'algorithme, les individus contenus dans l'archive sont retournés au décideur (ligne 42).

Algorithme 7.1 : pseudo-code détaillé de GISMOO

1: Initialiser l'archive A à ∅ et t à 0
2: Générer la population initiale POP_t aléatoirement ou avec des heuristiques gourmandes
3: Évaluer chaque individu $x \in POP_t$et mettre à jour A
4: Trier POP_t
5: Générer une population initiale Q_t d'enfants de même taille que la population Parent par croisement
6 : Évaluer chaque nouvel enfant créé et mettre à jour A
7: **Tant qu'**aucun critère d'arrêt n'est atteint
8 : Joindre Q_t *et* POP_t
9 : Calculer de la performance de chaque individu de la population combinée
10 : Évaluer le point idéal courant et le point Nadir courant à partir de A
11 : Calculer les distances normalisées entre chaque solution
12 : Trier $Q_t \cup POP_t$
13 : Conserver les N meilleurs individus pour former la population Parent POP_{t+1}
14: **Tant que**$| Q_t | < N/2$
15: Sélectionner deux parents P_1 et $P_2 \in POP_{t+1}$ par tournoi binaire
16: Créer 2 enfants Enf_1 et Enf_2 par croisement
17: Évaluer les enfants et mettre à jour A
18: Tirer aléatoirement un nombre *rnd* entre *0* et *1*
19: **Si** $rnd < p_m$ **alors**
20: Mute les enfants
21: **Fin si**
22: *Winner*= Compet (Enf_1 , Enf_2)
23: Ajouter *Winner* à Q_t
24: **Fin tant que**
25: Classer POP_{t+1} en front
26: Total = somme des facteurs d'isolement des individus non dominés
27 : **Pour** chaque individu x du 1er front **faire**
28 : nb_clones = $\lceil$ (Dist(x)* *N/2*)/Total $\rceil$
29 : cpt =0
30 : **Tant que** cpt< nb_clones
31 : (c_1, c_2)= Clone (x)
32 : c_1^{mut}= mutation_1(c_1)
33 : c_2^{mut}= mutation_2(c_2)
34 : Évaluer les clones mutés et mettre à jour A
35 : *Winner*= Compet (c_1^{hyp1}, c_2^{hyp2})
36: Ajouter *Winner* à Q_t
37: cpt = cpt +1
38: **Fin tant que**
39: **Fin pour**
40: $t = t + 1$
41: **Fin tant que**
42: **Retourner A**

Algorithme 7.2 : Procédure *Compet* (Individu x, Individu y)

2 : **Si** x domine y **alors**
3 : retourner x
4 : **Sinon**
5 : **Si** x est dominé par y **alors**
6 : retourner y
7 : **Sinon**
8 : **Si** Dist(x) > Dist(y) **alors**
9 : retourner x
10 : **Sinon**
11 : **Si** Dist(x) < Dist(y) **alors**
12 : retourner y
13 : **Sinon**
14 : choisir aléatoirement entre x et y et retourner l'individu choisi
15 : **Fin si**
16 **Fin si**
17 : **Fin si**
18 : **Fin si**

7.4 Le problème de sac à dos multi-objectifs en 0/1 (MOKP)

Le MOKP utilisé dans ce chapitre est celui proposé par Zitzler et Thiele [1999] et Zitzler *et al.* [2001] dans leurs expérimentations. Nous avons choisi ce problème car c'est l'un des problèmes les plus étudiés dans la communauté multi-objectifs. Ainsi, nous pourrons comparer les performances de notre approche avec celles des meilleurs algorithmes de la littérature.

Le *nobj*-objectifs problème de sac à dos en 0/1 est défini par un ensemble de J items (ou objets). À chacun des items sont associés Z valeurs de profit ainsi que Z poids. Formellement, le problème peut être exprimé de la manière suivante. Soit un ensemble de J objets et Z sacs à dos avec :

- $p_{z,j}$ - profit correspondant à l'objet j en fonction du sac à dos z,
- $w_{z,j}$ – poids correspondent à l'objet j en fonction du sac à dos z,
- cap_z - capacité totale du sac à dos z,

Il faut donc trouver un vecteur $x = \{x_1, x_2, \ldots, x_J\} \in \{0,1\}^J$ qui maximise

$$f_z(x) = \sum_{j=1}^{J} p_{z,j} x_j \qquad z = 1,\ldots, Z$$

sujet à :

$$\sum_{j=1}^{J} w_{z,j} x_j \leq cap_z \qquad z = 1,\ldots, Z$$

Le problème consiste donc à sélectionner un sous-ensemble d'objets x_j maximisant $f(x)=(f_1(x), f_2(x), \ldots, f_z(x))$ et pour lequel les contraintes de capacité sont respectées.

Comme pour sa version uni-objectif, le problème de sac à dos multi-objectifs peut être utilisé pour modéliser de nombreux problèmes réels, comme la répartition de budgets ou encore l'allocation de ressources. Plusieurs heuristiques ont été développées pour résoudre ce problème. En recensant la littérature, on retrouve, sans être exhaustif, des méthodes de recherche dans le voisinage comme le recuit simulé [Serafini 1994; Ulungu *et al.* 1999], la recherche avec Tabou [Gandibleux *et al.* 1996], des méthodes évolutionnaires comme le VEGA [Schaffer 1985], le NSGA [Srivinas et Deb 1994] et le NSGAII [Deb 2000], le SPEA [Zitzler et Thiele 1998] et le SPEA2 [Zitzler *et al.* 2001], le PMS^{MO} [Zinflou *et al.* 2006], ou encore le M-PAES [Knowles et Corne 2000]. Plus récemment, des approches hybrides combinant algorithme génétique et méthode de recherche locale ont été proposées pour résoudre le problème comme le *Multi-Objective Genetic Local Search* (MOGLS) [Jaszkiewicz 2000] ou le *Genetic Tabu Search for the Multi-Objective Knapsack Problem* (GTS^{MOKP}) [Barichard 2003]. Le MOGLS utilise une méthode de descente pure comme opérateur de recherche locale alors que le GTS^{MOKP}, de son côté, utilise un algorithme de recherche avec tabous comme opérateur de recherche locale. Ces deux algorithmes, contrairement à GISMOO, ne sont pas des approches Pareto et ils utilisent une fonction d'agrégation linéaire pondérée pour évaluer la performance des individus trouvés au cours des itérations. Mentionnons que parmi les approches hybrides pour le MOKP, le MOGLS est un des algorithmes les plus réputés et est souvent utilisé dans la littérature pour des fins de comparaison. Toutefois, à notre connaissance, les méthodes hybrides utilisées pour résoudre ce problème se limitent principalement à des approches hybridant algorithmes génétiques et méthodes de recherche locale.

7.4.1 Espace de recherche et représentation

Le MOKP utilisé dans cette partie de la thèse est un problème multi-objectifs en *0/1*. Une solution réalisable à ce problème est donc un vecteur binaire de n composantes satisfaisant les différentes contraintes du problème. Par conséquent, nous avons opté pour une représentation classique sous forme de chaîne de bits pour représenter chaque individu. L'espace de recherche S est constitué par l'ensemble de tous les vecteurs binaires, qui est clairement un sous-ensemble de $\{0, 1\}^n$.

7.4.2 Génération de la population initiale pour le MOKP

Comme mentionné à la Section 7.3, l'algorithme GISMOO commence en générant une population de départ constituée à 70% d'individus générés aléatoirement et à 30% d'individus générés en utilisant une heuristique gourmande. Pour construire chaque individu aléatoirement dans le cas du MOKP, on ajoute simplement à l'individu en cours de création un sous-ensemble d'objets pris au hasard. Si la somme des poids de tous les objets choisis aléatoirement excède la capacité d'un des sacs, on retire progressivement les objets influant le moins sur le profit de manière à respecter les contraintes de capacité. L'heuristique gourmande utilisée pour générer des individus est la même que celle proposée par Jaszkiewicz [2000] et consiste simplement à construire un individu x en insérant progressivement les objets influant le plus sur le profit tout en respectant les contraintes de capacité.

7.4.3 Opérateur de croisement pour le MOKP

Dans la version de GISMOO pour le MOKP, nous utilisons l'opérateur de croisement classique à un point de coupure [Holland 1962]. Notons que l'individu résultant d'un

croisement peut comporter beaucoup plus d'objets que chaque parent pris séparément. Par conséquent, il peut violer les contraintes de capacité du MOKP. Pour restaurer la faisabilité des individus créés par croisement, nous procédons de la même manière que lors de la création de la population initiale en retirant progressivement les objets influant le moins sur le profit.

7.4.4 Opérateur de mutation pour le MOKP

Pour résoudre le MOKP, nous utilisons deux types de mutation dans notre algorithme. Le premier opérateur de mutation est une inversion de bit simple. Cet opérateur de mutation consiste à inverser de manière aléatoire un gène de l'individu à muter (si le gène sélectionné est à 0, on le met à 1 et inversement). Le second opérateur de mutation est celui décrit par Knowles et Corne [2000]. Cet opérateur de mutation consiste à tirer un nombre aléatoire *nb* pour chaque gène du chromosome de l'individu à muter. Si le reste de la division entière de *nb* par le quart de la taille du chromosome est nul, on inverse la valeur du gène correspondant. Notons que seul le deuxième opérateur de mutation est utilisé dans la phase génétique de GISMOO alors que les deux sont appliqués dans la phase immune. Finalement, il est important de mentionner qu'après la mutation, la faisabilité des individus mutés est restaurée de la même manière que lors de la création de la population initiale.

7.5 Expérimentations numériques

L'algorithme GISMOO proposé 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.

7.5.1 Jeux d'essais

Les expérimentations numériques présentées dans ce chapitre sont réalisées à l'aide des neuf instances du problème de sac à dos multi-objectifs publiées par Zitzler et Thiele [1999]. Ces jeux d'essais ont 2, 3 et 4 objectifs combinés avec 250, 500 et 750 objets. De plus, pour chaque instance, le nombre de sacs est égal au nombre d'objectifs. Ces instances ont été construites aléatoirement par les auteurs. Il n'y a pas de corrélation entre les poids et les profits. De plus, les capacités totales de chaque sac sont réglées à environ la moitié de la somme des poids des objets pour le sac considéré. Il en découle que le nombre d'objets présents attendus dans les solutions optimales est d'environ la moitié du nombre total d'objets du problème [Zitzler et Thiele 1999].

7.5.2 Comparaison expérimentale

Dans cette section, nous comparons, dans un premier temps, GISMOO au PMS^{MO} et au NSGAII qui sont deux algorithmes Pareto de la littérature. Dans un deuxième temps, nous comparons les performances de notre approche à celles du MOGLS qui est un algorithme hybride réputé de la littérature comptant parmi les approches les plus performantes pour résoudre le MOKP. Ces trois algorithmes ont été choisis, d'une part, parce qu'ils illustrent bien l'état de l'art des algorithmes évolutionnaires en optimisation multi-objectifs et, d'autre part, leurs codes sources ont été rendus disponibles par leurs auteurs respectifs.
Comme pour GISMOO, les algorithmes PMS^{MO}, NSGAII et MOGLS ont été implémentés en C^{++}. Dans notre implémentation, les principales structures de données sont partagées par

tous les algorithmes. Nous obtenons ainsi une base de comparaison commune pour évaluer équitablement les performances des différents algorithmes. Dans toutes les expérimentations numériques effectuées dans cette partie, chaque instance du MOKP a été résolue à 30 reprises par chaque algorithme sur la même machine test. La taille N des populations est la même pour chaque algorithme et dépend du nombre d'objectifs et du nombre d'objets de l'instance à traiter. Le Tableau 7.1 indique la taille des populations pour tous les algorithmes en fonction du nombre d'objectifs et d'objets de l'instance.

Nombre d'objectifs	**Nombre d'objets**		
	250	**500**	**750**
2	150	200	250
3	200	250	300
4	250	300	400

Tableau 7.1 : Taille de la population selon le nombre d'objectifs et d'objets de l'instance à résoudre

7.5.2.1 Comparaison avec d'autres approches Pareto

Dans cette première série d'expérimentations numériques, nous allons comparer l'approche GISMOO avec les deux approches Pareto mentionnées à la section précédente, le NSGAII et le PMS^{MO}. Pour tous les problèmes testés dans cette partie, chaque algorithme a été exécuté pendant 500 générations. Pour les trois algorithmes, la probabilité de mutation a été fixée à 0.06. Notons ici que GISMOO ne nécessite pas de fixer une probabilité de croisement. Pour les deux autres algorithmes, la probabilité de croisement est fixée à 1.00. Finalement, la taille de l'archive locale du PMS^{MO} est fixée à 100. La valeur des différents paramètres du NSGAII et du PMS^{MO} a été fixée en accord avec les expérimentations effectuées par Zitzler *et al.*[2001] et par Zinflou *et al.*[2006].

Mentionnons aussi que, dans ce travail, nous ne comparons pas directement les performances du NSGAII par rapport à celles du PMS^{MO}. Le lecteur peut consulter [Zinflou *et al.* 2006] pour une comparaison directe entre les deux algorithmes.

Le Tableau 7.2 présente les résultats de la métrique d'hyper-volume ***H*** (présentée à la Section 3.8.3 du Chapitre 3) pour les trois algorithmes. Les deux premières colonnes du tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats moyens du NSGAII, du PMS^{MO} et de GISMOO pour la métrique ***H***. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. À partir des résultats du Tableau 7.2, nous observons que GISMOO obtient de meilleurs résultats que les deux autres algorithmes sur toutes les instances testées. En effet, la taille de l'espace dominé par GISMOO est plus importante que celle du NSGAII et du PMS^{MO}. On note toutefois que l'écart entre GISMOO et NSGAII est plus important qu'entre GISMOO et PMS^{MO}.

Nombre d'objectifs	Nombre d'objets	NSGAII	PMS^{MO}	GISMOO
2	250	9.47 E+7	9.84 E+7	9.86 E+7
	500	3.95 E+8	4.02 E+8	4.07 E+8
	750	8.57 E+8	8.66 E+8	8.93 E+8
3	250	8.06 E+11	9.03 E+11	9.28 E+11
	500	6.42 E+12	7.04 E+12	7.70 E+12
	750	2.26 E+13	2.45 E+13	2.71 E+13
4	250	6.36 E+15	7.13 E+15	7.94 E+15
	500	1.01 E+17	1.08 E+17	1.34 E+17
	750	5.19 E+17	5.47 E+17	7.15 E17

Tableau 7.2 : Moyenne de l'hyper-volume ***H*** du NSGAII, PMS^{MO} et GISMOO pour les instances du problème de sac à dos multi-objectifs

Si l'hyper-volume donne une bonne idée de la taille de l'espace dominé par un ensemble de solutions, cette métrique ne permet pas de comparer deux ensembles de solutions l'un

par rapport à l'autre. Pour y parvenir, nous utilisons la métrique ***C*** (décrite à la Section 3.8.2 du Chapitre 3). La Figure 7.1 présente les résultats des trois algorithmes selon la métrique ***C*** suivant la présentation adoptée dans [Zitzler et Thiele 1999]. Dans cette figure, chaque boîte correspond à la comparaison des solutions de l'algorithme *A* en ligne avec l'algorithme *B* en colonne. Ainsi, la valeur ***C***(A,B) indique le pourcentage d'éléments de *B* dominés par au moins un élément de *A*. Pour chaque boîte, l'échelle part de 0 en bas de la boîte pour aller à 1 en haut de celle-ci. Chaque boîte contient 9 graphiques correspondants respectivement, de gauche à droite, aux instances contenant 250, 500 et 750 objets regroupées en fonction du nombre d'objectifs (2, 3 et 4). Chaque barre horizontale noire correspond à la moyenne des différentes mesures ***C*** pour 30 exécutions et les barres verticales indiquent l'écart entre les valeurs maximales et minimales trouvées lors des différentes exécutions.

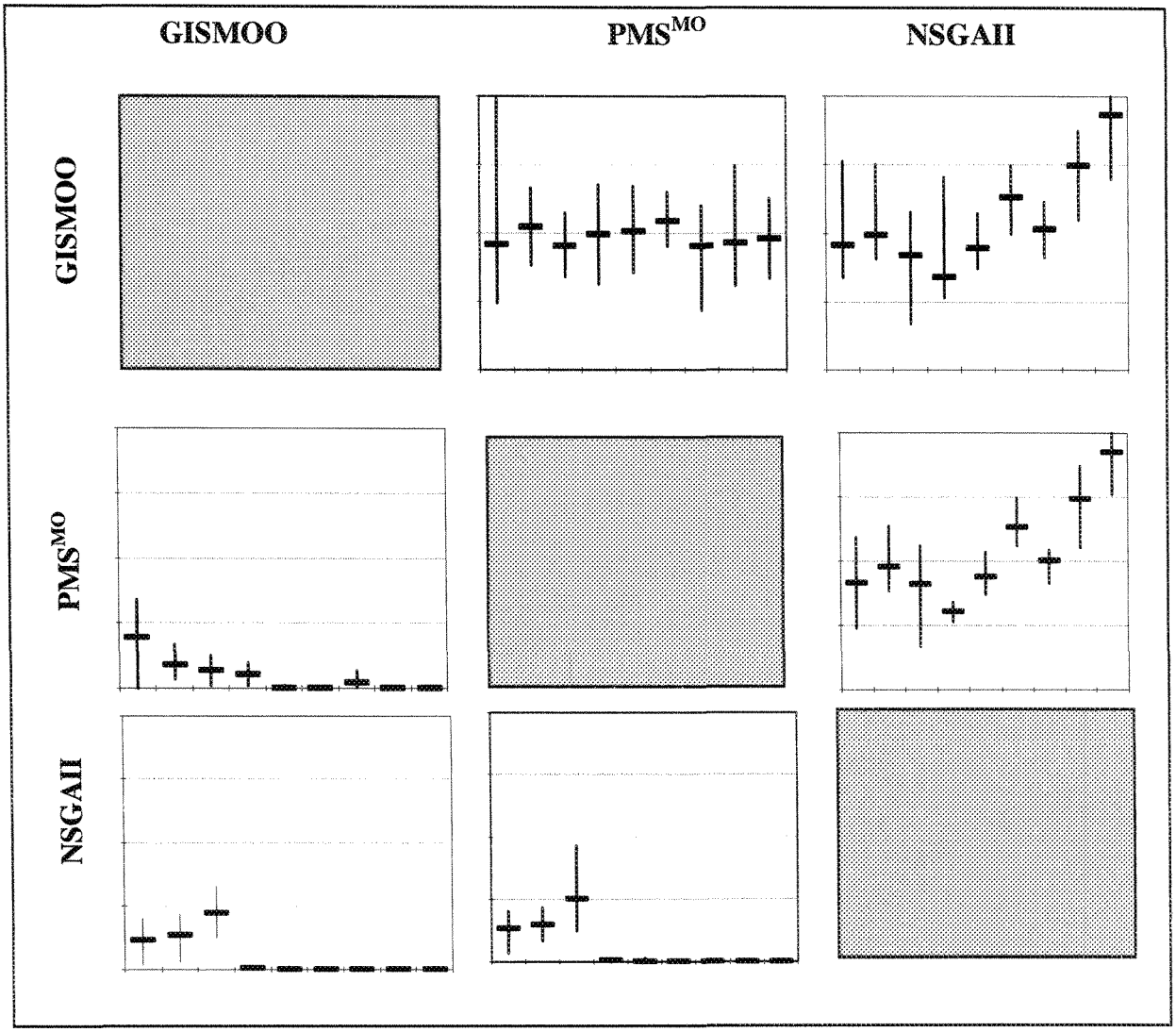


Figure 7.1 : Couverture moyenne C de GISMOO, PMS^{MO} et NSGAII pour le problème de sac à dos multi-objectifs

Les résultats obtenus avec la métrique C confirment les résultats obtenus avec l'hyper-volume, à savoir une meilleure performance de GISMOO comparativement au NSGAII et au PMS^{MO}. En effet, les valeurs de $C(GISMOO, PMS^{MO})$ et $C(GISMOO, NSGAII)$ sont respectivement supérieures à celles de $C(PMS^{MO}, GISMOO)$ et $C(NSGAII, GISMOO)$. On note aussi que la différence entre l'approche proposée dans ce chapitre et les deux

autres algorithmes Pareto semble augmentée en fonction du nombre d'objectifs. En effet, à l'exception des problèmes à deux objectifs, le rapport entre les résultats C(GISMOO, PMS^{MO}) et C(GISMOO, NSGAII) par rapport à C(PMS^{MO}, GISMOO) et C(NSGAII, GISMOO) est supérieur à 10 ce qui signifie que la grande majorité des solutions trouvées par GISMOO sont de qualité égale ou supérieure à celles du PMS^{MO} et du NSGAII. Toutefois, les résultats de la métrique ***C*** ne permettent pas de déterminer si les solutions d'un algorithme sont mieux distribuées que celle d'un autre.

Pour confirmer les résultats obtenus avec la métrique ***C*** et avoir une idée de la distribution des solutions dans l'espace de recherche, nous avons aussi comparé les trois algorithmes en utilisant la métrique ***Diff***, présentée à la Section 3.8.4 du Chapitre 3, qui permet de mesurer la différence de couverture entre les solutions de deux algorithmes. Les Tableaux 7.3 et 7.4 résument respectivement les résultats obtenus entre GISMOO et PMS^{MO} et entre GISMOO et NSGAII en utilisant cette métrique. Les deux premières colonnes de chaque tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats moyens obtenus pour la métrique ***Diff*** après 30 exécutions indépendantes de chacun des algorithmes à comparer. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris.

Nombre d'objectifs	Nombre d'objets	PMS^{MO}	GISMOO
2	250	1.96 E+4	2.07 E+5
	500	1.70 E+6	5.02 E+6
	750	3.99 E+4	2.74 E+7
3	250	4.83 E+8	2.56 E+10
	500	1.78 E+7	6.55 E+11
	750	1.23 E+7	2.63 E+12
4	250	1.67 E+12	8.13 E+14
	500	1.73 E+11	2.60 E+16
	750	3.31 E+8	1.68 E+17

Tableau 7.3 : Différence de couverture moyenne du PMS^{MO} et GISMOO pour les instances du problème de sac à dos multi-objectifs

Nombre d'objectifs	Nombre d'objets	NSGAII	GISMOO
2	250	1.85 E+4	3.94 E+6
	500	4.84 E+4	1.29 E+7
	750	5.48 E+4	3.56 E+7
3	250	1.77 E+8	1.22 E+11
	500	1 E+9	1.14 E+12
	750	1.77 E+9	4.27 E+9
4	250	2.49 E+12	1.58 E+15
	500	1.76 E+13	3.33 E+16
	750	3.14 E +13	1.96 E+17

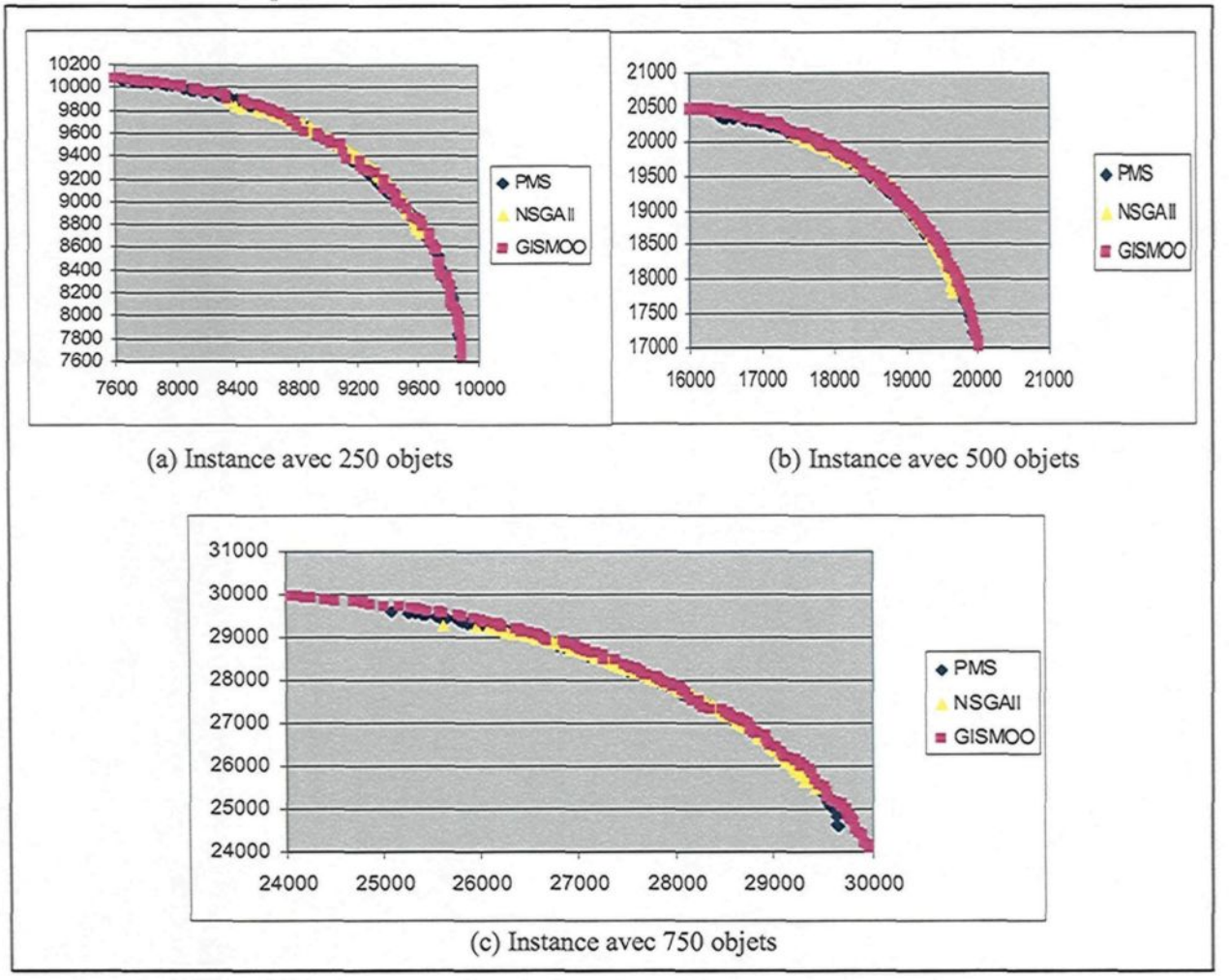
Tableau 7.4 : Différence de couverture moyenne du NSGAII et GISMOO pour les instances du problème de sac à dos multi-objectifs

En examinant les deux tableaux, on note, une fois de plus, que les résultats sont nettement en faveur de GISMOO. En effet, lorsqu'on compare, dans un premier temps, GISMOO au PMS^{MO} (Tableau 7.3), on note que GISMOO obtient une plus grande différence de couverture que le PMS^{MO} pour toutes les instances. On note aussi que l'écart entre les deux algorithmes semble être en relation avec la taille de l'instance et le nombre d'objectifs. Ainsi, l'écart de couverture entre les deux algorithmes augmente jusqu'à 10^9 pour le problème de 750 objets et 4 objectifs. On remarque aussi que malgré la différence de couverture entre les deux algorithmes et les résultats observés avec l'hyper-volume, les

deux algorithmes semblent explorer différentes régions de l'espace des objectifs. Cette situation s'explique sans doute par le fait que les deux algorithmes utilisent différents mécanismes d'exploration. En comparant, dans un deuxième temps, GISMOO et le NSGAII (Tableau 7.4), on remarque aussi un net avantage en faveur de GISMOO. En effet, GISMOO obtient une plus grande différence de couverture que le NSGAII sur toutes les instances testées. Toutefois, contrairement à ce que l'on avait observé au Tableau 7.3, l'écart entre les deux algorithmes ne semble pas être en relation avec le nombre d'objectifs, mais plutôt avec la taille des instances. Ces résultats laissent supposer que les solutions trouvées par GISMOO ont une meilleure distribution que celles trouvées par le NSGAII et le PMS^{MO}.

Pour nous en convaincre, nous avons représenté graphiquement les solutions des trois algorithmes pour les instances à deux objectifs. La Figure 7.2 (a, b, c) montre les ensembles Pareto obtenues par chacun des trois algorithmes pour les trois instances bi-objectives. Cette représentation graphique confirme les résultats des différentes mesures. Pour les trois instances, il apparaît clairement que l'ensemble Pareto proposé par le NSGAII est plus étriqué que celui proposé par GISMOO. En effet, les points des ensembles Pareto du NSGAII sont, pour la plupart, tous regroupés dans une même région de l'espace de recherche. Cette situation explique les résultats observés avec la métrique ***Diff***. En effet, les mécanismes d'exploration de GISMOO lui permettent d'obtenir une meilleure dispersion des solutions dans l'espace de recherche alors que le NSGAII semble obtenir une meilleure exploration d'une région particulière de cet espace. Pour l'instance avec 250 objets il est très difficile de différencier les ensembles proposés par GISMOO et le PMS^{MO} car les

courbes sont pratiquement confondues. Par contre, pour les deux autres instances, on remarque clairement que l'ensemble Pareto de GISMOO est plus étendu que celui calculé par le PMS^{MO}. On constate donc, une fois de plus, que GISMOO explore mieux l'espace de solutions que le PMS^{MO}.



(a) Instance avec 250 objets

(b) Instance avec 500 objets

(c) Instance avec 750 objets

Figure 7.2 : Exemple de surfaces de compromis trouvées par le PMS^{MO}, le NSGAII et GISMOO pour les trois instances à deux objectifs

Le Tableau 7.5 indique (en secondes), pour chacun des trois algorithmes et pour chaque instance du problème, les temps moyens d'exécution obtenus. On note, à l'aide de ce tableau, que GISMOO nécessite toujours moins de temps de calcul que le NSGAII et le PMS^{MO}.

Nombre d'objectifs	Nombre d'objets	NSGAII	PMS^{MO}	GISMOO
2	250	15	21	9
	500	20	41	13
	750	43	94	24
3	250	25	73	19
	500	53	89	35
	750	75	121	55
4	250	64	100	52
	500	103	150	85
	750	205	302	108

Tableau 7.5 : Moyenne des temps de calculs (secondes) du NSGAII, PMS^{MO} et GISMOO pour les instances du problème de sac à dos multi-objectifs

En complément, nous avons voulu comparer l'apport de chacune des deux phases sur la performance globale de GISSMMOO. La Figure 7.3 (a, b et c) compare graphiquement les ensembles de solutions obtenus par GISMOO, par GISMOO sans phase immune (Phase génétique) et par GISMOO sans phase génétique (Phase immune) pour les trois instances bi-objectives. À partir de cette figure, on note que les courbes proposées par chacune des deux phases de GISMOO exécutées séparément sont nettement dominées par l'ensemble Pareto trouvé par GISMOO. On note aussi, en comparant les deux phases l'une par rapport à l'autre, que la phase génétique effectue une meilleure intensification de la recherche tandis que la phase immune semble être capable d'effectuer une meilleure diversification. En effet, les courbes proposées par la phase immune sont plus étendues que celles de la phase génétique. Toutefois, ces courbes ne sont pas continues. À l'opposé, les courbes

proposées par la phase génétique sont continues, mais généralement plus étriquées. Les résultats de cette figure mettent en évidence la complémentarité des deux phases et expliquent les bons résultats de GISMOO.

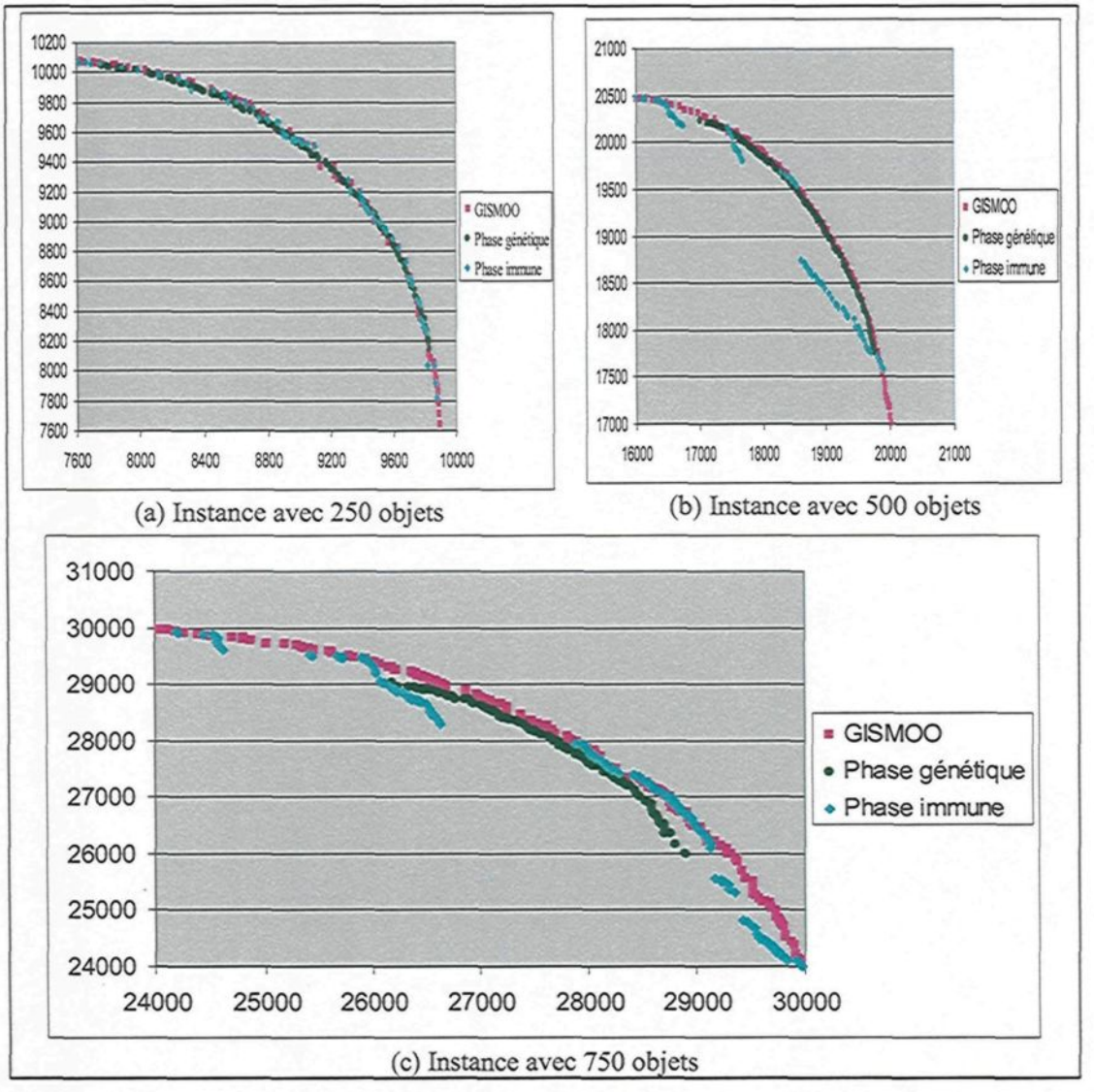


Figure 7.3 : Apport des différentes phases sur la performance globale de GISMOO

Les résultats précédents mettent en évidence l'efficacité de l'approche hybride proposée dans ce chapitre pour le problème de sac à dos multi-objectifs en 0/1. En particulier, GISMOO semble être capable d'identifier les régions moins explorées dans l'espace de solutions afin d'orienter la recherche vers ces régions. Ainsi, comparativement aux deux autres algorithmes Pareto utilisés dans cette partie, GISMOO parvient à proposer des ensembles Pareto avec des solutions mieux distribuées dans l'espace de recherche. Dans la pratique, les ensembles de solutions proposés par GISMOO sont plus intéressants pour un décideur que ceux du NSGAII ou du PMS^{MO} car ils contiennent des solutions diversifiées. Ainsi, les ensembles Pareto trouvés par GISMOO facilitent la prise de décision du gestionnaire en lui offrant une vision plus large et plus diversifiée des solutions potentielles à adopter.

Toutefois, les résultats obtenus laissent aussi entrevoir des possibilités de gains en terme d'intensification de la recherche dans certaines régions. Pour nous en convaincre, nous allons comparer les performances de GISMOO avec celle d'un algorithme non Pareto hybridant un algorithme génétique et une procédure de recherche locale : le MOGLS. Cet algorithme est réputé dans la littérature pour ses capacités d'intensification [Jaszkiewicz 2000] et compte aussi parmi les approches les plus performantes pour résoudre le MOKP [Barichard 2003].

7.5.2.2 Comparaison avec un algorithme non Pareto

Dans les expérimentations numériques suivantes, nous avons comparé GISMOO avec le MOGLS de Jaszkiewicz [2000] et une version de GISMOO avec recherche locale (GISMOO+LS). L'ajout d'une procédure de recherche locale à notre algorithme a pour but,

d'une part, de mesurer l'apport éventuel d'un mécanisme explicite d'intensification aux performances de notre approche. D'autre part, plusieurs auteurs [Knowles et Corne 2000; Berro 2001] ont relevé la difficulté à comparer des algorithmes mimétiques multi-objectifs avec des approches Pareto classiques du fait des différences de fonctionnement entre les deux type d'approches (nombre d'évaluations, nombre de générations, etc...). L'ajout de la même procédure de recherche locale utilisée par le MOGLS à GISMOO nous permettra de mieux comparer les mécanismes d'explorations des deux algorithmes. La méthode de recherche locale implémentée dans cette partie est une méthode de descente simple [Jaszkiewicz 2000]. Ainsi, à chaque itération de l'algorithme, on choisit aléatoirement un individu de la population Q et cet individu est amélioré localement dans 50% des cas. On note que, dans l'algorithme du MOGLS, la solution générée par croisement est améliorée à chaque itération. Il est important de préciser que, contrairement aux algorithmes présentés dans la section précédente, le critère d'arrêt du MOGLS ne correspond pas à un nombre maximum de générations. En effet, pour les algorithmes agrégatifs comme le MOGLS, la notion de génération n'est présente que pour garder la présentation classique des algorithmes évolutionnaires [Barichard 2003]. Dans ces expérimentations, Jaszkiewicz [2000] note que 50 générations du MOGLS représentent approximativement 500 générations d'un AG multi-objectifs standard comme le NSGA ou le SPEA. Plus récemment, Barichard [2003] utilisa le même nombre de générations pour comparer son algorithme agrégatif au NSGA. Conformément à ces deux expérimentations, nous avons fixé le nombre de générations du MOGLS à 50 de manière à être le plus équitable possible. Pour ce qui est de la taille de la population, elle est choisie en fonction du problème à

résoudre selon les valeurs du Tableau 7.1. Notons aussi que nous avons limité le nombre maximum de générations à 250 pour la version de GISMOO avec recherche locale. En diminuant ainsi le nombre de générations de GISMOO+LS, nous tenons compte des évaluations supplémentaires causées par l'addition de la procédure de recherche locale de manière à avoir approximativement le même nombre d'évaluations pour les deux versions de notre algorithme.

Le Tableau 7.6 présente les résultats de l'hyper-volume pour les trois algorithmes. Les deux premières colonnes du tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats moyens de 30 exécutions du MOGLS, de GISMOO et de GISMOO+LS pour la métrique H. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. À l'aide de ce tableau, on remarque, dans un premier temps, en comparant GISMOO et MOGLS, que les performances des deux algorithmes sont très proches l'une de l'autre sur la plupart des problèmes avec un avantage pour le MOGLS. On remarque aussi que, pour les instances à trois objectifs, l'écart entre les deux approches semble diminuer lorsque la taille de l'instance augmente. Lorsqu'on analyse maintenant l'effet de l'addition d'une procédure de recherche locale à l'algorithme GISMOO, on note que cet ajout améliore les performances de l'approche sur toutes les instances. En effet, à l'exception des instances à deux objectifs et contenant respectivement 250 et 750 objets où les trois algorithmes obtiennent les mêmes résultats moyens, GISMOO+LS obtient les meilleurs résultats sur toutes les autres instances. Ainsi, on note que l'ajout d'une procédure de recherche locale à notre approche permet d'obtenir des résultats supérieurs à ceux du MOGLS pour 5

instances tout en obtenant des résultats moyens identiques pour les quatre instances restantes.

Nombre d'objectifs	Nombre d'objets	MOGLS	GISMOO	GISMOO + LS
2	250	9.86 E+7	9.86 E+7	9.86 E+7
	500	4.08 E+8	4.07 E+8	4.08 E+8
	750	8.93 E+8	8.93 E+8	8.93 E+8
3	250	9.33 E+11	9.28 E+11	9.34 E+11
	500	7.74 E+12	7.70 E+12	7.74 E+12
	750	2.72 E+13	2.71 E+13	2.73 E+13
4	250	8.09 E+15	7.94 E+15	8.12 E+15
	500	1.36 E+17	1.34 E+17	1.37 E+17
	750	7.19 E+17	7.15 E17	7.29 E+17

Tableau 7.6 : moyenne de l'hyper-volume ***H*** du MOGLS, GISMOO et GISMOO +LS pour les instances du problème de sac à dos multi-objectifs

Les faibles écarts constatés entre GISMOO et MOGLS combinés aux résultats obtenus avec l'ajout d'une méthode d'intensification à notre approche hybride semblent confirmer les lacunes en terme d'intensification de la recherche de notre algorithme observées à la section précédente. Toutefois, ces résultats laissent supposer que, même si GISMOO n'intensifie pas la recherche comme le MOGLS, il parvient à trouver des solutions dans toutes les régions de l'espace de solutions ce qui explique sans doute les faibles écarts en terme d'hyper-volume entre les deux algorithmes.

La Figure 7.4 présente les résultats des trois algorithmes selon la métrique ***C*** de façon similaire à la Figure 7.1. Ainsi, chaque boîte de la figure correspond à la comparaison des solutions de l'algorithme *A* en ligne avec l'algorithme *B* en colonne. On observe, à partir de cette figure, que les résultats de GISMOO sont inférieurs à ceux du MOGLS et de GISMOO+LS pour toutes les instances. En comparant maintenant GISMOO+LS avec

MOGLS, on note qu'à l'exception des trois instances à deux objectifs, les résultats penchent en faveur de GISMOO+LS. Ainsi, GISMOO+LS obtient de meilleurs résultats que le MOGLS pour 5 instances, est inférieur pour trois instances tout en obtenant un résultat identique pour l'instance restante. En examinant les graphiques de la Figure 7.4, on remarque aussi que l'avantage de GISMOO+LS par rapport au MOGLS augmente en fonction du nombre d'objectifs et du nombre d'objets du problème à résoudre. Ces résultats laissent supposer que les mécanismes d'exploration de GISMOO+LS sont plus performants que ceux du MOGLS, en particulier lorsque le nombre d'objets et le nombre d'objectifs du problème à résoudre augmentent. Toutefois, pour les problèmes bi-objectifs, le MOGLS effectue une meilleure intensification de la recherche comparativement à GISMOO+LS.

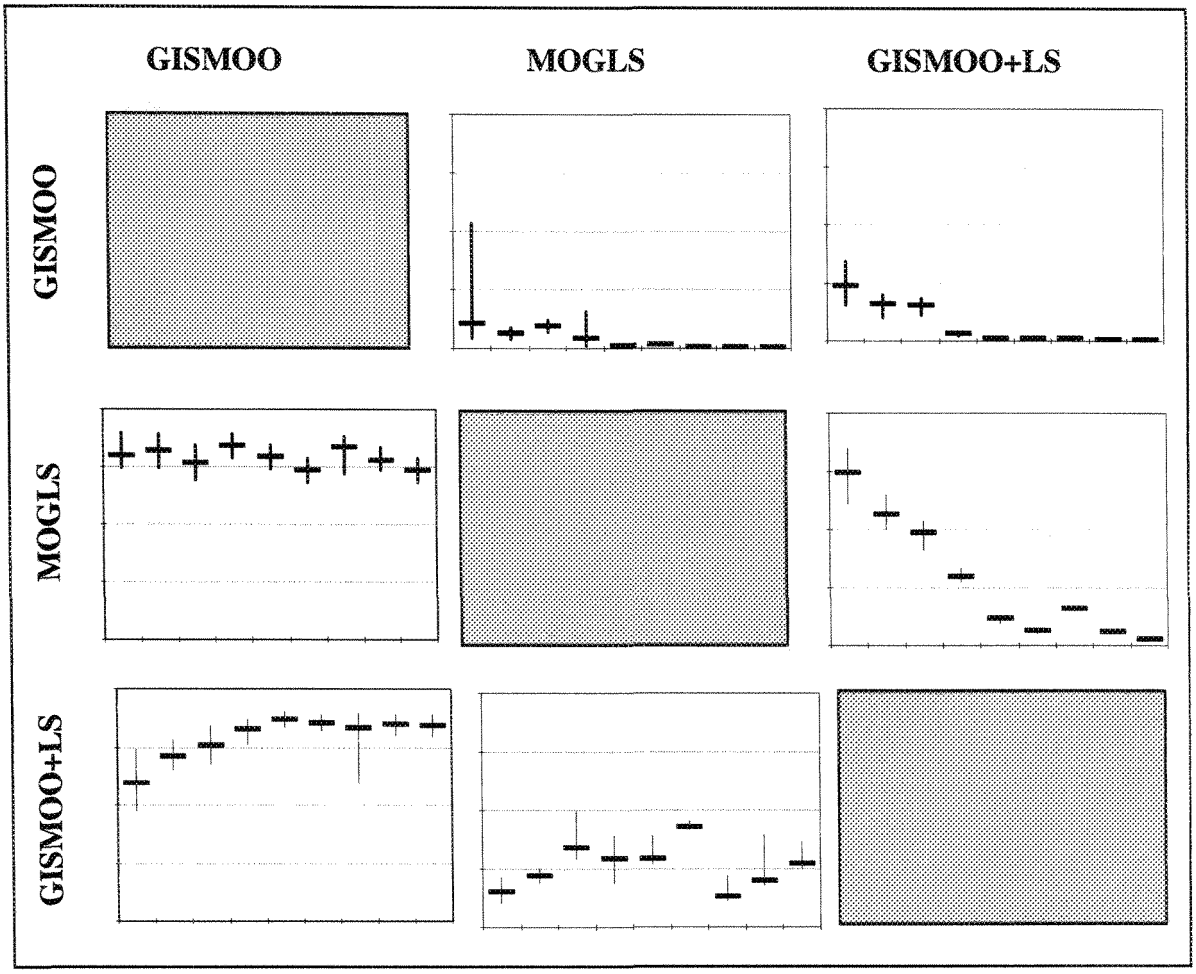


Figure 7.4 : Couverture moyenne ***C*** de GISMOO, MOGLS et GISMOO+LS pour le problème de sac à dos multi-objectifs

Pour confirmer les tendances observées à l'aide de la métrique ***C***, nous avons également comparé les trois algorithmes en utilisant la métrique ***Diff*** qui permet de mesurer la différence de couverture entre les solutions de deux algorithmes. Les Tableaux 7.7 à 7.9 résument respectivement les résultats obtenus entre GISMOO et MOGLS, entre GISMOO et GISMOO+LS et entre GISMOO+LS et MOGLS en utilisant cette métrique. Les deux premières colonnes de chaque tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats

moyens de 30 exécutions indépendantes de chacun des algorithmes pour la métrique ***Diff***. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris.

Nombre d'objectifs	Nombre d'objets	GISMOO	MOGLS
2	250	3.23 E+3	7.60 E+4
	500	4.25 E+4	2.16 E+5
	750	9.94 E+3	3.26 E+5
3	250	7.58 E+7	5.67 E+9
	500	7.37 E+8	3.94 E+10
	750	4.50 E+9	1.10 E+11
4	250	3.09 E+12	1.50 E+14
	500	4.87 E+13	2.23 E+15
	750	3.98 E+14	9.52 E+15

Tableau 7.7 : différence de couverture moyenne entre GISMOO et MOGLS pour les instances du problème de sac à dos multi-objectifs

Nombre d'objectifs	Nombre d'objets	GISMOO	GISMOO+LS
2	250	1.15 E+04	4.46 E+04
	500	4.70 E+4	1.61 E+5
	750	7.99 E+3	3.05 E+5
3	250	2.45 E+7	5.88 E+9
	500	1.01 E+8	4.66 E+10
	750	3.51 E+8	1.47 E+11
4	250	7.54 E+11	1.85 E+14
	500	4.62 E+12	2.98 E+15
	750	2.52 E+13	1.39 E+16

Tableau 7.8 : différence de couverture moyenne entre GISMOO et GISMOO+LS pour les instances du problème de sac à dos multi-objectifs

Nombre d'objectifs	Nombre d'objets	MOGLS	GISMOO+LS
2	250	4,36 E+4	4,03 E+3
	500	1.06 E+5	4.53 E+4
	750	5.42 E+4	3.58 E+4
3	250	5.78 E+08	8.56 E+8
	500	1.01 E+9	9.03 E+9
	750	1.25 E +9	4.3 E+10
4	250	6.54 E+12	3.97 E+13
	500	2.73 E+13	8 E+14
	750	8.99 E+13	4.39 E+15

Tableau 7.9 : différence de couverture moyenne entre le MOGLS et GISMOO +LS pour les instances du problème de sac à dos multi-objectifs

Les résultats obtenus avec la métrique ***Diff*** confirment les tendances observées précédemment. Ainsi, GISMOO+LS et le MOGLS obtiennent une meilleure différence de couverture que GISMOO pour toutes les instances (Tableaux 7.7 et 7.8). On note aussi que les écarts entre GISMOO et GISMOO+LS sont plus importants que ceux entre GISMOO et MOGLS pour les problèmes avec 3 et 4 objectifs. Par contre, pour les problèmes à 2 objectifs, l'écart entre GISMOO et MOGLS est plus important que celui entre GISMOO et GISMOO+LS. Lorsque l'on compare GISMOO+LS au MOGLS à l'aide du Tableau 7.9, on note que GISMOO+LS obtient les meilleurs résultats pour six des neuf instances et est inférieur seulement pour les instances à deux objectifs. Comme pour la métrique **C**, on observe que l'avantage de GISMOO+LS par rapport au MOGLS augmente en fonction du nombre d'objectifs et du nombre d'objets du problème à résoudre. Ainsi, pour les instances avec deux objectifs, l'écart entre les deux algorithmes a tendance à se réduire en fonction de l'augmentation du nombre d'objets. Pour les instances à trois et à quatre objectifs, l'écart entre les deux algorithmes à tendance à augmenter en relation avec l'augmentation du nombre d'objets. Ces résultats confirment que GISMOO+LS effectue une meilleure

exploration de l'espace de recherche que le MOGLS, en particulier lorsque le nombre d'objectifs et le nombre d'objets du problème augmentent. De plus, les mesures obtenues à l'aide de la métrique ***Diff*** nous permettent aussi de conclure que les surfaces de compromis calculées par GISMOO+LS sont généralement mieux distribuées que celles du MOGLS, en particulier pour les instances avec plus de deux objectifs.

La Figure 7.5 (a, b, c) représente graphiquement les surfaces des compromis générées par les trois algorithmes pour chacune des instances à deux objectifs. Cette représentation confirme les résultats obtenus avec les autres mesures. On note en observant les graphiques qu'il est très difficile de départager visuellement les trois algorithmes, car les surfaces de compromis sont quasiment confondues. On note toutefois que, pour l'instance à 250 objets, la surface de compromis du MOGLS est plus dense que celle de GISMOO et GISMOO+LS. Par contre, pour les deux autres instances, on observe que les surfaces de GISMOO et GISMOO+LS semblent légèrement plus étendues que celle du MOGLS. On note aussi que les solutions de GISMOO sont légèrement inférieures de celles fournies par MOGLS et GISMOO+LS. Ces observations expliquent les faibles écarts observés précédemment entre les performances des différents algorithmes sur les différentes mesures.

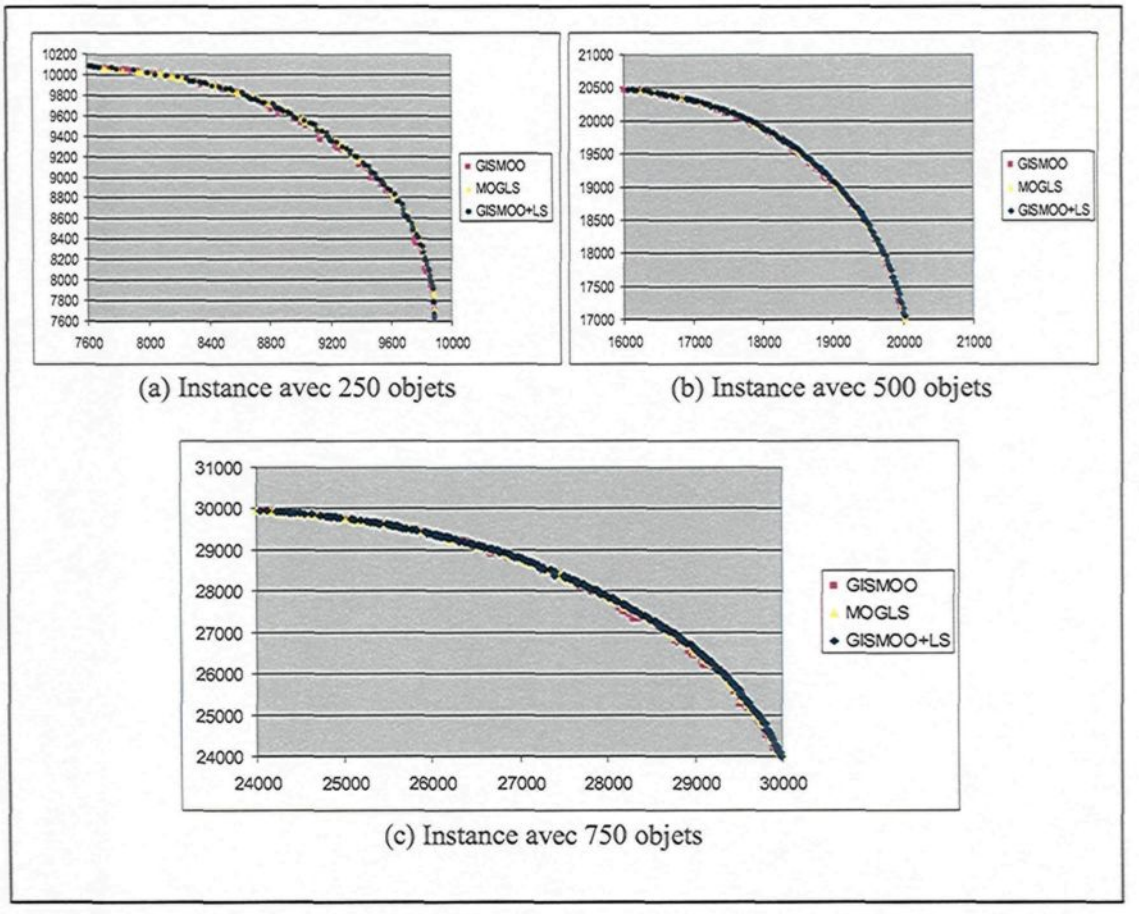


Figure 7.5 : Exemple de surfaces de compromis trouvées par GISMOO, MOGLS et GISMOO+LS pour les trois instances à deux objectifs

Le Tableau 7.10 indique (en secondes), pour chacun des trois algorithmes et pour chaque instance du problème, les temps moyens d'exécution obtenus. On note, à l'aide de ce tableau, que GISMOO nécessite toujours moins de temps de calcul que MOGLS qui lui nécessite moins de temps de calcul que GISMOO+LS. On note, ainsi que même si le nombre de générations de MOGLS est fixé à 50, le temps de calcul nécessaire à cet algorithme pour les réaliser est toujours supérieur à celui de GISMOO. Finalement, on remarque que l'ajout de la procédure de recherche locale à GISMOO entraîne une

augmentation du temps d'exécution de l'algorithme supérieure à 100% pour toutes les instances. Ces résultats sont intéressants sachant qu'à chaque itération le MOGLS n'effectue qu'un seul croisement. En effet, cela implique que le nombre d'opérations nécessaires pour effectuer ce croisement ainsi que le nombre d'amélioration locale effectuées par l'algorithme sont très important. Ceci explique pourquoi l'auteur de cet algorithme [Jaszkiewicz 2000] compare que 50 générations du MOGLS à 500 générations d'un algorithme évolutionnaire classique. Il est important de mentionner que les caractéristiques du MOKP font qu'il n'est pas nécessaire de réévaluer au complet une solution après chaque mouvement de recherche locale. L'écart entre les temps de calculs de GISMOO et du MOGLS serait donc encore plus important pour un problème ou une évaluation complète de la solution serait nécessaire après chaque mouvement.

Nombre d'objectifs	Nombre d'objets	MOGLS	GISMOO	GISMOO+LS
2	250	10	9	34
	500	19	13	48
	750	27	24	67
3	250	22	19	51
	500	53	35	97
	750	77	55	131
4	250	56	52	99
	500	107	85	177
	750	202	108	289

Tableau 7.10 : Moyenne des temps de calculs (secondes) du MOGLS, GISMOO et GISSMO+LS pour les instances du problème de sac à dos multi-objectifs

7.6 Conclusion

Dans ce chapitre, nous avons proposé un nouvel algorithme multi-objectifs Pareto (GISMOO) combinant des concepts issus des algorithmes génétiques avec des propriétés inspirées des systèmes immunitaires artificiels. Les expérimentations numériques

effectuées ont permis de démontrer l'efficacité de notre approche sur le problème de sac à dos multi-objectifs en 0/1. En effet, les performances de notre algorithme sont supérieures à celles obtenues par deux autres algorithmes Pareto représentatifs de l'état de l'art en optimisation multi-objectifs à l'aide d'algorithmes évolutionnaires : le NSGAII et le PMS^{MO}. De plus, les résultats expérimentaux montrent aussi que, lorsque l'on utilise des méthodes d'intensification équivalentes, les performances de GISMOO se comparent avantageusement par rapport à celles d'un algorithme mimétique comme le MOGLS. Une conclusion naturelle à ces expérimentations numériques est que, malgré sa simplicité, GISMOO est une alternative efficace pour résoudre des problèmes d'optimisation multi-objectifs. En effet, contrairement au NSGAII, au PMS^{MO} ou au MOGLS qui sont basés sur des modèles évolutionnaires complexes, GISMOO est un algorithme très simple à implémenter. En particulier, ces résultats mettent en évidence l'importance des mécanismes de diversification de la recherche sur l'efficacité globale d'un algorithme multi-objectifs. Les résultats de GISMOO sur ce benchmark classique de la littérature ont permis d'apprécier ses performances. Il est important de rappeler que GISMOO est un algorithme Pareto générique, ce qui signifie qu'il peut être adapté à la résolution de n'importe quel problème multi-objectifs.

CHAPITRE 8

APPLICATION DE GISMOO À LA RÉSOLUTION DU POVI

8.1 Introduction

Le chapitre précédent a permis de présenter GISMOO, un nouvel algorithme générique permettant de résoudre de manière efficace les PMO. GISMOO est un algorithme hybride qui combine des concepts issus des algorithmes génétiques avec des paradigmes empruntés aux systèmes immunitaires artificiels. Comme tout algorithme Pareto, le but de GISMOO est bien évidemment d'offrir une bonne approximation de la frontière Pareto, mais aussi d'obtenir une population de solutions non dominées la plus diversifiée possible, en explorant de manière intelligente l'espace de recherche. Les expérimentations numériques effectuées ont ainsi montré que l'approche proposée était particulièrement efficace pour approximer la frontière Pareto en une seule exécution. En particulier, les performances de GISMOO sur un « benchmark » classique en optimisation multi-objectifs (le MOKP) se comparent avantageusement avec celles d'autres algorithmes représentatifs de l'état de l'art. Contrairement à la plupart des algorithmes multi-objectifs de la littérature, GISMOO ne requiert pas la calibration de nombreux paramètres qui détermine son efficacité. En effet, la plupart des paramètres de l'approche proposée au chapitre précédent sont implicites et leur calibration est automatique.

Dans ce chapitre, nous allons maintenant appliquer GISMOO à la résolution du problème industriel d'ordonnancement de voitures. En effet, comme nous l'avons déjà mentionné à plusieurs reprises dans cette thèse, le POVI est un problème avec trois objectifs conflictuels à optimiser. Jusqu'à présent, ce problème a toutefois été traité en optimisant un seul objectif ou en considérant les trois objectifs de manière lexicographique. À notre connaissance, ce

problème n'a jamais été abordé d'un point de vue Pareto. Par conséquent, les principales approches Pareto n'ont jamais été utilisées pour résoudre ce problème.

L'objectif de ce chapitre est donc double. Il consiste, d'une part, à aborder pour la première fois le POVI en traitant les objectifs simultanément sans leur attribuer d'ordre ou de poids. D'autre part, il va permettre de valider les performances de GISMOO sur une problématique réelle multi-objectifs.

Le reste de ce chapitre est organisé de la manière suivante. La Section 8.2 présente les adaptations faites à GISMOO pour résoudre le POVI. Par la suite, à la Section 8.3, les résultats obtenus par notre approche sont analysés en utilisant les instances du POVI présentées au Chapitre 6. Comme au chapitre précédent, les tests effectués dans cette partie de la thèse sont réalisés en suivant les standards adoptés par la communauté multi-objectifs [Coello Coello *et al.* 2002].

8.2 Description de GISMOO pour le POVI

Dans cette section, nous présentons les différentes adaptations incorporées à GISMOO afin de l'appliquer à la résolution du POVI.

8.2.1 Représentation

Contrairement au MOKP utilisé au chapitre précédent, le POVI n'est pas un problème codé sous forme binaire. La représentation classique des AG apparaît donc peu adaptée pour ce problème. En effet, une solution réalisable du POVI est composée de deux vecteurs de *nc* composantes satisfaisant les différentes contraintes du problème. Chaque vecteur indique respectivement les classes et les couleurs des différentes voitures. Par conséquent,

nous avons opté pour une représentation sous forme de vecteurs d'entiers pour représenter chaque individu.

8.2.2 Génération de la population initiale

Comme mentionné à la Section 7.3 du Chapitre 7, l'algorithme GISMOO commence en générant une population de départ constituée à 70% d'individus générés aléatoirement et à 30% d'individus générés en utilisant une heuristique gourmande. Pour construire chaque individu aléatoirement, dans le cas du POVI, on ajoute de façon itérative à l'individu en cours de création la classe et la couleur d'une voiture choisie au hasard dans l'ensemble de voitures à placer. Les heuristiques gourmandes utilisées pour générer des individus sont les heuristiques *greedy_color* et *greedy_ratio* présentées à la Section 6.3.2 du Chapitre 6. On note que chaque fois qu'un individu doit être créé de manière gourmande, on choisit aléatoirement l'heuristique à appliquer.

8.2.3 Opérateur de croisement

Dans la version de GISMOO pour le POVI, nous utilisons l'opérateur de croisement $NCPX^{MO}$ que nous avons proposé au Chapitre 6. Cependant, au lieu d'utiliser la formule du *TWI* introduite au Chapitre 6, dans cette partie de la thèse, nous utilisons plutôt la notion d'*intérêt total* (TI) 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 :

$$TI_{v,color,i} = I_{v,i,HPO} + I_{v,i,LPO} + I_{v,i,COLOR} \qquad (8.1)$$

On note que la seule différence entre le calcul du TI et du TWI réside dans le fait que le calcul du TI ne fait intervenir aucune pondération des objectifs. De cette manière, il permet

une compensation entre les objectifs et autorise ainsi la recherche de solutions de compromis.

8.2.4 Opérateurs de mutation

Pour résoudre le POVI avec GISMOO, nous utilisons uniquement deux des quatre opérateurs de mutation présentés au Chapitre 6, à savoir : l'*inversion_simple* et l'*échange_aléatoire*. Pendant la phase génétique de GISMOO, les individus générés sont mutés en utilisant l'opérateur de mutation par inversion. Comme nous l'avons indiqué au chapitre précédent lors de la description de GISMOO, les deux opérateurs de mutation sont appliqués pendant la phase immune de l'algorithme.

8.2.5 Sélection

Comme pour le MOKP, plusieurs schémas de sélection auraient pu être envisagés pour cette version de GISMOO permettant de résoudre le POVI. Toutefois, de manière à rester fidèle à la description de l'algorithme faite au chapitre précédent, nous utilisons aussi une sélection par tournoi dans la phase génétique de l'algorithme. On rappelle que, dans GISMOO, le tournoi est effectué en considérant le facteur de dominance des individus participants au tournoi et en cas d'égalité, l'égalité est brisée en utilisant le facteur d'isolement.

8.2.6 Remplacement

La stratégie de remplacement utilisée par GISMOO dans cette partie est un remplacement déterministe de type $(\lambda+\mu)$, où λ indique la taille de la population Parent et μ la taille de la population Enfant. On rappelle que ce schéma de remplacement consiste

simplement à joindre et trier les populations Parent et Enfant pour ne conserver que les λ meilleurs individus afin de former la prochaine génération.

8.3 Expérimentations numériques

La version de GISMOO pour résoudre le POVI présentée dans ce chapitre a été implémentée 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.

8.3.1 Jeux d'essais

Les expérimentations numériques présentées dans ce chapitre sont réalisées à l'aide des jeux d'essais fournis par le constructeur Renault qui sont disponibles sur Internet (http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2005/). Il s'agit des mêmes instances que celles utilisées au Chapitre 6.

8.3.2 Comparaison expérimentale

Dans cette section, nous comparons les performances de GISMOO à celle du PMS^{MO} et du NSGAII. Comme pour GISMOO, les algorithmes PMS^{MO} et NSGAII ont été implémentés en C^{++}. Dans notre implémentation, les principales structures de données sont partagées par tous les algorithmes. Nous obtenons ainsi une base de comparaison commune pour évaluer équitablement les performances des différents algorithmes. Dans toutes les expérimentations numériques effectuées dans cette partie, chaque instance du POVI a été résolue à 30 reprises par chaque algorithme sur le même ordinateur. Pour cette série d'expérimentations numériques, nous utilisons les paramètres suivants :

- La taille N des populations est la même pour chaque algorithme et est fixée empiriquement à 100 individus pour toutes les instances.
- Pour les trois algorithmes, la probabilité de mutation a été fixée à 0.06.
- Le temps maximum alloué à chaque algorithme pour une exécution est fixé à 350 secondes. Ce temps d'exécution est le même que celui accordé pour les expérimentations effectuées au Chapitre 6 de cette thèse et correspond à la limite de temps fixée lors du Challenge ROADEF 2005.
- Pour NSGAII et PMS^{MO}, la probabilité de croisement est fixée à 100 %. La taille de l'archive locale du PMS^{MO} est quant à elle fixée à un maximum de 100 individus.

Les Tableaux 8.1 à 8.3 présentent les résultats obtenus par les trois algorithmes en ce qui concerne la métrique d'hyper-volume $\boldsymbol{H}$ (présentée à la Section 3.8.3 du Chapitre 3) pour les trois ensembles tests du POVI. Chaque ligne de ces tableaux donne respectivement le nom de l'instance, le nombre de voitures à ordonnancer (Taille) ainsi que les résultats moyens obtenus pour la métrique $\boldsymbol{H}$ pour chaque algorithme. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. De plus, mentionnons que nous avons indiqué par un (*) les instances pour lesquelles les trois algorithmes trouvent exactement le même ensemble de solutions à chaque exécution. Il est important de préciser ici, qu'étant dans un contexte de minimisation des objectifs, le calcul de l'hyper-volume ne se fait pas par rapport aux axes du repère, mais par rapport à un point de référence *ref* dans l'espace des objectifs [Zitzler 1999]. Dans la pratique, on prend souvent le point *Nadir* des ensembles à évaluer comme point de référence. Schématiquement, les coordonnées du point Nadir correspondent aux pires valeurs de chaque objectif des points

des ensembles à évaluer. La Figure 8.1 schématise le calcul de l'hyper-volume dans un contexte de minimisation.

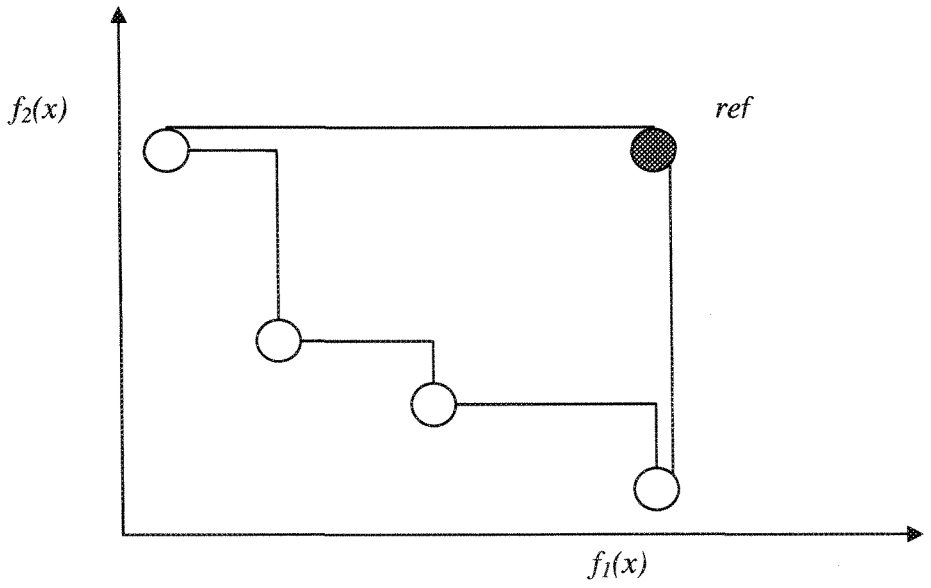


Figure 8.1 : Exemple d'application de la métrique ***H*** dans un contexte de minimisation

À partir des résultats des Tableaux 8.1 à 8.3, nous observons que GISMOO obtient les meilleurs résultats sur toutes les instances des trois ensembles. En effet, la taille de l'espace dominé par GISMOO est toujours identique ou plus importante que celle du NSGAII et du PMS^{MO}. En fait, lorsqu'on compare GISMOO au NSGAII, on constate qu'à l'exception de quatre instances de l'ensemble X où les deux algorithmes obtiennent des résultats identiques, GISMOO dépasse NSGAII sur les 38 instances restantes. Par rapport au PMS^{MO}, GISMOO obtient aussi de meilleurs résultats sur 38 instances tout en obtenant des résultats identiques sur les 4 instances restantes. Si on cherche à comparer NSGAII et PMS^{MO}, on remarque que les résultats sont plus partagés et que les performances de deux algorithmes sont souvent très proches l'une de l'autre.

Instance	Taille	NSGAII	PMS^{MO}	GISMOO
022_3_4	485	1.16 E+8	1.17 E+8	1.22 E+8
024_38_3	1260	6.05 E+6	6.47 E+6	8.7 E+6
024_38_5	1315	1.58 E+7	1.58 E+7	1.93 E+7
025_38_1	1004	1.41 E+8	1.65 E+8	1.76 E+8
039_38_4_ch1	954	7.41 E+6	7.40 E+6	8.30 E+6
048_39_1	600	5.27 E+7	5.7 E+7	6.16 E+7
064_38_2_ch1	875	3.46 E+7	3.48 E+7	3.94 E+7
064_38_2_ch2	335	1.45 E+7	1.52 E+7	1.65 E+7

Tableau 8.1 : Moyenne de l'hyper-volume ***H*** du NSGAII, PMS^{MO} et GISMOO pour les instances de l'ensemble *A* du POVI

Instance	Taille	NSGAII	PMS^{MO}	GISMOO
022_S22_J1	526	1.54 E+6	1.86 E+6	2.03 E+6
023_S23_J3	1110	6.46 E+6	6.98 E+5	7.26 E+6
024_V2_S22_J1	1270	7.6 E+7	8.71 E+7	1.04 E+8
025_S22_J3	1161	8.04 E+6	9.44 E+6	1.02 E+7
028_ch1_S22_J2	365	3.41 E+6	3.35 E+6	5.43 E+6
028_ch2_S23_J3	65	8.56 E+3	9.13 E+3	9.14 E+3
029_ S21_J6	730	2.93 E+7	2.89 E+7	3.56 E+7
035_ch1_S22_J3	128	8.64 E+7	8.79 E+7	9.14 E+7
035_ch2_S22_J3	269	1.39 E+8	1.45 E+8	1.56 E+8
039_ch1_S22_J4	1231	1.24 E+8	1.37 E+8	1.42 E+8
039_ch3_ S22_J4	1000	5.86 E+6	1.06 E+7	2.01 E+7
048_ch1_ S22_J3	591	6.61 E+7	7.2 E+7	7.41 E+7
048_ch2_ S22_J3	546	9.94 E+7	1.01 E+8	1.05 E+8
064_ch1_ S22_J3	825	4.24 E+7	4.14 E+7	5.12 E+7
064_ch2_ S22_J4	412	8.04 E+7	8.11 E+7	9.15 E+7

Tableau 8.2 : Moyenne de l'hyper-volume ***H*** du NSGAII, PMS^{MO} et GISMOO pour les instances de l'ensemble *B* du POVI

Instance	Taille	NSGAII	PMS^{MO}	GISMOO
022_S49_J2	704	1.17 E+8	1.18 E+8	1.2 E+8
023_S49_J2	1260	5.46 E+7	5.53 E+7	6.59 E+7
024_S49_J2	1319	1.02 E+8	1.02 E+8	1.1 E+8
025_ S49_J1	996	2.30 E+8	2.36 E+8	2.53 E+8
028_CH1_S50_J4	325	1.51 E+8	1.55 E+8	1.66 E+8
028_CH2_S51_J1*	65	1.00 E+3	1.00 E+3	1.00 E+3
029_S49_J5	780	1.54 E+8	1.34 E+8	1.72 E+8
034_VP_S51_J1_J2_J3	921	1.45 E+8	1.5 E+8	1.54 E+8
034_VU_S51_J1_J2_J3	231	9.05 E+7	9.11 E+7	1.00 E+8
035_CH1_S50_J4*	90	9.85 E+6	9.85 E+6	9.85 E+6
035_CH2_S50_J4	376	4.36 E+5	5.00 E+5	6.09 E+5
039_CH1_S49_J1*	1247	2.5 E+8	2.5 E+8	2.5 E+8
039_CH3_S49_J1	1037	1.59 E+8	1.67 E+8	1.81 E+8
048_CH1_S50_J4	519	7.32 E+7	7.36 E+7	8.71 E+7
048_CH2_S49_J5	459	8.5 E+7	9.03 E+7	9.62 E+7
064_CH1_S49_J1	875	3.53 E+7	3.53 E+7	4.19 E+7
064_CH2_S49_J4	273	9.76 E+7	9.76 E+7	9.77 E+7
655_CH1_S51_J2_J3_J4*	264	1.00 E+9	1.00 E+9	1.00 E+9
655_CH2_S52_J1_J2_S01_J1	219	2.08 E+6	2.17 E+6	2.37 E+6

Tableau 8.3 : Moyenne de l'hyper-volume ***H*** du NSGAII, PMS^{MO} et GISMOO pour les instances de l'ensemble *X* du POVI

Si par définition l'hyper-volume donne une bonne idée de la taille de l'espace dominé par un ensemble de solutions, cette métrique ne permet pas de comparer deux ensembles de solutions l'un par rapport à l'autre. Pour y parvenir, nous utilisons la métrique ***C*** (décrite à la Section 3.8.2 du Chapitre 3). La Figure 8.2 présente les résultats des trois algorithmes selon la métrique ***C*** suivant la présentation adoptée par Zitzler et Thiele [1999]. Dans cette figure, chaque boîte correspond à la comparaison des solutions de l'algorithme *A* en ligne avec l'algorithme *B* en colonne. Ainsi, la valeur ***C***(A,B) indique le pourcentage d'éléments de *B* dominés par au moins un élément de *A*. Pour chaque boîte, l'échelle part de 0 en bas de la boîte pour aller à 1 en haut de celle-ci. Chacune des boîtes contient 8 graphiques

correspondants respectivement aux 8 instances de l'ensemble A. Chaque barre horizontale noire indique la moyenne des différentes mesures C pour 30 exécutions et les barres verticales indiquent l'écart entre les valeurs maximales et minimales trouvées lors des différentes exécutions.

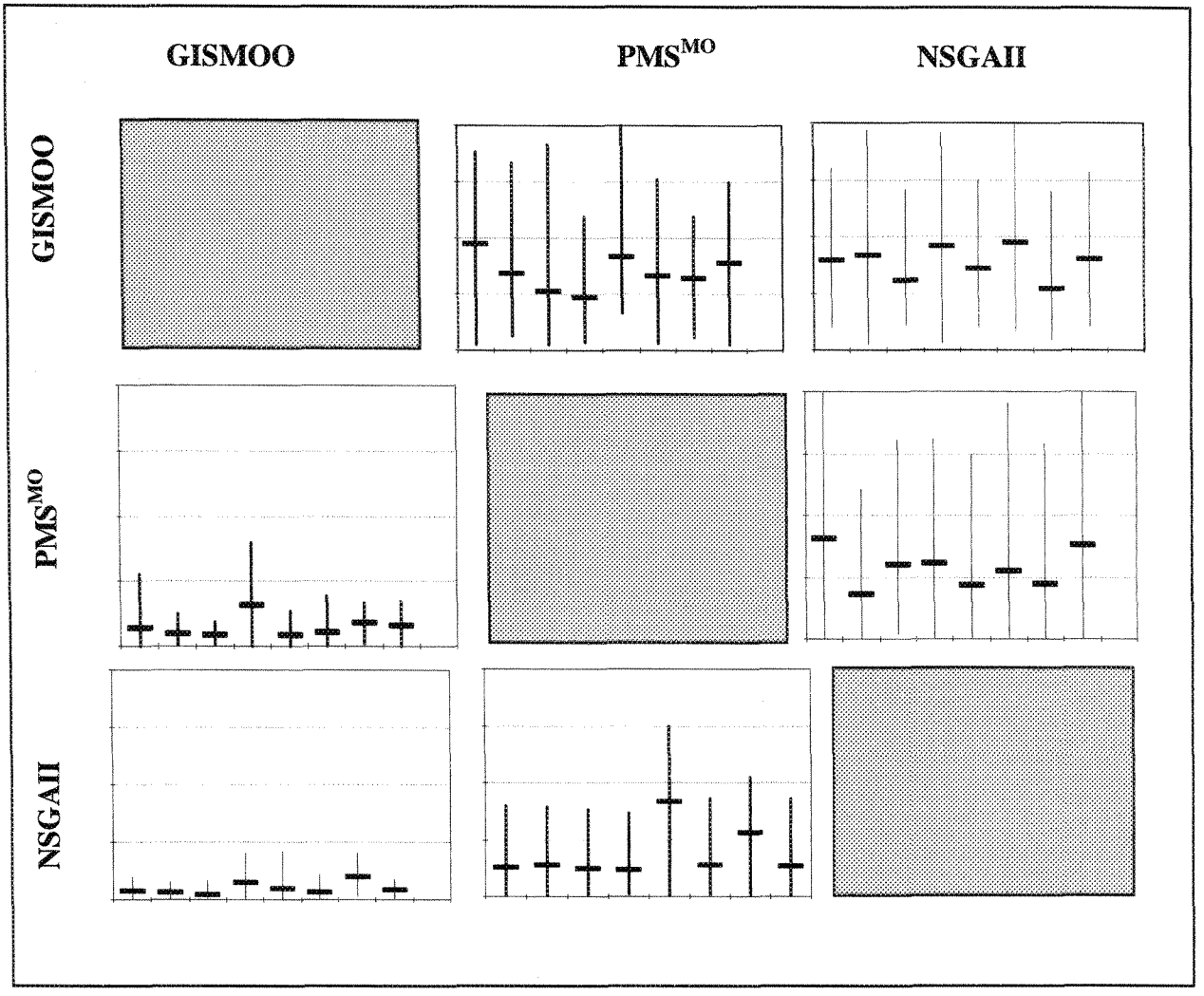


Figure 8.2 : Couverture moyenne C de GISMOO, PMS^{MO} et NSGAII pour les instances de l'ensemble A

De manière similaire à la Figure 8.2, la Figure 8.3 présente les résultats des trois algorithmes selon la métrique ***C*** pour les 15 instances de l'ensemble B.

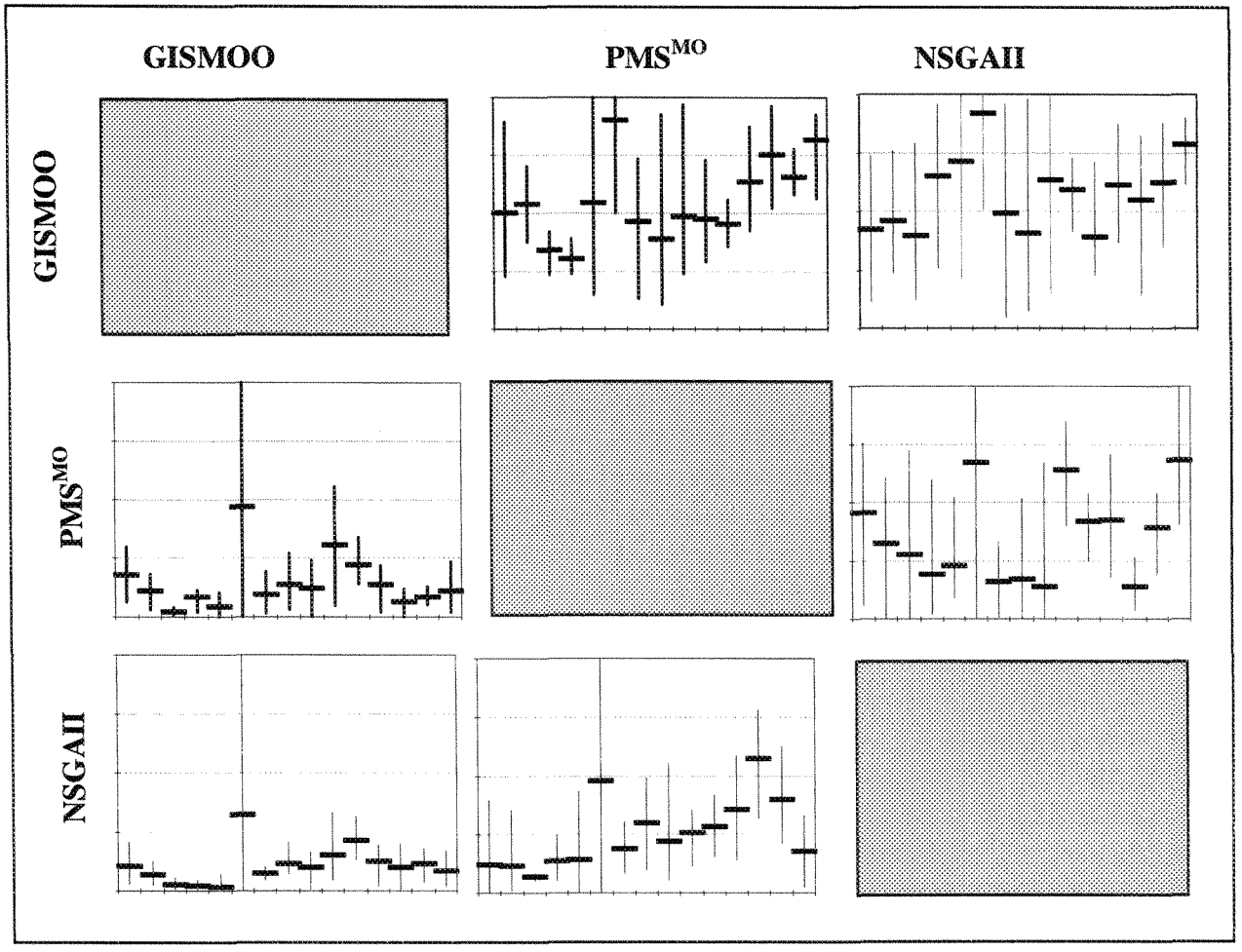


Figure 8.3 : Couverture moyenne ***C*** de GISMOO, PMS^{MO} et NSGAII pour les instances de l'ensemble B

Finalement, la Figure 8.4 présente les résultats des trois algorithmes selon la métrique ***C*** pour 15 instances des 19 instances de l'ensemble X. Comme les trois algorithmes obtenaient exactement les mêmes résultats à chaque exécution pour les quatre instances 028_CH2_S51_J1, 035_CH1_S50_J4, 039_CH1_S49_J1 et 655_CH1_S51_J2_J3_J4, les

résultats de ces instances ne sont pas présentés à la Figure 8.4 afin d'alléger le graphique. Le lecteur peut consulter l'annexe pour obtenir les résultats détaillés pour ces quatre instances particulières.

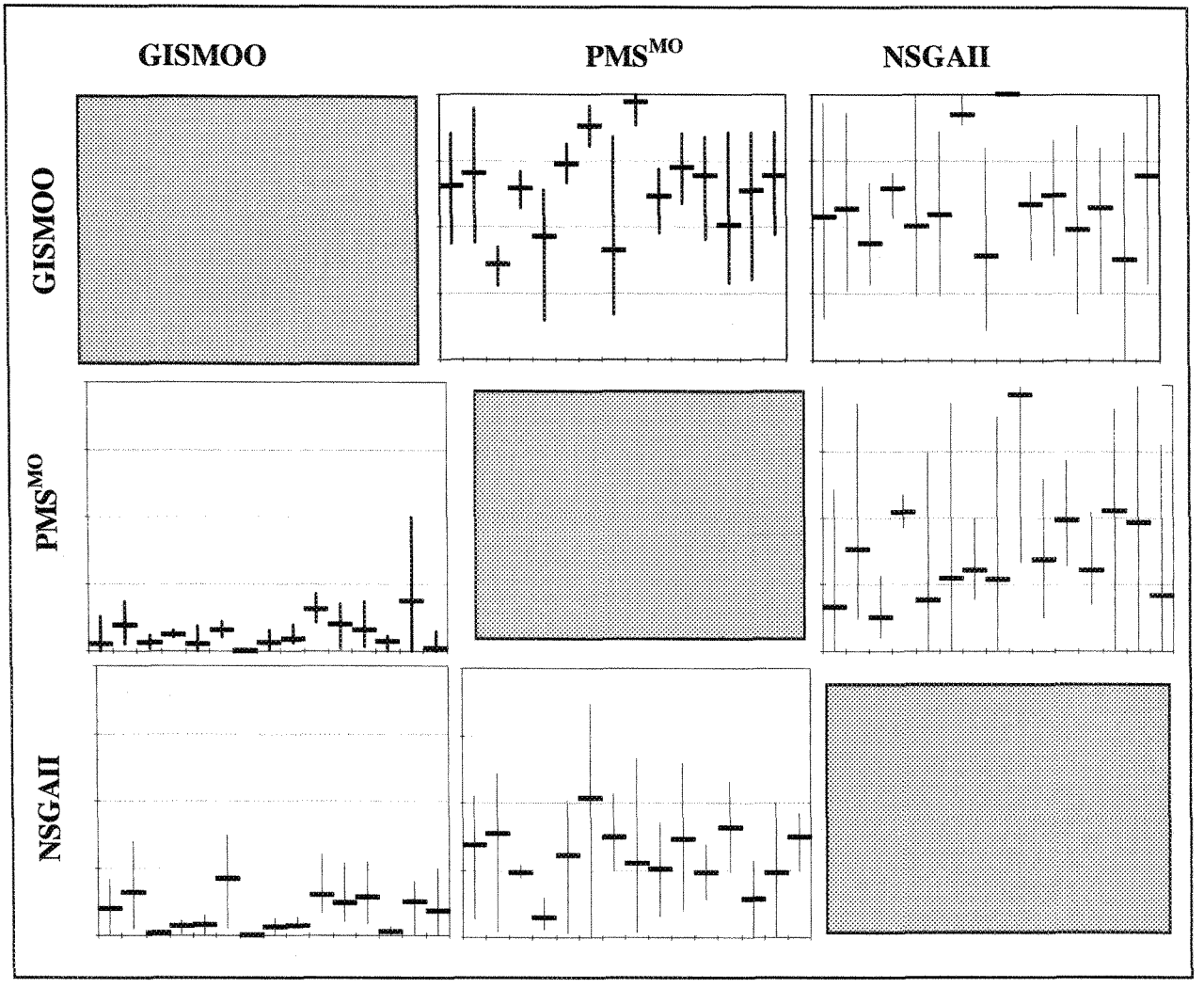


Figure 8.4 : Couverture moyenne C de GISMOO, PMS^{MO} et NSGAII pour les instances de l'ensemble X

Les résultats obtenus en utilisant cette métrique pour les trois ensembles d'instances du POVI sont, une fois de plus, clairement en faveur de GISMOO sur toutes les instances. En

effet, les résultats de C(NSGAII, GISMOO) sont toujours inférieurs à ceux de C(GISMOO, NSGAII). Il est important de noter que sur un grand nombre d'instances, le rapport entre les résultats des deux algorithmes est supérieur à 10, ce qui signifie que la quasi-totalité des solutions trouvées par GISMOO sont non-dominées ou dominent celles trouvées par NSGAII. Si on compare maintenant GISMOO au PMS^{MO}, on remarque que les résultats de C(PMS^{MO}, GISMOO) sont toujours inférieurs à ceux de C(GISMOO, PMS^{MO}). On note aussi que le rapport entre les résultats des deux algorithmes est supérieur à 10 pour la plupart des instances. On en conclut donc que la quasi-totalité des solutions trouvées par GISMOO sont non-dominées ou dominent celles trouvées par PMS^{MO}. Les résultats obtenus en utilisant la métrique C confirment donc les résultats obtenus avec l'hyper-volume.

En plus de l'hyper-volume et de la couverture, nous avons aussi comparé les trois algorithmes en utilisant la métrique ***Diff*** (présentée à la Section 3.8.4 du Chapitre 3) qui permet de mesurer la différence de couverture entre les solutions de deux algorithmes. Les Tableaux 8.4 à 8.9 résument respectivement les résultats obtenus entre GISMOO et NSGAII et entre GISMOO et PMS^{MO} en utilisant cette métrique pour chaque ensemble test. Chaque ligne de ces tableaux donne respectivement le nom de l'instance, le nombre de voitures à ordonnancer (Taille) et, pour chaque algorithme, les résultats moyens obtenus pour la métrique ***Diff*** après 30 exécutions indépendantes. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris.

Instance	Taille	NSGAII	GISMOO
022_3_4	485	3.09 E+6	9.10 E+6
024_38_3	1260	1.16 E+5	1.19 E+5
024_38_5	1315	1.06 E+8	1.09 E+8
025_38_1	1004	8.24 E+8	8.59 E+8
039_38_4_ch1	954	4.2 E+6	5.09 E+6
048_39_1	600	1.88 E+8	1.97 E+8
064_38_2_ch1	875	2.11 E+6	2.15 E+6
064_38_2_ch2	335	8.4 E+6	1.05 E+7

Tableau 8.4 : Différence de couverture moyenne du NSGAII et GISMOO pour les instances de l'ensemble A

Instance	Taille	PMS^{MO}	GISMOO
022_3_4	485	3.09 E+6	7.88 E+6
024_38_3	1260	1.16 E+5	1.19 E+5
024_38_5	1315	1.06 E+8	1.09 E+8
025_38_1	1004	8.25 E+8	8.35 E+8
039_38_4_ch1	954	4.2 E+6	5.1 E+6
048_39_1	600	1.88 E+8	1.93 E+8
064_38_2_ch1	875	2.11 E+6	2.15 +6
064_38_2_ch2	335	8.5 E+6	9.82 E+6

Tableau 8.5 : Différence de couverture moyenne du PMS^{MO} et GISMOO pour les instances de l'ensemble A

Instance	Taille	NSGAII	GISMOO
022_S22_J1	526	1.97 E+6	2.46 E+6
023_S23_J3	1110	2.77 E+6	2.85 E+6
024_V2_S22_J1	1270	3 E+6	3.05 E+6
025_S22_J3	1161	2.4 E+6	2.77 E+6
028_ch1_S22_J2	365	1.96 E+6	2.16 E+6
028_ch2_S23_J3	65	8.66 E+2	1.44 E+3
029_ S21_J6	730	5.89 E+6	5.96 E+6
035_ch1_S22_J3	128	3.36 E+7	3.86 E+7
035_ch2_S22_J3	269	3.44 E+7	3.61 E+7
039_ch1_S22_J4	1231	1.08 E+8	1.26 E+8
039_ch3_ S22_J4	1000	1.05 E+5	1.19 E+7
048_ch1_ S22_J3	591	1 E+6	1.09 E+6
048_ch2_ S22_J3	546	1.95 E+7	2.99 +7
064_ch1_ S22_J3	825	7.38 E+6	8.26 E+6
064_ch2_ S22_J4	412	3.35 E+7	4.46 E+7

Tableau 8.6 : Différence de couverture moyenne du NSGAII et GISMOO pour les instances de l'ensemble B

Instance	Taille	PMS^{MO}	GISMOO
022_S22_J1	526	1.98 E+6	2.14 E+6
023_S23_J3	1110	2.77 E+6	2.80 E+6
024_V2_S22_J1	1270	3.02 E+6	3.05 E+6
025_S22_J3	1161	2.6 E+6	2.76 E+6
028_ch1_S22_J2	365	1.96 E+6	2.17 E+6
028_ch2_S23_J3	65	8.7 E+2	8.9 E+2
029_ S21_J6	730	2.88 E+6	5.97 E+6
035_ch1_S22_J3	128	3.37 E+7	3.71 E+7
035_ch2_S22_J3	269	3.44 E+7	3.55 E+7
039_ch1_S22_J4	1231	1.09 E+8	1.13 E+8
039_ch3_ S22_J4	1000	1.04 E+6	1.14 E+7
048_ch1_ S22_J3	591	1.01 E +6	1.03 E +6
048_ch2_ S22_J3	546	1.95 E+7	2.99 E+7
064_ch1_ S22_J3	825	7.36 E+6	8.36 E+6
064_ch2_ S22_J4	412	3.38 E+7	4.39 E+7

Tableau 8.7 : Différence de couverture moyenne du PMS^{MO} et GISMOO pour les instances de l'ensemble B

Instance	Taille	NSGAII	GISMOO
022_S49_J2	704	4.8 E+6	7.82 E+6
023_S49_J2	1260	5.97 E+6	7.04 E+6
024_S49_J2	1319	1.4 E+7	1.48 E+7
025_ S49_J1	996	5.97 E+7	6.2 E+7
028_CH1_S50_J4	325	2.09 E+7	2.24 E+7
028_CH2_S51_J1*	65	0	0
029_S49_J5	780	7.84 E+7	9.62 E+7
034_VP_S51_J1_J2_J3	921	1.96 E+7	2.05 E+7
034_VU_S51_J1_J2_J3	231	2.51 E+7	3.45 E+7
035_CH1_S50_J4*	90	0	0
035_CH2_S50_J4	376	3.91 E+5	5.64 E+5
039_CH1_S49_J1*	1247	0	0
039_CH3_S49_J1	1037	1.94 E+7	2.16 E+7
048_CH1_S50_J4	519	2.88 E+7	3.02 E+7
048_CH2_S49_J5	459	1.54 E+7	1.65 E+7
064_CH1_S49_J1	875	8.31 E+6	8.97 E+6
064_CH2_S49_J4	273	2.34 E+6	2.37 E+6
655_CH1_S51_J2_J3_J4*	264	0	0
655_CH2_S52_J1_J2_S01_J1	219	2.63 E+5	2.92 E+5

Tableau 8.8 : Différence de couverture moyenne du NSGAII et GISMOO pour les instances de l'ensemble X

Instance	Taille	PMS^{MO}	GISMOO
022_S49_J2	704	4.82 E+6	7.21 E+6
023_S49_J2	1260	5.91 E+6	6.97 E+6
024_S49_J2	1319	1.4 E+7	1.48 E+7
025_ S49_J1	996	5.97 E+7	6.14 E+7
028_CH1_S50_J4	325	2.1 E+7	2.2 E+7
028_CH2_S51_J1*	65	0	0
029_S49_J5	780	7.85 E+7	1.16 E+8
034_VP_S51_J1_J2_J3	921	1.96 E+7	2 E+7
034_VU_S51_J1_J2_J3	231	2.5 E+7	3.39 E+7
035_CH1_S50_J4*	90	0	0
035_CH2_S50_J4	376	3.91 E+5	5.00 E+5
039_CH1_S49_J1*	1247	0	0
039_CH3_S49_J1	1037	1.94 E+7	2.08 E+7
048_CH1_S50_J4	519	2.88 E+7	3.01 E+7
048_CH2_S49_J5	459	1.54 E+7	1.6 E+7
064_CH1_S49_J1	875	8.31 E+6	8.97 E+6
064_CH2_S49_J4	273	2.34 E+6	2.37 E+6
655_CH1_S51_J2_J3_J4*	264	0	0
655_CH2_S52_J1_J2_S01_J1	219	2.63 E+5	2.83 E+5

Tableau 8.9 : Différence de couverture moyenne du PMS^{MO} et GISMOO pour les instances de l'ensemble X

En examinant les six tableaux, on note une fois de plus que les résultats sont nettement en faveur de GISMOO. Dans un premier temps, lorsqu'on compare GISMOO au NSGAII (Tableaux 8.4, 8.6 et 8.8), on observe que GISMOO obtient une plus grande différence de couverture que NSGAII pour toutes les instances à l'exception de quatre instances de l'ensemble X où les deux algorithmes obtiennent des résultats identiques. Dans un deuxième temps, en comparant GISMOO au PMS^{MO} (Tableaux 8.5, 8.7 et 8.9), on remarque aussi un net avantage en faveur de GISMOO. En effet, GISMOO obtient une plus grande différence de couverture que PMS^{MO} sur toutes les instances testées à l'exception de quatre instances de l'ensemble X où les deux algorithmes obtiennent des résultats identiques. Ces résultats confirment ceux observés avec l'hyper-volume. Cette concordance

entre les résultats des deux métriques s'explique par le fait que le calcul de la métrique ***Diff*** est basé sur le calcul de l'hyper-volume. Combinés aux résultats de l'hyper-volume, ces résultats laissent supposer que les solutions trouvées par GISMOO ont une meilleure distribution que celles trouvées par NSGAII et PMS^{MO}.

8.3.2 Analyse des résultats d'un point de vue décisionnel

Dans cette partie, nous avons voulu comparer la performance des trois algorithmes utilisés précédemment d'un point de vue décisionnel. Afin d'illustrer graphiquement les résultats obtenus, la comparaison entre les différents algorithmes est effectuée en ne considérant que les objectifs HPO et COLOR du POVI. En plus du NSGAII et du PMS^{MO}, nous comparons aussi les résultats de GISMOO avec les résultats obtenus par la meilleure équipe du challenge ROADEF 2005 (BEST_ROADEF). On rappelle que les résultats de BEST_ROADEF ont été obtenus en traitant les objectifs de manière lexicographique. De plus, il est important de préciser que les résultats du challenge ROADEF 2005 présentés dans cette partie ont été obtenus en optimisant les objectifs selon les hiérarchisations suivantes : HPO-COLOR-LPO ou COLOR-HPO-LPO.

La Figure 8.5 présente une comparaison visuelle entre GISMOO, NSGAII, PMS^{MO} et BEST_ROADEF pour une exécution type sur le problème 022_3_4 (un problème de l'ensemble A) qui est, selon Renault, un problème pour lequel les contraintes sur les options prioritaires sont « faciles » à satisfaire. Cette représentation graphique confirme les résultats des différentes mesures. Pour cette instance, il apparaît clairement que l'ensemble Pareto proposé par GISMOO domine ceux proposés par NSGAII et PMS^{MO}. Lorsqu'on compare maintenant les résultats de GISMMOO à ceux de BEST_ROADEF, on remarque

qu'une seule exécution de GISMOO permet d'obtenir les solutions obtenues avec deux exécutions distinctes de l'algorithme utilisé par l'équipe gagnante du challenge. On rappelle ici que le temps alloué à GISMOO correspond au temps limite accordé pour une seule exécution dans le cadre challenge. De plus, on constate que GISMOO permet de mettre en évidence un éventail de solutions alternatives qui sont ignorées par le traitement lexicographique des objectifs imposé par la donnée du Challenge. En effet, en accordant une trop grande importance à un objectif, le traitement lexicographique utilisé par les équipes du challenge ROADEF 2005 fait converger l'algorithme vers une zone restreinte de l'espace de recherche.

D'un point de vue décisionnel, l'ensemble de solutions proposé par GISMOO offre une plus grande latitude au gestionnaire en lui présentant 19 solutions alternatives comparativement aux deux solutions extrêmes proposées par BEST_ROADEF. Ainsi, pour un gestionnaire plutôt orienté vers l'objectif COLOR, on constate qu'en relâchant légèrement l'importance accordée à cet objectif, il peut sauver deux ou même quatre conflits sur l'objectif HPO. Dans ce cas, le nombre de changements de couleur passerait de 11 à 13 tandis que le nombre de conflits sur les options prioritaires passerait de 39 à 35. À l'opposé, un gestionnaire plutôt orienté vers la minimisation de l'objectif HPO serait probablement intéressé de voir l'effet d'un relâchement de l'importance accordée à cet objectif sur le nombre de purges. Grâce à l'ensemble de solutions présenté par GISMOO, il pourrait constater qu'un tel relâchement entraîne une diminution quasi proportionnelle du nombre de purges. En effet, en permettant trois conflits supplémentaires sur l'objectif HPO comparativement à la solution extrême, on sauve trois changements de couleur. Le nombre

de conflits en HPO passe alors de 0 à 3 tandis que le nombre de changement de couleur passe de 31 à 28. L'ensemble de solutions proposé par GISMOO permet aussi à un gestionnaire, n'ayant aucune préférence entre l'objectif HPO et l'objectif COLOR, de choisir une solution plus équilibrée qui entraîne sensiblement le même nombre de conflits en HPO (21) que de changement de couleur (20).

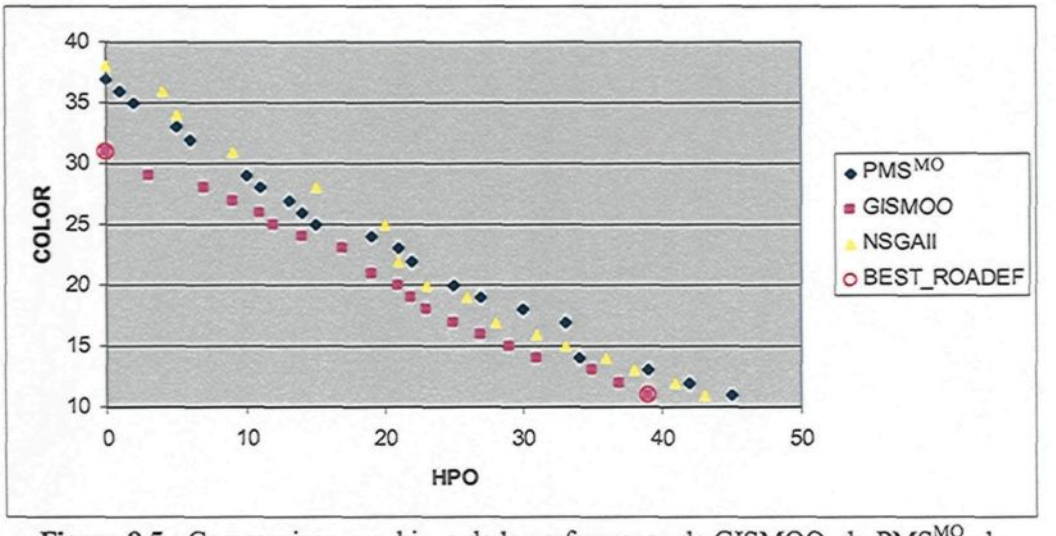


Figure 8.5 : Comparaison graphique de la performance de GISMOO, du PMS^{MO}, du NSGAII et de BEST_ROADEF pour l'instance 022_3_4

Il est important de préciser qu'il n'est toutefois pas toujours possible de générer des ensembles de solutions de compromis. En effet, parmi les différentes instances du POVI proposées par Renault, il en existe pour lesquelles chaque algorithme trouve une solution unique qui optimise tous les objectifs simultanément. C'est notamment le cas de l'instance 655_CH1_S51_J2_J3_J4 de l'ensemble X comme le montre le Tableau 8.10. Chaque ligne du tableau indique respectivement, pour chaque algorithme, le nombre de conflits obtenu en options prioritaires (HPO), le nombre de changement de couleur (COLOR) et le nombre de

conflits en options non prioritaires (LPO). En analysant ces résultats, on remarque que les quatre algorithmes obtiennent exactement la même solution. Il est important de préciser que cette solution est obtenue à chaque exécution de chaque algorithme.

GISMOO			PMS^{MO}			NSGAII			BEST_ROADEF		
HPO	**COLOR**	**LPO**	**HPO**	**COLOR**	**LPO**	**HPO**	**COLOR**	**LPO**	**HPO**	**COLOR**	**LPO**
0	30	0	0	30	0	0	30	0	0	30	0

Tableau 8.10 : Comparaison de la performance de GISMOO, du PMS^{MO}, du NSGAII et de BEST_ROADEF pour l'instance 655_CH1_S51_J2_J3_J4

La Figure 8.6 présente une comparaison visuelle entre GISMOO, NSGAII, PMS^{MO} et BEST_ROADEF pour une exécution type sur le problème 035_ch2_S22_J3 (un problème de l'ensemble B) qui est, selon Renault, un problème pour lequel les contraintes sur les options prioritaires sont « difficiles » à satisfaire. Comme pour l'exemple présenté à la Figure 8.4, il apparaît clairement que l'ensemble Pareto proposé par GISMOO domine globalement ceux proposés par NSGAII et PMS^{MO}. Cependant, on note que l'écart entre GISMOO et les deux autres algorithmes Pareto n'est pas aussi important que celui observé pour le problème 022_3_4. Cette situation peut s'expliquer par le fait que le problème 035_ch2_S22_J3 ne contient que 269 voitures à ordonnancer alors que le problème 022_3_4 en contenait lui 485. Toutefois, on note que ni NSGAII, ni PMS^{MO} ne parviennent à obtenir la solution extrême en HPO. Ainsi, les meilleures solutions trouvées par NSGAII et PMS^{MO}, si on considère l'objectif HPO comme prioritaire engendrent respectivement 448 et 438 conflits sur les options prioritaires, alors que la meilleure solution de GISMOO engendre seulement 385 conflits en HPO. On remarque d'ailleurs que GISMOO trouve

exactement les mêmes solutions extrêmes que BEST_ROADEF. On peut donc supposer que l'écart entre GISMOO et les deux autres algorithmes Pareto serait encore plus important pour un problème de plus grande taille avec des contraintes sur les options prioritaires qui sont difficiles à satisfaire. En plus de ces deux solutions extrêmes, GISMOO propose 70 autres solutions de compromis au décideur. Toutefois, contrairement à ce que l'on avait pu observer pour le problème 022_3_4, on remarque qu'il existe un écart important entre la solution extrême en HPO et le reste de l'ensemble de solutions proposé par GISMOO. On peut expliquer cet écart par la difficulté à satisfaire les contraintes sur les options prioritaires qui caractérisent ce problème.

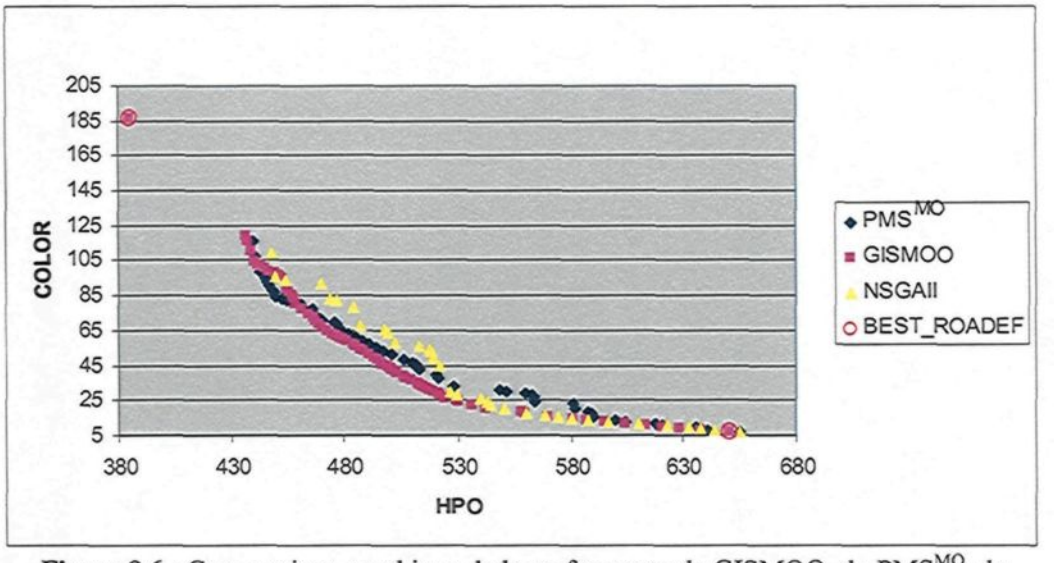


Figure 8.6 : Comparaison graphique de la performance de GISMOO, du PMS^{MO}, du NSGAII et de BEST_ROADEF pour l'instance 035_ch2_S22_J3

Finalement, la Figure 8.7 présente une comparaison visuelle entre GISMOO, NSGAII, PMS^{MO} et BEST_ROADEF pour une autre instance (048_ch2_ S49_J5) pour laquelle les contraintes sur les options prioritaires sont « difficiles » à satisfaire, mais qui contient cette

fois-ci 546 voitures à ordonnancer. Cette représentation graphique confirme les suppositions faites au graphique précédent. Ainsi, on observe que l'ensemble de solutions proposé par GISMOO domine nettement les deux ensembles de solutions du NSGAII et du PMS^{MO}. L'écart entre les courbes étant nettement plus important pour ce problème par rapport à celui observé pour le problème 035_ch2_S22_J3 qui contient deux fois moins de voitures que le problème 048_ch2_ S49_J5. Lorsqu'on compare maintenant GISMOO à BEST_ROADEF, on constate cette fois-ci que les deux solutions de BEST_ROADEF dominent les solutions de GISMOO. Toutefois, en observant plus attentivement les solutions des deux algorithmes, on remarque qu'elles ont exactement la même valeur sur l'objectif principal, mais que c'est sur l'objectif secondaire qu'elles se distinguent. Ainsi, on a 3 conflits en HPO et 93 changements de couleur pour la solution de BEST_ROADEF lorsque HPO est l'objectif principal contre 3 conflits en HPO et 135 changements de couleur pour la solution de GISMOO. Lorsque l'objectif COLOR est l'objectif principal, BEST_ROADEF obtient une solution avec 58 changements de couleur et 282 conflits en HPO contre 58 changements de couleur et 420 conflits en HPO pour la solution de GISMOO. On rappelle ici que le temps accordé à GISMOO pour générer ces ensembles de solutions correspond à une seule exécution de l'algorithme utilisé par BEST_ROADEF. En accordant à GISMOO le même temps que celui accordé à BEST_ROADEF pour générer ces deux solutions, c'est-à-dire le double du temps, l'écart entre les solutions des deux algorithmes se réduit considérablement comme le montre le graphique de la Figure 8.8. En examinant le graphique de la Figure 8.8, on observe que les solutions extrêmes de GISMOO sont pratiquement identiques à celles de BEST_ROADEF. Ainsi, on a

respectivement 3 conflits en HPO et 94 changements de couleur pour GISMOO contre 3 conflits et 93 changements de couleur pour BEST_ROADEF lorsque HPO est l'objectif prioritaire. Lorsque COLOR est l'objectif principal, on a 58 changements de couleur et 284 conflits en HPO pour GISMOO contre 58 changements de couleur et 283 conflits en HPO pour BEST_ROADEF. En plus de ces deux solutions extrêmes, GISMOO propose 7 solutions de compromis supplémentaires au décideur.

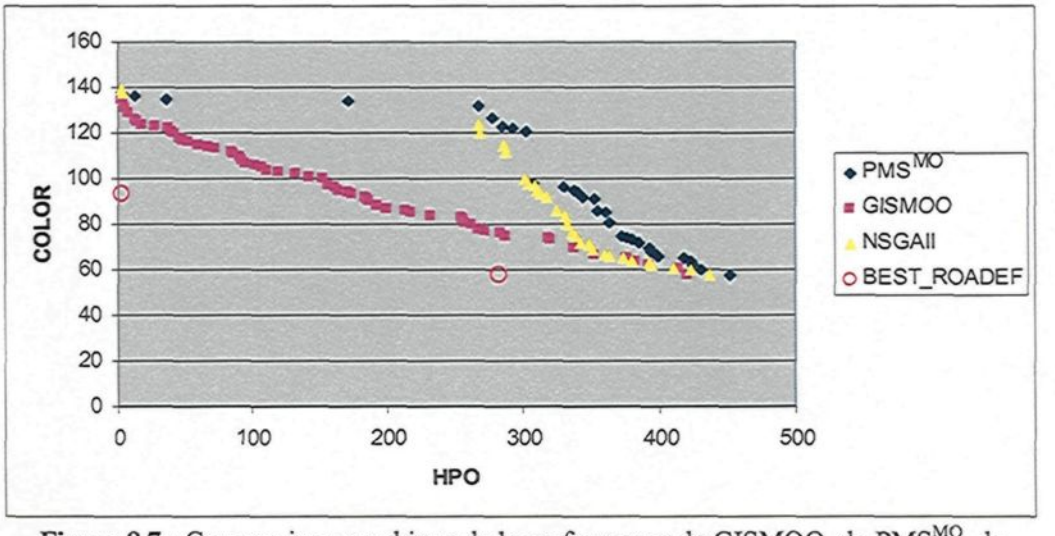


Figure 8.7 : Comparaison graphique de la performance de GISMOO, du PMS^{MO}, du NSGAII et de BEST_ROADEF pour l'instance 048_ch2_ S49_J5

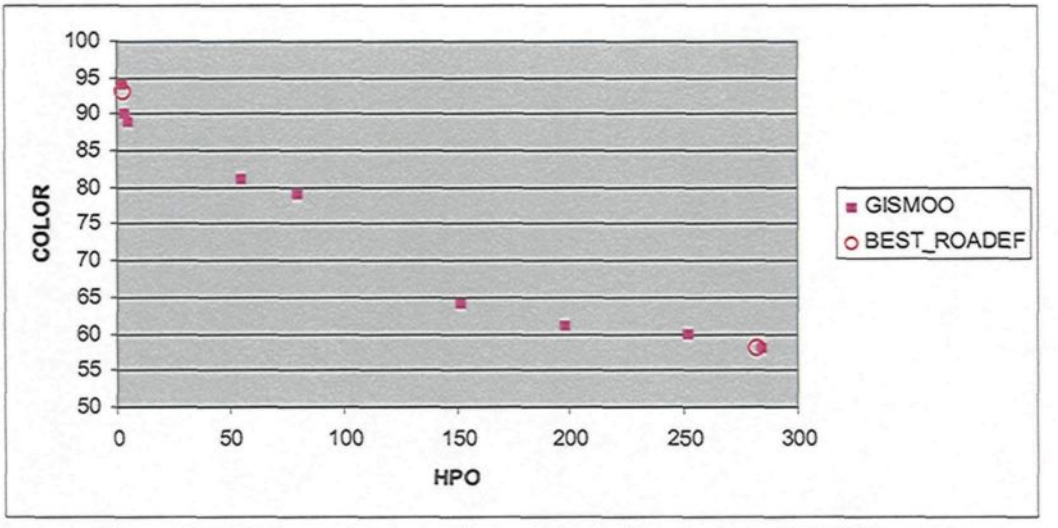


Figure 8.8 : Comparaison graphique de la performance de GISMOO et de BEST_ROADEF pour l'instance 048_ch2_ S49_J5 en accordant le même temps d'exécution aux deux algorithmes

8.4 Conclusion

Dans ce chapitre, nous avons appliqué GISMOO à la résolution du problème industriel d'ordonnancement de voitures. Une des principales originalités de l'utilisation de GISMOO pour résoudre le POVI réside dans le fait qu'en utilisant un algorithme Pareto, on traite les différents objectifs du problème dans leur intégralité sans leur attribuer d'ordre ou de poids. À notre connaissance, c'est la première fois que le problème est abordé d'un point de vue intégralement multi-objectifs. En effet, la revue de la littérature effectuée au Chapitre 2 a révélé que ce problème a été traité jusqu'à maintenant de manière uni-objectif ou en considérant les objectifs selon un ordre lexicographique. Les résultats obtenus par GISMOO sur les différentes instances du POVI ont permis de mettre en évidence de

nombreuses solutions de compromis qu'il n'était pas possible d'obtenir avec un traitement lexicographique des objectifs. Ainsi, au lieu de proposer au décideur une seule solution en fonction d'une hiérarchisation des objectifs choisie a priori, GISMOO propose plutôt un ensemble de solutions à partir duquel il peut choisir la solution la plus satisfaisante selon ses préférences.

Cette nouvelle façon d'aborder le problème ainsi que l'approche de résolution proposé dans ce chapitre a permis, d'une part, de mettre en évidence l'intérêt de résoudre le POVI en considérant les objectifs dans leur intégralité. D'autre part, le fait d'utiliser GISMOO pour résoudre le problème industriel contribue à réduire l'écart entre les approches de résolution proposées dans un contexte théorique et les situations pratiques. En effet, les résultats expérimentaux ont confirmé les bonnes performances de notre algorithme pour le traitement d'une problématique réelle. Ainsi, comme nous l'avions déjà observé au chapitre précédent, les performances de GISMOO sont supérieures à celles obtenues par deux autres algorithmes évolutionnaires Pareto représentatifs de l'état de l'art en optimisation multi-objectifs : le NSGAII et le PMS^{MO}.

CHAPITRE 9

CONCLUSION ET PERSPECTIVES

9.1 Conclusion

Dans cette thèse, nous avons abordé la résolution de problèmes d'ordonnancement industriel multi-objectifs à l'aide d'algorithmes évolutionnaires. Afin d'évaluer les différentes approches proposées, nous avons principalement réalisé nos expérimentations en utilisant la problématique de l'ordonnancement d'une chaîne d'assemblage de voitures. Trois raisons principales ont poussés à expérimenter les AE à l'aide de cette problématique. En premier lieu, le problème d'ordonnancement de voitures est un problème bien connu en optimisation combinatoire et a un intérêt dans le monde industriel comme le montre le Challenge ROADEF 2005 [Gagné *et al.* 2006; Benoit 2007; Briant *et al.* 2007; Estellon *et al.* 2007; Ribeiro *et al.* 2007; Solnon *et al.* 2007]. Toutefois, on note que très peu d'auteurs ont proposé des AG pour résoudre ce problème aussi bien dans sa version théorique que dans sa version industrielle. Cette situation laisse à penser que ce type d'algorithme est peut-être mal adapté aux spécificités du problème. En deuxième lieu, le POVI est, comme la plupart des problèmes industriels, un problème multi-objectifs. Cependant, un survol de la littérature [Chew *et al.* 1992; Lopez et Roubellat 2001; Nguyen et Cung 2005; Gagné *et al.* 2006; Benoit 2007; Briant *et al.* 2007; Estellon *et al.* 2007; Ribeiro *et al.* 2007; Solnon *et al.* 2007] révèle que ce problème a été traité jusqu'à maintenant de manière uni-objectif ou en considérant les objectifs selon un ordre lexicographique. Finalement, les problèmes d'ordonnancement multi-objectifs sont connus pour être particulièrement difficiles à résoudre, particulièrement dans des contextes industriels. Toutefois, plusieurs auteurs ont noté [Allahverdi *et al.* 1999; McKay et Wiers 1999; Portougal et Robb 2000] que la majorité des travaux en ordonnancement s'attardaient principalement à des contextes de

base. Malgré la complexité de ces contextes, ce sont souvent des versions simplifiées des situations pratiques. Il en résulte qu'il existe un écart entre les approches de résolution proposées pour les problèmes d'ordonnancement théoriques et les besoins réels pour aborder les situations pratiques [McKay et Wiers 1999; Portougal et Robb 2000; Gao 2004]. Les problèmes d'ordonnancement industriel multi-objectifs constituent donc une problématique de recherche actuelle. En proposant de nouvelles approches basées sur les AG permettant de supporter la prise de décision pour cette catégorie de problème, les travaux réalisés dans cette thèse ont permis d'enrichir les modèles existants en optimisation multi-objectifs à l'aide d'algorithmes évolutionnaires.

Plus spécifiquement, les travaux présentés dans cette thèse ont permis d'apporter plusieurs contributions de différents types par la réalisation des objectifs spécifiques fixés au début de ce travail de recherche. Le premier objectif consistait à *comprendre et isoler les mécanismes liés aux AG, afin de proposer une stratégie de résolution efficace pour la résolution d'un problème uni-objectif.* Dans notre contexte d'étude, le problème uni-objectif retenu est le POV. Pour atteindre cet objectif, nous avons présenté, dans un premier temps, les AE et, plus particulièrement, les AG au Chapitre 3. Cette présentation a permis d'identifier et de comprendre le rôle des différents mécanismes de cette méthode de résolution. Schématiquement, nous avons ainsi vu que l'opérateur de croisement a pour but d'explorer l'espace de solutions tandis que l'opérateur de mutation joue le rôle de bruit et tente d'empêcher une convergence trop hâtive. Il apparaît clairement que ces deux opérateurs influent grandement sur l'efficacité de l'algorithme. En particulier, l'opérateur de croisement oriente le processus de recherche d'un AG. Ainsi, l'utilisation d'une méthode

de croisement non adaptée au problème à résoudre peut fausser le processus de recherche de l'algorithme. En partant de ce constat, nous avons proposé, dans un deuxième temps, deux nouveaux opérateurs de croisement qui ont été présentés au Chapitre 4. La particularité de ces deux nouveaux opérateurs de croisement est qu'ils sont spécialisés pour résoudre la problématique du POV et qu'ils utilisent des informations liées au problème à résoudre pour effectuer le croisement. L'objectif derrière l'utilisation d'informations sur le problème dans le processus de croisement est d'orienter et d'intensifier le processus de recherche à l'aide de ces informations afin de le rendre plus performant. Les résultats obtenus en utilisant ces deux nouveaux opérateurs de croisement sur les instances du POV de la littérature ont mis en évidence leur efficacité. Ces résultats montrent l'importance d'adapter les différents opérateurs d'un AG à la problématique étudiée, et ce, même lorsque l'emploi d'opérateurs naturels est possible. En effet, les AG proposés jusqu'à présent se sont avérés très peu efficaces pour résoudre ce problème. Cette situation s'explique sans doute par le fait que les AG proposés dans la littérature utilisent tous des opérateurs de croisement classiques qui ne sont pas adaptés aux spécificités du POV.

Le deuxième objectif visait *à proposer des schémas de coopération originaux entre AG et différentes autres approches de résolution.* Cet objectif a été atteint au Chapitre 5 de cette thèse par la proposition de deux schémas d'hybridation entre AG et programmation linéaire en nombres entiers. Si l'hybridation de deux métaheuristiques est maintenant assez répandue dans la littérature, les méthodes hybridant une métaheuristique et une méthode exacte sont moins courantes. Les deux schémas de coopération introduits au Chapitre 5 proposent d'intégrer le programme linéaire en nombres entiers dans le processus de

croisement de l'AG présenté au Chapitre 4. Lors de la réalisation des différentes stratégies d'hybridation, nous avons montré l'apport de chacun des mécanismes de coopération proposés sur la qualité finale de la solution obtenue. Par la suite, au Chapitre 7, nous avons aussi présenté GISMOO, un nouvel algorithme hybridant systèmes immunitaires artificiels et AG. Nous reviendrons un peu plus loin sur cette contribution.

Pour *résoudre efficacement le POVI de manière lexicographique,* comme le mentionnait notre troisième objectif, nous avons adapté, au Chapitre 6, l'AG développé pour résoudre le POV afin de proposer un nouvel AG pour la résolution du problème industriel. Les résultats obtenus en utilisant les instances du Challenge ROADEF 2005 ont permis de montrer l'efficacité de notre approche. Ainsi, l'algorithme proposé permet d'obtenir des résultats compétitifs par rapport aux meilleurs résultats connus pour ce problème. À notre connaissance, c'est une des premières approches basées sur les AG qui permet de résoudre efficacement le POVI. Ces résultats montrent que les AG peuvent être des alternatives de résolution robustes et efficaces à condition d'incorporer des connaissances spécifiques au problème à résoudre lors de la conception de l'algorithme.

Finalement, le dernier objectif de cette thèse consistait à *développer un nouvel algorithme Pareto générique permettant de résoudre des problèmes d'optimisation combinatoire multi-objectifs.* Pour atteindre cet objectif, nous avons proposé GISMOO. Introduit au Chapitre 7, GISMOO est un algorithme Pareto hybride qui combine la métaphore immune avec un algorithme génétique afin d'identifier et mettre l'emphase sur les régions de l'espace de solution peu exploitées au cours des itérations de l'algorithme. À notre connaissance, c'est le premier algorithme Pareto à combiner AG et SIA dans cette

optique. Comme GISMOO a été développé comme un algorithme multi-objectifs générique, ses performances ont été présentées sur un benchmark classique en optimisation multi-objectifs : le problème de sac à dos multi-objectifs. Au Chapitre 8, GISMOO a été évalué en utilisant les différentes instances du POVI du Challenge ROADEF 2005. Dans les deux cas, les performances de notre approche ont été comparées avec celles de différents algorithmes représentatifs de l'état de l'art en optimisation multi-objectifs à l'aide d'algorithmes évolutionnaires. Il est important de préciser, qu'à notre connaissance les résultats présentés dans ce chapitre sont les premiers résultats intégralement multi-objectifs pour le POVI. En effet, le problème n'avait jusqu'à présent jamais été abordé d'un point de vue Pareto.

En résumé, ce travail de recherche a permis d'apporter des contributions que l'on peut regrouper en trois grandes catégories. La première catégorie de contributions concerne le contexte d'ordonnancement industriel utilisé dans cette thèse. En effet, peu d'effort a été fait jusqu'à maintenant pour résoudre le POV théorique ou industriel à l'aide d'algorithmes génétiques. En utilisant des AG, les travaux réalisés dans cette thèse permettent de mieux juger de l'apport de ce type d'algorithmes dans la résolution de problèmes d'ordonnancement industriel. Ces travaux mettent aussi en évidence l'importance d'adapter les composantes d'un AG aux spécificités du problème afin de le rendre efficace. Par ailleurs, une grande partie de ce travail traite le POV industriel en optimisant les différents objectifs de manière lexicographique puis en considérant les objectifs dans leur intégralité sans attribuer d'ordre ou de poids. À notre connaissance, c'est la première fois que le problème est abordé d'un point de vue intégralement multi-objectifs. Le traitement d'une

problématique réaliste permet de proposer des outils d'aide à la décision efficace en contexte industriel et transférables à d'autres problématiques.

La deuxième catégorie d'apports de cette thèse regroupe les contributions algorithmiques. Les contributions apportées dans cette catégorie se situent notamment au niveau de la définition d'opérateurs génétiques spécifiques, de l'hybridation d'algorithmes et du traitement multi-objectifs. Nous nous sommes concentrés, dans un premier temps, à l'élaboration de deux opérateurs génétiques qui exploitent des informations liées aux caractéristiques du problème à résoudre pour effectuer le croisement. Dans un deuxième temps, nous avons établi deux stratégies permettant l'utilisation d'une méthode exacte dans le processus de croisement. Applicables à d'autres fonctionnalités des AG, comme la mutation, ces deux schémas d'hybridation permettent d'augmenter les capacités d'intensification de la recherche des AG. Finalement, nous avons proposé un AG original permettant de résoudre, de manière efficace, le POVI en considérant les objectifs de manière lexicographique. Ainsi, les contributions algorithmiques se retrouvent, en partie, sous la forme d'algorithmes qui sont développés et spécifiquement adaptés pour le POV et le POVI

En plus des algorithmes dédiés à la problématique d'ordonnancement de voitures, nous avons aussi proposé GISMOO, un nouvel algorithme Pareto générique pour l'optimisation multi-objectifs. Contrairement à la plupart des AE et des SIA classiques, GISMOO ne nécessite pas la calibration de nombreux paramètres. La plupart des paramètres de GISMOO sont implicites et leur calibration est automatique. En fait, seulement trois paramètres sont explicitement définis. De plus, l'algorithme proposé ne nécessite pas

l'implémentation de procédures complexes pour réduire la taille de l'archive, pour effectuer des opérations de « clustering » ou encore pour mettre en place des techniques classiques de nichage. GISMOO est donc une approche de résolution très simple à implémenter et rivalise d'efficacité avec les meilleurs algorithmes multi-objectifs de la littérature. Ainsi, GISMOO s'est montré performant aussi bien pour un benchmark classique multi-objectifs (le MOKP) que pour un problème multi-objectifs industriel (le POVI).

Finalement, la dernière catégorie de contributions de cette thèse regroupe les aspects plus fondamentaux de nos travaux. Ainsi, les études réalisées nous ont permis de mettre en place une méthodologie de développement permettant d'adapter un AG à la résolution d'un problème d'optimisation donné. Les travaux réalisés lors de l'élaboration des schémas d'hybridation entre AG et méthodes exactes nous ont permis d'identifier les opérateurs génétiques pouvant influencer le processus d'intensification de la recherche par un AG. La revue de littérature sur les méthodes de résolution de PMO a mis en évidence l'importance des phases de diversification dans le succès d'un algorithme multi-objectifs. Cette caractéristique essentielle est mise en application dans GISMOO par le biais de la métaphore immune.

Cette nouvelle façon d'aborder le problème combinée aux approches proposées devrait permettre de rapprocher les modèles théoriques en ordonnancement et les situations rencontrées dans la pratique tout en proposant de nouvelles stratégies de résolution efficaces et transférables à d'autres secteurs industriels (aéronautique, navale, informatique, etc.). En effet, la plupart des problèmes rencontrés dans la pratique correspondent à des problématiques multi-objectifs. Le fait d'avoir utilisé, comme contexte d'illustration, le

POVI, un problème à trois objectifs, contribue à enrichir les modèles résolution existants en proposant des approches permettant de traiter un problème industriel réel et non un contexte de base comme c'est souvent le cas dans la littérature [Allahverdi *et al.* 1999; McKay et Wiers 1999; Portougal et Robb 2000]. De plus, même si une grande partie des approches proposées dans cette thèse ont été expérimentées en utilisant la problématique industrielle d'ordonnancement de voitures, leurs applications pourraient être étendues à la résolution d'autres problématiques industrielles. Nous sommes bien évidemment conscients que pour traiter convenablement diverses problématiques industrielles, des adaptations seront sûrement nécessaires afin de tenir compte des particularités de chaque problème à résoudre. Toutefois, une grande partie des mécanismes présentés dans cette thèse ont été définis de manière générique, de telle sorte qu'ils puissent être appliqués à d'autres problèmes. C'est en particulier le cas de GISMOO. En effet, nous avons déjà montré dans ce travail que notre approche multi-objectifs pouvait être facilement adaptée aussi bien à la résolution d'une problématique classique en optimisation multi-objectifs (le MOKP) qu'à la résolution d'un problème multi-objectifs pratique (le POVI) tout en obtenant des résultats compétitifs dans les deux cas. Selon Basseur [2004], pour répondre aux besoins des entreprises concernées par des problèmes multi-objectifs, une approche de résolution doit posséder trois qualités essentielles :

- le temps de calcul doit être raisonnable et adapté aux ressources disponibles pour les exécuter ;
- l'approche proposée doit être robuste de manière à fournir des solutions de bonne qualité malgré l'utilisation de paramètres qui font fluctuer les résultats; et

- les solutions proposées doivent offrir au décideur de bons compromis entre les différents objectifs.

Les expérimentations effectuées sur le MOKP et le POVI montrent que GISMOO possède ces trois qualités essentielles. En effet, notre approche a obtenu de manière constante de bons résultats sur toutes les instances tests pour ces deux types de problèmes. On note aussi que, malgré la difficulté de certaines instances, les temps de calcul de GISMOO se sont avérés égaux ou inférieurs à ceux des autres algorithmes en compétition pour toutes les instances.

D'un point de vue décisionnel, les nouvelles stratégies proposées dans cette thèse facilitent le processus de prise de décision en aidant le décideur à évaluer la situation, les diverses alternatives et leurs impacts éventuels sur le processus de production. Par exemple, on a montré au Chapitre 8 que l'utilisation de GISMOO pour résoudre le POVI a permis de proposer de nouvelles solutions alternatives par rapport aux solutions extrêmes obtenues dans la littérature jusqu'à maintenant. Ainsi, au lieu de proposer au décideur une seule solution en fonction d'une hiérarchisation des objectifs choisie a priori, on lui propose plutôt un ensemble de solutions à partir duquel il peut choisir la solution la plus satisfaisante selon ses préférences. Cette propriété est une qualité essentielle recherchée dans une méthode de résolution en milieu industriel et montre bien l'intérêt pratique de ce genre d'approche. On peut donc en conclure que les travaux réalisés dans cette thèse ont permis d'atteindre l'objectif général de ce travail de recherche à savoir : *enrichir les modèles existants et de proposer des approches de résolution efficaces basées sur des AG permettant de supporter la prise de décision pour des problèmes d'ordonnancement*

industriel multi-objectifs. De plus, l'utilisation de problématiques académiques (POV, MOKP) ainsi qu'une problématique industrielle (POVI) met bien en valeur le lien permettant de passer de la théorie à la pratique en optimisation multi-objectifs.

9.2 Perspectives de recherche

Plusieurs perspectives de recherche s'ouvrent à la suite de ce travail. Tout d'abord, la méthodologie utilisée dans cette thèse pour élaborer de nouveaux opérateurs de croisement pour le problème d'ordonnancement de voitures peut être généralisée et étendue aux autres opérateurs génétiques comme la mutation ou la sélection. Par la suite, il serait intéressant d'adapter et d'appliquer ces nouveaux opérateurs à la résolution d'autres types de problèmes d'optimisation combinatoire pour lesquels les opérateurs de croisement classiques se sont avérés inefficaces.

Au niveau des schémas d'hybridation, les expérimentations effectuées ont montré que si l'ajout d'une méthode exacte permettait des gains en terme de qualité de solutions par rapport à l'algorithme original, ces gains se faisaient au détriment du temps d'exécution. Il serait donc intéressant d'améliorer l'efficacité de ces stratégies en ce qui concerne les temps de résolution afin de les rendre encore plus compétitives. Ces améliorations peuvent, par exemple, passer par le développement de versions parallèles des schémas de coopération proposés ou encore l'utilisation de coopérations parallèles de type *modèle en îles*. Ce type de coopération semble être prometteur, surtout avec l'avènement de technologies permettant de développer des puissances de calculs importantes comme les grilles de calculs ou encore le calcul pair à pair. Outre ces améliorations, il serait intéressant

d'évaluer l'efficacité d'autres schémas d'hybridation ainsi qu'une éventuelle combinaison avec les schémas proposés dans cette thèse. Comme pour les nouveaux opérateurs de croisement, les schémas d'hybridation proposés dans ce document ne se limitent pas à leur application au POV. La généralisation de ces schémas à d'autres problématiques est donc une perspective envisageable. Finalement, une autre perspective ressortant de la partie hybridation de ces travaux est la définition d'un ensemble de méthodes coopérant entre elles ainsi que la mise en place d'un mécanisme adaptatif permettant de choisir quel type de coopération réaliser en fonction de l'évolution de l'algorithme.

D'un point de vue multi-objectifs, les résultats obtenus par GISMOO sur les deux problèmes testés dans cette thèse laissent entrevoir de nombreuses perspectives dans le domaine pour des travaux futurs. Tout d'abord, il serait intéressant d'étendre le champ d'application de GISMOO à la résolution d'autres problématiques multi-objectifs. En effet, GISMOO a été défini de manière générique, de telle sorte qu'il peut être facilement appliqué à la résolution d'autres problèmes multi-objectifs. Pour ces expérimentations, nous préconisons de privilégier deux types de problèmes :

- les problèmes à plus de deux objectifs. En effet, les expérimentations effectuées dans cette thèse sur le problème de sac à dos multi-objectifs ont montré que la qualité des mécanismes multi-objectifs, en particulier des mécanismes de diversification, était mieux évaluée dans les cas à plus de deux objectifs;
- les problèmes multi-objectifs réels. Ceci permettrait de confirmer l'apport de l'approche proposée en utilisant d'autres problématiques industrielles comportant des spécificités différentes du POVI.

De plus, nous pensons aussi que, en considérant les caractéristiques intrinsèques des SIA et des AE, l'utilisation de GISMOO pour des problèmes à variables continues ou des problèmes multi-objectifs en contexte dynamique peut être particulièrement utile. Un des atouts de GISMOO réside dans l'auto-calibration de certains de ces paramètres. Cette avenue de recherche est prometteuse, il serait donc intéressant d'approfondir les travaux dans cette direction afin d'incorporer des techniques d'intelligence artificielle dans GISMOO permettant une meilleure auto-calibration des paramètres ce qui ouvrirait la voie vers de nouveaux algorithmes plus adaptatifs.

Finalement, nous avons aussi constaté qu'en optimisation multi-objectifs, il était souvent difficile d'évaluer les performances des algorithmes exécutés sur un grand nombre d'instances d'un problème, et ce, malgré l'utilisation de plusieurs métriques. Il serait donc important d'approfondir les travaux existants afin de développer de meilleurs outils d'estimation de la qualité d'un ensemble Pareto. Ceci permettrait de perfectionner les standards afin de mieux évaluer les performances des nombreuses méthodes multi-objectifs développées actuellement.

BIBLIOGRAPHIE

[Allahverdi *et al.* 1999]
Allahverdi, A., J. N. D. Gupta et T. Adowaisan (1999). "A review of scheduling research involving setup considerations." Omega **27**: p .219-239.

[Banzhaf *et al.* 1998]
Banzhaf, W., P. Nordin, R. E. Keller et F. D. Francone (1998). Genetic Programming- An Introduction on Automatic Evolution Programs and its Applications, Morgan Kaufmann, dpunkt. Verlag.

[Barichard 2003]
Barichard, V. (2003). Approches hybrides pour les problèmes multiobjectifs. LERI. Thèse de doctorat, École doctorale d'Angers.

[Basseur 2004]
Basseur, M. (2004). Conception d'algorithmes coopératifs pour l'optimisation multi-objectifs : Application aux problèmes d'ordonnancement de type flow-shop. Thèse de doctorat. U.F.R. D'I.E.E.A., Université des sciences et technologies de Lille.

[Bautista *et al.* 1996]
Bautista, J., R. Companys et A. Corominas (1996). "Heuristics and exact algorithms for solving the Monden problem." European Journal of Operational Research: p101-103.

[Ben Hamida 2001]
Ben Hamida, S. (2001). Algorithmes Évolutionnaires: Prise en Compte des Contraintes et Application Réelle. Thèse de doctorat, CMAP École Polytechnique Paris 11.

[Benoit 2007]
Benoit, T. (2007). "Soft car sequencing with colors: Lower bounds and optimality proofs." European Journal of Operational Research: to appear doi:10.1016/j.ejor.2007.04.035.

[Berro 2001]
Berro, A. (2001). Optimisation multiobjectif et stratégies d'évolution en environnement dynamique. Thèse de doctorat, Toulouse, Université des Sciences Sociales Toulouse I: 170p.

[Blickle et Thiele 1996]
Blickle, T. et L. Thiele (1996). "A comparison of selection schemes used in evolutionary algorithms." Evolutionary Computation **4**(4): p. 361-394.

[Blum *et al.* 2005]
Blum, C., A. Roli et E. Alba (2005). An Introduction to Metaheuristics Techniques, In: Parallel Metaheuristics. E. Alba (Ed.), Wiley.

[Boivin 2005]
Boivin, S. (2005). Résolution d'un problème de satisfaction de contraintes pour l'ordonnancement d'une chaîne d'assemblage automobile, Mémoire de maîtrise, Université du Québec à Chicoutimi.

[Briant *et al.* 2007]
Briant, O., D. Naddef et G. Mounié (2007). "Greedy approach and multi-criteria simulated annealing for the car sequencing problem." European Journal of Operational Research: doi:10.1016/j.ejor.2007.04.052.

[Carter 2000]
Carter, J. H. (2000). "The immune system as a model for pattern recognition and classification." Journal of the American Medical Informatics Association **7**: 28–41.

[Carvalho et Freitas 2001]
Carvalho, D. R. et A. A. Freitas (2001). An immunological algorithme for discovering small-disjunct rules in data mining. Genetic and Evolutionary Computational Conference, San Francisco, California, USA, Morgan Kaufman.

[Chapman 1987]
Chapman, D. (1987). "Planning for Conjunctive Goals." Artificial Intelligence **32**: p.333-377.

[Charnes et Cooper 1961]
Charnes, A. et W. Cooper (1961). Management Models and Industrial Applications of Linear Programming,. New York, John Wiley.

[Chew *et al.* 1992]
Chew, T.-L., J.-M. David, A. Nguyen et Y. Tourbier (1992). Solving constraint satisfaction problems with simulated annealing; the car sequencing problem. Proceedings of the International Workshop on Expert Systems and their Application (publié pour Renault: Service Systèmes Experts), Avignon, France.

[Coello Coello et Cortés 2005]
Coello Coello, A. C. et N. C. Cortés (2005). " Solving Multiobjective Optimization Problems Using an Artificial Immune System." Genetic Programming and Evolvable Machines **6**(2): 163-190.

[Coello Coello et Pulido 2001]
Coello Coello, A. C. et G. T. Pulido (2001). Multiobjective Optimization using a Micro-genetic Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), p. 274-282, San Francisco, California.

[Coello Coello *et al.* 2002]
Coello Coello, C. A., D. Van Veldhuizen, A. et G. Lamont, B. (2002). Evolutionary Algorithms for solving multi-objective problems. Boston, Kluwer Academic.

[Collette et Siarry 2002]
Collette, Y. et P. Siarry (2002). Optimisation multiobjectif, Eyrolles, ISBN: 2-212-11168-1.

[Conover et Iman 1981]
Conover, W. J. et R. L. Iman (1981). "Rank Transformations as a Bridge between Parametric and Nonparametric Statistics." The American Statistician **35**: p. 124-129.

[Cordeau *et al.* 2007]
Cordeau, J. F., G. Laporte et F. Pasin (2007). "Iterated Tabu Search for the Car Sequencing Problem." European Journal of Operational Research: In press.

[Corne 2001]
Corne, D. W. (2001). PESA II : Region-based Selection in Evolutionary Muliobjective Optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), San Francisco, California.

[Cui *et al.* 2001]
Cui, X., M. Li et T. Fang (2001). Study of population diversity of multiobjective evolutionary algorithm based on immune and entropy principles. Congress on Evolutionary Computation (CEC'2001), New Jersey.

[Dasgupta et Attoh-Okine 1997]
Dasgupta, D. et N. Attoh-Okine (1997). Immune-based systems : A survey. IEEE International Conference on Systems, Man and Cybernetics, Orlando, IEEE Press.

[Davenport et Tsang 1999]
Davenport, A. et E. Tsang (1999). Solving constraint satisfaction sequencing problems by iterative repair. Proceeding of the 1th international conference on the pratical applications of constraint technologies and logic programming (PACLP). Practical Applications Company, London England. p. 345-357

[De Castro et Timmis 2002]
De Castro, L. N. et J. Timmis (2002). Artificial Immune Systems: A New Computational Intelligence Approach. London.

[De Castro et Von Zuben 1999]
De Castro, L. N. et F. J. Von Zuben (1999). Artificial Immune Systems : Part I : Basic Theory and Applications. Technical Report TR-DCA 01/99, Department of Computer Engineering and Industrial Automation, Department of Computer Engineering and

Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil.

[De Castro et Von Zuben 2000]
De Castro, L. N. et F. J. Von Zuben (2000). Artificial Immune Systems : Part II - A Survey of Applications. Technical Report DCA-RT 02/00, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil.

[De Castro et Von Zuben 2000]
De Castro, L. N. et F. J. Von Zuben (2000). An evolutionary immune network for data clustering. IEEE SBRN'00 (Brazilian Symposium on Artificial Neural Networks).

[de Castro et Von Zuben 2002]
de Castro, L. N. et F. J. Von Zuben (2002). "Learning and Optimization Using the Clonal Selection Principle." IEEE Transactions on Evolutionary Computation **6**(3): 239–251.

[De Jong 1975]
De Jong, K. A. (1975). An Analysis of the Behaviour of a Class of Genetic Adaptive Systems, PhD thesis, University of Michigan.

[Deb 1999]
Deb, K. (1999). "Multi-objective Genetic Algorithms : Problem Difficulties and Construction of Test Problems." Evolutionary Computation **7**(3): p. 205-230.

[Deb 2000]
Deb, K. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multiobjective Optimization : NSGA II. Parallel problem Solving form Nature – PPSN VI, M. Schoenauer et al. (Eds), Springer Lecture Notes in Computer Science, p. 849-858.

[Deb 2001]
Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. New York, N.Y., J. Wiley and Sons.

[Delaval 1997]
Delaval, M. (1997). Séquencement des lignes d'assemblage à modèles mélangés, Université des Sciences et Technologies de Lille.

[Dimistrescu et Stützle 2003]
Dimistrescu, I. et T. Stützle (2003). "Combinations of local search and exact algorithms." Lecture Notes in Computer Science **2611**: p 211-223.

[Dincbas *et al.* 1988]
Dincbas, M., H. Simonis et P. van Hentenryck (1988). Solving the car sequencing problem In: *Proceedings of the European Conference on Artificial Intelligence* (ECAI-88), London, Pitmann Publishing, Kodratoff Y (ed), p. 90–295.

[Dorigo 1992]
Dorigo, M. (1992). Optimization, learning and natural algorithms. Ph.D.Thesis, Italy, Politecnico di Milano.

[Dréo et Siarry 2003]
Dréo, J. et P. Siarry (2003). Métaheuristiques pour l'optimisation difficile. Paris, Eyrolles.

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

[Edgeworth 1881]
Edgeworth, F. Y. (1881). Mathematical Psychics: An essay on the application of mathematics to the moral sciences.

[Ehrgott 2000]
Ehrgott, M. (2000). "Multicriteria optimization." Lecture Notes in Economics and Mathematical Systems **491**: Springer-Verlag.

[Estellon *et al.* 2005]
Estellon, B., F. Gardi et K. Nouioua (2005). Ordonnancement de véhicules: une approche par recherche locale à grand voisinage. Actes of JFPC, Lens, France.

[Estellon *et al.* 2007]
Estellon, B., F. Gardi et K. Nouioua (2007). "Large neighborhood improvements for solving car-sequencing problems." RAIRO - Operations Research: **40**(4), p. 355-379.

[Estellon *et al.* 2007]
Estellon, B., F. Gardi et K. Nouioua (2007). "Two local search approaches for solving real-life car sequencing problem." European Journal of Operational Research(doi:10.1016/j.ejor.2007.04.043).

[Fogel *et al.* 1966]
Fogel, L. J., A. J. Owens et M. J. Walsh (1966). "Artificial Intelligence through Simulated Evolution." Wiley, New York.

[Fonseca et Fleming 1993]
Fonseca , C. M. et P. J. Fleming (1993). Genetic Algorithm for Multiobjective Optimization : Formulation, Discussion and Generalization. In Proceedings of the Fifth

International Conference on Genetic Algorithms, Morgan Kauffman Publishers, San Mateo, California.

[Fonseca et Fleming 1995]
Fonseca , C. M. et P. J. Fleming (1995). "An overview of evolutionary algorithms in multiobjective optimization." Evolutionary Computation **3**(1): p. 1-16.

[Forrest et Perelson 1991]
Forrest, S. et A. S. Perelson (1991). Genetic algorithms and the immune system. Lecture Notes in Computer Science, Vol 496: p.319-325, Berlin, H.-P. Schwefel and R. Männer.

[Fourman 1985]
Fourman (1985). Compaction of Symbolic Layout using Genetic Algorithms. In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithm, p.141-153.

[Francisci 2002]
Francisci, D. (2002). Algorithmes évolutionnaires et optimisation multi-objectifs en data mining, Laboratoire informatique, signaux et systèmes de Sophia Antipolis UMR 6070.

[Gagné *et al.* 2008]
Gagné, C., M. Gravel, S. Morin et W. L. Price (2008). "Impact of the pheromone trail on the performance of ACO algorithms for solving the car-sequencing problem." Journal of the Operational Research Society, Vol. 59, p. 1077-1090.

[Gagné *et al.* 2006]
Gagné, C., M. Gravel et W. L. Price (2006). "Solving real car sequencing problems with ant colony optimization." European Journal of Operational Research **174**(3): 1427-1448.

[Gandibleux *et al.* 1996]
Gandibleux, X., N. Mezdaoui et A. Fréville (1996). A tabu search procedure to solve multiobjective combinatorial optimization problems. Proceeding volume of MOPGP '96, Springer-Verlag.

[Gao 2004]
Gao, Z. (2004). Theory vs. Pratice: The Challenges From Industry. in the 2004 American Control Conference, Boston, Massachusset , June 30-July 2.

[Garey et Johnson 1979]
Garey, M. S. et D. S. Johnson (1979). Computer and Intractability : A Guide to the Theory of NP-Completeness. New York, W.H. Freeman and Co.

[Gavranovic 2007]
Gavranovic, H. (2007). "Local search and suffix tree for car-sequencing problem with colors." European Journal of Operational Research **In Press, Corrected Proof**.

[Gent et Walsh 1999]
Gent, I. P. et T. Walsh (1999). CSPLib: a benchmark library for constraints. in proceeding of the fifth International Conference on Principles and Practice of Constraint Programming, Alexandria, Virginia.

[Glover 1986]
Glover, F. (1986). "Future paths for integer programming and links to artificial intelligence." Computers and Operations Research **5**: p. 533-549.

[Goldberg 1989]
Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning,. Massachusetts,, Addison-Wesley,Reading.

[Goldberg et Lingle 1985]
Goldberg, D. E. et R. Lingle (1985). Alleles, Loci and the Traveling Salesman Problem. in proceeding of the first Int. Conf. on Genetic Algorithms (ICGA'85), J. J. Grefenstette Editor, Carnegie-Mellon University, Pittsburgh, PA, p. 154-159.

[Goldberg et Richardson 1987]
Goldberg, D. E. et J. J. Richardson (1987). Genetic Algorithm with Sharing for Multimodal Function Optimization. Genetic Algorithms and their Applications : Proceedings of the Second ICGA, Lawrence Erlbaum Associates,Hillsdales, p. 41-49.

[Gottlieb *et al.* 2003]
Gottlieb, J., M. Puchta et C. Solnon (2003). "A study of greedy, local search and ant colony optimization approches for car sequencing problems." Applications of Evolutionary Computing, Lecture Notes in Computers Science: G.R. Raidl *et al.* (Eds). Springer-Verlag Heidelberg. p.246-257.

[Gravel *et al.* 2008]
Gravel, M., C. Gagné, S. Noël et A. Zinflou (2008). "Combining metaheuristics and ILP for solving the car-sequencing problem." Revue d'Intelligence Artificielle **22**(2): p.209-235.

[Gravel *et al.* 2005]
Gravel, M., C. Gagné et 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**: p.1287-1295.

[Grefenstette 1986]
Grefenstette, J. J. (1986). "Optimization of Control parameters for genetic algorithms." IEEE Transactions on Systems, Man, and Cybernetics **SMC-16**(1): p.122-128.

[Hajela et Lin 1992]
Hajela, P. et C.-Y. Lin (1992). "Genetic search strategies in multicriterion optimal design." Structural Optimization **4**: p.99-107.

[Holland 1962]
Holland, J. (1962). "Outline for a logical theory of adaptive systems." Journal of the association of computing machinery **3**.

[Holland 1975]
Holland, J. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor.

[Hopfield et Tank 1985]
Hopfield, J. J. et D. W. Tank (1985). "Neural computation of decisions in optimization problems." Biological Cybernetics **52**: p.141-152.

[Horn *et al.* 1994]
Horn, J., N. Nafpliotis et D. E. Goldberg (1994). A niched pareto genetic algorithm for multiobjective optimization. In Proceeding of the first IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation, Volume 1, Piscataway, NJ, p. 82-87.

[Hwang et Masud 1980]
Hwang, C.-L. et A. S. M. Masud (1980). Multiple Objectives Decision Making- Methods and Application, Springer- Verlag, Berlin.

[Ignizio 1981]
Ignizio, J. P. (1981). "The Determination of a Subset of Efficient Solutions via Goal Programming." Computing and Operations Research 3: p. 9-16.

[Ishibuchi et Murata 1996]
Ishibuchi, H. et T. Murata (1996). Multi-Objective Genetic Local Search Algorithm., In Proceedings of the 1996 International Conference on Evolutionary Computation, p. 119-124, Nagoya, Japan.

[Jaszkiewicz 2000]
Jaszkiewicz, A. (2000). On the performance of multiple objective genetic local search on the 0/1 knapsack problem: a comparative experiment. Technical Report RA-002, Research report Institute of Computing Science,Poznan University of Technology.

[Jaszkiewicz *et al.* 2004]
Jaszkiewicz, A., M. Kubiak et P. Kominek (2004). The Application of the genetic local search algorithm to Car Sequencing Problem. 7th National Conference on Evolutionary Algorithms and Global Optimization, Kazimierz Dolny, Poland.

[Joslin et Clements 1998]
Joslin, D. et D. P. Clements (1998). Squeaky wheel optimization. in Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI.

[Jussien et Laburthe 2004]
Jussien, N. et F. Laburthe (2004). "Hybrid Optimization Techniques." Annals of Operations Research **130**(1-4), p.17-18.

[Kirkpatrick *et al.* 1983]
Kirkpatrick, S., J. C. D. Gelatt et M. P. Vecchi (1983). "Optimization by simulated annealing." Science **220**: p. 671-680.

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

[Knowles et Corne 1999]
Knowles, J. D. et D. W. Corne (1999). The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Multiobjective Optimisation. Congress on Evolutionary Computation, Washington.

[Knowles et Corne 2000]
Knowles, J. D. et D. W. Corne (2000). M-PAES : A Memetic Algorithm for Multiobjective Optimization. In Proceedings of the 2000 Congress on Evolutionary Computation.

[Knowles et Corne 2000]
Knowles, J. D. et D. W. Corne (2000). The Pareto-Envelope based Selection Algorithm for Multiobjective Optimization. In Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI), p. 839-848, Berlin.

[Koza 1992]
Koza, J. R. (1992). Genetic Programming : On the programming of computers by Means of Natural Selection. MIT Press. Cambridge, MA, USA.

[Lee *et al.* 2000]
Lee, W., R. A. Nimbalkar, K. K. Yee, S. B. Patil, P. H. Desai, T. T. Tran et S. J. Stolfo (2000). "A data mining and CIDF based approach for detecting novel and distributed intrusions." Lecture Notes in Computer Science **1907**: 49–65.

[Leu *et al.* 1996]
Leu, Y., L. A. Matheson et L. P. Rees (1996). "Sequencing mixed model assembly lines with genetic algorithms." Computers and Industrial Engineering: p.1027-1036.

[Lopez et Roubellat 2001]
Lopez, P. et F. Roubellat (2001). Ordonnancement de la production. Paris, Hermès science publications.

[Macaskill 1973]
Macaskill, J. L. (1973). "Computer simulation for mixed production lines." Management Science: p.341-348.

[Mahfoud 1995]
Mahfoud, S. W. (1995). Niching Methods for Genetic Algorithms, IlliGAL TR. n° 95001, University of Illinois at Urbana-Champaign, Urbana.

[McKay et Wiers 1999]
McKay, K. N. et V. C. S. Wiers (1999). "Unifying the theory and practice of production scheduling." Journal of Manufacturing Systems **18**(4): p. 241-255.

[Michalewicz 1994]
Michalewicz, Z. (1994). Genetic algorithms + data structures = evolution programs. Berlin, Springer.

[Monden 1998]
Monden, Y. (1998). Toyota Production System: An Integrated Approach to Just-In-Time, Engineering & Management Press.

[Morin *et al.* 2006]
Morin, S., C. Gagné, M. Gravel et W. L. Price (2006). Trace de phéromone spécialisée dans un algorithme de fourmi pour le problème de car sequencing. Actes de MOSIM 2006, M. Gourgand et F. Riane Eds.

[Nasraoui *et al.* 2002]
Nasraoui, O., F. Gonzalez et D. Dasgupta (2002). The fuzzy artificial immune system: Motivations, basic concepts, and application to clustering and web profiling. IEEE International Conference on Fuzzy Systems.

[Nedzela 1990]
Nedzela, M. (1990). Introduction a la science de la gestion, Sillery : Presses de l'Université du Québec.

[Neveu *et al.* 2004]
Neveu, B., G. Trombettoni et F. Glover (2004). Id Walk: A candidate list strategy with a simple diversification device. Proceedings of CP'2004, Springer.

[Nguyen et Cung 2005]
Nguyen, A. et V.-d. Cung (2005). Le problème du Car Sequencing Renault et le challenge ROADEF' 2005, Actes JFPC 2005.

[Obayashi *et al.* 1998]
Obayashi, S., S. Takahashi et Y. Takeguchi (1998). Niching and elitist models for mogas. Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V), Berlin, Germany, p. 241-249, Springer.

[Osman et Laporte 1996]
Osman, I. H. et G. Laporte (1996). "Metaheuristics: A bibliography." Annals of Operations Research(63): p.513-623.

[Othomani 1998]
Othomani, I. (1998). Optimisation multicritère : Fondements et Concepts, PhD thesis, Université de Grenoble.

[Overill 2002]
Overill, R. (2002). Computational immunology for fraud detection. Fourth Coordination Workshop and RETRIEVE end-of-project event.

[Parello 1988]
Parello, B. D. (1988). "Car Wars : the (almost) birth of an expert system." AI expert: p.60-64.

[Parello *et al.* 1986]
Parello, B. D., W. C. Kabat et L. Wos (1986). "Job-shop scheduling using automated reasoning: a case study of the car sequencing problem." Journal of Automated Reasoning **2**: p.1-42.

[Pareto 1896]
Pareto, V. (1896). Cours d'économie politique. Lausanne, F. Rouge.

[Parks et Miller 1998]
Parks, G. T. et Miller (1998). Selective breeding in multiobjective genetic algorithm. Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V), Berlin, Germany, p.250-259, Springer.

[Perron et Shaw 2004]
Perron, L. et P. Shaw (2004). Combining forces to solve the car sequencing problem. Proceedings of CP-AI-OR'2004, Lecture Notes in Computer Science 3011, Springer Berlin / Heidelberg, p. 225-239.

[Portougal et Robb 2000]
Portougal, V. et D. J. Robb (2000). "Production Scheduling Theory: Just where is it applicable?" Interface **30**(6): p64-76.

[Potvin 1996]
Potvin, J.-Y. (1996). "Genetic Algorithms for the Traveling Salesman Problem." Annals of Operations Research **63**: p. 339-370.

[Prandtstetter et Raidl 2007]
Prandtstetter, M. et G. R. Raidl (2007). "An Integer Linear Programming Approach and a Hybrid variable Neighborhood search for the Car Sequencing Problem." European Journal of Operational Research: to appear doi:10.1016/j.ejor.2007.04.044.

[Puchinger et Raidl 2005]
Puchinger, J. et G. R. Raidl (2005). Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. Institute of Computer Graphics and Algorithms, Vienna University of Technology,Vienna, Austria.

[Rechenberg 1965]
Rechenberg, I. (1965). Cybernetic Solution Path of an Experimental Problem. R. A. E. L. Translation.

[Rechenberg 1973]
Rechenberg, I. (1973). Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Stuttgart.

[Régin et Puget 1997]
Régin, J.-C. et J.-F. Puget (1997). A filtering Algorithm for global sequencing Constraints. Proceeding CP, Linz, Austria, Smolka G (ed), Springer, p. 32–46

[Ribeiro *et al.* 2007]
Ribeiro, C. C., D. Aloise, T. F. Noronha, C. Rocha et S. Urrutia (2007). "An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem." European Journal of Operational Research **In Press, Corrected Proof**.

[Ribiero *et al.* 2007]
Ribiero, C. C., D. Aloise, T. F. Noronha, C. Rocha et S. Urrutia (2007). "A hybrid heuristic for a multi-objective real-life car sequencing." European Journal of Operational Research(doi:10.1016/j.ejor.2007.04.034).

[Rudolph 1998]
Rudolph, G. (1998). On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In IEEE International Conference on Evolutionary Computation (ICEC'98), Piscataway, NJ, p. 511-516.

[Schaffer 1985]
Schaffer, D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithm. In genetic Algorithm and their Applications: Proceedings of the First International Conference on Genetic Algorithm, p.93-100.

[Schewfel 1981]
Schewfel, H.-P. (1981). Numerical Optimization of computer Models, Wiley, 398 p, ISBN: 0471099880.

[Schott 1995]
Schott (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Department of Aeronautics and Astronautics Massachusetts Institute of Technology.

[Serafini 1994]
Serafini, P. (1994). Simulated annealing for multiple objective optimization problems. In : Tzeng G.H., Wang H.F., Wen V.P., Yu P.L. (eds). Multiple Criteria Decision Making. Expand and Enrich the Domains of Thinking and Application, Springer Verlag: p 283-292.

[Siala 2005]
Siala, W. (2005). Un algorithme génétique pour le problème d'ordonnancement de voitures Mémoire de maîtrise, HEC, Montréal.

[Silverman 1986]
Silverman, B., W. (1986). Density estimation for statistics and data analysis, Chapman and Hall, London.

[Smith 1997]
Smith, B. (1997). Succeed-first or Fail-first: A Case Study in Variable and Value Ordering Heuristics. Proceedings of PACT'97, third international Conference on the Pratical Applications of Constraint Technology. London, UK, p. 321-330

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

[Smith *et al.* 1992]
Smith, R. E., S. Forrest et A. S. Perelson (1992). Searching for diverse, cooperative populations with genetic algorithms, Technical Report TCGA No. 92002, University of Alabama, Tuscaloosa.

[Smith *et al.* 1993]
Smith, R. E., S. Forrest et A. S. Perelson (1993). "Population diversity in an immune system model: Implications for genetic search." Foundation of Genetic Algorithms **2**: 153-165.

[Solnon 2000]
Solnon, C. (2000). Solving Permutation Constraint Satisfaction Problems with artificial Ants. Proceedings of the 14th European Conference on Artificial Intelligence, Amsterdam, Netherlands.

[Solnon 2006]
Solnon, C. (2006). Des fourmis pour le problème d'ordonnancement de voitures. Actes JFPC 2006.

[Solnon *et al.* 2007]
Solnon, C., V.-d. Cung et C. Artigues (2007). "The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem." European Journal of Operational Research (to appear).

[Soulard 2002]
Soulard, N. (2002). La méthode kanban dans l'industrie automobile actuelle, Conservatoire National des Arts et Métiers Centre Régional associé de Versailles.

[Srivinas et Deb 1994]
Srivinas, N. et K. Deb (1994). "Multiobjective Optimization using Non-dominated Sorting in Genetic Algorithms." Evolutionary Computation **2**(3): p. 221-248.

[Sumichrast *et al.* 1992]
Sumichrast, R. T., R. S. Russell et B. W. Taylor (1992). "A comparative analysis of sequencing procedures for mixed-model assembly lines in a just-in-time production system." International Journal of Production Research: p.199-214.

[Talbi 1999]
Talbi, E.-G. (1999). Métaheuristiques pour l'optimisation combinatoire multi-objectif: état de l'art, Rapport CNET (France Telecom).

[Talbi 2002]
Talbi, E.-G. (2002). " A Taxonomy of Hybrid Metaheuristics." Journal of Heuristics (8): p. 541-564.

[Tavakkoli-Moghaddam *et al.* 2007]
Tavakkoli-Moghaddam, R., A. R. Rahimi-Vahed et A. H. Mirzaei (2007). "Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm." The International Journal of Advanced Manufacturing Technology **36**: p.969-981.

[Taylor 1911]
Taylor, F. W. (1911). The Principles of Scientific Management. New York.

[Teller et Veloso 1996]
Teller, A. et M. Veloso (1996). PADO: a new learning architecture for object recognition. Symbolic Visual Learnig, Oxford University Press.

[Terada *et al.* 2006]
Terada, J., H. Vo et D. Joslin (2006). Combining Genetic Algorithms with Squeaky-Wheel Optimization. Genetic and Evolutionary Computation COnference (GECCO) 2006, Seattle Washington, p. 1329-1336.

[Thomopoulos 1967]
Thomopoulos, N. T. (1967). "Line balancing-sequencing for mixed model assembly." Management Science: p.59-75.

[Timmis et Knight 2001]
Timmis, J. et T. Knight (2001). Artificial immune systems: Using the immune system as inspiration for data mining. Data mining: a heuristic approach, Hussein A.Abbass, Ruhul A. Sarker and Charles S. Newton editors.

[Twycross et Cayzer 2002]
Twycross, J. et S. Cayzer (2002). An immune-based approach to document classification, Technical Report HPL-2002-292, HP Laboratories,USA.

[Ulungu *et al.* 1999]
Ulungu, E. L., J. Teghem, P. Fortemps et D. Tuyttens (1999). "MOSA method: a tool for solving multiobjective combinatorial optimization problems." Journal of Multi-Criteria Decision Analysis **8**: 221-236.

[Valenzuela-Rendon et Uresti-Charre 1997]
Valenzuela-Rendon, M. et E. Uresti-Charre (1997). A Non-Generational Genetic Algorithm for Multiobjective Optimization. Proceedings of the Seventh International Conference on Genetic Algorithms, p. 658-665.

[Van Veldhuizen 1999]
Van Veldhuizen, D. A. (1999). Multiobjective, Evolutionary Algorithms: Classification, Analyses and New Innovation, Air Force Institute of Technology, United States.

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

[Xiaobo et Ohno 1997]
Xiaobo, Z. et K. Ohno (1997). "Algorithms for sequencing mixed models on an assembly line in a JIT production system." Computers and Industrial Engineering: p. 47-56.

[Yang et Liao 1999]
Yang, W.-H. et C.-J. Liao (1999). "Survey of scheduling research involving setup times." International Journal of Systems Science **30**(2): p. 143-155.

[Yoo et Hajela 1999]
Yoo, J. et P. Hajela (1999). "Immune Network Simulations in Multicriterion Design." Structural Optimization **18**: 85-94.

[Zinflou *et al.* 2008]
Zinflou, A., C. Gagné, M. Gravel et W. L. Price (2008). "Pareto memetic algorithm for multiple objective optimization with an industrial application." Journal of Heuristics, **14**: p.313-333.

[Zitzler 1999]
Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: methods and applications. Ph.D. thesis, Zurich, Swiss Federal Institute of Technology..

[Zitzler 2001]
Zitzler, E. (2001). Evolutionary Algorithms for Multiobjective Optimization. EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems.

[Zitzler *et al.* 1999]
Zitzler, E., K. Deb et L. Thiele (1999). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,, Tech. Report n° 70, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology (ETH), Zurich, 1999.

[Zitzler *et al.* 2001]
Zitzler, E., M. Laumanns et L. Thiele (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Technical Report 103, Computer Engineering and Communication Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

[Zitzler et Thiele 1998]
Zitzler, E. et L. Thiele (1998). An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach, TIK-Report n° 43.

[Zitzler et Thiele 1999]
Zitzler, E. et L. Thiele (1999). "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach." IEEE Transactions on Evolutionary Computation **3**: p.257-271.

ANNEXE

RÉSULTATS NUMÉRIQUES DE LA MÉTRIQUE *C* POUR LES TROIS ENSEMBLES D'INSTANCES DU POVI

Dans cette annexe, nous présentons à l'aide de tableaux les résultats détaillés obtenus par GISSMOO, PMS^{MO} et NSGAII selon la métrique ***C*** pour les trois ensembles d'instances du POVI.

GISSMOO

PMS^{MO}

NSGAII

GISSMOO

C(GISSMOO, PMS^{MO})			
Instance	Max	Min	Moyenne
022_3_4	0,882353	0,027027	0,47
024_38_3	0,833333	0,0625	0,34
024_38_5	0,913043	0,025641	0,26
025_38_1	0,593583	0,037037	0,23
039_38_4_ch1	1	0,166667	0,41
048_39_1	0,759036	0,03125	0,33
064_38_2_ch1	0,594059	0	0,59
064_38_2_ch2	0,0263158	0	0,75

C(GISSMOO,NSGAII)			
Instance	Max	Min	Moyenne
022_3_4	0,8	0,1	0,40
024_38_3	0,967213	0,025641	0,42
024_38_5	0,710526	0,108108	0,30
025_38_1	0,959184	0,0294118	0,46
039_38_4_ch1	0,75	0,1	0,36
048_39_1	1	0,0833333	0,48
064_38_2_ch1	0,0447761	0	0,70
064_38_2_ch2	0,105	0,000	0,786

PMS^{MO}

C(PMS^{MO}, GISSMOO)			
Instance	Max	Min	Moyenne
022_3_4	0,275591	0	0,07
024_38_3	0,126437	0,0045045	0,05
024_38_5	0,0960187	0,00442478	0,04
025_38_1	0,396635	0,00222222	0,16
039_38_4_ch1	0,137931	0	0,04
048_39_1	0,192737	0	0,05
064_38_2_ch1	0,0100806	0	0,17
064_38_2_ch2	0,00E+00	0,000	1,72E-01

C(PMSMO,NSGAII)			
Instance	Max	Min	Moyenne
022_3_4	1	0	0,41
024_38_3	0,606557	0	0,18
024_38_5	0,807692	0,0217391	0,30
025_38_1	0,8125	0	0,31
039_38_4_ch1	0,75	0	0,22
048_39_1	0,95122	0	0,28
064_38_2_ch1	0,000	0,000	0,791
064_38_2_ch2	1	0	0,75

NSGAII

C(NSGAII, GISSMOO)			
Instance	Max	Min	Moyenne
022_3_4	0,097561	0	0,04
024_38_3	0,0780488	0	0,03
024_38_5	0,0814607	0,00595238	0,02
025_38_1	0,204444	0,00263158	0,07
039_38_4_ch1	0,206897	0	0,05
048_39_1	0,110831	0	0,03
064_38_2_ch1	0,0179949	0	0,21
064_38_2_ch2	5,99E-03	0,00E+00	0,09

C(NSGAII,PMSMO)			
Instance	Max	Min	Moyenne
022_3_4	0,4	0	0,12
024_38_3	0,395349	0	0,14
024_38_5	0,382353	0	0,12
025_38_1	0,369231	0,00896861	0,12
039_38_4_ch1	0,75	0	0,42
048_39_1	0,433735	0	0,14
064_38_2_ch1	0,000	0,000	0,525
064_38_2_ch2	0	0	0,43

Figure A.1 : Résultats numérique selon la métrique ***C*** de GISSMOO, PMS^{MO} et NSGAII pour les instances de l'ensemble A

GISSMOO

PMS^MO

NSGAII

C(GISSMOO, PMS^MO)

Instance	Max	Min	Moyenne
022_S22_J1	0,892857	0,229358	0,50
023_S23_J3	0,7	0,377358	0,54
024_V2_S22_J1	0,422449	0,233333	0,34
025_S22_J3	0,391304	0,241379	0,30
028_ch1_S22_J2	1	0,147059	0,54
028_ch2_S23_J3	1	0,5	0,90
029_S21_J6	0,731884	0,132353	0,46
035_ch1_S22_J3	0,925926	0,107527	0,39
035_ch2_S22_J3	0,969173	0,239544	0,48
039_ch1_S22_J4	0,727273	0,294118	0,47
039_ch3_S22_J4	0,557692	0,355556	0,45
048_ch1_S22_J3	0,875	0,425532	0,63
048_ch2_S22_J3	0,964286	0,521739	0,75
064_ch1_S22_J3	0,77551	0,578947	0,66
064_ch2_S22_J4	0,923077	0,564103	0,81

C(GISSMOO,NSGAII)

Instance	Max	Min	Moyenne
022_S22_J1	0,737705	0,113208	0,42
023_S23_J3	0,757576	0,238095	0,46
024_V2_S22_J1	0,791667	0,122449	0,40
025_S22_J3	0,956522	0,26	0,65
028_ch1_S22_J2	1	0,206897	0,71
028_ch2_S23_J3	1	0,5	0,92
029_S21_J6	0,97	0,0454545	0,49
035_ch1_S22_J3	0,980	0,073	0,405
035_ch2_S22_J3	1	0,153846	0,63
039_ch1_S22_J4	0,727273	0,416667	0,59
039_ch3_S22_J4	0,710526	0,225806	0,39
048_ch1_S22_J3	0,875	0,366667	0,61
048_ch2_S22_J3	0,826087	0,142857	0,55
064_ch1_S22_J3	0,880952	0,347826	0,62
064_ch2_S22_J4	0,9	0,615385	0,78

C(PMS^MO, GISSMOO)

Instance	Max	Min	Moyenne
022_S22_J1	0,29798	0,0604027	0,18
023_S23_J3	0,180645	0,0300188	0,11
024_V2_S22_J1	0,0438757	0,00731707	0,02
025_S22_J3	0,116848	0,0205479	0,09
028_ch1_S22_J2	0,102222	0	0,04
028_ch2_S23_J3	1	0	0,47
029_S21_J6	0,193103	0,0181818	0,09
035_ch1_S22_J3	2,71E-01	0,029	1,38E-01
035_ch2_S22_J3	0,24026	0	0,12
039_ch1_S22_J4	0,555556	0,047619	0,30
039_ch3_S22_J4	0,338235	0,142857	0,22
048_ch1_S22_J3	0,219048	0,021978	0,14
048_ch2_S22_J3	0,118644	0	0,06
064_ch1_S22_J3	0,131868	0,0519481	0,08
064_ch2_S22_J4	0,238095	0,02	0,11

C(PMS^MO,NSGAII)

Instance	Max	Min	Moyenne
022_S22_J1	0,754098	0,0555556	0,45
023_S23_J3	0,607143	0	0,32
024_V2_S22_J1	0,722222	0	0,27
025_S22_J3	0,6	0,0217391	0,19
028_ch1_S22_J2	0,52	0,0909091	0,23
028_ch2_S23_J3	1	0	0,67
029_S21_J6	0,333	0,000	0,156
035_ch1_S22_J3	0,515152	0	0,17
035_ch2_S22_J3	0,666667	0	0,14
039_ch1_S22_J4	0,846154	0,4	0,64
039_ch3_S22_J4	0,535714	0,25	0,41
048_ch1_S22_J3	0,702703	0,176471	0,42
048_ch2_S22_J3	0,26087	0,0344828	0,13
064_ch1_S22_J3	0,534884	0,189655	0,39
064_ch2_S22_J4	1	0,40625	0,68

C(NSGAII, GISSMOO)

Instance	Max	Min	Moyenne
022_S22_J1	0,206751	0,030303	0,10
023_S23_J3	0,128959	0,0258065	0,07
024_V2_S22_J1	0,0576037	0,00548446	0,02
025_S22_J3	0,048	0,0028169	0,02
028_ch1_S22_J2	0,0729614	0	0,02
028_ch2_S23_J3	1	0	0,33
029_S21_J6	0,104545	0,0444444	0,07
035_ch1_S22_J3	2,10E-01	7,25E-02	0,12
035_ch2_S22_J3	0,166667	0,00344828	0,10
039_ch1_S22_J4	0,333333	0,0454545	0,15
039_ch3_S22_J4	0,31746	0,135593	0,22
048_ch1_S22_J3	0,2	0,0208333	0,13
048_ch2_S22_J3	0,204545	0	0,10
064_ch1_S22_J3	0,184211	0,0340909	0,12
064_ch2_S22_J4	0,169811	0,02	0,08

C(NSGAII,PMS^MO)

Instance	Max	Min	Moyenne
022_S22_J1	0,396694	0	0,12
023_S23_J3	0,356164	0,00980392	0,11
024_V2_S22_J1	0,084507	0,0510204	0,07
025_S22_J3	0,25	0,0517241	0,13
028_ch1_S22_J2	0,432432	0	0,14
028_ch2_S23_J3	1	0	0,48
029_S21_J6	0,302	0,083	0,190
035_ch1_S22_J3	0,491228	0,101266	0,30
035_ch2_S22_J3	0,556122	0,0591398	0,22
039_ch1_S22_J4	0,357143	0,117647	0,25
039_ch3_S22_J4	0,416667	0,155556	0,28
048_ch1_S22_J3	0,586957	0,142857	0,36
048_ch2_S22_J3	0,78125	0,321429	0,57
064_ch1_S22_J3	0,625	0,214286	0,40
064_ch2_S22_J4	0,333333	0,025641	0,18

Figure A.2 : Résultats numérique selon la métrique *C* de GISSMOO, PMS^MO et NSGAII pour les instances de l'ensemble B

GISSMOO

PMS^{MO}

NSGAII

C(GISSMOO, PMS^{MO})

Instance	Max	Min	Moyenne
022_S49_J2	0,853659	0,435897	0,65
023_S49_J2	0,949153	0,4375	0,70
024_S49_J2	0,423077	0,277778	0,36
025_S49_J1	0,707865	0,571429	0,64
028_CH1_S50_J4	0,641026	0,146341	0,46
028_CH2_S51_J1*	1	1	1,00
029_S49_J5	0,8125	0,666667	0,74
034_VP_S51_J1_J2_J3	0,954545	0	0,95
034_VU_S51_J1_J2_J3	0,171429	0	0,84
035_CH1_S50_J4*	1	1	1,00
035_CH2_S50_J4	1	0,882353	0,97
039_CH1_S49_J1*	1	1	1,00
039_CH3_S49_J1	0,719298	0,473684	0,61
048_CH1_S50_J4	0,852459	0,586957	0,72
048_CH2_S49_J5	0,84	0,451613	0,69
064_CH1_S49_J1	0,857143	0,285714	0,50
064_CH2_S49_J4	0,857143	0,3	0,64
655_CH1_S51_J2_J3_J4*	1	1	1,00
655_CH2_S52_J1_J2_S01_J1	0,857143	0,470588	0,69

C(GISSMOO,NSGAII)

Instance	Max	Min	Moyenne
022_S49_J2	0,964286	0,16129	0,54
023_S49_J2	0,931034	0,258065	0,57
024_S49_J2	0,666667	0,285714	0,44
025_S49_J1	0,704762	0,535714	0,64
028_CH1_S50_J4	1	0,243243	0,50
028_CH2_S51_J1*	1	1	1,00
029_S49_J5	0,860465	0,240741	0,54
034_VP_S51_J1_J2_J3	0,882353	0	1,00
034_VU_S51_J1_J2_J3	0,111	0,000	0,800
035_CH1_S50_J4*	1	1	1,00
035_CH2_S50_J4	1	1	1,00
039_CH1_S49_J1*	1	1	1,00
039_CH3_S49_J1	0,708333	0,375	0,58
048_CH1_S50_J4	0,825	0,390244	0,62
048_CH2_S49_J5	0,882353	0,173913	0,49
064_CH1_S49_J1	0,8	0,25	0,57
064_CH2_S49_J4	0,857143	0	0,38
655_CH1_S51_J2_J3_J4*	1	1	1,00
655_CH2_S52_J1_J2_S01_J1	1	0,285714	0,69

C(PMS^{MO}, GISSMOO)

Instance	Max	Min	Moyenne
022_S49_J2	0,127907	0	0,03
023_S49_J2	0,181818	0,0241935	0,10
024_S49_J2	0,0585774	0,00738007	0,03
025_S49_J1	0,0760408	0,0512295	0,06
028_CH1_S50_J4	0,0939394	0	0,02
028_CH2_S51_J1*	1	1	1,00
029_S49_J5	0,112245	0,049505	0,08
034_VP_S51_J1_J2_J3	0	0	0,00
034_VU_S51_J1_J2_J3	6,36E-03	0,000	7,89E-02
035_CH1_S50_J4*	1	1	1,00
035_CH2_S50_J4	0,0967742	0,0277778	0,04
039_CH1_S49_J1*	1	1	1,00
039_CH3_S49_J1	0,212766	0,108911	0,15
048_CH1_S50_J4	0,175258	0,0102041	0,10
048_CH2_S49_J5	0,181818	0,0169492	0,08
064_CH1_S49_J1	0,0574713	0	0,04
064_CH2_S49_J4	0,5	0	0,19
655_CH1_S51_J2_J3_J4*	1	1	1,00
655_CH2_S52_J1_J2_S01_J1	0,0714286	0	0,01

C(PMS^{MO},NSGAII)

Instance	Max	Min	Moyenne
022_S49_J2	0,607143	0	0,16
023_S49_J2	0,931034	0,119403	0,38
024_S49_J2	0,285714	0,0454545	0,13
025_S49_J1	0,589286	0,467532	0,52
028_CH1_S50_J4	0,744681	0	0,19
028_CH2_S51_J1*	1	1	1,00
029_S49_J5	0,930233	0	0,27
034_VP_S51_J1_J2_J3	0,192	0,000	0,500
034_VU_S51_J1_J2_J3	0	0	0,88
035_CH1_S50_J4*	1	1	1,00
035_CH2_S50_J4	1	0,333333	0,96
039_CH1_S49_J1*	1	1	1,00
039_CH3_S49_J1	0,645833	0,125	0,34
048_CH1_S50_J4	0,714286	0,317073	0,49
048_CH2_S49_J5	0,521739	0,173913	0,30
064_CH1_S49_J1	0,909091	0	0,53
064_CH2_S49_J4	1	0	0,48
655_CH1_S51_J2_J3_J4*	1	1	1,00
655_CH2_S52_J1_J2_S01_J1	0,777778	0	0,21

C(NSGAII, GISSMOO)

Instance	Max	Min	Moyenne
022_S49_J2	0,209302	0	0,10
023_S49_J2	0,344828	0,0267857	0,16
024_S49_J2	0,0163399	0	0,01
025_S49_J1	0,0536913	0,00489396	0,03
028_CH1_S50_J4	0,0783784	0	0,04
028_CH2_S51_J1*	1	1	1,00
029_S49_J5	0,373737	0,0272727	0,21
034_VP_S51_J1_J2_J3	0	0	0,00
034_VU_S51_J1_J2_J3	2,46E-03	0,00E+00	0,06
035_CH1_S50_J4*	1	1	1,00
035_CH2_S50_J4	0,0666667	0,0277778	0,04
039_CH1_S49_J1*	1	1	1,00
039_CH3_S49_J1	0,297872	0,0851064	0,15
048_CH1_S50_J4	0,268041	0,049505	0,12
048_CH2_S49_J5	0,272727	0,0422535	0,14
064_CH1_S49_J1	0,0348837	0	0,01
064_CH2_S49_J4	0,2	0	0,13
655_CH1_S51_J2_J3_J4*	1	1	1,00
655_CH2_S52_J1_J2_S01_J1	0,25	0	0,09

C(NSGAII, PMS^{MO})

Instance	Max	Min	Moyenne
022_S49_J2	0,52381	0,0666667	0,34
023_S49_J2	0,610169	0,0192308	0,39
024_S49_J2	0,269231	0,222222	0,24
025_S49_J1	0,147727	0,0280899	0,07
028_CH1_S50_J4	0,510638	0,0151515	0,30
028_CH2_S51_J1*	1	1	1,00
029_S49_J5	0,868421	0	0,52
034_VP_S51_J1_J2_J3	0,250	0,000	0,536
034_VU_S51_J1_J2_J3	0,0178571	0	0,67
035_CH1_S50_J4*	1	1	1,00
035_CH2_S50_J4	0,428571	0,0769231	0,25
039_CH1_S49_J1*	1	1	1,00
039_CH3_S49_J1	0,649123	0,1	0,36
048_CH1_S50_J4	0,347826	0,139535	0,24
048_CH2_S49_J5	0,583333	0,24	0,41
064_CH1_S49_J1	0,285714	0	0,14
064_CH2_S49_J4	0,5	0	0,24
655_CH1_S51_J2_J3_J4*	1	1	1,00
655_CH2_S52_J1_J2_S01_J1	0,461538	0,25	0,37

Figure A.3 : Résultats numérique selon la métrique ***C*** de GISSMOO, PMS^{MO} et NSGAII pour les instances de l'ensemble X