UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

par
Arnaud Zinflou

Système interactif d'aide à la décision basé sur des algorithmes génétiques pour l'optimisation multi-objectifs

29 juin 2004

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 de nombreux secteurs de l'industrie, les décideurs sont confrontés à des problèmes complexes, de grande dimension et multi-objectifs. Prendre une décision, pour ce genre de problèmes, nécessite en général l'optimisation simultanée de plusieurs objectifs souvent contradictoires. Malheureusement, la complexité des problèmes industriels, le nombre sans cesse croissant d'objectifs à optimiser simultanément et la rapidité des changements de l'environnement raccourcissent considérablement les délais de prise de décision tout en rendant cette tâche plus difficile pour les gestionnaires. Des outils informatiques comme les systèmes interactifs d'aide à la décision (SIAD) s'avèrent donc d'une grande utilité pour le décideur car ils favorisent une répartition évolutive des compétences entre l'utilisateur et la machine et offrent une bonne intégration de l'homme et de la machine dans le processus de décision. Les SIAD permettent donc au décideur d'évaluer la situation, les diverses alternatives et leurs impacts éventuels.

Récemment, les techniques d'optimisation multi-objectifs à l'aide d'algorithmes évolutionnaires, et plus particulièrement d'algorithmes génétiques, suscitent de plus en plus d'intérêt auprès des chercheurs 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.

Ce mémoire propose d'intégrer les systèmes interactifs d'aide à la décision, l'optimisation multi-objectifs et les algorithmes génétiques afin de proposer un outil performant permettant la résolution de problèmes d'optimisation multi-objectifs. Dans le SIAD proposé, le traitement d'un problème multi-objectifs se fera en deux phases. La première phase consiste à approximer l'ensemble Pareto optimal. Cette étape sera réalisée à l'aide d'un nouvel algorithme génétique multi-objectifs hybride. Cette approche combine un algorithme génétique basé sur les concepts d'élitisme, de niche et de dominance Pareto avec des opérateurs de recherche locale. La deuxième phase utilise l'expérience du décideur afin d'approfondir la recherche dans une zone plus spécifique de l'ensemble pseudo Pareto Optimal en fonction des préférences exprimées par celui-ci. Pour cela, une approche générique de recherche de solutions de compromis est combinée avec un algorithme génétique. Le SIAD proposé est un outil flexible et facile d'utilisation grâce à son interface homme-machine conviviale. Cet outil ne constitue qu'un support à la prise de décision, la décision finale restant du ressort du planificateur. Un exemple d'application du SIAD proposé a été réalisé pour aborder un problème d'ordonnancement industriel rencontré dans une entreprise de production d'aluminium. Cette application montre bien l'intérêt pratique de ce genre de système.

Bien qu'ayant produit des résultats très encourageants, ce travail de recherche représente surtout une première exploration des possibilités offertes par la combinaison de trois domaines de recherche en constante évolution : les SIAD, l'optimisation multi-objectifs et les algorithmes génétiques. L'union de ces trois champs de recherche laisse entrevoir des possibilités intéressantes pouvant mener à la conception de nouveaux outils de résolution permettant l'élaboration de scénarios pour éclairer la prise de décision. Ce travail peut donc être considéré comme une contribution vers l'élaboration et l'implantation de ce genre d'outils.

REMERCIEMENTS

Je tiens tout d'abord à remercier Mme Caroline Gagné, ma directrice de recherche pour sa disponibilité, son support et son implication dans ce travail de recherche. En particulier pour avoir été attentive à tous mes commentaires ou idées et pour avoir su répondre à mes nombreuses interrogations. Son optimisme et ses conseils ont été une aide précieuse sans laquelle ce travail de recherche n'aurait sans doute pas vu le jour.

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 Daniel Bouchard avec qui j'ai partagé un bureau pendant une grande partie de ma maîtrise et qui, comme moi, est un grand amateur de jeux vidéo.

Je remercie aussi M. Marc Gravel , M. Wilson Price et M. Bernard Lefebvre pour avoir accepté d'évaluer ce travail ainsi que pour leurs judicieux commentaires.

Un travail de recherche ne pourrait être réalisé sans support financier. À ce titre, je voudrais remercier les fonds institutionnels de l'UQAC ainsi que les organismes subventionnaires CRSNG et FQRNT pour leur appui.

Je tiens aussi à remercier Mlle Chitra Sewsagur pour sa patience, sa compréhension et son soutien. Je la remercie également pour avoir contribué à améliorer la qualité du français de ce document.

J'aimerais aussi remercier sincèrement mes parents, mes deux frères, ma sœur et toute ma famille pour leur soutien et leur confiance. Je tiens, en particulier, à remercier mes parents qui m'ont toujours poussé à aller de l'avant et qui ont toujours cru en moi même quand moi-même je n'y croyais plus. Sans leur soutien et leur amour, ce travail n'aurait

jamais pu être réalisé. Je me dois aussi de dire un merci particulier à ma petite sœur Corinne qui a, elle aussi, pris la peine de lire ce mémoire afin de m'aider à corriger les nombreuses fautes d'orthographe qui s'y étaient glissées.

En terminant, j'ai une pensée spéciale pour ma tante Mme Marguerite Gbdamassi et ma grand-mère Mme Élisabeth Zinflou qui nous ont quitté toutes les deux pendant que je poursuivais mon séjour au Canada. Ce travail vous est dédié. Que la terre vous soit légère.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

INTRODUCTION

Le concept de Systèmes Interactifs d'Aide à la Décision (SIAD) s'est formalisé dans la littérature dans les années 70 [Little, 1970; Gorry et Scott-Morton, 1971]. Ce sont des outils spécifiquement développés pour supporter la prise de décision. La conception de tels systèmes implique l'utilisation de techniques issues de divers domaines comme l'informatique, la recherche opérationnelle, l'intelligence artificielle, l'ingénierie logicielle, l'interaction homme-machine et les télécommunications. Les SIAD s'avèrent particulièrement utiles pour aider à trouver une solution appropriée à des problèmes complexes, de grande dimension et ayant des objectifs fortement dépendants des préférences de l'utilisateur. La plupart des problèmes d'optimisation font partie de cette catégorie de problèmes.

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 de l'objectif défini. Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes dans différents secteurs de l'industrie : mécanique, chimie, télécommunications, environnement, transport, et autres. 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. Pour ce genre de problèmes, 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. La nécessité de trouver rapidement des solutions acceptables à plusieurs de ces problèmes a entraîné le développement de plusieurs algorithmes d'approximation dont font partie les métaheuristiques.

Parmi les métaheuristiques, les algorithmes génétiques s'avèrent être une approche particulièrement intéressante pour résoudre des problèmes d'optimisation multi-objectifs. Cet intérêt s'explique notamment par la faculté pour ce type d'algorithme à exploiter de vastes espaces de recherche et à générer plusieurs solutions de compromis en une seule étape d'optimisation.

Les algorithmes génétiques sont des programmes informatiques qui 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. Ce sont à la base des algorithmes d'optimisation stochastiques, mais qui peuvent également servir pour l'apprentissage

automatique. Bien que les principes sous-jacents soient simples, ces algorithmes s'avèrent être des mécanismes de recherche généraux, puissants et robustes.

L'objectif général de ce travail de recherche est de spécifier et concevoir un système interactif d'aide à la décision basé sur des algorithmes génétiques pour l'optimisation multi-objectifs afin de mieux comprendre les bénéfices que ce type de systèmes peut apporter dans la résolution de problèmes d'optimisation multi-objectifs. L'atteinte de cet objectif passe par une connaissance adéquate des systèmes interactifs d'aide à la décision, des problèmes d'optimisation multi-objectifs et des algorithmes génétiques. Le contenu de ce mémoire est organisé en cinq chapitres.

Dans le Chapitre 2, une revue de la littérature sur les systèmes interactifs d'aide à la décision est présentée en détaillant les notions les plus pertinentes pour la réalisation de ce travail. Ce chapitre présente également les principaux facteurs de succès ou d'échec ayant freiné l'implantation des systèmes interactifs d'aide à la décision ainsi qu'un bref aperçu de l'étendue de leur champ d'applications.

Le Chapitre 3 porte sur les algorithmes génétiques et l'optimisation multi-objectifs. Tout d'abord, les concepts de bases d'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 génétiques sont ensuite décrits comme une technique récente d'optimisation qui possède des caractéristiques intéressantes pour résoudre des problèmes multi-objectifs. Finalement, un certain nombre d'algorithmes génétiques multi-objectifs sont présentés en essayant d'apporter un regard critique sur chacun d'eux.

Le Chapitre 4 présente les travaux effectués dans ce mémoire pour développer et implanter les différentes approches de résolution constituant la base de modèle du système interactif d'aide à la décision proposé. La mise en œuvre de ces différentes approches sera illustrée à l'aide d'un problème d'ordonnancement industriel multi-objectifs rencontré dans une entreprise de production d'aluminium. Ce chapitre est organisé en trois parties. Dans un premier temps, le problème d'ordonnancement est énoncé. Dans un deuxième temps, un algorithme génétique uni-objectif permettant la résolution du problème est présenté. Finalement, la dernière partie de ce chapitre porte sur l'adaptation et le développement de trois approches de résolution du problème, mais cette fois-ci, d'un point de vue multi-objectifs. En particulier, nous y présentons l'AG^{MOPI} un algorithme génétique multi-objectifs permettant d'approximer en une seule exécution l'ensemble Pareto.

Le chapitre 5 présente le système interactif d'aide à la décision proposé dans son ensemble. Un accent particulier étant mis sur l'interface homme-machine et la visualisation des solutions.

Il faut également souligner que ce travail fait suite à une série de travaux effectués dans le but d'établir des méthodes de résolution performantes à des problèmes d'ordonnancement industriel. Cependant, ce travail apporte un nouveau point de vue en intégrant différentes notions issues de l'optimisation multi-objectifs, des SIAD et des algorithmes génétiques. Plus particulièrement, il propose un environnement de résolution complet permettant une répartition évolutive des compétences entre l'utilisateur et la machine, en fonction du problème à résoudre, et offrant une bonne intégration de l'homme et de la machine dans le processus de décision.

En définitive, ce travail de recherche permet de mieux évaluer l'intérêt et le potentiel d'outils comme les SIAD dans un contexte industriel. La conclusion de ce mémoire présente le bilan de ce travail et permet d'identifier de futures avenues de recherche.

CHAPITRE 2

LES SYSTÈMES INTERACTIFS D'AIDE À LA DÉCISION

2.1 Introduction

Comprendre les faits, prévoir les événements, anticiper les actions, c'est ce que les hommes ont de tout temps désiré qu'il s'agisse de phénomènes naturels, d'administration, d'un projet de conquête, de rejet d'envahisseurs ou autres. Pourtant, le processus de décision est l'un des plus complexes par sa nature. En théorie, une décision doit être prise en connaissance de cause, donc à partir d'informations suffisamment pertinentes pour la construire, en éliminant le plus possible les risques d'échec. De plus, une décision, entraînant une ou plusieurs actions, engage celui qui la prend.

Dans la vie quotidienne, nos décisions sont souvent prises sur la base d'intuitions et d'expériences passées. Cependant, ce type de stratégies ne peut s'appliquer qu'à des problèmes familiers. Lorsque nous sommes confrontés à des situations nouvelles, la tâche de prise décision devient beaucoup plus difficile.

De nos jours, l'environnement des organisations est de plus en plus complexe, évolue rapidement et la tendance est plutôt à l'accroissement de cette complexité. Cet accroissement est dû à plusieurs facteurs comme la technologie, une complexité structurelle des décisions et une plus forte compétition nationale comme internationale causée par la mondialisation des économies. Par ailleurs, les conséquences financières et humaines d'une erreur de décision peuvent s'avérer dramatiques à cause des réactions en chaîne qu'elles pourraient engendrer sur les différentes parties de l'organisation. Il importe donc pour le décideur de pouvoir prévoir les conséquences éventuelles des décisions qu'il compte prendre.

Dans un tel contexte, un Système Interactif d'Aide à la Décision (**SIAD**) permettrait à l'usager d'évaluer la situation, les diverses alternatives et leurs impacts éventuels. Le concept de SIAD est apparu dans la littérature dans les années 70 [Little, 1970; Gorry et Scott-Morton, 1971]. Ce sont des outils spécifiquement développés pour supporter la prise de décision. Les SIAD se caractérisent principalement par leur interactivité, leur flexibilité et leur adaptabilité. Le développement de tels systèmes implique l'utilisation de techniques issues de divers domaines comme l'informatique, la recherche opérationnelle, l'intelligence artificielle, l'ingénierie logicielle, l'interaction homme-machine et les télécommunications. Les SIAD sont particulièrement utiles pour aider à résoudre des problèmes complexes de grande dimension et ayant des objectifs fortement dépendants des préférences de l'utilisateur.

Le domaine des SIAD est très vaste. Quelques auteurs proposent des états de l'art relativement complets [Davis *et al.*, 1986; Lévine et Pomerol, 1990]. Ce chapitre introduit en premier lieu la notion de prise de décision ainsi que quelques définitions liées à celle-ci. Cette présentation ne se prétend pas exhaustive, mais a uniquement pour but d'introduire un minimum d'éléments nécessaires à la présentation des SIAD. Par la suite, les différents concepts liés aux SIAD sont présentés ainsi que les principales barrières ayant freiné leur succès. Finalement, nous terminons en présentant un bref aperçu du champ d'application des SIAD dans une organisation.

2.2 La prise de décision

Nous allons dans un premier temps donner quelques définitions importantes à propos de la prise de décision. Pour Holtzman [1989], prendre une décision signifie concevoir et s'engager dans une stratégie d'allocation irrévocable de ressources. Le processus de prise de décision n'inclut pas l'allocation de ces ressources qui est appelée une *action*. De manière plus générale, on peut dire que tout individu placé devant plusieurs alternatives mutuellement exclusives choisit l'une d'entre elles à la suite d'un processus mental appelé *décision*.

On peut associer à toute décision un *domaine* qui correspond à son champ d'application. Par exemple, distinguer les décisions médicales, des décisions militaires ou industrielles. Ces domaines peuvent être eux-mêmes divisés en sous-domaines. Le domaine d'une décision est par définition générique, c'est-à-dire que plusieurs décisions partagent le même domaine. Par contre, certains facteurs d'une décision sont uniques et dépendent de la décision et du décideur. L'ensemble de ces facteurs est appelé la *situation* d'une décision. Cette situation se compose du *contexte* et des *préférences* du décideur. Chaque décision a un contexte qui lui est associé et qui affecte fortement l'intérêt et la disponibilité des différents choix pour le décideur. Ce contexte inclut l'état des informations détenues par le décideur et constitue sa perception des conséquences possibles de ses actions. Chaque décideur a des désirs particuliers qui sont exprimés par ses préférences sur les résultats possibles de sa décision. Par exemple, pour l'achat d'une voiture, le contexte d'un décideur pourrait inclure le fait d'avoir une famille, d'avoir l'habitude d'effectuer de longs trajets pour son travail et d'habiter dans une région désertique. Ses préférences peuvent inclure

son enthousiasme pour les voitures 4 X 4, les embrayages automatiques ou la transmission intégrale avec ABS. À toute décision est toujours associée des *alternatives* parmi lesquelles le décideur doit choisir. Sans alternatives, il n'y a pas de décision. À chaque alternative est associée un *résultat* ou bénéfice espéré (au sens large du terme) qui peut guider le choix entre les alternatives.

Dans la section suivante, nous allons décrire les différentes phases d'un processus de décision ainsi que les différents niveaux de structuration d'un problème de décision.

2.2.1 Le processus de décision

Un processus de décision, dans le cadre défini par la gestion des organisations, se compose de quatre phases : une phase d'information, une phase de conception, une phase de choix et une phase d'évaluation du choix [Lévine et Pomerol, 1990].

Lors de la phase d'information, il s'agit d'identifier les objectifs ou buts du décideur, c'est-à-dire de définir le problème à résoudre. Pour cela, il est nécessaire de rechercher les informations pertinentes en fonction des questions que se pose le décideur. Ensuite, la décision est classée parmi les différentes catégories connues. Pendant cette phase, l'acquisition d'informations pertinentes peut se poser elle-même en terme de décision. Pour certains problèmes, il est parfois difficile de trouver des informations pertinentes. Or, ce sont elles qui sont à l'origine du processus de décision et leur choix est crucial car elles influencent fortement les autres phases puisque tous les choix suivants en découlent.

La phase de conception comprend la génération, le développement et l'analyse des différentes suites possibles d'actions. Il va donc être nécessaire de choisir un ou plusieurs modèles de décision en fonction de la complexité du problème à traiter. Pour le ou les

modèles choisis, il faut également déterminer les variables de décision, les variables incontrôlables, les variables résultats ainsi que les relations mathématiques, symboliques ou qualitatives [Crossland et Wynne, 1995] entre ces variables et construire les différentes alternatives.

Lors de la phase de choix, le décideur choisit entre les différentes suites d'actions qu'il a été capable de construire et d'identifier pendant la phase précédente. Cette phase inclut la recherche, l'évaluation et la recommandation d'une solution appropriée au modèle.

Finalement, la phase d'évaluation du choix correspond à une évaluation *a posteriori* du choix du décideur. Cette évaluation permettra éventuellement de corriger les petites erreurs. Comme le montre la Figure 2.1, ce processus n'est pas obligatoirement séquentiel, il peut y avoir des retours en arrière. Par exemple, pendant la seconde ou la troisième phase, le décideur peut être amené à générer une nouvelle alternative ou encore à rechercher de nouvelles informations, et ensuite à modifier le ou les modèles choisis. La présence de ces retours en arrière pendant le processus de décision dépend du niveau de structuration du problème de décision.

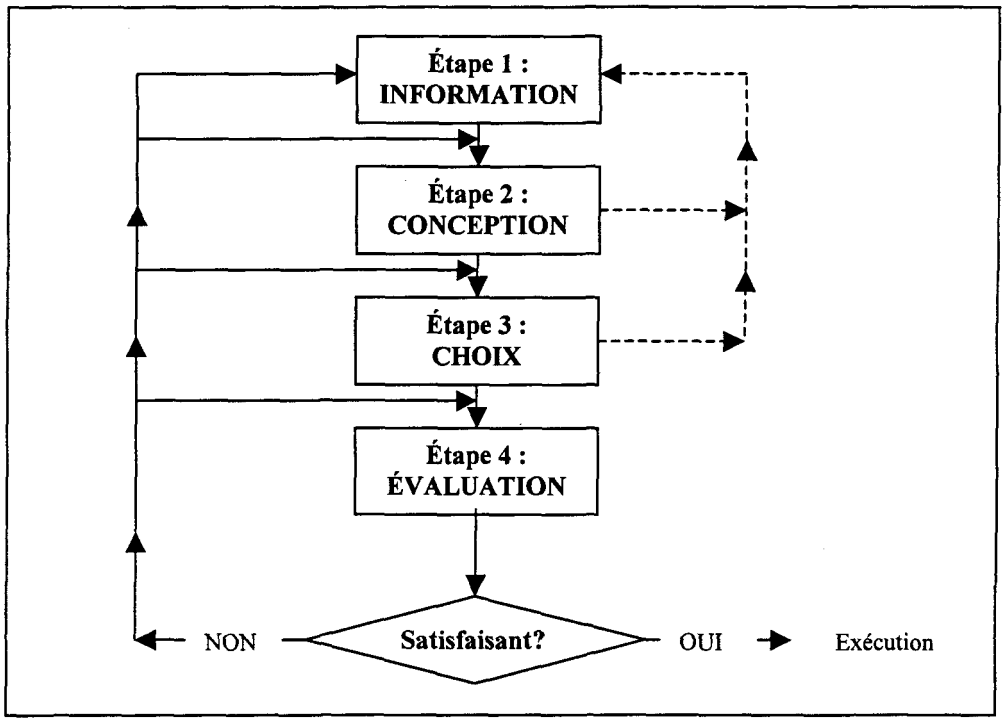


Figure 2.1 : Modèle du processus décisionnel [Reix, 2000]

2.2.2 Niveau de structuration des problèmes de décisions

On distingue généralement *les décisions programmables* (correspondant aux problèmes structurés) *des décisions non programmables* (correspondant aux problèmes peu ou mal structurés). En pratique, il n'existe toutefois pas de dichotomie entre les problèmes structurés et les problèmes non structurés mais un continuum du degré de structuration allant des moins structurés aux plus structurés [Garlatti, 1996].

Un problème de décision est dit structuré s'il peut être exprimé par un modèle calculable, stable et s'il est accompagné d'une règle de choix invariante. Cette dernière fait en sorte qu'une décision structurée présente souvent un caractère répétitif. Le processus de

prise de décision se réduit dans ce cas à l'exécution d'un calcul. Il s'agit alors de pseudo-décision de type algorithmique [Reix, 2000].

À l'opposé, un problème de décision peu ou mal structuré est un problème qui va nécessiter un effort important pour être formalisé. Ce genre de problèmes de décision possède en général trois principales caractéristiques [Klein et Tixier, 1971]. Premièrement, leur résolution est fortement dépendante des préférences, des jugements, de l'intuition et de l'expérience du décideur. Deuxièmement, les objectifs poursuivis lors de la prise de décision sont nombreux, en conflits et fortement dépendants de la perception de l'utilisateur. La recherche d'une solution pour ce genre de problème implique un mélange de recherche d'informations, de formulation du problème, de calcul et de manipulation de données. Finalement, ce sont des problèmes qui évoluent rapidement et dont la solution doit être obtenue dans un temps limité. Un des aspects les plus importants est que, pour cette classe de problème de décision, l'homme prend l'avantage sur la machine contrairement aux problèmes structurés. Résoudre le problème exige de faire appel à l'intuition et au savoir-faire du décideur qui devient l'élément prépondérant du couple Homme/Machine. Dans ce cas, le contrôle de la recherche de solutions doit être laissé, en totalité ou en partie, au décideur.

Les SIAD, présentés dans la section suivante, ont été conçus pour résoudre la seconde catégorie de problèmes de décision [Eierman et Niederman, 1995].

2.3 Les SIAD

Selon Power [1999], le concept de SIAD a évolué à partir de deux domaines de recherche : les études sur la prise de décision dans les organisations du Carnegie Institute of Technology à la fin des années 1950 et les travaux sur les systèmes informatiques interactifs du Massachusetts Institute of Technology dans les années 1960. Le concept de SIAD est devenu un domaine de recherche en soit dans le milieu des années 1970, avant de prendre plus d'ampleur au cours des années 1980 [Hättenschwiler, 1999]. Il apparaît clairement que les SIAD reposent sur des bases multidisciplinaires, incluant notamment l'informatique, la recherche opérationnelle, l'intelligence artificielle, l'ingénierie logicielle, l'interaction homme-machine et de plus en plus les télécommunications [Gachet, 2001].

Le concept de SIAD est extrêmement vaste et sa définition varie selon le point de vue de l'auteur [Druzdzel et Flynn, 1999]. De nombreuses définitions des SIAD ont été proposées comme celle de Peaucelle [Davis *et al.*, 1986]:

> L'expression systèmes interactifs d'aide à la décision désigne les systèmes qui servent dans le processus de la prise de décision. Ces systèmes aident mais ne remplacent pas le décideur. Dans cette perspective, l'aide généralement automatisée permet au décideur d'avoir accès aux données et de tester différents choix possibles pour la résolution du problème à traiter.
>
> ...l'efficacité du processus de résolution du problème décisionnel est amplifiée par l'interaction entre l'humain et la machine, chacun d'eux étant utilisé dans leurs champs distinctifs de compétence.

De son coté, Finlay [1994] définit simplement les SIAD comme « des systèmes informatiques supportant la prise de décision ». De son côté, Probst [1984] pense qu'un SIAD est :

> Un logiciel conçu pour faciliter la préparation d'informations pertinentes sur la base desquelles une décision motivée peut être prise... donc un ensemble de moyens informatiques organisés pour améliorer le processus décisionnel. Ces systèmes fourniraient un cadre normatif à la démarche devant aboutir à une prise de décision.

Turban [1993] définit pour sa part un SIAD comme :

> un système d'information interactif, flexible, adaptable et spécifiquement développé pour aider la résolution d'un problème de décision en améliorant la prise de décision. Il utilise des données, fournit une interface utilisateur simple et autorise l'utilisateur à développer ses propres idées ou points de vue.

Finalement, Schroff [1998] et Keen [1981] estiment qu'il est impossible de donner une définition précise qui inclurait toutes les facettes d'un SIAD. S'il n'existe pas une définition universelle des SIAD, les différents auteurs s'accordent sur les différentes caractéristiques ou fonctionnalités recherchées dans ce type de système. Parmi celles-ci, mentionnons les suivantes [Garlatti, 1996] :

a. ils doivent apporter principalement une aide pour les problèmes peu ou mal structurés en connectant ensemble des jugements humains et des informations calculées;
b. ils doivent posséder une interface simple et conviviale afin d'éviter que l'usager ne soit perdu devant la complexité du système;
c. ils doivent fournir une aide pour différentes catégories d'utilisateurs ou différents groupes d'utilisateurs;
d. ils doivent supporter des processus interdépendants ou séquentiels;
e. ils doivent être adaptatifs dans le temps. Le décideur peut être réactif, être capable de confronter des conditions changeant rapidement et d'adapter le SIAD

pour faire face aux nouvelles conditions. Un SIAD doit être suffisamment flexible pour que le décideur puisse ajouter, détruire, combiner, changer et réarranger les variables du processus de décision ainsi que les différents calculs fournissant ainsi une réponse rapide à des situations inattendues;

f. ils doivent laisser le contrôle de toutes les étapes du processus de décision au décideur pour que celui-ci puisse remettre en cause à tout moment les recommandations faites par le SIAD. Un SIAD doit aider le décideur et non se substituer à lui;

g. ils doivent utiliser des modèles. La modélisation permet d'expérimenter différentes stratégies sous différentes conditions. Ces expériences peuvent apporter de nouvelles vues sur le problème et un apprentissage;

h. les SIAD les plus avancés utilisent un système à base de connaissances qui apporte notamment une aide efficace et effective dans des problèmes nécessitant une expertise;

i. ils doivent permettre la recherche heuristique ; et

j. ils ne doivent pas être des outils de type *boite noire*. Le fonctionnement d'un SIAD doit être fait de manière à ce que le décideur le comprenne et l'accepte.

Pour satisfaire tous ces critères, un SIAD se compose d'au moins trois composants : un module de dialogue ou interface homme-machine (IHM), un module contenant les données (base d'informations) et un module contenant les procédures de calcul ou modèles (base de modèle) [Sage, 1991]. À ces trois modules peut éventuellement s'ajouter une base de connaissances [Garlatti, 1996]. Comme l'indique la Figure 2.2, le module de dialogue est

interconnecté avec les autres modules. Il constitue l'interface entre l'utilisateur et le reste du système. Les modules représentés en pointillés constituent les modules optionnels d'un SIAD.

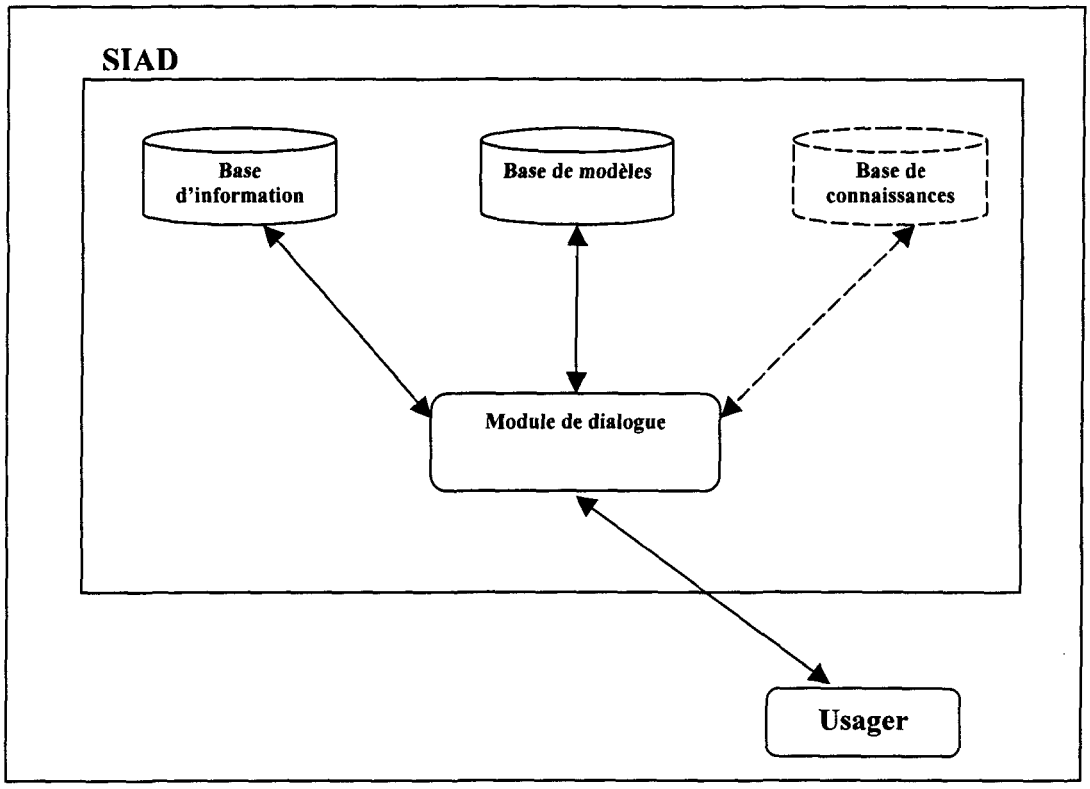


Figure 2.2 : Composition d'un SIAD

Le module de dialogue représente le module charnière du système permettant d'établir une collaboration entre le décideur et la machine. Pour Courbon et Stabell [1986], le module de dialogue est au centre du SIAD et sa réalisation est primordiale. En fait, une étude [Myers, 1995] a montré qu'au moins 50 % du code des applications interactives correspond à l'IHM et 50 % du temps du développement est dépensé pour son implantation. C'est par l'intermédiaire des interfaces gérées par ce module que le décideur accède aux données et aux fonctions de calcul du SIAD. Une fois les manipulations

demandées par le décideur effectuées, le système lui renvoie le résultat via les interfaces du module de dialogue. Les échanges sont d'autant plus favorisés que les représentations des résultats, tout comme le mode de questionnement du système, correspondent aux représentations mentales du décideur. Ainsi, le décideur peut exercer son contrôle et effectuer sa recherche heuristique dans de bonnes conditions. Un bon module de dialogue doit permettre d'afficher les informations sous différentes formes (graphiques 2D ou 3D, textes, vidéo ou autres). Il doit aussi fournir une aide à l'usager pour que ce dernier mène à bien sa tâche et il doit le guider à l'aide d'exemples précis tout en étant suffisamment flexible pour s'adapter aux besoins des différents usagers. Un autre point à souligner est que l'interface homme-machine doit permettre le choix entre différents modes ou styles de fonctionnement [Lévine et Pomerol, 1990]. Parmi ces différents modes de fonctionnement, Courbon [1986] distingue notamment le mode *assisté* (toutes les possibilités du dialogue sont disponibles), le mode *expert* (l'assistance est très limitée, ce mode s'applique dans le cas d'un décideur expérimenté), le mode *automatique* (le dialogue se déroule tout seul, suppression de la plus grande partie de l'interactivité), le mode *procédure* (des séquences entières sont exécutées à partir d'une instruction) et le mode *apprentissage* (l'accès au dialogue suit une progression pédagogique).

La base d'informations assure la fonction de mémoire, elle stocke non seulement les données, de façon permanente ou passagère, mais elle gère aussi l'enregistrement de données volatiles ainsi que l'effacement de ces mêmes données selon le souhait de l'utilisateur. Ces données volatiles correspondent aux résultats obtenus lors de traitements de données. Les données que nous avons qualifiées de permanentes sont les informations

statistiques ou autres données qui décrivent les situations courantes et passées. Parmi ces données, il peut aussi y avoir des estimations concernant l'évolution de certains paramètres environnementaux.

La base de modèle se compose d'un ensemble de modèles et d'un système de gestion de ceux-ci. Les modèles peuvent être : des outils de recherche opérationnelle, des modèles statistiques ou autres. Pour avoir davantage de flexibilité, un SIAD doit posséder plusieurs modèles [Chabbat, 1997]. Dans cette optique, le SIAD organise les liens et le passage de paramètres entre les différents modèles, de même qu'il gère le module de dialogue [Lévine et Pomerol, 1990].

La base de connaissance regroupe pour sa part un ensemble de connaissances sur le domaine du problème, sur les modèles et sur les stratégies de constructions des modèles. Elle permet d'apporter une aide active à la résolution du problème de décision pendant toutes les phases du processus [Klein, 1988]. Elle introduit la notion d'apprentissage dans le SIAD. La base de connaissance peut aussi jouer dans certains cas le rôle de base de modèles [Hansen *et al.*, 1995].

Un SIAD peut assister le décideur lors des trois premières étapes du processus décisionnel [Davis *et al.*, 1986] décrit à la Section 2.2.1. À la phase d'information, le système permet l'exploration de l'environnement afin d'identifier les conditions et les situations exigeant une prise de décision. Dans la phase de conception, le système permet de préciser la situation avec les diverses hypothèses, de générer les solutions possibles et de tester leur faisabilité. Finalement, lors de la phase de choix, le SIAD peut suggérer certaines

solutions au décideur parmi les différentes alternatives développées au cours de la phase précédente. Cependant, la décision finale revient toujours au décideur.

Jusqu'à maintenant, les SIAD ont été présentés pour être utilisés par un seul usager à la fois. Ils peuvent également être utilisés pour la prise de décision de groupe. En effet, de nombreuses études comme celle de Keen [1981], Liberatore et Titus [1983] ou, plus récemment, Totton et Flavin [1991] montrent que la plupart des décisions dans une organisation sont prises par un groupe d'individus plutôt que par un décideur isolé. Plus une organisation devient complexe, moins les décisions sont prises par une seule personne [Gannon, 1979]. Les SIAD de groupe constituent une spécialisation des SIAD qui reflète cet aspect de la prise de décision.

2.4 Barrières au succès des SIAD

Bien que les SIAD aient été le sujet de nombreuses recherches depuis plus d'une vingtaine d'années, ils n'ont pas nécessairement connu beaucoup de succès, en particulier du point de vu des utilisateurs [Rudnicka et Madey, 2001]. Le domaine des SIAD est trop vaste pour tenter de dresser une liste exhaustive des raisons qui font que ce type de systèmes suscite moins d'intérêt en pratique. Cependant, nous pouvons diviser les différents facteurs d'échec des SIAD en trois grandes catégories [Gachet, 2001] : *les facteurs humains*, *les facteurs conceptuels* et *les facteurs techniques*.

En ce qui concerne les facteurs humains, Ghasemzadeh et Archer [2000] font remarquer qu'en général le décideur n'est pas assez impliqué dans le processus de prise de décision avec un SIAD. Ce manque d'implication risque de provoquer une certaine réticence et un

manque de confiance du décideur vis-à-vis du système. Un autre facteur responsable du manque d'intérêt des usagers par rapport aux SIAD traditionnels peut être expliqué par l'intérêt croissant pour « l'informatique de l'utilisateur final » [Kreie *et al.*, 2000]. Ce concept fait référence aux personnes qui développent des logiciels pour leur usage personnel ou pour d'autres sans être des spécialistes de l'informatique. Le développement de cette tendance s'explique, d'une part, par le coût du matériel informatique qui est devenu de plus en plus bas tout en étant de plus en plus performant, et d'autre part, par les environnements de programmation qui sont de plus en plus intuitifs [Gachet, 2001]. Même si ce type de systèmes est généralement de qualité inférieure aux SIAD traditionnels [Kreie *et al.*, 2000], les utilisateurs préfèrent cette manière de travailler qui augmente leur satisfaction et élimine les problèmes de communication avec un spécialiste de l'informatique. Finalement, les SIAD sont généralement des systèmes complexes et difficiles à utiliser pour des non-spécialistes de l'informatique [Gachet, 2001]. Or, un bon SIAD doit être compréhensible et facile d'utilisation pour les décideurs [Sprague et Watson, 1993], sinon ils auront tendance à garder leurs distances vis-à-vis de tels systèmes.

Les facteurs conceptuels font référence aux problèmes rencontrés par les SIAD à cause d'un mauvais choix lors de leur design. Les SIAD sont utiles pour rechercher de l'information, l'évaluation et le choix d'une décision, mais ils sont moins utiles pour des tâches comme l'exploration d'un problème [Huber, 1982]. Une des tâches principales d'un SIAD consiste à modéliser l'environnement ou le contexte du problème à résoudre [Gachet, 2001]. Cependant, Hättenschwiler [1993] constate plusieurs lacunes relatives à cette modélisation dans la plupart des SIAD :

- un manque de standards et de concepts pour la modélisation;
- un manque de supports pour la réutilisation des modèles existants; et
- un manque de convivialité permettant la modélisation.

En dépit des progrès technologiques considérables de ces dernières années, le développement et l'exploitation des SIAD continuent de connaître des problèmes tant au niveau matériel que logiciel. Le temps de développement inhérent à la construction d'un SIAD est souvent trop long et sa complexité non négligeable [Gachet, 2001]. Construire un SIAD requiert de l'expertise dans des domaines aussi variés que le design d'interface, la programmation et l'analyse de décision ce qui, dans certains cas, peut être compliqué à réunir. Hättenschwiler *et al.* [1998] attribuent le manque d'intérêt pour les SIAD aux techniques traditionnelles de conception qu'ils jugent trop rigides. Ceci entraîne alors des coûts de développement trop élevés, une architecture statique et un manque de réutilisabilité du système. Comme toutes applications interactives, ces systèmes présentent souvent des dépassements de budgets avec des coûts de maintenance très élevés [Palanque *et al.*, 1994]. Certaines évaluations d'applications interactives [Bohm 1976] ont révélé que leur maintenance peut représenter jusqu'à 70 % du coût total de développement. Ce coût de maintenance est occasionné principalement par trois types de problèmes [Farenc *et al.*, 1994]:

- la correction des erreurs apparues lors de l'utilisation de l'application ;
- l'addition de nouvelles fonctionnalités ou la modification de fonctionnalités existantes de l'application; et

- l'amélioration de l'IHM suite aux réticences exprimées par les utilisateurs ou suite aux difficultés qu'ils éprouvent lors de l'utilisation d'une ou de plusieurs fonctionnalités.

Construire une interface homme/machine pour un SIAD constitue un autre facteur technique sensible [Gachet, 2001]. Un système avec une interface usager surchargée, inadaptée ou pas assez claire sera généralement inutilisable en pratique [Druzdzel et Flynn, 1999].

Finalement, les SIAD sont des systèmes complexes souvent composés de plusieurs sous-systèmes hétérogènes (base de données, librairies mathématiques, etc.) qu'il est difficile d'intégrer dans un seul système productif [Gachet, 2001].

2.5 Exemples de domaine d'application des SIAD

Depuis une dizaine d'années, de nombreux SIAD ont été développés pour des secteurs aussi variés que les télécommunications, le transport aérien et ferroviaire, la santé, l'ordonnancement et la gestion de projet. En fait, si l'on souhaitait résumer l'ensemble des domaines d'application des SIAD, on pourrait dire que les SIAD peuvent être mis en place au niveau de chaque activité humaine nécessitant un processus de décision élaboré. Dans cette section, nous présentons quatre exemples de SIAD fonctionnant au quotidien dans des entreprises.

2.5.1 Un système d'allocation de wagon

À la SNCF (société nationale de chemin de fer français), la répartition des wagons de marchandises est effectuée, sur le plan national, par un service « central de répartition »

[Lévine et Pomerol, 1990]. Des experts, au sein de ce service, appelés répartiteurs, reçoivent par téléphone les demandes de wagons vides en provenance des zones. Ils centralisent alors les informations sur les wagons vides disponibles et assurent la répartition entre les zones déficitaires et les zones excédentaires. Autrement dit, ils prennent des wagons dans les zones excédentaires pour les envoyer vers les zones utilisatrices.

Outre les demandes ou les offres téléphoniques, les répartiteurs disposent des données de GCTM (Gestion Centralisée du Transport Marchandise) qui fournissent un état de la situation à trois heures du matin, état qui sort sur l'imprimante vers huit heures du matin à l'arrivée des répartiteurs. Cette situation donne un état des besoins et des wagons présents dans les zones ainsi que la moyenne des besoins et des ressources des jours précédents.

Le SIAD mis en place dans ce cadre a pour objectif d'aider le répartiteur dans sa tâche. Il est basé sur l'utilisation d'un système à base de connaissances. La minimisation des coûts de déplacement à vide, le respect des délais et la satisfaction des clients sont les critères principaux qui régissent le travail des répartiteurs et qui doivent se retrouver dans le SIAD.

Il s'agit là d'un premier exemple de SIAD qui peut être généralisé à l'ensemble des problèmes d'allocation de ressources en extrayant les caractéristiques propres à chaque problème.

2.5.2 La gestion de la production

Un autre type de SIAD pouvant être utile dans l'industrie concerne les systèmes de gestion de production. Un projet de ce type a été mené par le Carnegie-Mellon Robotics Institute sous la direction de M.S. Fox [1981]. Dans ce genre d'application, il s'agit d'optimiser la production en influant sur certains critères : changer les techniques de

conception, changer le matériel, anticiper les pannes en proposant des solutions de remplacement et synchroniser les différentes étapes de production. Le système mis en place permet de faire un suivi de la production. Il permet aussi d'effectuer des simulations par rapport à certaines modifications de paramètres et ainsi de trouver des solutions de rechange en cas de problèmes.

2.5.3 Un système de surveillance pour les sites industriels à hauts risques

Les SIAD peuvent aussi être utiles dans la gestion des risques industriels. La présence de sites industriels à très hauts risques technologiques (centrale nucléaire, centrale thermique et autres) en milieu urbain représente, comme leur nom l'indique, un risque non négligeable pour la population [Boukachour *et al.*, 2000]. Une solution serait l'installation d'un réseau de sirènes qui doit permettre le confinement des populations. Il s'agit pour un opérateur préposé à la détection des risques majeurs de déclencher ces sirènes au moment jugé opportun. Pour prendre sa décision, il doit être en mesure d'avoir une vision globale de la situation et des évènements survenus ainsi que de leurs conséquences (réaction en chaîne). La connaissance concernant l'évolution d'une situation est dispersée auprès de différents experts. Pour avoir une évaluation de la situation, on va devoir interroger des experts en chimie, en météorologie, en traitement des situations d'urgence (gestion des populations, traitement des blessés par les services concernés, traitement des accidents routiers par les pompiers, et autres). L'ensemble des informations à traiter et à synthétiser par l'opérateur, qui prendra la décision finale de déclencher ou non les sirènes, est vaste. Le SIAD devra donc permettre de mettre en évidence l'information pertinente en ayant recours à cette multi-expertise.

2.5.4 Pioneer Natural Resources

Dans l'industrie du pétrole et du gaz, un grand nombre de variables sont liées à l'exploitation d'une société d'énergie, notamment les coûts du développement et de la production. À cause de la relation complexe entre toutes ces variables, les gestionnaires ont du mal à déterminer la rentabilité de leurs décisions. Pioneer Natural Resources (PNR) de Las Colina dans l'état du Texas aux États-Unis a décidé de créer un SIAD qui pourrait fournir des informations plus précises à cet égard.

En 1995, les cadres de cette entreprise ont commencé à repérer toutes les variables de gestion et ont tracé des diagrammes de tous les processus de l'entreprise pour créer un modèle qui pourrait montrer les conséquences du changement d'une ou de plusieurs de ces variables. L'entreprise a ensuite construit un prototype de SIAD. Les cadres ont d'abord testé le système en simulant la situation instable de PNR sur la côte du Golfe du Mexique, dont le temps de production était toujours long. Devant l'efficacité du système, les responsables de la société estiment que chacune des cinq divisions de la compagnie pourrait accroître ses revenus de 25 à 40 % en utilisant un outil similaire.

2.6 Conclusion

Dans ce chapitre, les notions qui ont été présentées se réfèrent à la prise de décision et aux systèmes interactifs d'aide à la décision. En résumé, un SIAD peut être vu comme un système permettant à un décideur de prendre une décision en mettant à sa disposition la meilleure information possible. Il assiste le décideur sans jamais se substituer à lui. Le

chapitre suivant présentera un domaine où l'utilisation de SIAD peut s'avérer très utile : l'optimisation multi-objectifs.

CHAPITRE 3

ALGORITHMES GÉNÉTIQUES ET OPTIMISATION MULTI-OBJECTIFS

3.1 Introduction

De nombreux secteurs de l'industrie mécanique, chimique, des télécommunications, de l'environnement, des transports et autres sont concernés par des problèmes complexes, de grande dimension et souvent de nature combinatoire pour lesquels la résolution par des méthodes exactes s'avère souvent difficile. De plus, ce genre de problèmes ont généralement des objectifs fortement dépendants des préférences de l'utilisateur. En fait, les problèmes d'optimisation rencontrés en pratique sont rarement uni-objectif. La plupart d'entre eux nécessitent l'optimisation simultanée de plusieurs objectifs souvent contradictoires. L'optimisation multi-objectifs s'intéresse à la résolution de ce type de problèmes. 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 dans les sciences de la gestion avant d'être graduellement utilisée dans d'autres domaines comme l'informatique ou l'ingénierie.

L'optimisation multi-objectifs vise donc à optimiser simultanément plusieurs objectifs. Contrairement à l'optimisation uni-objectif où la solution optimale est unique et souvent clairement définie, les Problèmes Multi-Objectifs (PMO) présentent un ensemble de solutions dites optimales. Cet ensemble est généralement nommé 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 consiste à choisir une solution à partir de cet ensemble en fonction des préférences exprimées par le décideur. Le choix d'une solution par rapport à une autre nécessite une bonne connaissance du problème et des différents facteurs liés au problème. Ainsi, une

solution choisie par un décideur peut ne pas être acceptable pour un autre décideur. Le choix d'une solution peut aussi être remis en cause dans un environnement dynamique. Il est donc utile d'avoir plusieurs alternatives dans le choix d'une solution PO.

Ce chapitre introduit, dans un premier temps, les concepts de base et les principes 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 génétiques sont ensuite décrits comme une technique récente d'optimisation qui possède des caractéristiques intéressantes pour résoudre un PMO. Un large éventail d'algorithmes génétiques multi-objectifs est présenté en essayant d'apporter un regard critique sur chacun d'eux. Pour terminer, les objectifs de ce travail de recherche sont décrits en relation avec la revue de littérature effectuée dans les Chapitres 2 et 3.

3.2 Optimisation multi-objectifs

3.2.1 Problème 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 sera laissé au décideur.

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

$$PMO \begin{cases} \min\ F(x)=\left(f_1(x),f_2(x),\ldots,f_k(x)\right) \\ s.c.\ x\in C \end{cases} \quad (1)$$

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

3.2.2 Notion de dominance

Soit une solution $x \in E$ (ensemble de solutions), x *domine* une autre solution $x' \in E$ si $\forall i \in [1..k]$ $f_i(x) \leq f_i(x')$ et pour au moins un j tel que $f_j(x) < f_j(x')$. La Figure 3.1 présente les relations de dominance pour deux objectifs de minimisation. Le rectangle gris clair contient les solutions dominant la solution représentée par le point B. Le rectangle gris foncé entoure la région contenant les solutions dominées par B. Toutes les autres solutions ne se trouvant ni dans le rectangle clair, ni dans le rectangle foncé, sont indifférentes à la solution représentée par B.

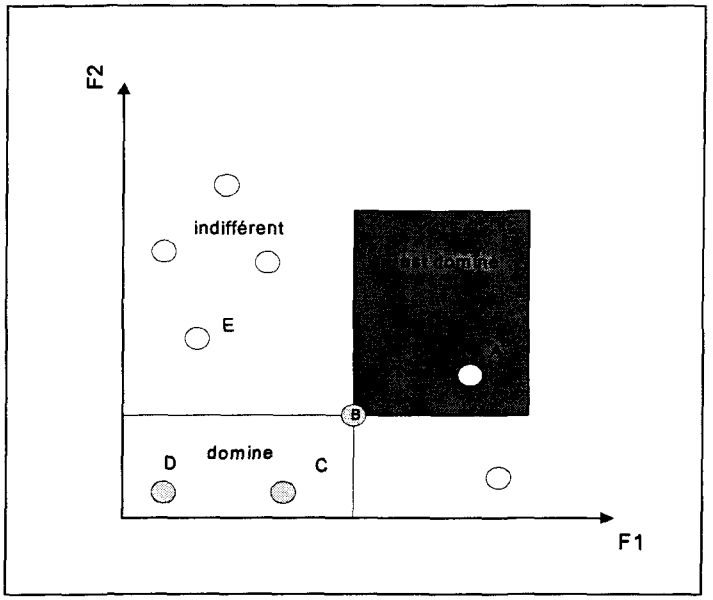


Figure 3.1 : Illustration des relations de dominance entre les solutions pour deux objectifs de minimisation [Zitzler, 2001]

3.2.3 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 $f_i(x) \leq f_i(x^*)$ pour tout $i = 1,\ldots, k$ et qu'il existe au moins un j tel que $f_j(x) < f_j(x^*)$. Ces points sont également appelés solutions *non inférieures* ou *non dominées*. L'ensemble de tous les points Pareto optimaux constitue la *frontière Pareto*.

À la Figure 3.2, 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.

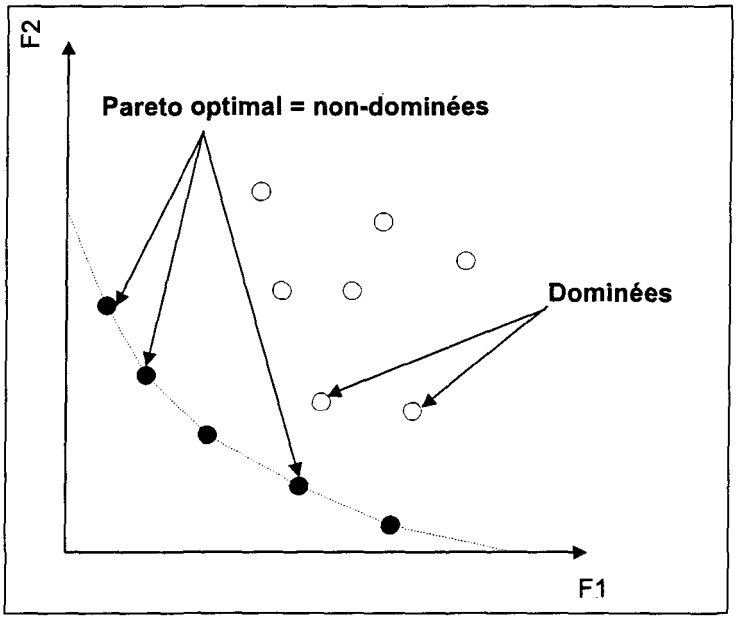


Figure 3.2 : Illustration du concept d'optimum Pareto [Zitzler, 2001]

3.2.4 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 », les décisions sont considérées comme un compromis entre les objectifs et les choix spécifiques du décideur. Dépendamment 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 objectifs en une fonction d'utilité suivant les préférences du décideur. Dans ce cas, le décideur est supposé connaître *a priori* le poids de chaque objectif afin de les combiner dans une fonction unique. Cela revient à résoudre un problème simple objectif. 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.

Dans les méthodes *a posteriori*, la recherche de solutions est effectuée sans qu'aucune information de préférence soit fournie. Le résultat du processus de recherche est un ensemble, idéalement l'ensemble Pareto optimal, de solutions candidates 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 Pareto optimal serait 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 sera obtenue après un nombre fini d'itérations.

La « classification concepteur » qui sera utilisée dans le reste du mémoire 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]:

$$U = U(f_1, f_2, \ldots, f_k). \qquad (2)$$

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

$$U = \sum_{i=1}^{k} U_i(f_i) \qquad (3)$$

et le modèle multiplicatif :

$$U=\prod_{i=1}^{k} U_i(f_i). \qquad (4)$$

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 :

$$\min \sum_{i=1}^{k} w_i f_i(x) \qquad (5)$$

où w_i représente le poids affecté à l'objectif i, $w_i \geq 0$ et $\sum_{i=1}^{k} w_i=1$. 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 T_i à atteindre pour chaque objectif f_i [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 :

$$\min \sum_{i=1}^{k} |f_i(x)-T_i| \qquad (6)$$

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* utilisent 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 seront présentées plus en détail dans la Section 3.6 du présent chapitre.

Classiquement, les PMO sont résolus à l'aide de technique 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 Pareto optimal [Francisci, 2002].

Récemment, les algorithmes génétiques 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 génétiques sont peu sensibles à la forme de la frontière Pareto. Les algorithmes génétiques 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] dont les principes de base sont présentés brièvement à la section suivante.

3.3 Les algorithmes génétiques

Les algorithmes génétiques (AG) 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 algorithmes génétiques 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 reste néanmoins très récente.

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 P est donc un ensemble de N éléments (individus) de E : $P = (x_1, x_2, \ldots, x_n)$. La mesure du degré d'adaptation de chaque individu constitue la fonction de performance F (*fitness*). L'objectif d'un AG est de faire évoluer cette population P afin de trouver le meilleur individu x^*. Dans ce but, à chaque génération t, les

individus de la population $P(t)$ 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 du mémoire.

3.3.1 Représentation des individus

Dans la nature, les structures géniques sont codées en base 4, dont les « chiffres » sont les quatre bases azotées : l'adénine (A), la thymine (T), la cytosine (C) et la guanine (G). Dans le cadre des AG, ce type de codage est bien difficile à utiliser et n'a donc pas été retenu. Nous présenterons 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 :

$$x = (a_1, a_2, \ldots, a_L) \in \{0,1\}^L, \qquad (7)$$

Où L est la taille du vecteur (nombre de bits). L'espace de recherche est alors $\{0,1\}^L$. Ce type de représentation présente 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 a_i. 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.

3.3.2 La sélection

Comme son nom l'indique, la sélection permet d'identifier les meilleurs individus d'une population et d'éliminer partiellement les mauvais. C'est un des principaux opérateurs utilisé 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, les paragraphes suivants décriront 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 performance comme l'illustre la Figure 3.3. La roue étant lancée, l'individu sélectionné est celui sur lequel la roue s'est arrêté. Cette méthode favorise les meilleurs individus, mais tous les individus conservent néanmoins des chances d'être sélectionnés.

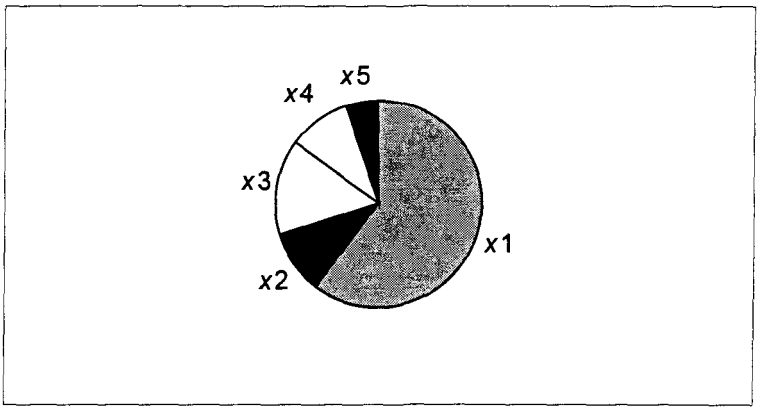


Figure 3.3 : Roulette pour une population de 5 individus avec $f(x_i) = \{60{,}10, 15{,}10, 5\}$. Pour tirer un individu, on lance la roue, et si elle s'arrête sur le segment i, x_i est sélectionné

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, de sélectionner d'une manière déterministe le meilleur dans ce groupe, et de recommencer l'opération jusqu'à l'obtention du nombre d'individus requis. Au cours d'une génération, il y 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.

3.3.3 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 N parents pour produire K 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 les paragraphes suivants, nous allons considérer trois opérateurs de croisement : le *croisement à un point*, le *croisement partiellement tracé (PMX)* et *le croisement par recombinaison d'arêtes (ER).*

Le *croisement à un point* est une des premières stratégies de croisement utilisées par les AG [Holland, 1962]. Son but est d'échanger un fragment de gènes entre deux individus. Il 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. Notons que le croisement s'effectue directement au niveau binaire, et non pas au niveau des gènes. Un chromosome peut donc être coupé au milieu d'un gène. À la Figure 3.4, le point de coupure entre les deux parents se situe à la position 5.

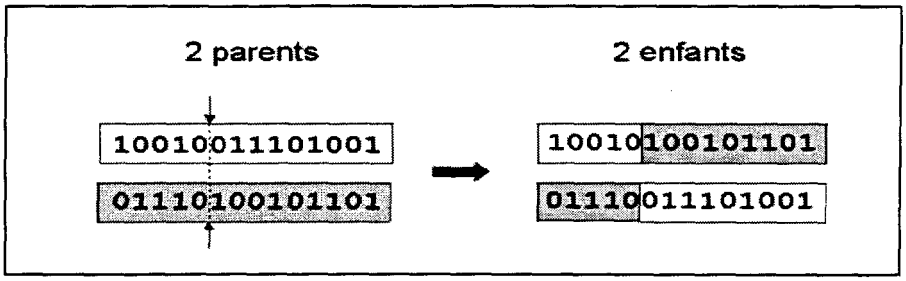


Figure 3.4 : Illustration du croisement à un point

Le *PMX* est une technique introduite par Goldberg et Lingle en 1985 pour le problème du voyageur de commerce. Cet opérateur de croisement sélectionne dans un premier temps deux points de coupure sur chaque parent. Afin de créer un enfant, la sous-chaîne comprise entre les deux points de coupure du premier parent remplace la sous-chaîne correspondante dans le deuxième parent. Ensuite, un remplacement inverse est effectué à l'extérieur des deux points de coupure de manière à éliminer les doublons et restaurer tous les gènes. La Figure 3.5 illustre la création d'un enfant par cette technique. Dans un premier temps, la sous-chaîne 236 du parent 2 est remplacée par la sous-chaîne 564 provenant du parent 1 (étape 1). Comme les gènes 4 et 5 sont maintenant dupliqués dans l'enfant, ils seront remplacés par les gènes compris entre les points de coupure du parent 2 qui ne sont pas encore présents dans l'enfant. Plus précisément, le gène 2 remplacera le gène 5 et le gène 3 remplacera le 4 (étape 2).

Parent 1 :	1 2 | **5 6 4** | 3 8 7
Parent 2 :	1 4 | 2 3 6 | 5 7 8
Enfant(étape 1) :	1 4 | **5 6 4** | 5 7 8
Enfant(étape 2) :	1 3 | **5 6 4** | 2 7 8

Figure 3.5 : Illustration du PMX

Cette technique de croisement essaie de préserver la position absolue des gènes. Dans les faits, le nombre de gènes qui ne vont pas conserver la position qu'ils avaient 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, seuls les gènes 2 et 3 ne conservent pas leurs positions absolues de l'un des deux parents.

L'opérateur de *croisement par recombinaison d'arêtes* (*ER*), introduit par Whitley *et al.* [1989], tente de préserver les plus d'arêtes possibles provenant des parents de façon à minimiser l'introduction d'arêtes additionnelles. Pour cela, une « carte » d'arêtes contenant toutes les connexions entre les différents gènes est construite pour les deux parents. La présence d'arêtes indique une connexion entre deux gènes. Pour construire un enfant, un gène courant est choisi entre les gènes de départ des parents. La carte d'arêtes est alors utilisée pour choisir le prochain gène courant. Parmi les gènes liés au gène courant dans un des parents, on choisit celui ayant le moins de liens disponibles dans la carte d'arêtes. En cas d'égalité entre deux gènes ou plus, on en choisit un aléatoirement. Lorsqu'un gène est choisi, la carte d'arêtes est mise à jour. La Figure 3.6 illustre la carte d'arêtes initiale pour deux individus x_1 = 13564287 et x_2 = 14236578 dans le cas d'un problème de voyageur de commerce [Potvin, 1996].

Gène 1 a des arêtes vers : 3 4 7 8
Gène 2 a des arêtes vers : 3 4 8
Gène 3 a des arêtes vers : 1 2 5 6
Gène 4 a des arêtes vers : 1 2 6
Gène 5 a des arêtes vers : 3 6 7
Gène 6 a des arêtes vers : 3 4 5
Gène 7 a des arêtes vers : 1 5 8
Gène 8 a des arêtes vers : 1 2 7

Figure 3.6 : Carte d'arêtes initiale

Les opérations successives de cet opérateur de croisement sont illustrées à la Figure 3.7 [Potvin, 1996] à partir de la carte d'arêtes initiale de la Figure 3.6. Si on suppose que le gène 1 est choisi comme point de départ, toutes les arêtes incidentes au gène 1 doivent être effacées de la carte d'arêtes initiale. À partir du gène 1, on peut choisir les gènes 3, 4, 7 ou 8. Le gène 3 a trois arêtes actives tandis que les gènes 4, 7 et 8 ont en deux comme illustré par la carte d'arête (a) de la Figure 3.7. Un gène est alors choisi aléatoirement entre les gènes 4, 7 et 8. Supposons que c'est le gène 8 qui est choisi. À partir du gène 8, on peut choisir les gènes 2 ou 7. Comme l'indique la carte d'arêtes (b), le gène 2 a deux arêtes actives alors que le gène 7 en a une seule, donc c'est lui qui sera choisi. À partir du gène 7, seul le gène 5 peut être choisi. De ce point, la carte d'arête (d) offre le choix entre les gènes 3 et 6 qui ont chacun deux arêtes actives. Si c'est le gène 6 qui est retenu, on peut choisir entre les gènes 3 et 4 qui ont chacun une arête active. Supposons que c'est le gène 4 qui est choisi aléatoirement. À partir du gène 4, seul le gène 2 peut être sélectionné et, à partir de ce point, le gène 3 doit être choisi. L'enfant généré à partir de x_1 et x_2 avec cette technique sera x'= 18756423.

Le gène 1 est choisi

(a) Gène 2 a des arêtes vers : 3 4 8
Gène 3 a des arêtes vers : 2 5 6
Gène 4 a des arêtes vers : 2 6
Gène 5 a des arêtes vers : 3 6 7
Gène 6 a des arêtes vers : 3 4 5
Gène 7 a des arêtes vers : 5 8
Gène 8 a des arêtes vers : 2 7

Le gène 8 est choisi

(b) Gène 2 a des arêtes vers : 3 4
Gène 3 a des arêtes vers : 2 5 6
Gène 4 a des arêtes vers : 2 6
Gène 5 a des arêtes vers : 3 6 7
Gène 6 a des arêtes vers : 3 4 5
Gène 7 a des arêtes vers : 5

Le gène 7 est choisi

(c) Gène 2 a des arêtes vers : 3 4
Gène 3 a des arêtes vers : 2 5 6
Gène 4 a des arêtes vers : 2 6
Gène 5 a des arêtes vers : 3 6
Gène 6 a des arêtes vers : 3 4 5

Le gène 5 est choisi

(d) Gène 2 a des arêtes vers : 3 4
Gène 3 a des arêtes vers : 2 6
Gène 4 a des arêtes vers : 2 6
Gène 6 a des arêtes vers : 3 4

Le gène 6 est choisi

(e) Gène 2 a des arêtes vers : 3 4
Gène 3 a des arêtes vers : 2
Gène 4 a des arêtes vers : 2

Le gène 4 est choisi

(f) Gène 2 a des arêtes vers : 3
Gène 3 a des arêtes vers : 2

Le gène 2 est choisi

(e) Gène 3 a des arêtes vers :

Le gène 3 est choisi

Figure 3.7: Évolution de la carte d'arêtes

3.3.4 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 3.8, la position 9 est sélectionnée et le gène est inversé, il passe de 1 à 0.

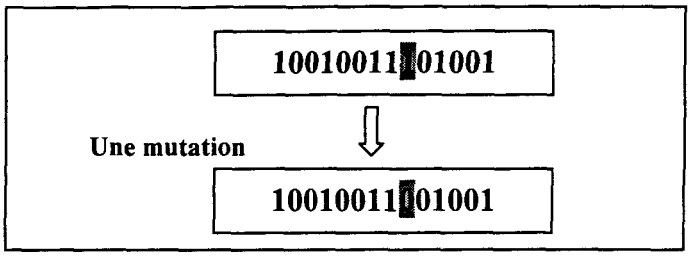


Figure 3.8 : Illustration de la mutation par inversion

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 3.9, les gènes aux positions 4 et 6 sont échangés.

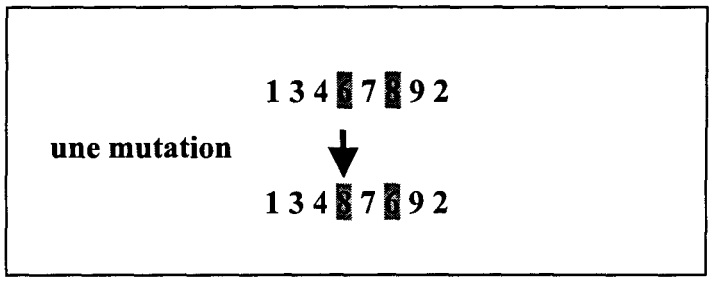


Figure 3.9 : Illustration de la mutation par échange

3.3.5 Les grandes lignes de l'algorithme

Le fonctionnement général d'un AG est illustré à la Figure 3.10.

Générer aléatoirement une population de *n* individus
TANT QUE Nombre de générations ≤ Nombre de générations Maximum **FAIRE**
Évaluation : Calculer la performance de chaque individu
Sélection : Établir de façon probabiliste les paires d'individus qui vont se reproduire en accordant une meilleure chance aux meilleurs individus
Reproduction : Appliquer les opérateurs génétiques aux paires sélectionnées. Les nouveaux individus produits représentent la nouvelle population

Figure 3.10 : Un algorithme génétique

L'AG commence en générant aléatoirement une population initiale de n individus. Ensuite, l'algorithme passe d'une génération à une autre en appliquant les mécanismes d'évaluation de sélection et de reproduction jusqu'à l'obtention d'un critère d'arrêt.

3.3.6 Le nichage

Une des principales difficultés à surmonter lorsque l'on veut utiliser un algorithme génétique 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 performance* et la *méthode de remplacement*.

Le *partage de performance* est une technique introduite par Goldberg et Richardson [1987]. Dans cette méthode, la performance d'un individu va être 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 performance partagée f' d'un individu x_i est donnée par :

$$f'(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} sh\ (d(x_i, x_j))} \qquad (8)$$

où *sh* désigne la fonction de partage qui a comme paramètre d'entrée la distance *d* 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] :

$$sh(d)=\begin{cases} 1-\left(\frac{d}{\sigma_{sh}}\right)^{\alpha} & si\ d(x,y)<\sigma_{sh} \\ 0 & \text{sinon} \end{cases} \qquad (9)$$

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 *d* peut se faire à l'aide de distance de Hamming ou des distances Euclidiennes.

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 AG que le partage de performance [Blickle et Thiele, 1996].

3.3.6 AG et optimisation multi-objectifs

C'est dans le milieu des années 80 [Schaffer, 1985] que, pour la première fois, les AG ont été utilisés pour l'optimisation multi-objectifs. Depuis, plusieurs implémentations différentes des AG 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 AG 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 ont été développées [Zitzler et Thiele, 1998; Knowles et Corne, 2000; Coello Coello et Pulido, 2001] . Les sections suivantes présentent un éventail des méthodes de résolution de PMO basées sur les AG en les regroupant selon la « classification concepteur » décrite à la section 3.2.4.

3.4 Les approches basées 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], par exemple, 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].

3.5 Les approches 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 3.11, l'idée est simple. Si nous avons k objectifs et une population de n individus, une sélection de n/k individus est effectuée pour chaque objectif. Ainsi, k sous-populations vont être créées, chacune d'entre elles contenant les n/k meilleurs individus pour un objectif particulier. Les k sous-populations sont ensuite mélangées afin d'obtenir une nouvelle population de taille n. Le processus de sélection se termine par l'application des opérateurs génétiques de croisement et de mutation.

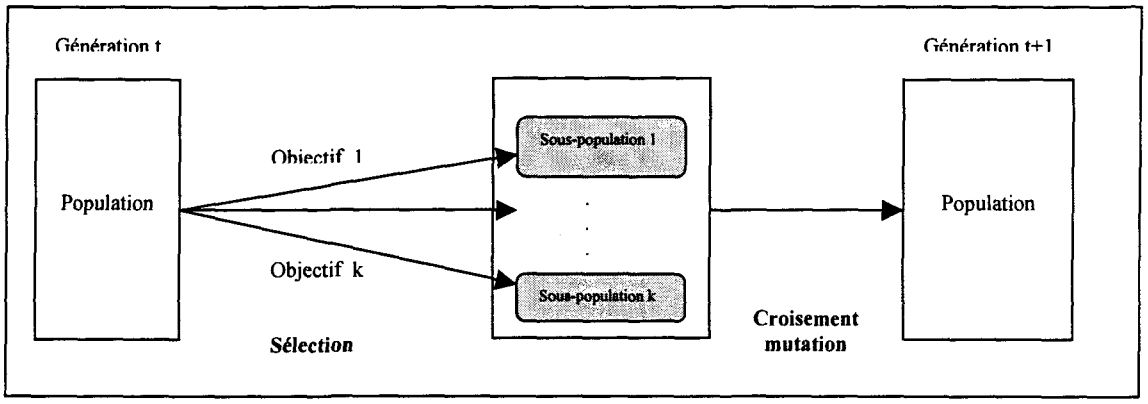


Figure 3.11 : Sélection parallèle dans l'algorithme VEGA [Talbi, 1999]

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 performance générale acceptable mais ne possèdent aucun objectif fort [Talbi, 1999].

3.6 Les approches Pareto

L'idée d'utiliser la dominance au sens de Pareto a été proposée par Goldberg [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 génétiques et utilisant cette notion. Cette présentation sera 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.

3.6.2 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.

3.6.2.1 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 performance 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 $p(t)$ individus. Le rang de cet individu est :

$$\text{rang}(x, t) = 1 + p(t) \qquad (10)$$

La procédure de sélection utilise ensuite ces rangs pour sélectionner ou éliminer des blocs d'individus.

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 performance afin de mieux répartir les solutions le long de la frontière Pareto. Mais les performances de l'algorithme sont dépendantes de la valeur du paramètre σ_{sh} utilisé dans la fonction de partage.

3.6.2.2 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 performance 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 performance 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}.

3.6.2.3 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 performance 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 performance que ceux du Front 2 qui eux auront une meilleure performance que ceux du Front 3 et ainsi de suite. Pour la sélection, elle applique ensuite le partage de performance au niveau de chaque front pour maintenir la diversité. La Figure

3.12 montre la classification d'une population par front dans un contexte de minimisation. L'ensemble des solutions non dominées est représenté par les solutions du front 1.

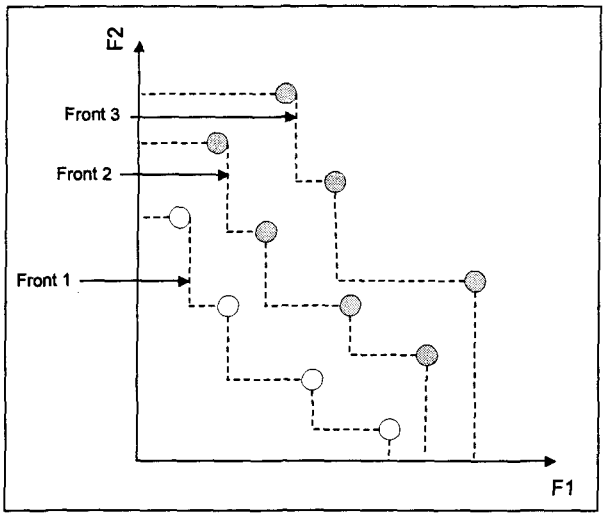


Figure 3.12 : Illustration du classement par front dans le NSGA [Zitzler *et al.*, 1999]

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 performance est important ($O(kN^3)$ avec k: le nombre d'objectifs et N: 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 de 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.

3.6.3 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.

3.6.3.1 NSGAII

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

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 $O(kN^2)$.

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 performance 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 premier front et ainsi de suite;

- $i_{distance}$ 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 3.13, pour estimer la densité au voisinage d'une solution i, on calcule la distance moyenne sur chaque objectif, entre les deux points les plus proche 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 i sans inclure un autre point de la population. L'algorithme de calcul est de complexité $O(kN\log(N))$. Cette distance de remplacement va être utilisée pour guider le processus de sélection.

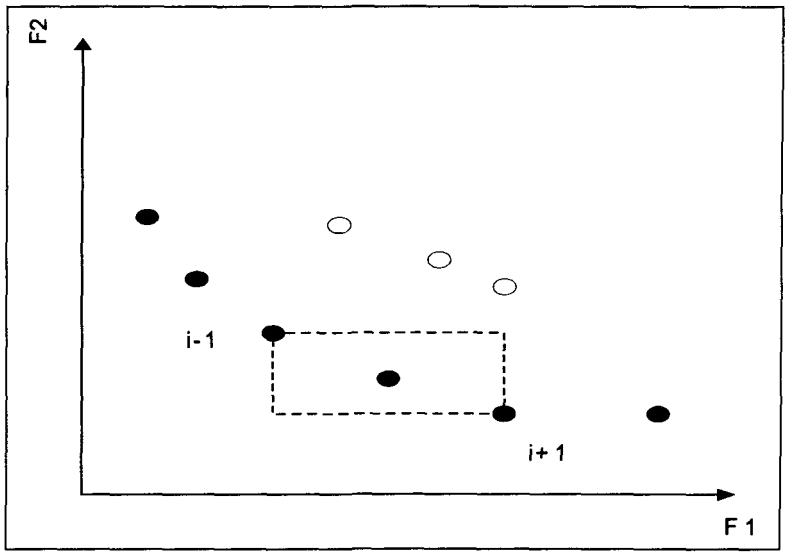


Figure 3.13 : Illustration du calcul de la $i_{distance}$ du NSGAII [Deb, 2000]

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. Mais 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 3.14 illustre le fonctionnement du NSGAII. À chaque itération t de l'algorithme, une population d'enfant Q_t de taille N est générée à partir d'une population P_t de parents de taille identique. Les deux populations sont alors combinées dans une population R_t de taille $2N$. 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 P_{t+1} de taille N est créée à partir des solutions des différents fronts de R_t. On commence avec les solutions du premier front et ainsi de suite. Comme la taille de R_t est de $2N$ et que la taille de la nouvelle population est juste de N, tous les fronts ne pourront pas être contenus dans P_{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.

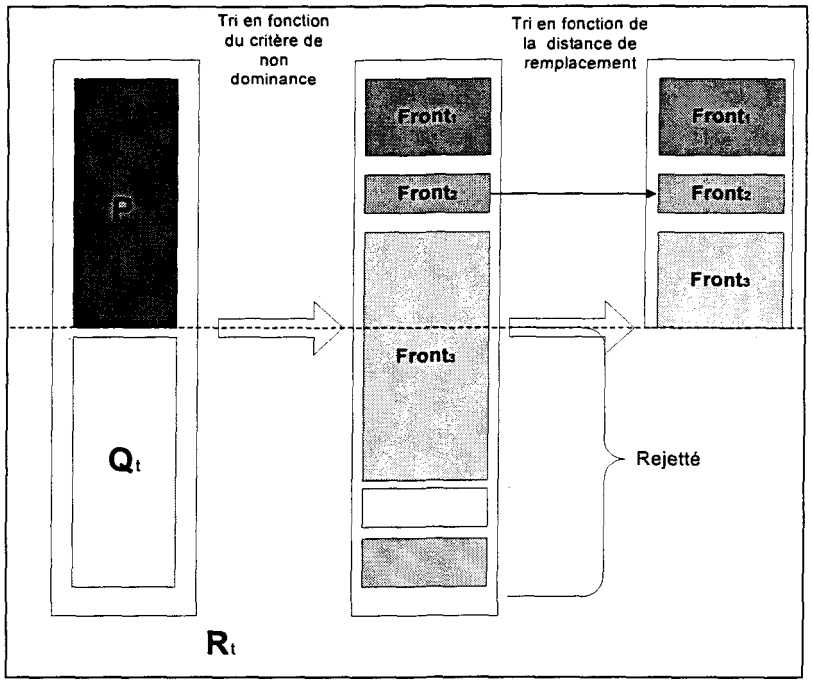


Figure 3.14 : Illustration du fonctionnement du NSGAII [Deb, 2001]

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

3.6.3.2 Strength Pareto Evolutionary Algorithm (SPEA)

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 N, le SPEA utilise une population externe, appelée archive, de taille M 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 performance 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.

La mise à jour de la performance s'effectue en deux étapes : dans un premier temps, on calcule pour chaque individu i de l'archive une force, S_i, correspondant au nombre de solutions de la population qu'il domine, n_i, divisé par la taille de la population plus un.

$$S_i = \frac{n_i}{N+1} \qquad (11)$$

La force S_i représente en même temps la performance F_i des individus de l'archive.
Dans un deuxième temps, on calcule la performance, F_j, d'un individu j de la population en additionnant les S_i de tous les membres de l'archive qui dominent ou qui sont égales à j, et en ajoutant un à la fin.

$$F_i = 1 + \sum S_i \qquad (12)$$

Cette méthode distribue efficacement les solutions sur la frontière de Pareto. La technique utilisée pour le calcul de la performance permet de bien échantillonner les individus dans l'espace. De plus, le concept de force associé à la technique de regroupement entraîne la formation de niches où la performance des individus dépend de leur position par rapport aux individus Pareto optimaux [Zitzler *et al.*, 1999]. Cependant, le calcul de la performance dépend de la taille de l'archive. Dans le cas où l'archive ne contiendrait qu'un individu, tous les membres de la population auraient la même performance. Dans ce cas, le

SPEA se comporterait comme un algorithme de recherche aléatoire. Par ailleurs, la méthode de regroupement utilisée pour réduire l'archive peut occasionner la perte de solutions non dominées [Deb, 2001].

3.6.3.3 SPEA 2

Une version révisée du SPEA a été récemment proposée pour corriger les lacunes de l'ancienne version : 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 performance, 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 performances identiques, le mécanisme permettant l'assignation de la performance 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 i de l'archive $\overline{P}$ de taille $\overline{N}$ et de la population P_t de taille N, on calcule la force $S(i)$ représentant le nombre de solutions qu'il domine. La performance brute $R(i)$ d'un individu i correspondant à la somme des forces des individus qui dominent i.

Bien que la performance 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 i et tous les individus j de l'archive et de la population est calculée et stockée dans une liste. Après avoir trié la liste en ordre croissant,

le $k^{ième}$ élément donne la distance σ_i^k recherchée. Dans l'algorithme, les auteurs utilisent $k=\sqrt{N+\overline{N}}$. La densité $D(i)$ correspondant à i est donc :

$$D(i)=\frac{1}{\sigma_i^k+2} \qquad (13)$$

Au dénominateur, on ajoute deux pour s'assurer que celui-ci sera supérieur à zéro et que $D(i)<1$. Finalement, en ajoutant $D(i)$ à la performance brute $R(i)$ on obtient la performance $F(i)$ de l'individu i.

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 3.15 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 est 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.

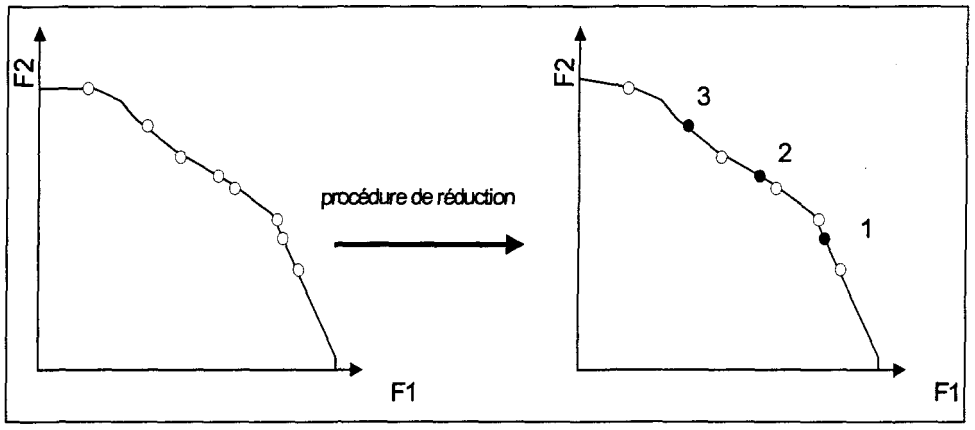


Figure 3.15 : Illustration de la méthode de réduction de l'archive utilisé par le SPEA 2 [Zitzler *et al.*, 2001].

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 Objectifs de la recherche

Les Chapitres 2 et 3 ont permis d'introduire différentes notions sur les SIAD, l'optimisation multi-objectifs ainsi que différentes méthodes comme les AG permettant de résoudre ce type de problème d'optimisation. L'objectif général de ce travail de recherche est de spécifier et concevoir un SIAD pour l'optimisation multi-objectifs afin de mieux comprendre les bénéfices que ce type de systèmes peut apporter dans la résolution de PMO. La mise en œuvre du SIAD sera illustrée à partir d'un contexte réel d'ordonnancement industriel. Les objectifs spécifiques de cette recherche sont :

a. de développer un algorithme génétique multi-objectifs efficace pour le problème d'ordonnancement industriel qui permet d'approximer l'ensemble PO en une seule exécution;
b. concevoir un SIAD cohérent doté d'une interface graphique conviviale permettant de guider le décideur lors de son processus de prise de décision.

La démarche utilisée dans ce travail consiste premièrement à concevoir un algorithme génétique uni-objectif pour la résolution du problème d'ordonnancement industriel afin de démontrer son efficacité et de justifier le choix de cette méthode de résolution. Deuxièmement, implémenter l'algorithme du NSGAII pour résoudre ce même problème. Cet algorithme a été choisi car il ne nécessite pas le réglage de nombreux paramètres et il

semble bien se comporter sur différents PMO [Deb, 2000; Zitzler *et al.*, 2001]. Par la suite, en s'inspirant notamment du NSGAII et du SPEA 2, concevoir un algorithme génétique multi-objectifs pour le contexte étudié. Pour en vérifier la performance, les résultats de cet algorithme seront évalués en relation avec les ensembles de références et les résultats obtenus par le NSGAII. Finalement, les différents éléments de l'interface graphique seront développés et intégrés avec les autres modules afin d'obtenir un ensemble cohérent.

Plus spécifiquement, le système devra comporter :

a. Une interface graphique facile d'utilisation et conviviale afin de permettre une prise en main rapide par l'utilisateur;
b. Une base de modèle constituée d'algorithmes génétiques qui fonctionnera en deux temps :
    - dans un premier temps, il s'agira d'approximer de manière efficace la frontière Pareto afin de fournir un ensemble de solutions au décideur et
    - dans un deuxième temps, une recherche sera effectuée dans une région de la frontière en fonction des préférences exprimées par le décideur.

3.8 Conclusion

Dans les chapitres 2 et 3, les notions fondamentales sur les SIAD et l'optimisation multi-objectifs ainsi qu'un état de l'art des méthodes de résolution de PMO ont été présentés. Les objectifs de recherche de ce travail ont également été définis. La suite de ce travail sera

consacrée à la conception d'un système interactif d'aide à la décision basé sur des algorithmes génétiques pour la résolution d'un problème d'ordonnancement industriel. La performance du système sera donc étudiée dans ce contexte réel.

CHAPITRE 4

BASE DE MODÈLE DU SIAD : AG MULTI-OBJECTIFS POUR UN CONTEXTE D'ORDONNANCEMENT INDUSTRIEL

4.1 Introduction

Les problèmes d'ordonnancement sont des problèmes d'optimisation combinatoire dont la complexité est maintenant établie. Pour la majorité de ce ceux-ci, il n'existe pas d'algorithmes connus pour les résoudre de façon optimale dans un temps polynomial ce qui en fait des problèmes définis *NP-difficiles* [Garey et Johnson, 1979; Chapman, 1987]. Ces problèmes ont fait l'objet de nombreux travaux dans le domaine de la recherche opérationnelle. Une des principales raisons justifiant cet effort s'explique par le fait que de nombreuses situations pratiques correspondent à des problèmes d'ordonnancement pour lesquels il serait intéressant d'appliquer les méthodes de résolution découlant des divers travaux sur le sujet. Cependant, plusieurs auteurs ont noté [Allahverdi *et al.*, 1999; Yang et Liao, 1999], en recensant la littérature, que les chercheurs s'attardaient principalement à des contextes de base. De plus, la majorité des travaux en ordonnancement traite généralement ces problèmes sur la base de l'optimisation d'un objectif unique. Cependant, les situations pratiques sont souvent plus complexes que les problèmes théoriques classiques et correspondent rarement à l'optimisation d'un seul objectif. Pour de tels contextes, des méthodes de résolutions à la fois rapides, efficaces et flexibles s'avèrent d'une grande utilité.

Les métaheuristiques se sont avérées des stratégies de résolution particulièrement performantes autant pour les problèmes d'optimisation combinatoire en général que pour les problèmes d'ordonnancement théoriques et pratiques. Par exemple, dans Gravel et *al.* [2002], les auteurs présentent une méthode basée sur la métaheuristique d'Optimisation par Colonies de Fourmis (OCF) pour la résolution d'un problème réel d'ordonnancement multi-

objectifs rencontré dans l'industrie de production de l'aluminium. Des travaux supplémentaires ont permis d'intégrer de nouveaux éléments à la métaheuristique d'OCF et leur efficacité a été démontrée dans Gagné *et al.* [2002a] à l'aide de problèmes théoriques de nature similaire au problème industriel. Ces ajouts ont par la suite été adaptés à l'OCF afin de résoudre le problème industriel [Gagné *et al.*, 2002b]. Certaines de ces approches ont été implantées dans un logiciel utilisé dans plusieurs usines de l'entreprise.

Toutefois, même si cet algorithme s'est avéré une solution efficace au problème, l'approche utilisée considère les objectifs selon un ordre lexicographique. Comme mentionnée à la Section 3.5 du chapitre précédent, cette manière d'aborder le problème a le désavantage de diriger la recherche dans une direction extrême et de considérer les autres objectifs seulement en cas d'égalité. Pour éviter cet inconvénient, les auteurs ont proposé une approche générique de recherche de compromis. Un exemple d'application de cette procédure a été présenté à l'aide de la recherche avec tabous hybridé avec la recherche par voisinage variable pour traiter un problème d'ordonnancement bi-objectifs sur une machine unique avec réglages dépendant de la séquence [Gagné *et al.*, 2004a]. Cette procédure générique a aussi été intégrée à l'OCF afin de résoudre le problème d'ordonnancement industriel [Gagné *et al.*, 2004b]. Cependant, cette approche nécessite plusieurs étapes d'optimisation pour obtenir une approximation de l'ensemble Pareto optimale.

Comme il a été expliqué au chapitre précédent, les métaheuristiques et, plus particulièrement, les algorithmes génétiques se sont révélés être des approches particulièrement intéressantes pour la résolution de PMO grâce à leur faculté à exploiter de vastes espaces de recherche et à générer des compromis multiples en une seule étape

d'optimisation. Pour ces raisons, il s'avère intéressant d'étudier l'apport de ce type d'algorithme pour la résolution du problème d'ordonnancement industriel décrit par Gravel *et al.*[2002].

Ce chapitre présente la démarche et les résultats de l'étude effectuée. Les approches de résolution présentées dans cette partie sont des composantes importantes du SIAD, car elles constituent la base de modèle. En premier lieu, le problème d'ordonnancement industriel sera décrit sommairement pour assurer la compréhension du lecteur. Par la suite, un algorithme génétique uni-objectif pour la résolution du problème industriel est proposé. Le reste du chapitre étant consacré à l'adaptation, au développement de trois stratégies de résolution du problème d'un point de vue multi-objectifs basées sur les algorithmes génétiques et à la présentation des résultats obtenus.

4.2 Exemple d'application : un problème d'ordonnancement industriel

Le problème traité dans ce travail est un problème réel d'ordonnancement industriel rencontré dans une aluminerie. Pour une période de production donnée, un carnet de commandes de lingots possédant différentes caractéristiques doit être produit en usine sur une machine à coulée horizontale qui est illustrée à la Figure 4.1. Lors de la production d'une commande, la machine à coulée est alimentée par les fours de préparation. Ceux-ci sont utilisés pour établir le mélange désiré en ajoutant différents ingrédients d'alliage à de l'aluminium pur de manière à obtenir les caractéristiques spécifiques du produit. Le moule utilisé permet de donner la dimension voulue au lingot et ce dernier est découpé, à l'aide

d'une scie, en pièces de la longueur désirée à la fin du processus de production. La machine à coulée traite les commandes en série, l'une à la suite de l'autre.

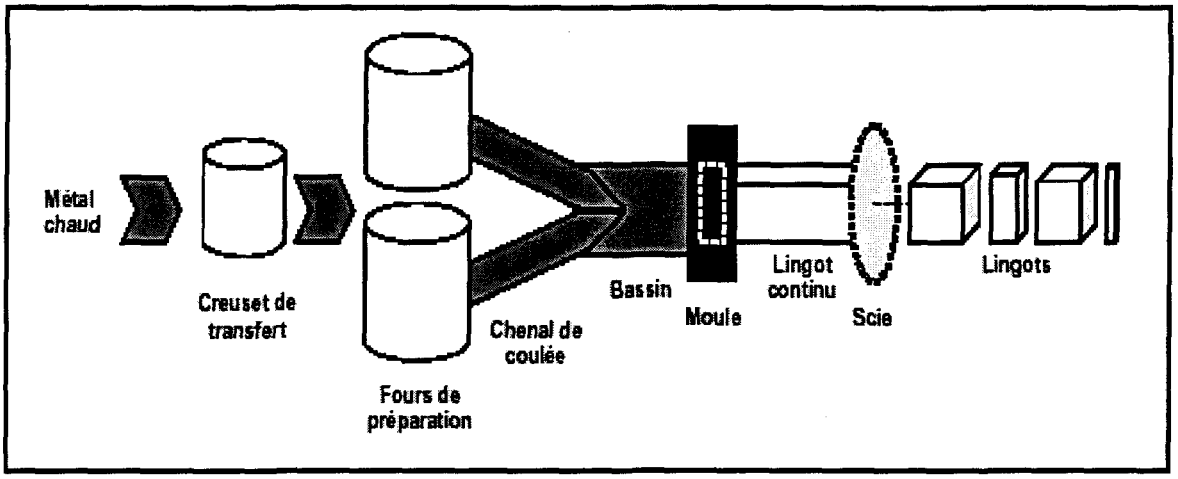


Figure 4.1 : Le processus de coulée horizontale [Gagné, 2001]

La fabrication successive de deux commandes d'alliages différents peut, dans certains cas, entraîner des procédures de drainage et de nettoyage des fours de préparation. De plus, la fabrication successive de deux commandes de lingots de dimensions différentes nécessite un arrêt de la machine à couler pour réaliser un changement de moule. Ces procédures, appelées réglages, représentent des pertes de capacité de production et doivent être autant que possible évitées. De plus, ces deux types de réglages sont dépendants de la séquence de production.

Le processus de production des commandes doit être planifié de façon à réduire le plus possible les pertes de capacité du système de production qui correspond aux temps de réglages et aux temps morts, mais aussi le retard des commandes. Chaque commande doit être livrée à son client respectif à une certaine date fixée préalablement. Un troisième

objectif est également visé et cherche à maximiser la capacité de livraison en regroupant les commandes d'une même destination tout en respectant une capacité cible du moyen de transport utilisé.

Ce problème consiste donc à ordonnancer un carnet de n commandes, c'est-à-dire à déterminer l'ordre dans lequel les commandes vont être traitées sur la machine à couler, de façon à minimiser les trois objectifs que sont la perte de capacité de la machine à couler (f_1), le retard total de l'ensemble des commandes (f_2) et la perte de capacité de livraison (f_3). Nous retrouvons ainsi le vecteur suivant d'objectifs à optimiser :

$$\textit{minimiser}\ \{f_1(x), f_2(x), f_3(x)\} \qquad (14)$$

Il faut noter que les objectifs de minimisation de la perte de capacité de livraison et de retard total sont mesurés en jours tandis que la perte de capacité de livraison est mesurée en tonnes métriques. Cette situation industrielle a été associée à un problème théorique d'ordonnancement de n commandes sur une machine unique avec temps de réglages dépendants de la séquence. Du et Leung [1990] ont démontré que ce problème classique est NP-difficile [Garey et Johnson, 1979] lorsque les temps de réglages sont dépendants de la séquence. Le problème industriel s'avère donc davantage complexe en raison des deux types de réglages et des différentes contraintes technologiques qui lui sont spécifiques.

Plusieurs travaux [Gagné, 2001; Gravel *et al.*, 2002; Gagné *et al.*, 2002b], incluant une thèse de doctorat, ont mené à l'élaboration d'une méthode de résolution performante pour ce problème multi-objectifs. Cette méthode de résolution servira, dans la suite de ce travail, comme base de comparaison pour la version uni-objectif de l'AG présenté dans la section suivante.

4. 3 Traitement uni-objectif à l'aide d'un algorithme génétique (AG^U)

La première composante de la base de modèle du SIAD est un AG uni-objectif pour le problème d'ordonnancement industriel. Cette section présente l'AG proposé ainsi que les résultats obtenus. Il sera ainsi démontré que cet algorithme permet de résoudre efficacement le problème et ainsi de justifier le choix de cette approche comme technique de résolution.

4.3.1 Description de l'algorithme

Comme il a été expliqué au chapitre précédent, les algorithmes génétiques suivent tous, en général, un principe commun et la présente implantation n'y fait pas défaut : une phase de constitution de la population initiale suivie de phases de génération de nouvelles populations grâce aux opérateurs génétiques de sélection, de mutation et de croisement. Chaque génération produit, en principe, un ordonnancement de commandes plus performant que les précédents. Plus le nombre de générations évolue, plus les solutions trouvées par l'algorithme se raffinent et la dernière génération devrait contenir une « bonne » solution, même si on ne peut garantir son optimalité.

Pendant le processus de création de l'algorithme proposé, deux populations existent en parallèle : la population courante de taille *N*, aussi appelée population *Parent*, et la nouvelle génération en voie de réalisation ou population *Enfant,* elle aussi de taille *N*. À la fin de chaque génération, les populations *Parent* et *Enfant* sont combinées, triées et seuls les *N* meilleurs individus sont conservés pour former la prochaine génération de *Parent*. La Figure 4.2 présente un résumé des différentes étapes de l'algorithme génétique uni-objectif

conçu pour résoudre la problématique de l'ordonnancement industriel et noté AG^U dans la suite de ce travail. Les définitions et la notation utilisée sont précisées au Tableau 4.1.

```
Initialiser les différentes variables de l'algorithme
Générer aléatoirement la population initiale P0 de taille N
Effectuer une amélioration locale pour chaque individu x ∈ P0
Trier P0
Mettre à jour le meilleur individu α
Tant que critère d'arrêt non rencontré faire
    Sélectionner dans Pt et création de Qt par application des opérateurs de croisement et
    de mutation
    Effectuer une amélioration locale pour chaque individu x ∈ Qt
    Combiner les deux populations de Qt et Pt et inclure les N meilleurs individus dans la
    prochaine génération Pt+1
    Mettre à jour le meilleur individu α
FinTantQue
Effectuer une amélioration complète sur α
Retourner α
```

Figure 4.2 : Pseudo-code de l'algorithme génétique uni-objectif (AG^U)

P_t	: Population parent à l'instant t
Q_t	: Population enfant à l'instant t
N	: Taille des populations P et Q
α	: Meilleur individu trouvé par l'algorithme
x	: individu

Tableau 4.1 : Notation de l'AG^U

Le problème d'illustration choisi est un problème d'ordonnancement, une représentation classique sous forme de chaîne de bits s'avère donc mal adaptée. Une manière naturelle de représenter les solutions d'un problème d'ordonnancement est la représentation par chemin présentée à la Section 3.3.1. C'est cette représentation qui a été utilisée dans l'AG^U.

L'algorithme commence par l'initialisation des différentes variables. Par la suite, la population initiale est générée de manière aléatoire en veillant à ce que les individus

produits soient des solutions valides. C'est-à-dire que le nombre de gènes de chaque individu est égal au nombre de commandes et chaque commande n'est répétée qu'une et une seule fois dans un individu.

À chaque itération, lors de la phase de génération de la population d'enfants, la première étape effectuée par l'AG^U consiste à sélectionner des individus pour le croisement. La procédure de sélection que nous avons utilisée est une sélection par tournoi, car cette technique de sélection s'avère efficace et n'est pas coûteuse à mettre en œuvre et à exécuter. Cette procédure est décrite plus en détail à la section 3.3.2 du chapitre précédent.

Une fois sélectionnés, les individus sont croisés selon une certaine probabilité (p_c). Deux opérateurs de croisement ont été retenus dans l'AG^U : le PMX et l'ER. La combinaison de ces deux opérateurs permet d'introduire une certaine diversité dans la population. Le premier exploite les similarités au niveau de la position relative des gènes alors que le deuxième tente de préserver le plus d'arcs possible des deux parents de façon à minimiser l'introduction d'arcs additionnels. Le fonctionnement de ces deux techniques de croisement est décrit plus en détail à la Section 3.3.3 du chapitre précédent. Chaque méthode a une chance sur deux d'être utilisée par l'algorithme. Comme l'opérateur ER ne produit qu'un enfant et le PMX en produit deux, lorsque que c'est le PMX qui est choisi, nous ne conservons que le meilleur des deux enfants.

Après avoir été généré par croisement, les enfants subissent une mutation selon une certaine probabilité (p_m). Dans l'AG^U, l'opérateur de mutation retenu est la mutation par échange. Cette technique de mutation est décrite plus en détail à la Section 3.3.4. Il est

important de préciser que dans notre implémentation le nombre de gènes à échanger est proportionnel à la taille du problème à résoudre.

Une façon d'améliorer les performances d'une métaheuristique ou de combler certaines de ses lacunes consiste souvent à la combiner avec une autre métaheuristique [Talbi, 2000]. Ce principe général appelé *hybridation* a été utilisé dans l'AG^U. Dans ce cas, l'AG a été combiné avec des procédures de recherche locale. Les procédures de recherche locale sont des méthodes qui résolvent les problèmes d'optimisation de manière itérative. Elles font évoluer la configuration courante en la remplaçant par une autre issue de son voisinage, ce changement de configuration est couramment appelé un *mouvement*. À la Figure 4.2, on remarque que tous les individus de la population initiale ainsi que chaque enfant produit sont améliorés à l'aide d'une procédure de recherche locale. De plus, la solution finale α est, elle aussi, améliorée avant d'être envoyée à l'utilisateur. Deux méthodes de recherche locale ont été utilisées dans l'AG^U : la méthode d'échanges d'arcs successifs *3-Opt* restreint [Johnson et McGeoch, 1997] et le *Or-Opt* [Or, 1976]. Il est toutefois à noter que chacune de ces méthodes est limitée à un nombre maximum de 30 modifications d'une solution par échanges d'arcs en raison du temps de calcul qu'elles nécessitent. Le choix de la méthode à utiliser pour améliorer un enfant est fait aléatoirement, chacune des deux méthodes ayant une chance sur deux d'être choisie.

Notre but étant l'optimisation d'un problème d'ordonnancement industriel rencontré dans un centre de coulée d'aluminium, l'évaluation précise d'une séquence de commandes se doit de reproduire la suite des évènements de préparation et de coulée des fours en tenant compte de la disponibilité du métal. Cette procédure s'avère toutefois complexe et exige

un temps de calcul important. De façon à limiter ce temps de calcul, l'évaluation des individus dans l'AG^U est réalisée à l'aide d'une fonction d'évaluation simplifiée qui traite la séquence de commandes sans tenir compte de tous les détails ce qui permet d'obtenir une estimation de l'évaluation des objectifs en un temps moins important que celui de l'évaluation complète. Cependant, il est important de noter qu'avant d'être retournée à l'usager, la solution finale trouvée par l'algorithme est évaluée de manière complète.

D'un point de vue informatique, l'algorithme a été implémenté en C++ selon une approche objet.

4.3.2 Essais numériques et résultats obtenus

Pour juger de la performance de l'algorithme présenté, neuf carnets de commandes représentatifs du contexte industriel de taille variant entre 10 et 140 commandes ont été testés. Dans Gagné *et al.* [2002b], les huit premiers problèmes ont été résolus à partir d'un algorithme d'OCF. Pour tous les problèmes les paramètres N, p_c, p_m , *NbGen*, *Kmax* qui représentent respectivement la taille de la population, la probabilité de croisement, la probabilité de mutation, le nombre maximum de générations et le nombre maximum de générations sans amélioration, sont fixés à (100, 0.8, 0.3, 100, 30). L'algorithme s'arrête après 100 générations ou après 30 générations successives sans amélioration. Pour tous les AG présentés dans ce chapitre, l'expérimentation numérique a été réalisée sur un PC Pentium IV 1.8Mhz avec 512Mo de RAM utilisant Windows XP.

Le Tableau 4.2 (a, b, c) présente les résultats obtenus pour les huit premiers problèmes lorsque les objectifs sont, tour à tour, désignés comme objectif principal. Dans ce tableau, les objectifs sont numérotés de 1 à 3 en fonction de leur ordre de priorité. En considérant la

perte de capacité comme objectif principal, on retrouve dans la première partie du Tableau 4.2 (a) les résultats obtenus par l'OCF [Gagné *et al.*, 2002b] et dans la seconde partie les résultats obtenus avec l'AG^U. Chaque problème a été résolu à dix reprises par les deux algorithmes et nous présentons le résultat moyen et l'écart type obtenu. On peut constater que la performance de l'AG^U sur l'objectif prioritaire est comparable à celle de l'OCF sur les quatre premiers problèmes. Pour les autres problèmes, l'OCF produit de meilleurs résultats et est moins variable que l'AG^U. Toutefois, pour cet objectif, les différences de performances entre les deux algorithmes sont minimes.

Au Tableau 4.2 (b), en considérant le retard comme l'objectif principal, on observe cette fois-ci une meilleure performance de l'AG^U pour tous les problèmes. Dans ce cas, les améliorations sur l'objectif principal sont plus substantielles sur certains problèmes et particulièrement pour les problèmes de 40 à 80 commandes. Ces améliorations atteignent près de 19% pour les problèmes de 70 et 80 commandes.

Au Tableau 4.2 (c), on peut constater que l'objectif de minimisation de la perte de capacité de transport est un objectif facile à optimiser pour les deux algorithmes qui fournissent des performances similaires. Cependant, sur le deuxième objectif, la perte de capacité de la machine à couler, on constate que l'OCF produit de meilleurs résultats et est moins variables que l'AG^U. Toutefois, les différences de performances entre les deux algorithmes sont encore cette fois minimes.

Taille du problème	OCF [Gagné *et al.*, 2002b]			AG^U		
	Capacité (1)	Retard (2)	Transport (3)	**Capacité** (1)	Retard (2)	Transport (3)
10	**1.64**	6.97	12.71	**1.64**	6.97	13.66
	(0.00)	(0.00)	(0.00)	**(0.00)**	(0.00)	(1.00)
20	**3.69**	49.75	0.00	**3.69**	43.72	4.32
	(0.00)	(2.15)	(0.00)	**(0.00)**	(0.38)	(4.56)
30	**3.69**	83.48	0.74	**3.69**	72.97	0.58
	(0.00)	(4.15)	(0.64)	**(0.00)**	(2.73)	(0.48)
40	**3.69**	131.38	17.75	**3.69**	119.81	17.90
	(0.00)	(3.22)	(0.00)	**(0.00)**	(2.19)	(0.48)
50	**4.75**	200.32	72.88	**4.82**	163.78	42.76
	(0.00)	(11.26)	(19.19)	**(0.10)**	(26.41)	(15.70)
60	**4.75**	285.78	63.46	**4.76**	251.81	51.04
	(0.00)	(15.40)	(21.92)	**(0.04)**	(33.42)	(11.74)
70	**4.75**	416.48	62.07	**4.84**	361.83	48.83
	(0.00)	(14.81)	(24.71)	**(0.12)**	(33.62)	(6.82)
80	**4.75**	637.53	36.37	**4.83**	582.80	26.85
	(0.00)	(33.44)	(14.30)	**(0.18)**	(38.41)	(1.99)

Tableau 4.2 (a) : Résultats comparatifs entre l'OCF et l'AG^U lorsque la perte de capacité de la machine à coulée est l'objectif principal. Le premier résultat de chacune des colonnes correspond à la moyenne de dix essais (le second résultat est l'écart type).

Taille du problème	OCF [Gagné *et al.*, 2002b]			AG^U		
	Capacité (2)	**Retard** (1)	Transport (3)	Capacité (2)	**Retard** (1)	Transport (3)
10	2.23	**6.78**	9.54	2.37	**6.73**	8.69
	(0.31)	**(0.10)**	(1.68)	(0.00)	**(0.00)**	(0.00)
20	3.97	**16.97**	0.00	4.06	**15.55**	0.86
	(0.11)	**(0.89)**	(0.00)	(0.11)	**(0.30)**	(2.73)
30	4.95	**26.16**	8.03	5.54	**24.39**	9.38
	(0.22)	**(1.10)**	(6.66)	(0.23)	**(0.81)**	(10.45)
40	6.23	**43.44**	17.68	7.43	**33.06**	21.48
	(0.39)	**(3.81)**	(14.40)	(0.45)	**(1.46)**	(6.85)
50	7.72	**19.64**	64.94	8.70	**15.95**	43.06
	(0.40)	**(1.48)**	(10.05)	(0.33)	**(2.90)**	(12.35)
60	9.21	**29.88**	70.45	10.36	**22.69**	53.94
	(0.31)	**(2.19)**	(12.07)	(0.38)	**(2.84)**	(8.07)
70	9.10	**106.05**	64.31	10.20	**86.27**	53.59
	(0.50)	**(6.04)**	(12.20)	(3.92)	**(3.92)**	(8.07)
80	9.56	**167.55**	42.98	10.78	**136.36**	39.22
	(0.44)	**(4.90)**	(11.17)	(0.63)	**(5.62)**	(8.57)

Tableau 4.2 (b) : Résultats comparatifs entre l'OCF et l'AG^U lorsque le retard est l'objectif principal. Le premier résultat de chacune des colonnes correspond à la moyenne de dix essais (le second résultat est l'écart type).

Taille du problème	OCF [Gagné *et al.*, 2002b]			AG^U		
	Capacité (2)	Retard (3)	**Transport** (1)	Capacité (2)	Retard (3)	**Transport** (1)
10	1.64 (0.00)	15.03 (0.00)	**0.00 (0.00)**	1.64 (0.00)	15.03 (0.00)	**0.00 (0.00)**
20	3.69 (0.00)	50.35 (2.92)	**0.00 (0.00)**	3.87 (0.12)	47.76 (2.82)	**0.00 (0.00)**
30	3.77 (0.00)	53.75 (1.95)	**0.00 (0.00)**	3.93 (0.09)	75.53 (14.12)	**0.00 (0.00)**
40	3.79 (0.00)	89.40 (8.90)	**0.00 (0.00)**	4.05 (0.12)	72.01 (7.10)	**0.00 (0.00)**
50	4.75 (0.00)	207.63 (18.86)	**0.00 (0.00)**	5.21 (0.21)	187.01 (35.52)	**0.00 (0.00)**
60	4.75 (0.00)	349.63 (32.39)	**0.00 (0.00)**	5.33 (0.26)	232.06 (45.34)	**0.00 (0.00)**
70	4.75 (0.00)	481.32 (71.94)	**0.00 (0.00)**	5.37 (0.16)	304.03 (19.19)	**0.00 (0.00)**
80	4.78 (0.00)	711.71 (121.83)	**0.00 (0.00)**	5.70 (0.41)	420.93 (60.92)	**0.00 (0.00)**

Tableau 4.2 (c) : Résultats comparatifs entre l'OCF et l'AG^U lorsque la perte de capacité de livraison est l'objectif principal. Le premier résultat de chacune des colonnes correspond à la moyenne de dix essais (le second résultat est l'écart type).

L'OCF proposé par Gagné *et al.* [2002b] n'a pas été testé pour des problèmes comportant plus de 80 commandes, il serait cependant intéressant de mesurer la performance de l'AG^U pour des problèmes de plus grande taille. Le Tableau 4.3 présente les résultats obtenus par l'AG^U pour un carnet de 140 commandes en conservant les paramètres de l'algorithme lorsque les objectifs sont, tour à tour, désignés comme objectif principal. Comme pour les huit problèmes précédents, le problème de 140 commandes a été résolu à dix reprises par l'AG^U et nous présentons le résultat moyen et l'écart type obtenu. Ce tableau permet de constater que l'objectif de perte de capacité de transport est un objectif assez facile à optimiser en présentant la valeur optimale à chaque exécution. Ce plus gros problème servira ultérieurement pour la comparaison de performances des différents algorithmes multi-objectifs.

Objectif principal : Capacité			**Objectif principal : Retard**			**Objectif principal : Transport**		
Capacité	Retard	Transport	Capacité	**Retard**	Transport	Capacité	Retard	**Transport**
5.10	2807.44	44.23	9.85	**1286.86**	50.26	5.74	2573.10	**0.00**
(0.12)	(41.25)	(8.78)	(0.33)	**(19.10)**	(8.49)	(0.45)	(544.03)	**(0.00)**

Tableau 4.3 : Résultats obtenus par l'AG^U pour un carnet de 140 commandes. Le premier résultat de chacune des colonnes correspond à la moyenne de dix essais et le second résultat est l'écart type.

Le Tableau 4.4 présente les meilleurs résultats connus sur chaque objectif. Pour les trois premiers problèmes, il a été possible de connaître les valeurs optimales à l'aide d'une méthode de séparations et d'évaluations progressives. Ces solutions sont indiquées par un *. Pour ces trois problèmes, l'AG^U permet d'obtenir la valeur optimale sur chacun des objectifs f_1, f_2 et f_3. Notons que sur les cinq derniers problèmes, l'AG^U a réussi à abaisser les meilleurs résultats connus pour l'objectif f_2 (indiqué par $^+$) et obtenus par l'OCF [Gagné *et al.*, 2002b] tout en obtenant un résultat identique au minimum connu sur les deux autres objectifs.

Taille du problème	**Capacité (f_1)**	**Retard (f_2)**	**Transport (f_3)**
	Meilleur résultat connu	**Meilleur résultat connu**	**Meilleur résultat connu**
10	1.64*	6.73*	0.00*
20	3.69*	15.05*	0.00*
30	3.69*	22.33*	0.00*
40	3.69	30.79$^+$	0.00*
50	4.75	11.76$^+$	0.00*
60	4.75	18.30$^+$	0.00*
70	4.75	80.51$^+$	0.00*
80	4.75	123.93$^+$	0.00*

Tableau 4.4 : Meilleures valeurs connues pour les trois objectifs; * indique une solution optimale et $^+$ solution trouvée par l'AG^U.

Le Tableau 4.5 donne le temps de calcul moyen obtenu par l'AG^U pour chacun des neuf problèmes. Ne disposant pas d'une base de comparaison équitable, ces résultats ne seront cependant pas comparés à ceux de l'OCF.

Taille du Problème	Temps de calcul (secondes)		
	Capacité	Retard	Transport
10	6.8	7.6	4
20	13	17.4	11
30	22.5	31.9	20
40	41.2	52.6	33
50	60	90.5	52
60	98.7	156	75
70	187.4	256	140
80	256.7	378.6	150
140	1186	1186	1186

Tableau 4.5 : Moyenne des temps de calcul de l'AG^U pour les trois objectifs (secondes)

De façon générale, les résultats présentés dans les paragraphes précédents montrent que l'AG^U proposé obtient des performances intéressantes en fournissant des résultats au moins comparables à ceux de l'OCF et en se démarquant plus particulièrement sur l'objectif de minimisation du retard total. On peut alors convenir que cette technique d'optimisation s'avère être une méthode de résolution efficace pour le problème d'ordonnancement industriel.

4.4 Traitement multi-objectifs à l'aide d'algorithmes génétiques

Maintenant que la performance de l'AG^U a été démontrée pour optimiser les objectifs selon un ordre lexicographique, nous allons nous intéresser à la résolution simultanée des différents objectifs du problème. Pour cela, l'algorithme du NSGAII est adapté afin de l'appliquer à la résolution du contexte d'ordonnancement industriel et servira pour fins de

comparaisons. Par la suite, un nouvel algorithme génétique multi-objectifs pour le problème industriel est proposé. Cet algorithme, noté AG^{MOPI} dans le reste du travail, sera utilisé comme méthode d'approximation de l'ensemble Pareto par le SIAD. L'AG^{MOPI} est un algorithme basé sur l'AG^{U} qui combine des éléments empruntés au NSGAII et au SPEA2 deux algorithmes présentés au chapitre précédent. Finalement, l'AG^{U} sera intégré à la procédure générique de recherche de compromis proposée par Gagné *et al.* [2004a] de façon à permettre d'orienter plus précisément la recherche en fonction des préférences exprimées par le décideur.

Afin de vérifier la qualité des solutions produites par les différents algorithmes génétiques multi-objectifs implémentés, des *ensembles de référence* ont été créés pour chacun des neuf problèmes tests évalués précédemment par l'AG^{U} pour permettre une comparaison. Les ensembles de référence proviennent de plusieurs exécutions de divers algorithmes uni-objectif et multi-objectifs et constituent une approximation des ensembles Pareto.

4.4.1 Adaptation du NSGAII pour fins de comparaisons

La première étape pour résoudre le problème d'un point de vue multi-objectifs consiste à reproduire l'algorithme du NSGAII et à l'adapter au contexte d'ordonnancement industriel. Cet algorithme a été choisi parce qu'il nécessite peu de réglage de paramètres. De plus, de récentes études [Deb, 2000; Zitzler *et al.*, 2001] montrent qu'il obtient de bonnes performances sur différents PMO.

4.4.1.1 Description de l'algorithme

L'algorithme du NSGAII utilisé est en définitive le même que celui décrit à la Section 3.6.3.1 mis à part l'ajout d'une procédure d'amélioration locale lors de la création de la population d'enfants et l'utilisation d'une archive permettant de conserver les solutions non dominées. Comme dans le cas de l'AG^U, la procédure d'amélioration locale permet d'intensifier la recherche. De son côté, l'archive est utilisée comme un outil de stockage pour nous assurer que les solutions non dominées trouvées par l'algorithme ne seront pas perdues durant la recherche. Ces deux ajouts ont pour but de permettre une comparaison équitable entre le NSGAII et l'AG^{MOPI}. Le fonctionnement de l'algorithme du NSGAII, présenté au chapitre précédent, est rappelé à la Figure 4.3. Les définitions et la notation utilisées sont précisées au Tableau 4.6. Les procédures d'évaluation, de croisement et de mutation utilisées sont les mêmes que celles de l'AG^U. Au niveau de l'amélioration locale, nous utilisons aussi, comme pour l'AG^U, les méthodes d'échange d'arcs successifs 3-Opt restreint et Or-Opt.

Initialiser aléatoirement la population initiale P_0 de taille N
Initialiser $A = \varnothing$
Calculer les différents fronts F_i de la population P_0
Mettre à jour A avec les individus du premier front
Sélectionner dans P_0 et création de Q_0 par application des opérateurs de croisement et de mutation
Tant que critère d'arrêt non rencontré **faire**
    Créer $R_t = P_t \cup Q_t$
    Calculer les différents fronts F_i de la population R_t
    Mettre à jour A avec les individus du premier front
    Mettre $P_{t+1} = \varnothing$ et $i = 0$,
    Tant que $|P_{t+1}| + |F_i| < N$ **faire**
    Inclure dans P_{t+1} les individus de F_i
    FinTantQue
    Inclure dans P_{t+1} les $(N - |P_{t+1}|)$ individus de F_i les mieux répartis au sens de la $i_{distance}$
Sélectionner dans P_{t+1} et création de Q_{t+1} par application des opérateurs de croisement et de mutation
Pour chaque individu $x \in Q_{t+1}$ **faire**
    Effectuer une amélioration locale sur x
Finpour
FinTantQue

Figure 4.3 : Algorithme du NSGAII pour le problème d'ordonnancement industriel

R	: Population intermédiaire de taille 2* N
F_i	: $i^{ème}$ Front
$i_{distance}$	: distance de remplacement
A	: Archive

Tableau 4.6 : Notation du NSGAII

Pour l'archive, nous avons choisi d'utiliser une structure de données arborescente appelée *quad-tree* [Sun et Steuer, 1996]. Cette structure de données sert à la gestion de solutions non dominées en ne conservant dans les nœuds constituant l'arborescence que les solutions non dominées et en limitant le nombre de comparaisons à effectuer pour établir la dominance. Comme pour l'AG^U, l'algorithme du NSGAII a été implémenté en C++ en apportant les changements discutés précédemment.

4.4.1.2 Essais numériques et résultats obtenus

L'expérimentation a été faite à partir des neuf problèmes évalués par l'AG^U à la section 4.4.3. Dans un premier temps, seulement deux des trois objectifs sont considérés pour permettre la représentation visuelle des résultats obtenus pour les neuf problèmes tests à l'aide de graphiques en deux dimensions. Dans un deuxième temps, le problème de 80 commandes est repris dans sa globalité pour démontrer la généralisation à plusieurs objectifs. Les paramètres N, p_c, p_m, $NbGen$ de l'algorithme sont fixés à (100,100, 0.3, 100). Notons que, pour cet algorithme, le seul critère d'arrêt utilisé est le nombre de générations.

Illustration de la performance du NSGAII sur deux objectifs

Les deux objectifs considérés pour l'illustration des résultats sont la minimisation de la perte de capacité en usine (f_1) et le retard total (f_2). Une comparaison visuelle entre les solutions proposées par le NSGAII et les points de l'ensemble de référence est présentée à

la Figure 4.4 (a à i). Pour les problèmes de 10 à 30 commandes, les solutions de l'ensemble de référence constituent la frontière Pareto. Pour ces trois problèmes, il a été possible d'énumérer entièrement l'espace de solutions de façon à en extraire l'ensemble Pareto optimal. Les solutions présentées à l'aide d'un triangle sont les solutions de l'ensemble de référence et les solutions représentées par un cercle sont celles trouvées par le NSGAII. La Figure 4.4 (a à i) montre que, de manière générale, l'algorithme du NSGAII fournit des solutions très proches de celles de l'ensemble de référence. Pour le problème de 10 commandes, la courbe proposée par le NSGAII est même identique à celle de l'ensemble de référence. Pour les problèmes de 20 et 60 commandes, les courbes du NSGAII et de l'ensemble de référence sont pratiquement superposées. Cependant, pour les autres problèmes, on remarque soit un écart appréciable entre certaines portions des deux courbes (Figure 4.4 (c, e, g et h)), soit qu'il y a des régions entières de l'ensemble de référence que le NSGAII n'est pas capable d'aller chercher (Figure 4.4 (d et i)). Ceci est peut-être dû au fait que l'algorithme du NSGAII éprouve, dans certains cas, des difficultés à répartir les solutions sur la frontière Pareto. On peut penser que pour ces problèmes, la répartition en front et l'utilisation de la distance de remplacement ($i_{distance}$) ne permettent pas de sélectionner les individus les plus isolés.

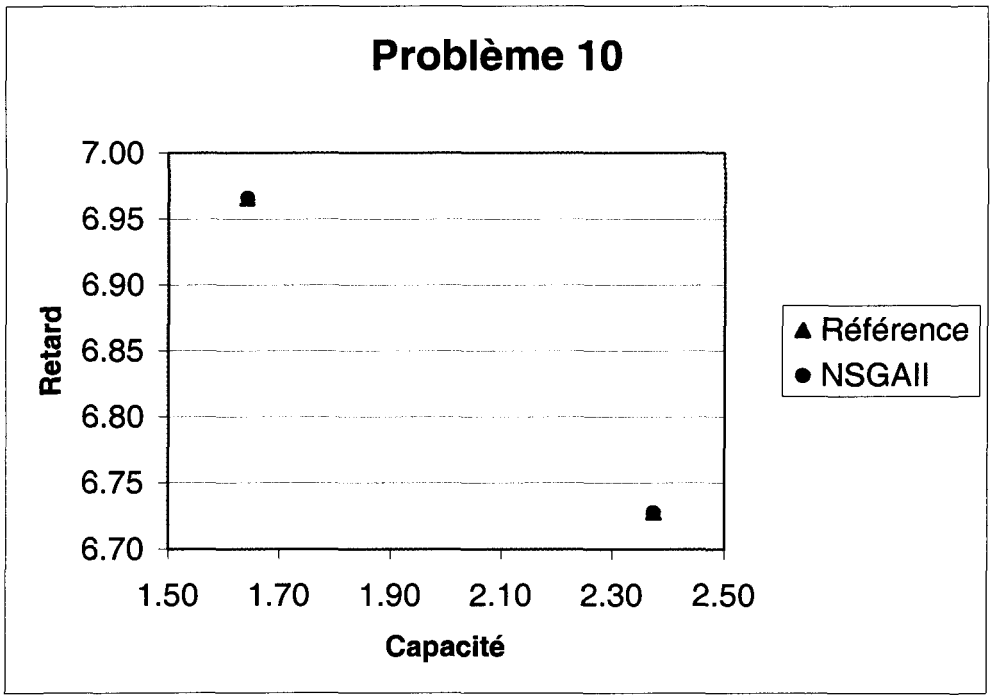


Figure 4.4 (a) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 10 commandes.

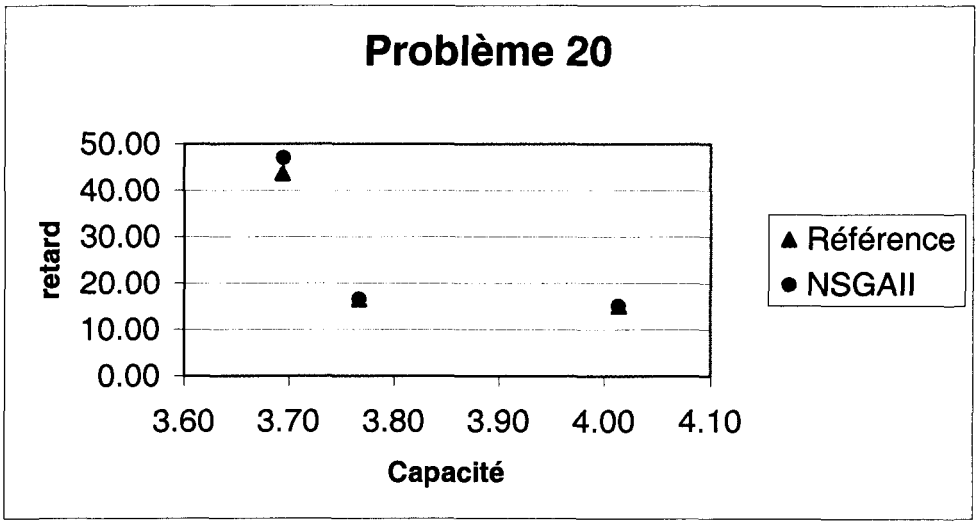


Figure 4.4 (b) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 20 commandes.

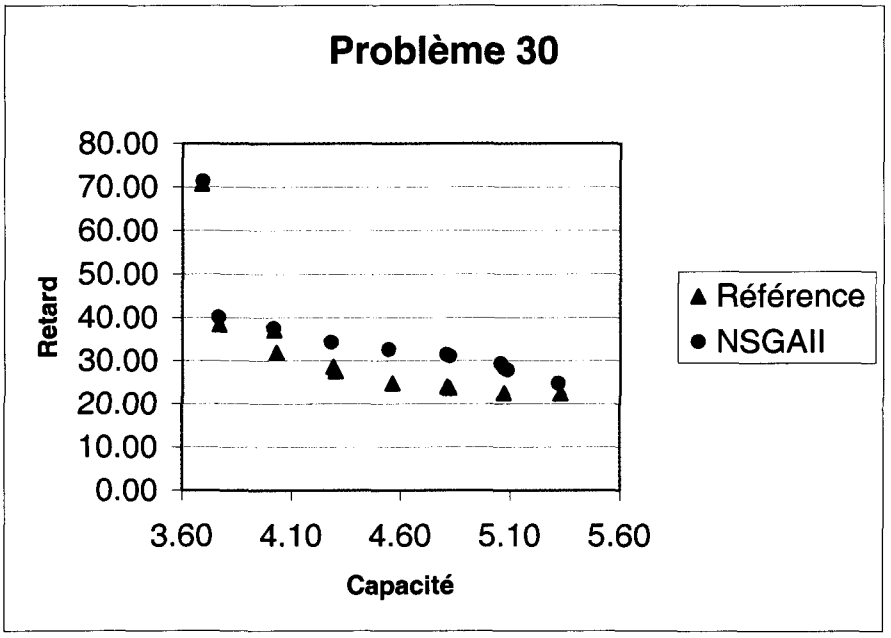


Figure 4.4 (c) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 30 commandes.

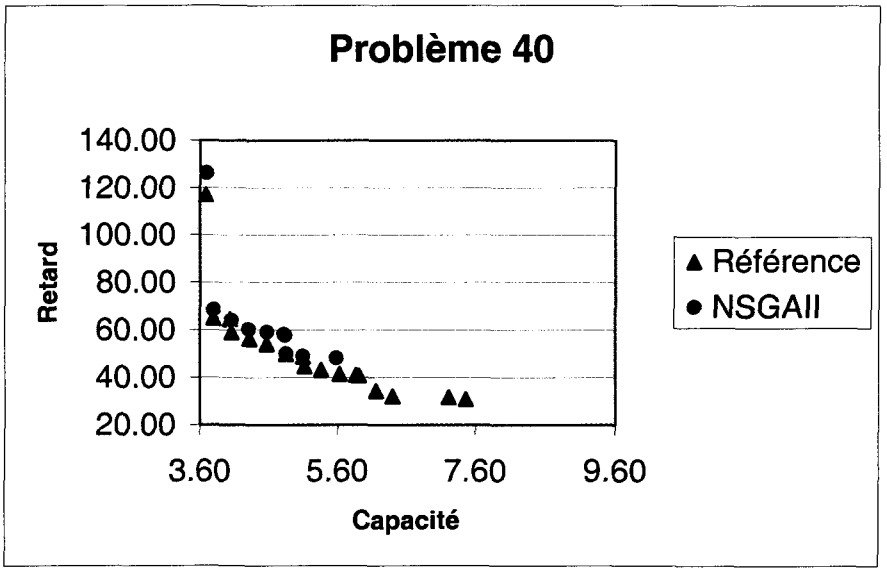


Figure 4.4 (d) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 40 commandes.

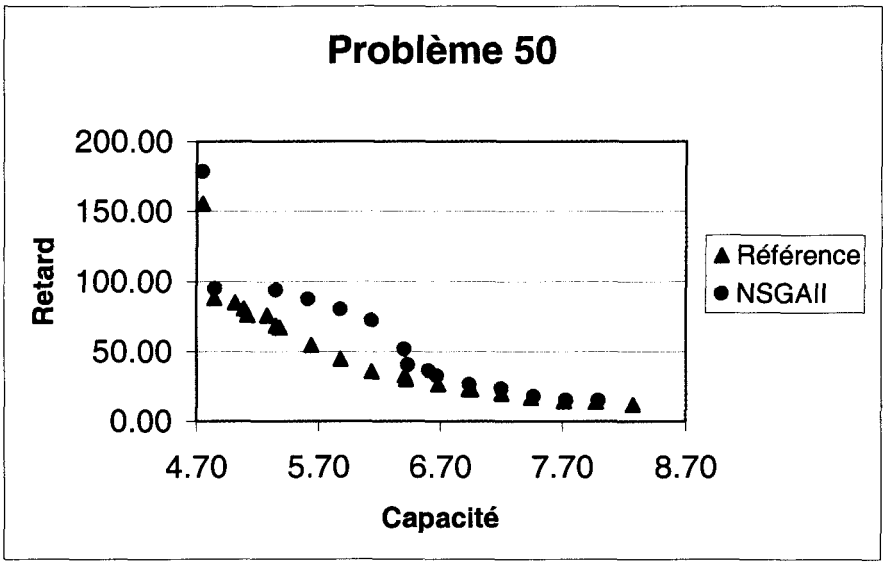


Figure 4.4 (e) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 50 commandes.

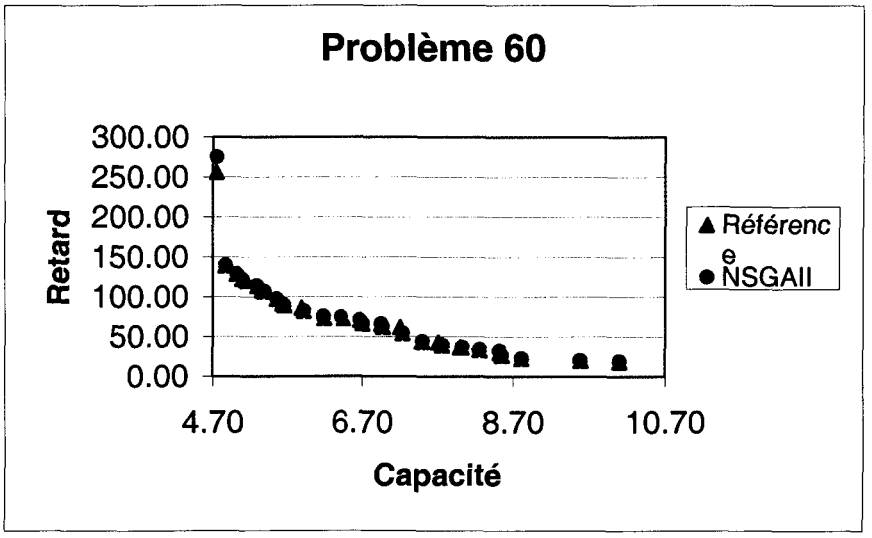


Figure 4.4 (f) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 60 commandes.

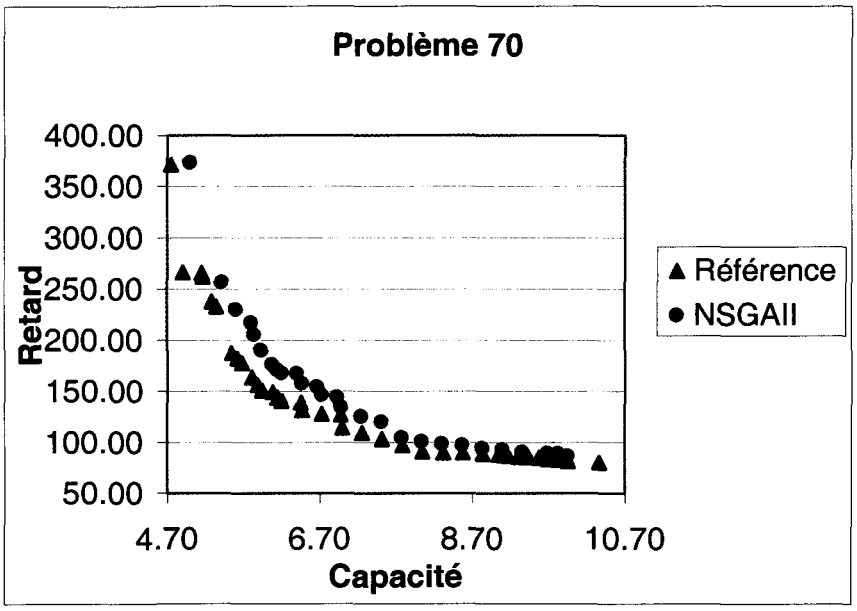


Figure 4.4 (g) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 70 commandes.

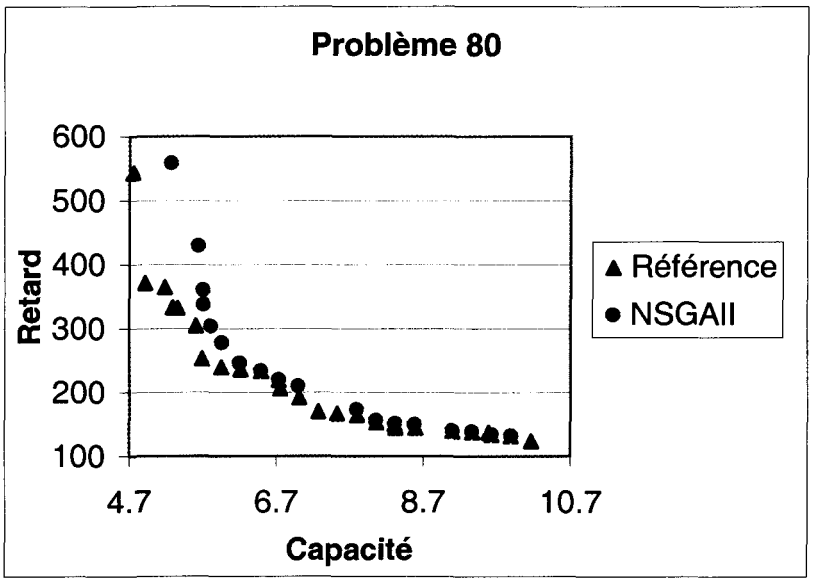


Figure 4.4 (h) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 80 commandes.

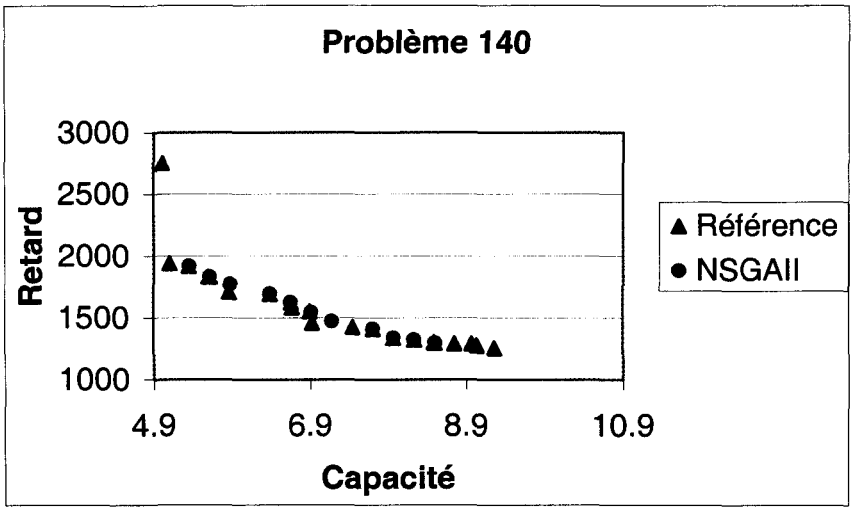


Figure 4.4 (i) : Comparaison graphique de la performance entre le NSGAII et l'ensemble de référence. Problème de 140 commandes

Illustration de la performance du NSGAII sur trois objectifs

Le problème industriel de 80 commandes est maintenant repris avec les trois objectifs pour évaluer la performance du NSGAII. L'ensemble de référence a été généré de façon analogue à ce qui a été fait pour deux objectifs. Un ensemble de 46 solutions non dominées a ainsi été obtenu. Le problème de 80 commandes a été résolu par l'algorithme du NSGAII sur le même ordinateur que celui utilisé pour l'AG^U en 450 secondes. Le Tableau 4.7 présente une comparaison entre les solutions de l'ensemble de référence et les solutions obtenues par le NSGAII. Dans la première partie du tableau, on retrouve les solutions de l'ensemble de référence et dans la seconde les 28 solutions obtenues par le NSGAII. On constate une fois de plus que l'ensemble de solutions proposées par le NSGAII ne couvre pas les différentes parties de l'ensemble de référence.

Ensemble de référence			NSGAII		
Capacité	**Retard**	**Transport**	**Capacité**	**Retard**	**Transport**
4.75	554.02	26.23	8.26	566.16	0
4.75	566.27	0	8.43	400.92	3.58
4.91	371.44	13.68	8.53	406.59	0
5.18	365	45.04	8.71	391.39	21.77
5.44	436.03	0	8.9	336.38	8.16
5.68	321.12	0	8.97	323.22	21.77
5.68	307.03	21.77	9.16	292.26	15.23
5.68	319.75	4.46	9.16	309.81	0
5.95	308.94	0	9.43	300.7	0
5.95	305.68	41.67	9.69	285.36	0
6.19	305.87	3.93	9.69	284.28	2.55
6.19	307.36	0	9.8	277.18	42.63
6.21	302.03	21.77	9.94	284.27	29.92
6.21	287.63	36.32	9.95	278.17	0
6.34	299.21	0	10.22	274.3	0
6.47	290.52	21.77	10.29	251.47	31.59
6.6	284.75	0.43	10.29	269.84	21.77
6.6	295.58	0	10.48	267.32	0
6.75	269.67	0	10.51	260.27	0
6.86	257.65	0	10.55	258.01	0
6.86	255.5	6.29	10.55	248.21	21.77
6.86	256.84	4.46	10.75	251.24	6.03
6.86	254.51	21.77	10.75	254.13	0
7.13	248.61	0	10.75	251.06	9.82
7.13	240.66	28.06	10.82	247.69	22.22
7.39	246.53	0	10.82	248.12	21.77
7.59	237.41	0	10.99	250.22	5.87
7.8	232.23	28.9	11.01	242.88	0
7.86	235.59	0			
8.04	229.97	3.93			
8.06	210.72	0			
8.32	207.68	0			
8.32	207.61	9.82			
8.57	206	10.1			
8.79	205.12	5.87			
8.79	206.89	3.93			
8.79	207.42	0			
8.9	204.44	0			
8.91	197.15	0			
9.33	176.09	40.36			
9.56	178.77	0			
9.58	172.74	0			
9.9	140.49	39.88			
10.17	123.93	36.55			
10.69	142.57	36.33			
10.94	138.55	26.66			
10.96	134.25	30.47			

Tableau 4.7 : Comparaison de la performance entre l'ensemble de référence et le NSGAII sur les 3 objectifs. Problème de 80 commandes.

Dans l'ensemble, les résultats expérimentaux ont montré les bonnes performances du NSGAII sur les différentes instances du problème d'ordonnancement industriel. Sur certains problèmes, les résultats du NSGAII rivalisent même avec ceux des ensembles de référence. Cependant, sur d'autres instances, il existe un écart appréciable entre les ensembles de référence et les solutions proposées par le NSGAII notamment en termes de diversité de solutions. On peut alors convenir que cette technique de résolution, bien qu'étant efficace, semble avoir de la difficulté à bien couvrir l'ensemble des solutions.

4.4.2 Phase 1 : proposition d'un nouvel AG (AG^{MOPI})

Cette section présente l'algorithme multi-objectifs proposé, l'AG^{MOPI}, pour la résolution du problème d'ordonnancement industriel. Cet algorithme est basé sur l'AG^{U} et utilise les concepts de niche et de dominance Pareto. L'AG^{MOPI} permet, dans une première phase d'optimisation, d'approximer l'ensemble Pareto en une seule exécution.

4.4.2.1 Description de l'algorithme

L'AG^{MOPI}, comme l'AG^{U}, est un algorithme élitiste hybridant un algorithme génétique avec des procédures de recherche locale. Le pseudo-code de l'AG^{MOPI} est donné à la Figure 4.5. Comme l'indique cette dernière, l'AG^{MOPI} gère l'élitisme de deux façons. Tout d'abord, l'AG^{MOPI} utilise le principe d'archive pour conserver les solutions non dominées rencontrées durant la recherche comme l'algorithme du SPEA2. Contrairement à ce dernier, l'AG^{MOPI} n'utilise pas une mais deux archives : une *archive primaire,* notée A et une *archive secondaire*, notée A'. L'archive primaire consiste en une population annexe de taille déterminée $\overline{N}$ permettant de stocker les solutions temporairement Pareto-Optimales. Toutes les solutions de l'archive primaire participent au processus de sélection. Lorsque le

nombre de solutions non dominées dans l'archive primaire dépasse $\overline{N}$, 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 secondaire et ne participent plus au processus de sélection. L'utilisation de ce double niveau d'archivage garantit qu'aucune solution non dominée ne sera perdue lors du processus de recherche. De plus, de façon similaire au NSGAII, l'AG^{MOPI} assure qu'à chaque nouvelle génération les meilleurs individus rencontrés sont conservés dans la population courante.

```
Initialiser A0 et A0' à ∅
Initialiser aléatoirement la population initiale P0 de taille N
Evaluer chaque individu x ∈ P0
Calculer la performance de chaque individu x ∈ P0
Mettre à jour A0
Trier P0
Tant que critère d'arrêt non rencontré faire
        Sélectionner des individus dans Pt et créer Qt par croisement et mutation
        Évaluer chaque individu x ∈ Qt
        Effectuer une amélioration locale sur chaque individu x∈ Qt
        Mettre à jour At avec chaque individu non dominé ∈ Qt
        Combiner les deux populations Qt et Pt
        Calculer la performance de chaque individu dans Qt ∪ Pt et dans At
        Si | At | > N alors
                Réduire At et mettre à jour At'
        FinSi
        Trier Qt ∪ Pt
        Copier les N premières solutions de Qt ∪ Pt dans Pt+1
FinTantQue
Mettre à jour At' avec les solutions de At
Retourner A'
```

Figure 4.5 : Pseudo-code de l'AG^{MOPI}

La première étape de l'algorithme consiste à générer la population de départ aléatoirement comme dans l'AG^{U}. Par la suite, à chaque itération de l'algorithme, on retrouve les étapes classiques de *sélection*, *croisement*, *mutation*. Pour ces trois étapes, l'AG^{MOPI} utilise les mêmes opérateurs que l'AG^{U} et le NSGAII à savoir une sélection par tournoi, l'ER et le PMX comme méthode de croisement et une mutation par inversion. A chaque itération de l'algorithme, chaque nouvel individu non dominé vient s'ajouter à l'archive primaire et les individus de l'archive primaire dominés par le nouvel arrivant sont supprimés. Comme le SPEA2, le mécanisme d'assignation de la performance de l'AG^{MOPI} tient en compte à la fois du nombre d'individus dominés par une solution i et du nombre d'individus qui domine la solution i. Pour cela, la première étape consiste à assigner à chaque individu i de l'archive primaire A_t et de la population P_t une force $S(i)$ correspondant au nombre de solutions dominées par i, tel que :

$$S(i)=\left|\left\{ j \,\middle|\, j\in P_t\cup A_t,\ i\succ j \right\}\right| \qquad (15)$$

où | | représente la cardinalité de l'ensemble et le symbole $\succ$ indique une relation de dominance de i par rapport à j. À partir de la valeur de $S(i)$, la performance brute d'un individu i, notée $R'(i)$, est déterminée dans l'AG^{MOPI} à l'aide de l'équation suivante :

$$R'(i)=\begin{cases} \dfrac{S(i)}{1+2*S(i)} & si \sum_{j\in P_t\cup A_t,\ j\succ i} S(j)=0 \\ \sum_{j\in P_t\cup A_t,\ j\succ i} S(j) & sinon \end{cases} \qquad (16)$$

Cette approche diffère de celle du SPEA2 dans le sens où les individus non dominés pour lesquels $\sum_{j \in P_t \cup A_t,\ j \succ i} S(j) = 0$ n'auront pas tous une performance brute identique et égale à 0 mais une performance brute comprise entre 0 et 0.5. Pour les solutions non dominées, cette méthode d'assignation de la performance brute permet de mieux tenir compte de la distribution des solutions dominées dans l'espace de recherche. Finalement, la dernière étape consiste à estimer la densité *D(i)* de chaque individu *i* (selon l'Équation 13 de la Section 3.6) à partir du concept du $k^{\text{ième}}$ plus proche voisin [Silverman, 1986] et à l'additionner à la performance brute précédemment calculée pour obtenir la performance globale de l'individu.

Une fois la population d'enfant Q créée, celle-ci est combinée avec la population courante. Les individus de la nouvelle population ainsi créée sont ensuite triés en fonction de leurs performances globales et seuls les N premiers sont conservés pour former la prochaine génération courante. A la fin de l'algorithme, les individus contenus dans l'archive primaire vont mettre à jour l'archive secondaire. Cet ensemble de solutions non dominées est ensuite retourné à l'usager .

Comme pour les deux algorithmes précédents, l'AG^{MOPI} a été implanté en C^{++}. L'AG^{MOPI} utilise les mêmes procédures d'évaluation, de croisement, de mutation et d'amélioration locale que le NSGAII. De cette manière, les principales structures de données sont partagées par les deux algorithmes. Ce qui nous permet d'obtenir une base solide pour une comparaison la plus équitable possible entre les deux algorithmes.

4.4.2.2 Essais numériques et résultats obtenus

Nous comparons ici les résultats de l'AG^{MOPI} avec ceux du NSGAII et des ensembles de référence. Comme pour le NSGAII, l'expérimentation a été faite à partir des neuf problèmes évalués par l'AG^{U} à la Section 4.4.3. Dans un premier temps, seulement deux des trois objectifs sont considérés pour permettre la représentation visuelle des résultats obtenus pour les neuf problèmes tests à l'aide de graphiques en deux dimensions. Dans un deuxième temps, le problème de 80 commandes est repris dans sa globalité pour démontrer la généralisation à plusieurs objectifs. Pour tous les problèmes, les paramètres N, p_c, p_m, *NbGen* et $\overline{N}$ qui représentent respectivement la taille des populations parents et enfants, la probabilité de croisement, la probabilité de mutation le nombre maximum de générations et la taille de l'archive primaire, sont fixés à (100, 0.8, 0.3, 100,100). Comme pour le NSGAII, le seul critère d'arrêt utilisé par l'AG^{MOPI} est le nombre maximum de générations.

Illustration des performances de l'AG^{MOPI} sur deux objectifs

Comme pour le NSGAII, les deux objectifs considérés pour l'illustration des résultats sont la minimisation de la perte de capacité en usine (f_1) et le retard total (f_2). Le Tableau 4.8 donne, pour chaque instance de problème, les temps de calcul obtenu par l'AG^{MOPI} et le NSGAII. Notons que l'AG^{MOPI} demande généralement plus de temps de calcul que le NSGAII. Cet écart s'explique cependant par l'utilisation de deux archives dans l'algorithme. On note toutefois que les temps de calculs seraient acceptables dans un contexte réel en étant inférieurs ou égaux à 3 minutes pour les deux algorithmes sur les sept premiers problèmes. Cependant, pour les deux problèmes de plus grande taille, ces temps augmentent à 6 minutes 6 secondes pour le problème de 80 commandes et à 1h33 pour le

problème de 140 commandes avec le NSGAII. En ce qui concerne l'AG^{MOPI}, les temps augmentent à 10 minutes 2 secondes et à 3h19.

Taille du problème	Temps de calcul (en secondes)	
	NSGAII	AG^{MOPI}
10	10	9
20	55	34
30	35	81
40	35	37
50	83	180
60	155	168
70	134	145
80	340	577
140	4802	11492

Tableau 4.8 : Moyenne des temps de calculs du NSGAII et de l'AG^{MOPI}

Une comparaison visuelle entre les solutions proposées par l'AG^{MOPI}, le NSGAII et les ensembles de référence est présentée à la Figure 4.7 (a à i). Les solutions présentées à l'aide d'un triangle sont les solutions de l'ensemble de référence, les solutions représentées par un cercle sont celles trouvées par le NSGAII et les solutions représentées à l'aide d'un losange sont celles de l'AG^{MOPI}. De manière générale, on observe à l'aide de la Figure 4.7 que les courbes proposées par l'AG^{MOPI} sont comparables à celles des ensembles de référence sur tous les problèmes. Les courbes proposées par l'AG^{MOPI} étant même identiques à celles de l'ensemble de référence pour les problèmes de 10 et 20 commandes. En comparant l'AG^{MOPI} au NSGAII, on remarque que l'AG^{MOPI} couvre mieux l'espace des solutions et réussit à abaisser les courbes proposées par le NSGAII pour 6 des 9 instances (problèmes 20,30 40, 50, 70 et 80) et obtient des résultats similaires sur trois instances (problèmes 10,

60 et 140). En particulier, on remarque que l'AG^{MOPI} parvient à couvrir les différentes régions des ensembles de référence que le NSGAII ne parvenait pas à obtenir.

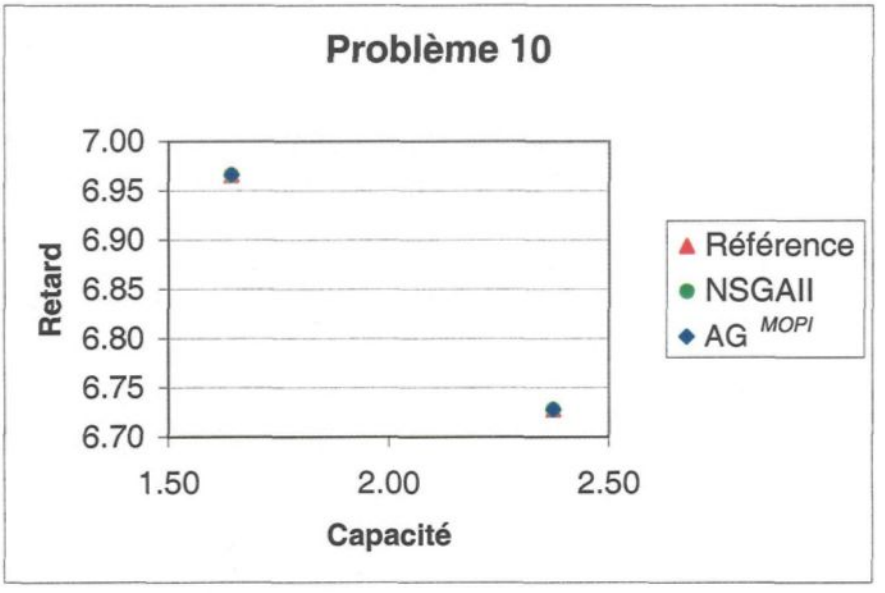


Figure 4.7 (a) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 10 commandes.

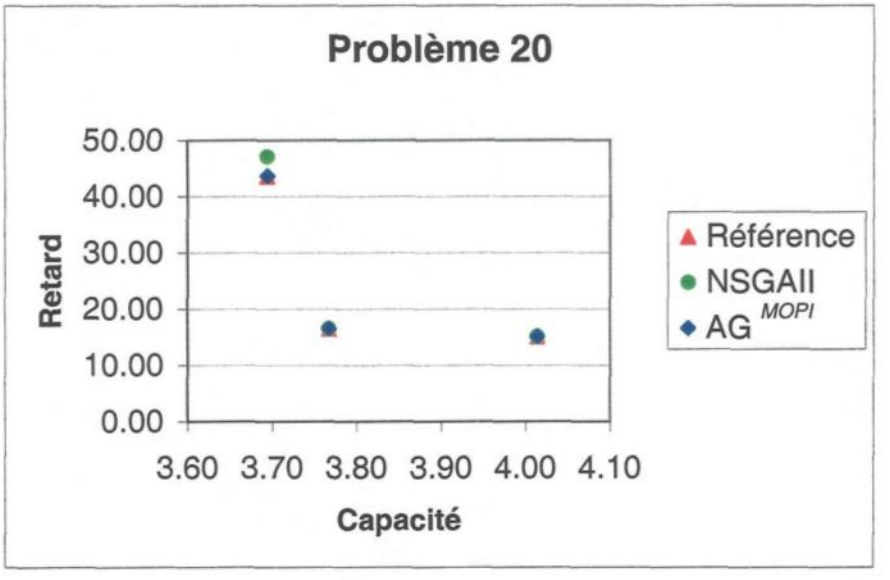


Figure 4.7 (b) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 20 commandes.

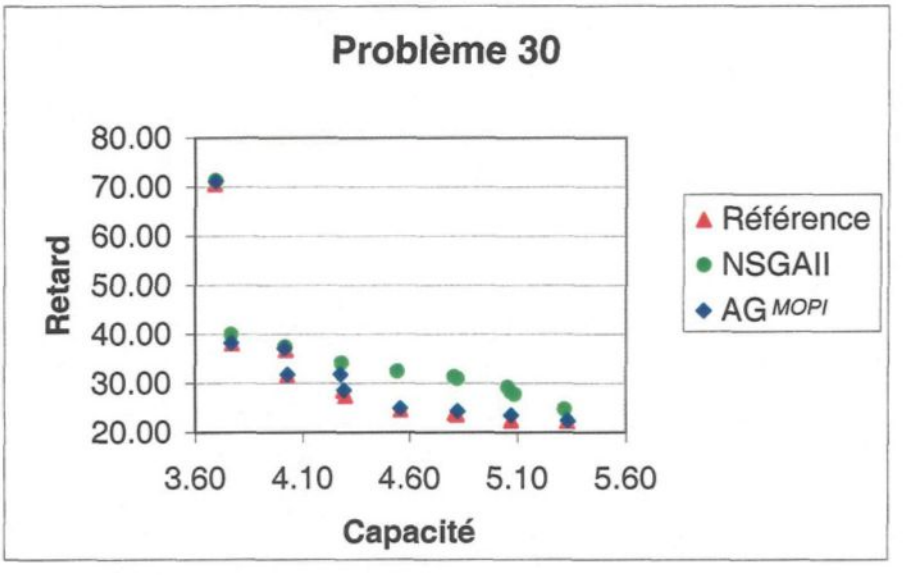


Figure 4.7 (c) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 30 commandes.

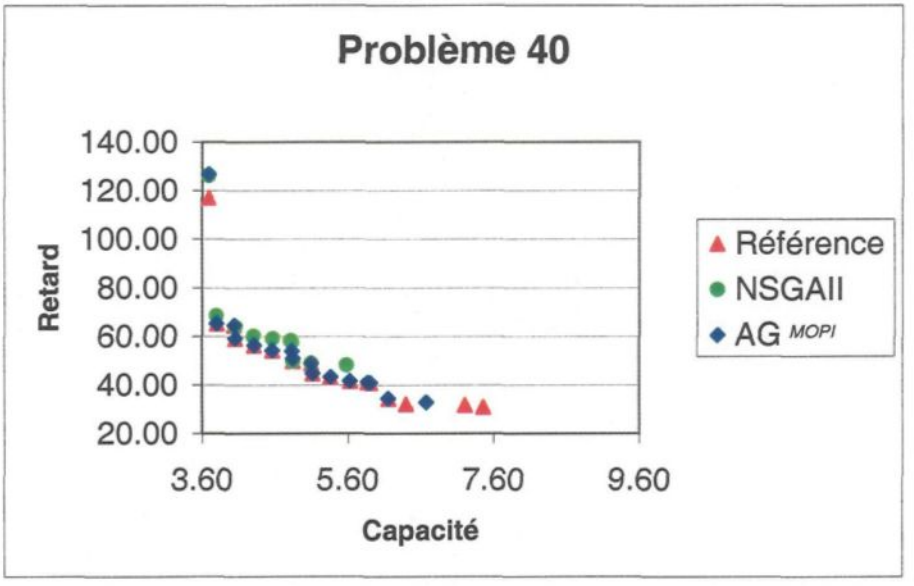


Figure 4.7 (d) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 40 commandes.

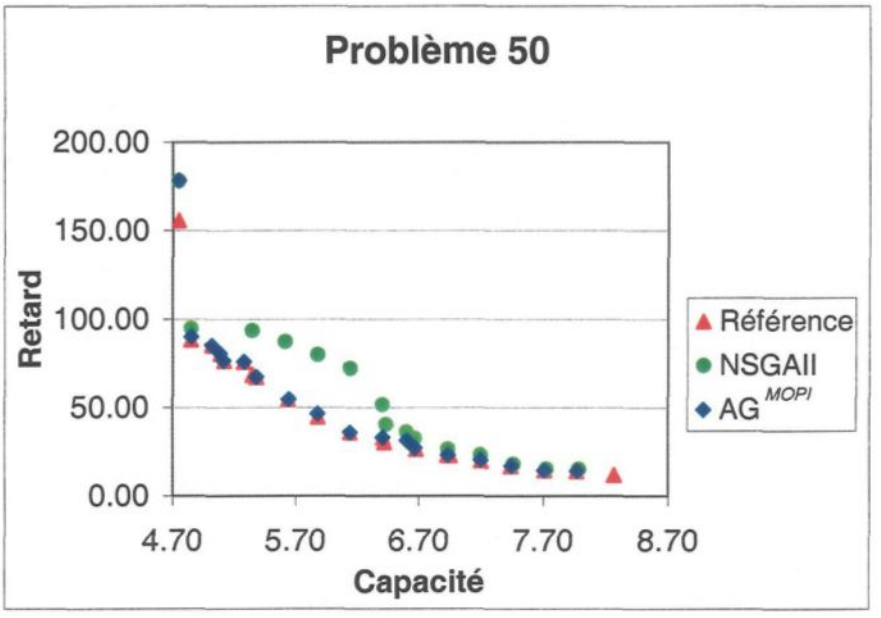


Figure 4.7 (e) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 50 commandes.

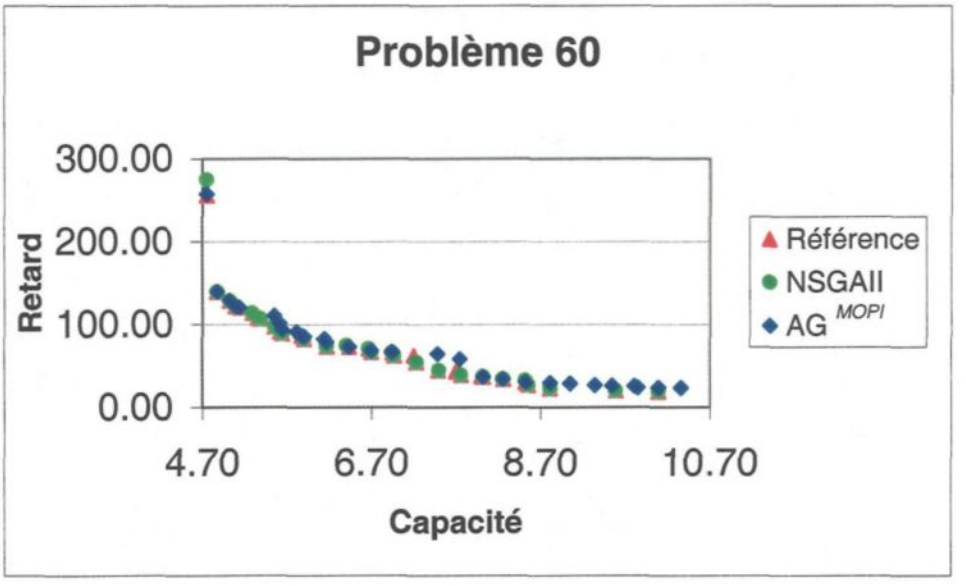


Figure 4.7 (f) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 60 commandes.

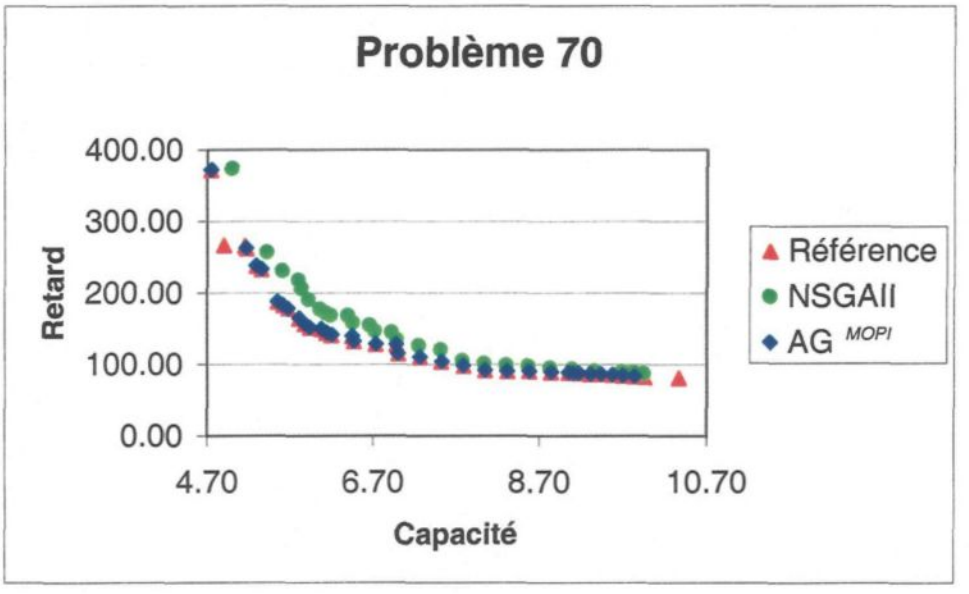


Figure 4.7 (g) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 70 commandes.

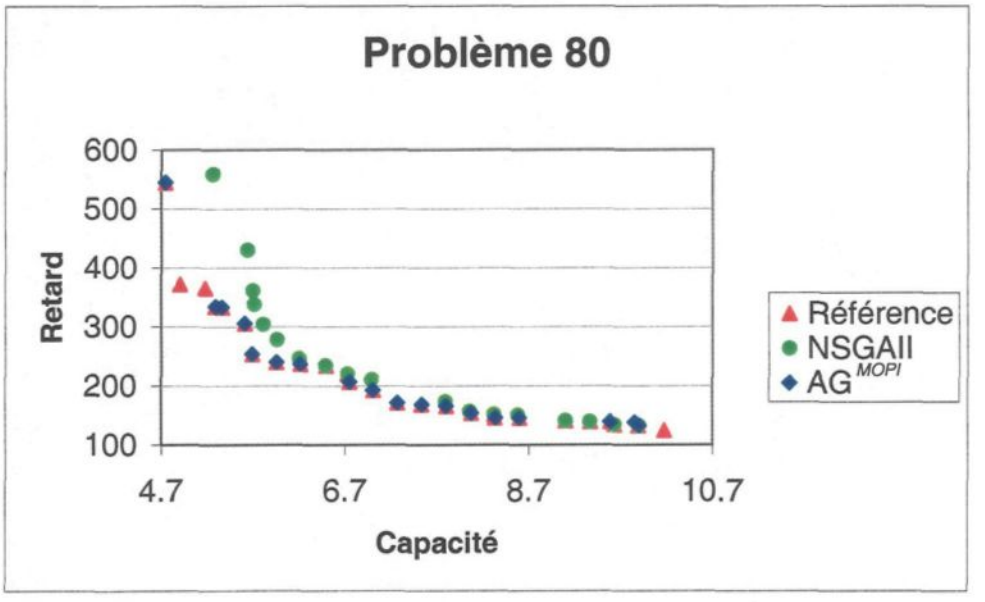


Figure 4.7 (h) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 80 commandes.

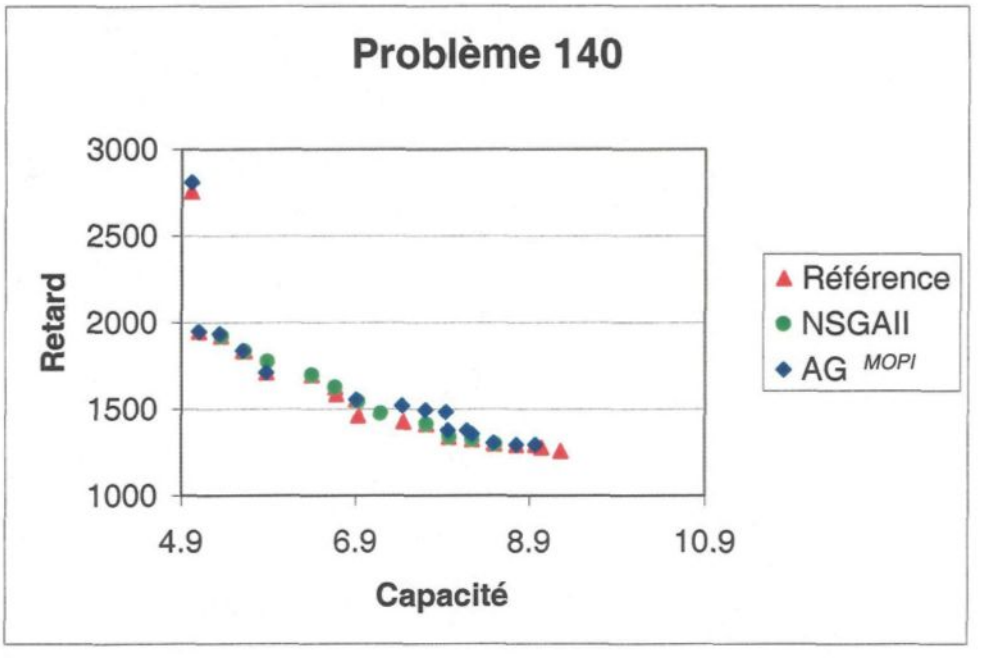


Figure 4.7 (i) : Comparaison graphique de la performance entre l'AG^{MOPI}, le NSGAII et l'ensemble de référence. Problème de 140 commandes.

Illustration de la performance de l'AG^{MOPI} sur trois objectifs

Afin de mieux évaluer la performance de l'AG^{MOPI}, le problème industriel de 80 commandes est maintenant repris dans son ensemble. Le Tableau 4.9 présente une comparaison entre les solutions de l'ensemble de référence, les solutions obtenues par le NSGAII et les solutions de l'AG^{MOPI}. Une fois de plus, on constate que l'AG^{MOPI} obtient de meilleures performances que le NSGAII avec une surface couvrant mieux l'ensemble des solutions et plus proche de l'ensemble de référence.

Pour nous en convaincre, nous avons aussi comptabilisé pour chacun des deux algorithmes, le nombre de solutions trouvées qui appartiennent au front pseudo Pareto représenté par l'ensemble de référence. On constate que 34 des 35 solutions trouvées par l'AG^{MOPI}, soit 97.17%, sont des solutions de l'ensemble de référence alors que ce taux est de 0% pour le NSGAII qui ne trouve aucune solution de l'ensemble de référence.

Ensemble de référence			NSGAII			AG^{MOPI}		
Capacité	**Retard**	**Transport**	**Capacité**	**Retard**	**Transport**	**Capacité**	**Retard**	**Transport**
4.75	554.02	26.23	8.26	566.16	0	5.53	567.58	21.77
4.75	566.27	0	8.43	400.92	3.58	5.68	319.75	4.46
4.91	371.44	13.68	8.53	406.59	0	5.68	321.12	0
5.18	365	45.04	8.71	391.39	21.77	5.68	307.03	21.77
5.44	436.03	0	8.9	336.38	8.16	5.95	308.94	0
5.68	321.12	0	8.97	323.22	21.77	6.19	305.87	3.93
5.68	307.03	21.77	9.16	292.26	15.23	6.19	307.36	0
5.68	319.75	4.46	9.16	309.81	0	6.21	302.03	21.77
5.95	308.94	0	9.43	300.7	0	6.34	299.21	0
5.95	305.68	41.67	9.69	285.36	0	6.47	290.52	21.77
6.19	305.87	3.93	9.69	284.28	2.55	6.6	284.75	0.43
6.19	307.36	0	9.8	277.18	42.63	6.6	295.58	0
6.21	302.03	21.77	9.94	284.27	29.92	6.75	269.67	0
6.21	287.63	36.32	9.95	278.17	0	6.86	254.51	21.77
6.34	299.21	0	10.22	274.3	0	6.86	256.84	4.46
6.47	290.52	21.77	10.29	251.47	31.59	6.86	257.65	0
6.6	284.75	0.43	10.29	269.84	21.77	6.86	255.5	6.29
6.6	295.58	0	10.48	267.32	0	7.13	240.66	28.06
6.75	269.67	0	10.51	260.27	0	7.13	248.61	0
6.86	257.65	0	10.55	258.01	0	7.39	246.53	0
6.86	255.5	6.29	10.55	248.21	21.77	7.59	237.41	0
6.86	256.84	4.46	10.75	251.24	6.03	7.86	235.59	0
6.86	254.51	21.77	10.75	254.13	0	8.04	229.97	3.93
7.13	248.61	0	10.75	251.06	9.82	8.06	210.72	0
7.13	240.66	28.06	10.82	247.69	22.22	8.32	207.61	9.82
7.39	246.53	0	10.82	248.12	21.77	8.32	207.68	0
7.59	237.41	0	10.99	250.22	5.87	8.57	206	10.1
7.8	232.23	28.9	11.01	242.88	0	8.79	205.12	5.87
7.86	235.59	0				8.79	207.42	0
8.04	229.97	3.93				8.79	206.89	3.93
8.06	210.72	0				8.9	204.44	0
8.32	207.68	0				8.91	197.15	0
8.32	207.61	9.82				9.33	176.09	40.36
8.57	206	10.1				9.56	178.77	0
8.79	205.12	5.87				9.58	172.74	0
8.79	206.89	3.93						
8.79	207.42	0						
8.9	204.44	0						
8.91	197.15	0						
9.33	176.09	40.36						
9.56	178.77	0						
9.58	172.74	0						
9.9	140.49	39.88						
10.17	123.93	36.55						
10.69	142.57	36.33						
10.94	138.55	26.66						
10.96	134.25	30.47						

Tableau 4.9 : Comparaison de la performance entre l'ensemble de référence, le NSGAII et l'AG^{MOPI} sur les 3 objectifs. Problème de 80 commandes.

En résumé, nous avons présenté un algorithme génétique multi-objectifs (AG^{MOPI}) performant pour résoudre le problème industriel de planification de la coulée d'aluminium.

Les performances de l'AG^{MOPI} ont été démontrées sur un ensemble de neuf problèmes du contexte d'ordonnancement industriel instances et comparées avec celles du NSGAII et des ensembles de référence. Les résultats expérimentaux montrent que l'AG^{MOPI} parvient à bien approximer les ensembles de référence et obtient des résultats au moins égaux ou supérieurs à ceux du NSGAII pour les 9 instances testées. Cependant, il est bon de noter que le NSGAII demande un effort de calcul moins important que l'AG^{MOPI}.

L'AG^{MOPI} est donc l'algorithme qui sera utilisé par le SIAD dans une première phase d'optimisation pour permettre d'approximer la frontière Pareto. La section suivante présente comment l'AG^{U} a été intégré à la procédure générique de recherche de compromis de façon à permettre, dans une deuxième phase, d'orienter plus précisément la recherche en fonction des préférences exprimées par le décideur.

4.4.3 Phase 2 : application de la procédure de recherche de compromis (AG^{CP})

La frontière Pareto approximée à l'aide de l'AG^{MOPI} permet au décideur d'avoir une idée de l'ensemble des solutions réalisables pour un problème donné. Le décideur peut alors préciser ses préférences pour les différents objectifs afin d'explorer des régions spécifiques de l'ensemble des solutions proposé par l'AG^{MOPI}. C'est dans cette optique que l'AG^{U} a été incorporé à la procédure de recherche de compromis proposée dans Gagné *et al.* [2004a] dans la base de modèle du SIAD. L'approche générique pour la recherche de compromis est une procédure agrégative qui fait partie de la catégorie des *approches basées sur la transformation du problème en un problème uni-objectif* présentée au Chapitre 3. Son but est de rechercher dans l'espace de solutions réalisables, la solution minimisant la *distance*

normalisée et *pondérée* au *point idéal* $F^* = \{F_1^*, F_2^*, \ldots, F_k^*\}$ où F_i^*, dans un contexte de minimisation, représente les valeurs minimales pour chacun des objectifs *i* dans l'ensemble des solutions pseudo Pareto optimales. La normalisation proposée s'effectue à l'aide de l'étendue du domaine. Pour cela, à l'opposé du point idéal, cette approche utilise aussi la notion de point *nadir* qui représente les valeurs maximales pour chacun des objectifs dans l'ensemble des solutions pseudo Pareto optimales.

La procédure générique de recherche de compromis se déroule en trois phases : *une phase de recherche des points idéal et nadir*, *une phase de recherche de solutions de compromis* et *une phase de présentation des solutions efficaces au décideur*. La première phase dans notre contexte est réalisée par l'AG^{MOPI} pour rechercher les points pseudo-idéal et pseudo-nadir. La deuxième phase consiste à rechercher les solutions de compromis en fonction de l'importance relative que le décideur accorde à chacun des objectifs. Le processus itératif de l'AG reste sensiblement le même que celui de l'AG^U mis à part qu'après l'évaluation d'un individu sur chacun des objectifs, la distance normalisée et pondérée de la solution par rapport au point idéal est calculée et cette distance devient la mesure de performance de cet individu. Pour chaque solution générée, les relations de dominance sont vérifiées afin de ne conserver que les solutions non dominées. À cet effet, comme pour le NSGAII et l'AG^{MOPI}, une structure de données arborescente appelée quad-tree [Sun et Steuer, 1996] est utilisée afin de limiter le nombre de comparaisons à effectuer. Une fois le processus itératif de l'AG achevé, une légère perturbation est appliquée sur la pondération des objectifs et le processus itératif est relancé. L'application de cette

perturbation permet d'élargir légèrement l'espace de recherche et ainsi d'atteindre des compromis intéressants se trouvant à proximité.

Finalement, la dernière phase consiste à présenter les résultats de la recherche au décideur. Pour cela, les solutions dont la distance normalisée et pondérée par rapport au point idéal est trop grande peuvent être négligées lors de la présentation des résultats. L'algorithme résultant de l'utilisation de la procédure de recherche de compromis est noté AG^{CP} dans le reste de ce travail.

Le lecteur peut consulter Gagné *et al.* [2004a; 2004b] pour une description plus détaillée de chacune des phases de la procédure générique et de son utilisation avec les métaheuristiques.

4.4.3.1 Essais numériques et résultats obtenus

Afin d'illustrer et de démontrer la pertinence de l'AG^{CP}, des essais numériques ont été effectués sur le problème de 140 commandes qui est la plus grande instance de problème dont nous disposons. Ce carnet de commandes a été choisi car, sur les plus petites instances, l'apport de cette deuxième phase de recherche ne pourrait pas être aussi bien mis en évidence compte tenu du fait de la proximité entre les ensembles de références et les ensembles de solutions proposées par l'AG^{MOPI}. L'illustration des performances de la procédure se fera seulement sur deux des trois objectifs à savoir la perte de capacité en usine et le retard total. Le problème ayant été résolu par l'AG^{MOPI}, on peut considérer la phase 1 du processus générique comme étant réalisée. Le point pseudo-idéal obtenu est alors $\{F_1^*, F_2^*\}= \{5.01, 1253.55\}$ et le point pseudo-nadir est $\{F_1^{nad}, F_2^{nad}\}= \{9.26,2208.39\}$. L'expérimentation numérique a été effectuée en 1181 secondes sur le

même ordinateur que celui utilisé pour les autres algorithmes. Une comparaison visuelle entre les solutions proposées par l'AG^{CP} et l'AG^{MOPI} est disponible à la Figure 4.8 (a et b) pour différentes possibilités de préférences du décideur exprimés à l'aide de pondération sur les objectifs. Les points encadrés à l'aide d'un carré et d'un cercle représentent respectivement les solutions de l'AG^{CP} et de l'AG^{MOPI} qui possèdent la distance normalisée et pondérée minimale au point idéal en fonction des préférences exprimées. La Figure 4.8 (a et b) montre que l'AG^{CP} suggère au décideur des actions représentatives de l'AG^{MOPI}. En fait, pour les pondérations choisies, les solutions suggérées par l'AG^{CP} ont une distance minimale au point idéal quasiment identique, voir même inférieure à celle de l'AG^{MOPI}.

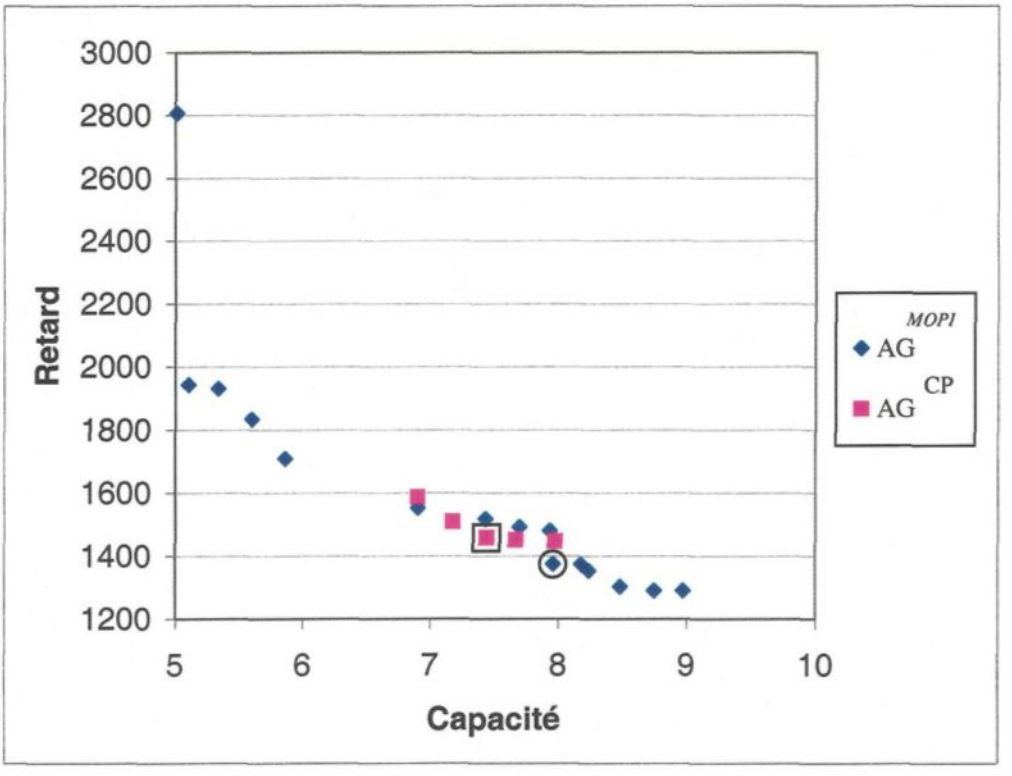


Figure 4.8 (a) : Comparaison de la performance entre l'AG^{CP} et l'AG^{MOPI}. Problème de 140 commandes, pondération capacité = 0.2, pondération retard = 0.8

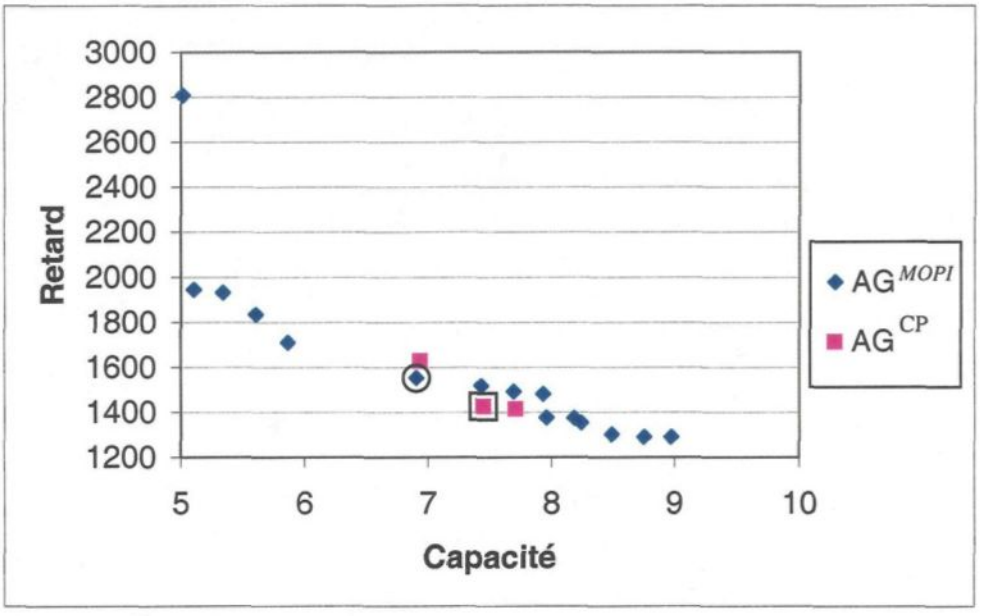


Figure 4.8 (b) : Comparaison de la performance entre l'AG^{CP} et l'AG^{MOPI}. Problème de 140 commandes, pondération capacité = 0.3, pondération retard = 0.7

Ces résultats démontrent la performance de cette approche et surtout de sa pertinence dans l'optique d'une deuxième phase visant à approfondir la recherche en fonction des préférences exprimées par le décideur.

4.5 Conclusion

Dans ce chapitre, les différents algorithmes qui vont constituer la base de modèle du SIAD ont été présentés. En particulier, l'AG^{MOPI}, un algorithme génétique multi-objectifs performant qui combine les concepts de niche, de dominance Pareto et d'élitisme. Les performances l'AG^{MOPI} ont été démontrées sur un ensemble de neuf carnets de commandes et comparés avec celles du NSGAII et des ensembles de référence. Les résultats expérimentaux ont montré que l'AG^{MOPI} obtenait des solutions représentatives des

ensembles de référence et parvenait à bien couvrir l'espace des solutions comparativement au NSGAII pour l'optimisation de deux et de trois objectifs. En fait, pour l'optimisation de deux objectifs, mis à part une instance, l'AG^{MOPI} obtient des résultats au moins identiques ou supérieurs au NSGAII sur les huit autres instances. Cependant, on note que l'utilisation de deux archives ainsi que la technique d'estimation de la densité engendrent une augmentation du temps de calcul de l'algorithme. De plus, nous avons aussi mis en évidence, sur quelques exemples, la pertinence de l'implantation de la procédure générique de compromis comme complément à l'AG^{MOPI} afin d'utiliser les préférences du décideur et ainsi d'approfondir la recherche dans une région plus spécifique de l'ensemble pseudo Pareto optimal proposé par l'AG^{MOPI}.

Le chapitre suivant présente les différents éléments de l'interface graphique du SIAD et intègre les différents modules afin d'obtenir un ensemble cohérent.

CHAPITRE 5

PRÉSENTATION DU SIAD POUR LE CONTEXTE D'ORDONNANCEMENT INDUSTRIEL

5.1 Introduction

De nos jours, l'environnement des entreprises se transforme. La concurrence mondiale, les développements technologiques font que l'accent est de plus en plus mis sur la productivité, la qualité et la satisfaction de la clientèle. Pour arriver à de bons résultats, il est devenu vital de prendre rapidement de bonnes décisions. Cependant, la complexité des problèmes industriels, le nombre sans cesse croissant d'objectifs à optimiser simultanément et la rapidité des changements de l'environnement raccourcissent considérablement les délais de prise de décision tout en rendant cette tâche plus difficile pour les gestionnaires. Dans un tel contexte, des outils informatiques comme les systèmes interactifs d'aide à la décision s'avèrent d'une grande utilité pour le décideur car ils lui permettent d'évaluer la situation, les diverses alternatives et leurs impacts éventuels. Cependant, les SIAD classiques traitent généralement les problèmes sur la base de l'optimisation d'un seul objectif ce qui correspond rarement à la réalité des situations pratiques.

Ce chapitre présente le SIAD multi-objectifs proposé qui met en œuvre, dans un environnement interactif, les différents algorithmes présentés dans le chapitre précédent pour résoudre un PMO. La présentation du système sera illustrée à l'aide du problème d'ordonnancement industriel multi-objectifs rencontré dans une entreprise de production d'aluminium décrit au Chapitre 4. Le fait d'avoir choisi un contexte réel, qui incorpore souvent des contraintes supplémentaires par rapport à un problème théorique, permet de mieux exploiter le SIAD proposé et ainsi de mieux mesurer les apports que ce type d'outils informatique peut apporter dans la résolution de PMO.

Dans un premier temps, nous présentons les différentes technologies retenues pour le développement du SIAD. Dans un deuxième temps, l'architecture générale du système est présentée. Finalement, les trois dernières sections décrivent de manière plus détaillée les modules du SIAD avec un accent mis en particulier sur l'interface personne-machine.

5.2 Choix technologiques

Afin d'assurer un développement uniforme et prévisible, qualités essentielles à mesure que croît la complexité d'un système, nous avons choisi d'adopter une approche objet pour la modélisation du SIAD. Cette modélisation utilise la norme UML (*Unified Modeling Language*) développée par Booch, Rumbaugh et Jacobson. UML unifie les concepts provenant de *OMT* de Booch *et al.* [1995], les *use case* de Jacobson *et al.* [1992], le *CRC* (Class-Responsibility-Collaboration) de la communauté Smalltalk, les *Statecharts* de Harel *et al.* [1987] et plusieurs autres. C'est un langage de modélisation qui contient les éléments du modèle, les concepts fondamentaux, la sémantique de modélisation, la notation, la représentation graphique des éléments du modèle, les directives générales et l'usage de mots clefs dans des contextes définis. Un des points forts de l'approche objet est qu'elle fournit une plate-forme permettant un développement et une maintenance plus rapide favorisant ainsi la réutilisabilité du système. Comme il a été expliqué au Chapitre 2, le manque de réutilisabilité est une des causes principales d'échec des SIAD.

L'implantation de la base de modèle a été effectuée en C^{++} à l'aide du compilateur Borland C^{++} Builder 6. L'interface graphique, quant à elle, a été réalisée en Delphi 5 en utilisant la librairie Glscene pour l'affichage Opengl. Le choix de ces deux langages de

programmation s'explique par le fait que ceux-ci sont des langages orientés objet offrant des possibilités intéressantes et utiles pour le développement d'un SIAD. Le C^{++} offre un bon compromis entre rapidité d'exécution et confort de programmation. De son côté, Delphi permet de développer rapidement des interfaces graphiques conviviales et interagit bien avec d'autres applications.

5.3 Présentation globale du système

De façon générale, le SIAD proposé doit permettre à l'usager de planifier la production de lingots d'aluminium en minimisant les trois objectifs que sont la perte de capacité de la machine à couler (f_1), le retard total de l'ensemble des commandes (f_2) et la perte de capacité de livraison (f_3) tout en respectant les différentes contraintes technologiques relatives à la disponibilité d'équipements.

Pour cela, le SIAD développé est constitué des trois modules essentiels décrits au Chapitre 2 : une base d'informations, une base de modèles et une interface personne-machine. Ces différents composants sont illustrés à la Figure 5.1 qui présente l'architecture générale du SIAD proposé et font l'objet des prochaines sections. L'usager de son côté représente simplement les différents utilisateurs qui interagissent avec le système afin de résoudre un problème.

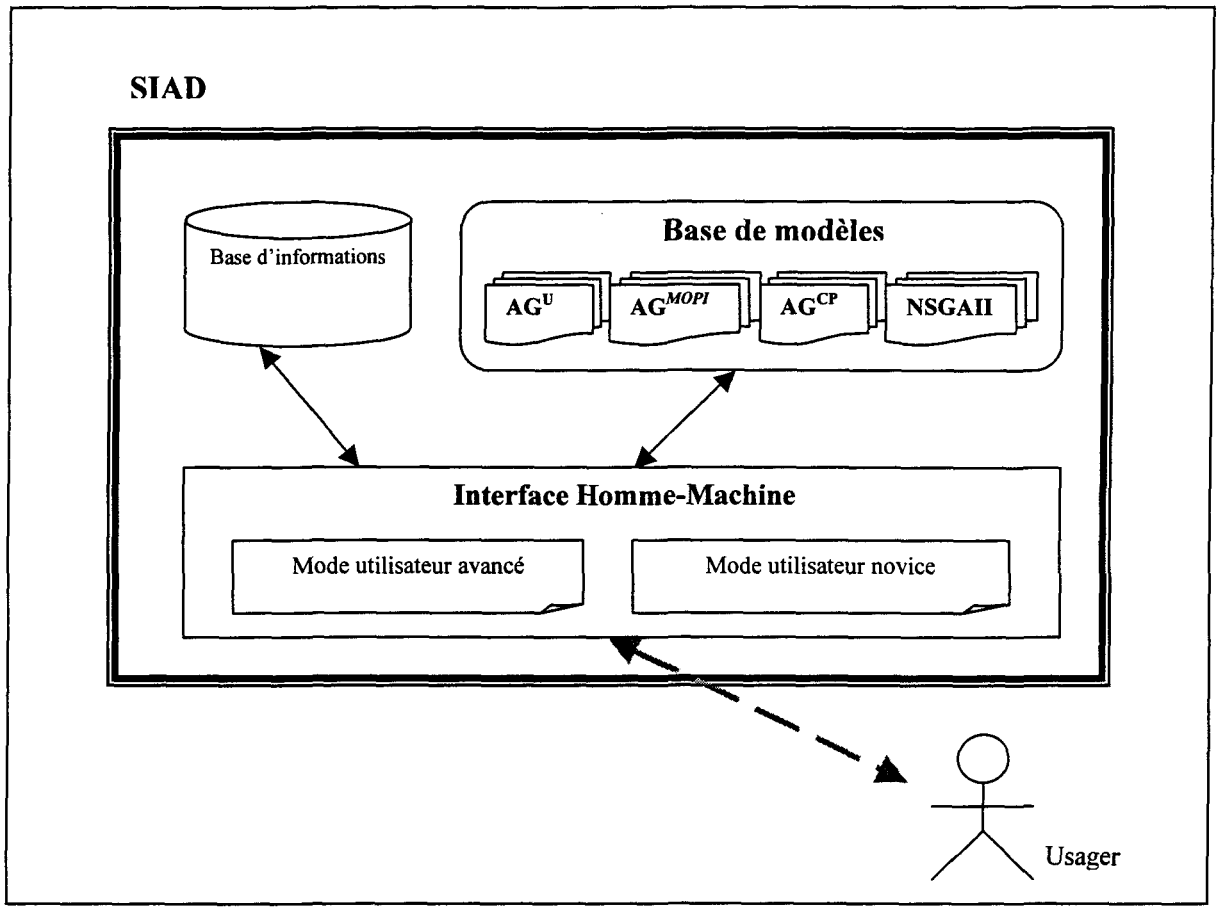


Figure 5.1 : Architecture générale du SIAD pour le contexte industriel

5.4 La base d'informations

La base d'informations du SIAD est constituée d'un ensemble de données techniques relatives à l'environnement de production de l'entreprise comme les différents types d'alliages, les numéros de moules et autres. Elle contient aussi certains résultats sur l'exécution des modèles. Il est important de noter cependant que la plupart de ces données ont été reprises telles quel de l'ancien logiciel sous forme de système de fichiers et, de ce fait, elles n'ont donc été l'objet d'aucune modification.

5.5 La base de modèles

Comme il a été expliqué au Chapitre 2, la base de modèle constitue un des éléments charnières de tous SIAD et doit en général comporter plusieurs modèles. Dans l'outil proposé, la base de modèle permet à l'utilisateur de déterminer sa stratégie de résolution grâce à des approches heuristiques basées sur les algorithmes génétiques. Elle est constituée de quatre algorithmes génétiques : l'AG^{U}, le NSGAII, l'AG^{MOPI} et l'AG^{CP}. Ces différents algorithmes permettent de résoudre de manière efficace le problème industriel et supportent les différents modes d'interaction du système. L'implantation et le fonctionnement des différents algorithmes de la base de modèle ont été décrits en détail au Chapitre 4. Le diagramme de classes de la base de modèle est présenté à l'Annexe 1.

5.6 L'interface Homme-Machine

Dans toutes les applications interactives, l'interface homme-machine ou module de dialogue représente un des éléments les plus importants. C'est à travers elle que va s'établir la collaboration entre le décideur et la machine. Cependant, pour qu'une collaboration efficace s'établisse, il faut tenir en compte des niveaux de connaissance sur les approches de résolution pour les différentes catégories d'utilisateurs qui peuvent interagir avec le SIAD. Différents modes d'interaction doivent donc être mis en place afin de favoriser une interaction homogène entre ces différentes catégories d'usagers et le SIAD. Dans le prototype de SIAD proposé, par souci de simplification, les différents types d'utilisateurs ont été regroupés en deux catégories distinctes à savoir les *utilisateurs novices* qui ont besoin d'être guidés lors de l'utilisation du SIAD et les *utilisateurs avancés* pour lesquels le

niveau d'assistance doit être réduit au strict minimum. Afin de répondre aux besoins spécifiques de ces deux catégories d'usagers, le SIAD proposé offre au démarrage le choix entre deux modes d'interaction comme l'illustre la Figure 5.2. Ces deux modes sont le *mode semi-assisté* et le *mode utilisateur avancé.*

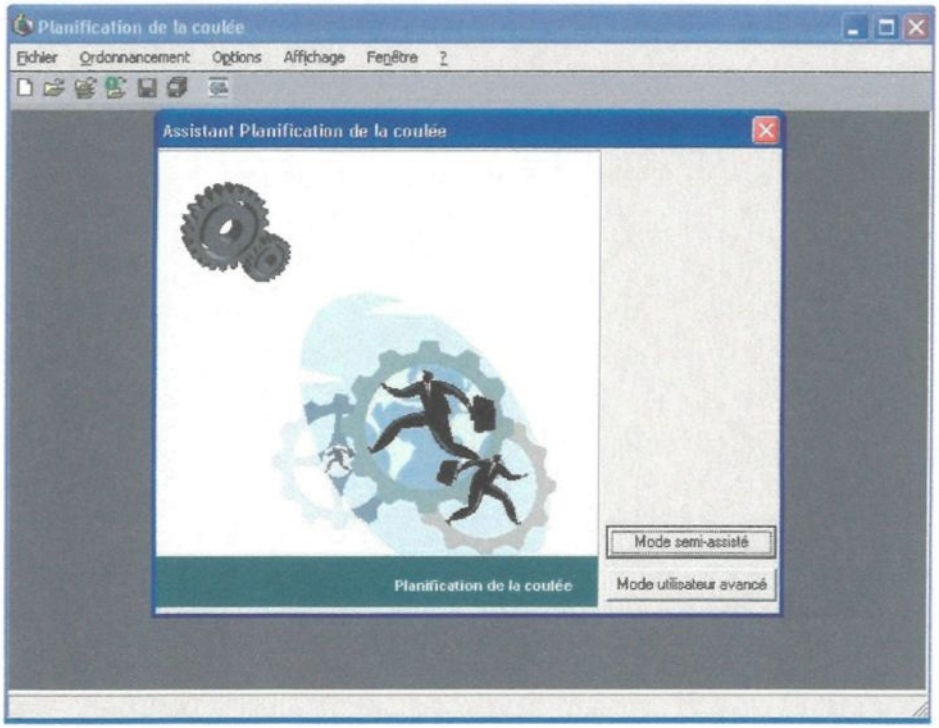


Figure 5.2 : Les différents modes d'interactions du SIAD

5.6.1 Le mode semi-assisté

Le mode semi-assisté s'adresse aux usagers non expérimentés ou non familiers avec le système. Ce mode guide l'utilisateur tout au long de son processus de décision à l'aide d'une approche par questions-réponses. Dans la première série de questions, l'usager sera

amené à définir le type d'actions qu'il envisage d'effectuer. L'utilisateur a le choix entre quatre options, à savoir :

- ouvrir un espace de travail existant;
- effectuer une optimisation uni-objectif ;
- approximer la frontière Pareto;
- chercher des solutions de compromis.

La première option de ce mode, *ouvrir un espace de travail existant*, permet à l'utilisateur d'ouvrir un environnement d'optimisation préalablement enregistré afin de visualiser les résultats alors obtenus ou encore d'effectuer une nouvelle optimisation.

La seconde option, *effectuer une optimisation uni-objectif*, indique au SIAD que l'utilisateur désire effectuer une optimisation des objectifs selon un ordre lexicographique. Lorsque l'usager choisit cette option, le système l'invite à choisir le carnet de commandes à optimiser et à assigner une priorité aux différents objectifs en les numérotant de 1 à 3. Une fois cette étape réalisée, le SIAD lance automatiquement la phase d'optimisation en utilisant l'AG^{U} et retourne les résultats à l'usager à la fin de l'optimisation.

La troisième option, *approximer la frontière Pareto*, permet d'effectuer une optimisation multi-objectifs en utilisant l'AG^{MOPI}. Une fois cette option choisie, l'usager doit indiquer au SIAD le carnet de commandes sur lequel il veut travailler ainsi que le nombre d'objectifs à optimiser. Notons que si dans notre contexte d'illustration l'usager peut choisir d'optimiser au minimum 2 objectifs et au maximum 3, l'approche pourrait cependant être généralisée pour plus de 3 objectifs. Par la suite, le SIAD demandera à l'usager d'indiquer le nombre de solutions à afficher. Celui-ci devra aussi indiquer au

système s'il désire arrêter le processus de recherche à la fin de cette étape d'optimisation ou approfondir la recherche en sélectionnant une solution de l'ensemble de solutions qui lui aura été présenté par le SIAD. Une fois toutes ces informations recueillies, l'optimisation du carnet de commandes est automatiquement lancée. Au cours de cette étape d'optimisation, le SIAD met à jour les informations relatives au point idéal et au point nadir pour le problème concerné dans la base d'informations. Si l'usager n'a pas choisi d'approfondir la recherche, le processus d'assistance se termine lorsque le SIAD retourne les résultats de l'optimisation à l'utilisateur. Dans le cas contraire, lorsque l'usager « clique » sur une solution de l'ensemble des solutions présenté, le SIAD calcule automatiquement la pondération accordée aux différents objectifs de la solution indiquée par le décideur. Pour effectuer ce calcul, le SIAD va rechercher dans la base d'informations les données sur le point idéal et le point nadir du problème traité afin de déterminer la pondération qui minimisera la distance normalisée au point idéal. Une fois cette pondération déterminée, la deuxième phase d'optimisation est lancée en fonction de la pondération calculée et en utilisant l'AG^{CP}. La Figure 5.3 illustre le lancement de la seconde phase d'optimisation.

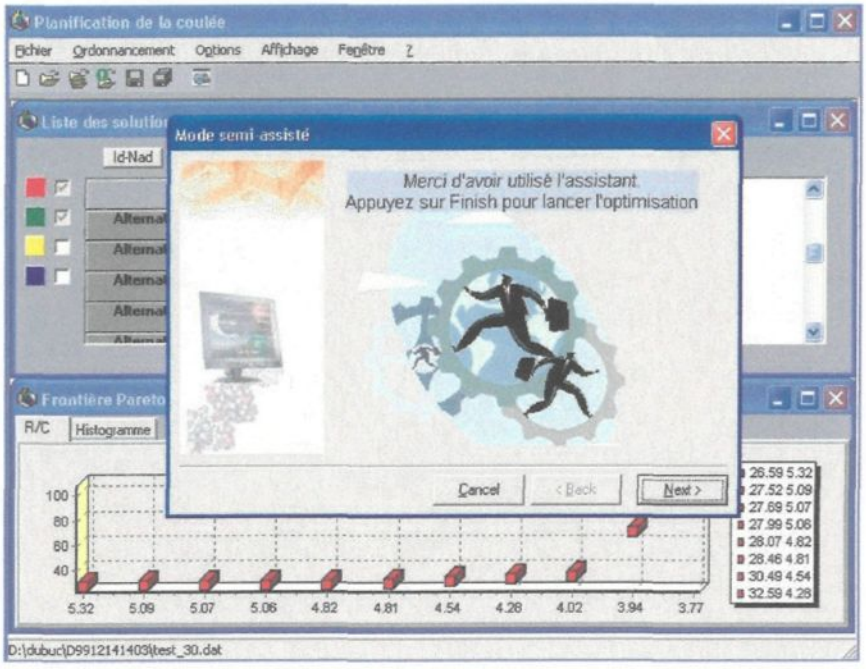


Figure 5.3 : Lancement automatique de la recherche de compromis

Finalement, la quatrième et dernière option du mode semi-assisté, *chercher des solutions de compromis*, permet à l'usager d'effectuer la recherche de solutions de compromis sans avoir à effectuer au préalable une approximation de la frontière Pareto. Cependant, il est important de noter ici qu'il faut, pour que cette option soit disponible, que le SIAD dispose au préalable des données relatives au point idéal et nadir du problème. Comme pour l'option précédente, l'usager est tout d'abord amené à indiquer le carnet de commandes ainsi que le nombre d'objectifs qu'il souhaite optimiser. Par la suite, le SIAD demande à l'usager de définir l'importance relative de chacun des objectifs. Pour cela, une jauge graduée de *Pas important* à *Très important* est présentée à l'utilisateur, comme l'indique la Figure 5.4. Lorsque l'usager fait varier cette jauge, le SIAD inscrit

automatiquement le poids correspondant au niveau d'importance de l'objectif sélectionné en s'assurant que la somme des poids de tous les objectifs est égale à un.

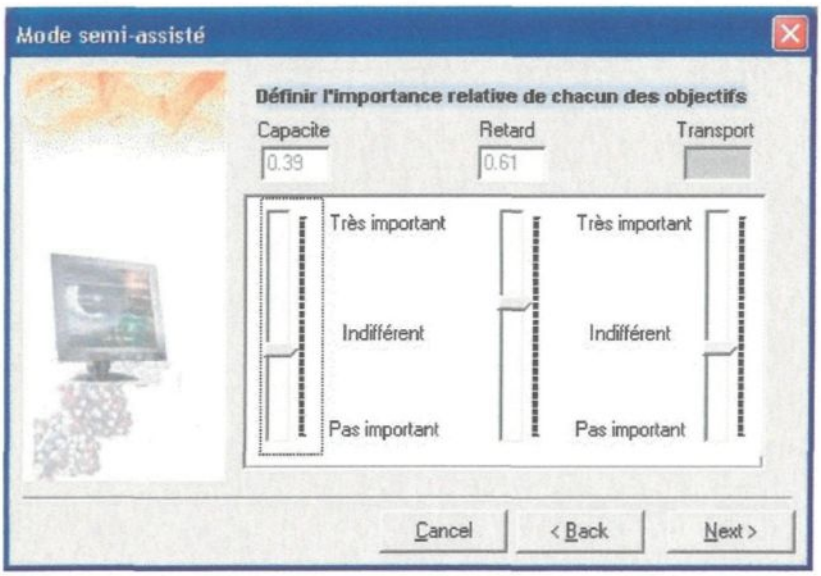


Figure 5.4 : Mécanisme de définition de l'importance relative de chacun des objectifs du SIAD

Une fois cette étape achevée, le SIAD lance le processus de recherche de solutions de compromis en utilisant l'AG^{CP}.

5.6.2 Le mode utilisateur avancé

Le mode utilisateur avancé s'adresse au décideur expérimenté. Dans ce mode, l'assistance est réduite au strict minimum et n'intervient qu'à la demande de l'usager. Ce mode permet, entre autres, à l'usager de paramétrer lui-même le logiciel, de choisir l'approche de résolution du problème qu'il envisage d'utiliser et de choisir le type d'algorithme à appliquer lors des différentes phases d'optimisation. De plus, un modèle de

résolution supplémentaire est disponible à partir de ce mode : l'algorithme du NSGAII. Il est important de noter que le passage d'un mode à l'autre peut être effectué à tout moment.

5.6.3 Visualisation des résultats

Une fois la résolution terminée, il est maintenant temps de visualiser les résultats obtenus. Afin de mieux synthétiser les résultats des différentes phases d'optimisation et ainsi de faciliter le processus de décision, le SIAD proposé permet d'afficher les informations obtenues sous différentes formes. On retrouve, par exemple, des graphiques à deux et à trois dimensions, des diagrammes de Gantt, des nuages de points. Dans cette section, nous nous intéresserons plus particulièrement à la représentation des solutions d'un point de vue multi-objectifs. La Figure 5.5 illustre une représentation de l'ensemble des solutions trouvées par le SIAD pour un carnet de 50 commandes du problème industriel présenté au chapitre précédent. La fenêtre principale est composée de deux parties, numérotée 1 et 2 à la Figure 5.5, qui offrent des fonctionnalités différentes.

Le haut de la fenêtre (partie 1) affiche l'ensemble des solutions trouvées par le système sous forme de grille, chaque colonne correspondant à un objectif donné. Les différents objectifs sont identifiés à l'aide d'une couleur, rouge pour le retard, vert pour la perte de capacité de la machine à couler et jaune pour la perte de capacité de livraison. En « double cliquant » sur une solution de la grille, l'utilisateur peut voir en détail, à l'aide d'un diagramme de Gantt, l'ordre de planification des commandes dans le temps proposé par le SIAD pour la solution sélectionnée.

Dans la partie 2 de la fenêtre, le même ensemble de solutions est représenté cette fois-ci à l'aide de courbes en deux dimensions qui tiennent en compte les objectifs deux à deux.

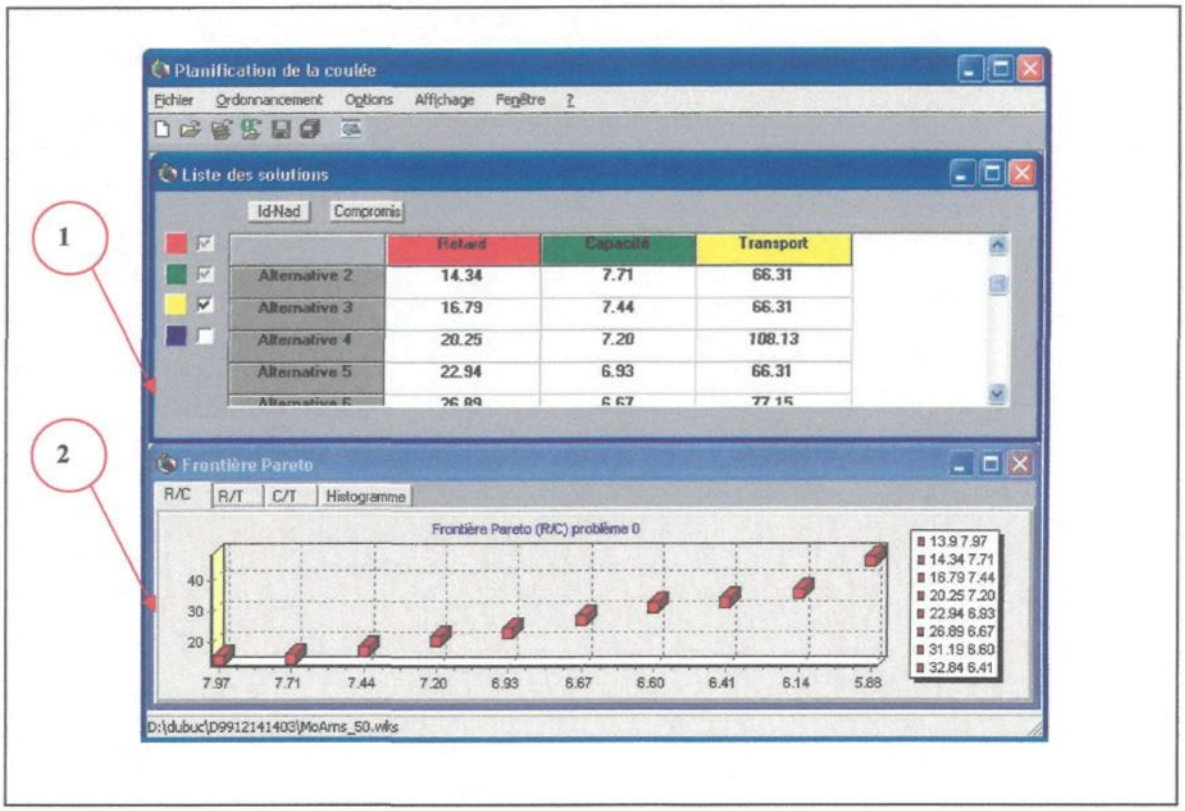


Figure 5.5 : Représentation de la frontière pseudo Pareto par le SIAD. Carnet de 50 commandes du contexte industriel.

Mis à part les courbes en deux dimensions, l'utilisateur peut aussi visualiser l'ensemble des solutions sous forme d'une série d'histogrammes en « cliquant » sur l'onglet *histogramme* de la partie inférieure de la Figure 5.5. La Figure 5.6 illustre cette forme de représentation pour un carnet de 50 commandes. Dans ce cas, chaque solution de l'ensemble trouvé par le système est représentée à l'aide d'un histogramme où chaque barre indique la valeur de la solution pour un objectif donné.

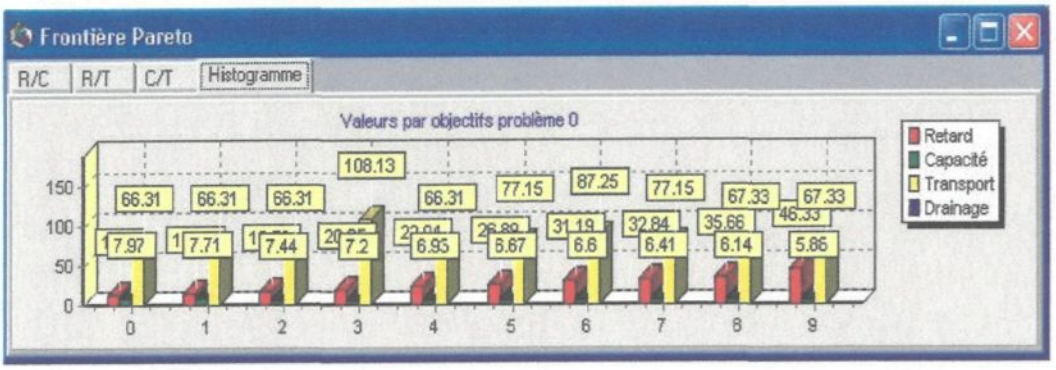


Figure 5.6 : Exemple de représentation des solutions sous forme d'histogramme

Jusqu'à présent, les graphiques proposés utilisaient des représentations en deux dimensions pour présenter les résultats de l'optimisation au décideur. Ce type de représentation est très utile pour représenter des problèmes bi-objectifs. Cependant, au-delà de passé deux objectifs, ce genre de diagramme devient très rapidement inadapté et il faut avoir recours à d'autres types de graphiques pour représenter les différentes solutions trouvées par les différentes phases d'optimisation.

Le problème d'ordonnancement industriel étudié est un problème multi-objectifs nécessitant la minimisation de trois objectifs. Lorsque l'optimisation est effectuée sur les trois objectifs, l'ensemble des solutions peut, comme précédemment, être visualisé sous forme d'une grille telle qu'illustrée dans la partie 1 de la Figure 5.5. Le système permet aussi de représenter les solutions dans leur ensemble à l'aide de différents graphiques en trois dimensions. À la Figure 5.7, l'ensemble des solutions est représenté sous forme de nuage de points en trois dimensions, chaque point étant relié à l'autre à l'aide d'un algorithme de triangulation complète. Pour connaître la valeur d'une solution pour chaque

objectif, il suffit à l'usager de « cliquer » sur un des points du graphique et le SIAD indiquera alors la valeur de chacun des objectifs pour la solution sélectionnée.

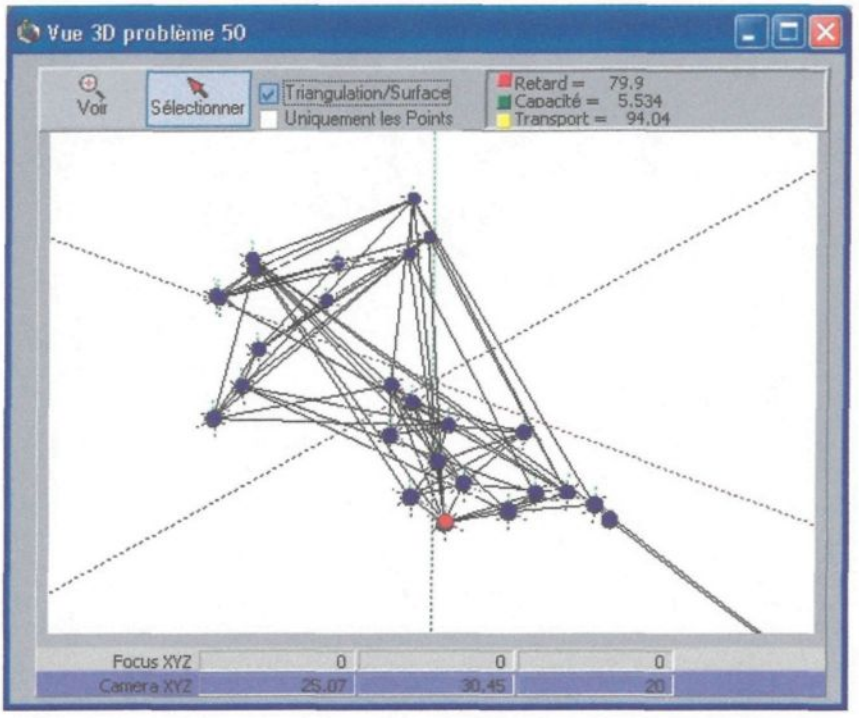


Figure 5.7 : Représentation de l'ensemble des solutions en 3 dimensions par le SIAD. Problème de 50 commandes du contexte industriel.

Dans le cas où l'usager s'intéresse à un groupe de solutions en particulier qu'il veut mettre en relation, le SIAD permet de représenter ce groupe de solutions sous forme d'axes X, Y, Z, chacun des axes indiquant la valeur d'un des trois objectifs de la solution. De cette manière, le décideur peut comparer les différentes solutions du groupe une par rapport à l'autre et ce, objectif par objectif. La Figure 5.8 illustre la représentation d'un groupe de trois solutions à l'aide d'un axe en trois dimensions. Les valeurs de chaque solution pour les différents objectifs sont indiquées dans le cadre en haut à droite de la figure. Notons que

dans le prototype proposé, le nombre maximum de solutions qui peuvent être mises en relation est limité à trois pour des raisons de visibilité.

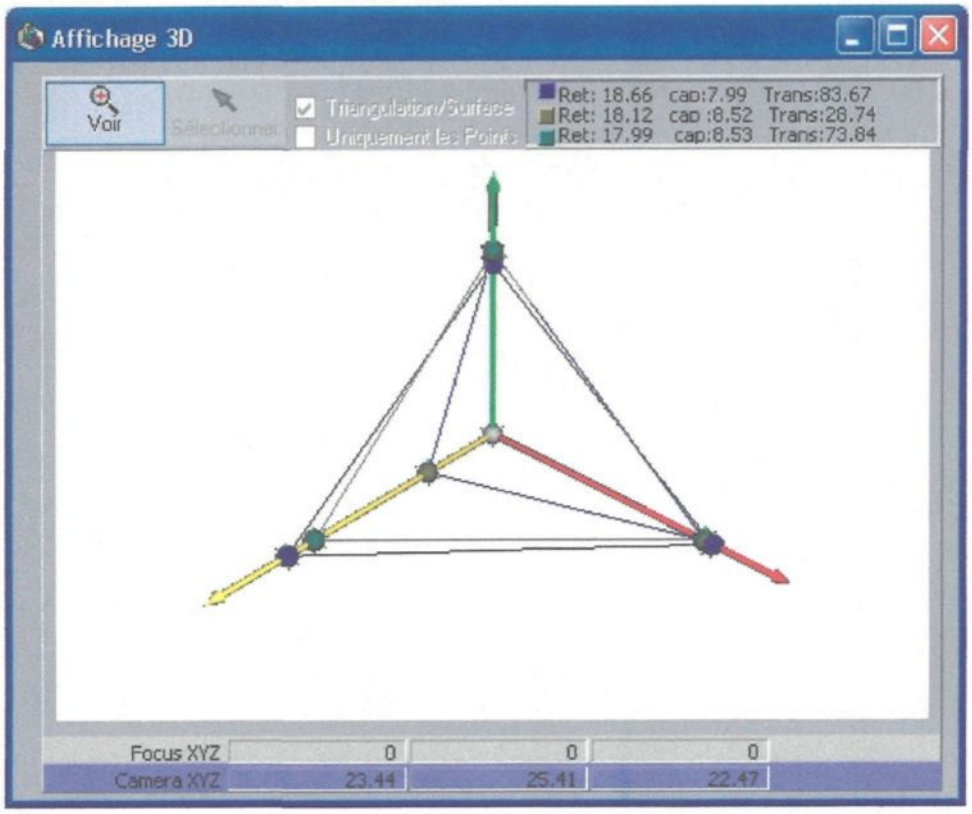


Figure 5.8 : Comparaison visuelle de trois solutions à l'aide d'un axe en 3 dimensions. Problème de 50 commandes du contexte industriel.

Notre exemple d'illustration ne comportait que trois objectifs, cependant le SIAD proposé permet l'affichage de solutions pour des problèmes comportant jusqu'à quatre objectifs. En effet, des graphiques comportant un axe en quatre dimensions ont été intégrés au SIAD et permettent, le cas échéant, d'afficher les solutions d'un tel problème.

5.7 Conclusion

Dans le présent chapitre, un système interactif d'aide à la décision permettant la résolution de problème multi-objectifs a été présenté. Il est composé d'une interface utilisateur, d'une base d'informations et d'une base de modèles composée de quatre algorithmes génétiques. Grâce à ces différents modèles, l'utilisateur peut résoudre le problème de différentes façons selon ses besoins. Le SIAD proposé assiste l'utilisateur tout au long de son processus de décision sans jamais se substituer à lui.

Le fonctionnement du SIAD a été illustré à l'aide d'un problème d'ordonnancement industriel rencontré dans un centre de production d'aluminium. L'utilisation d'un contexte réel, qui incorpore des contraintes additionnelles par rapport aux problèmes théoriques, a permis de mieux montrer l'apport que ce type de système peut avoir dans le milieu industriel.

De manière à éviter le manque de réutilisabilité qui est une des principales causes d'échec des SIAD, le prototype proposé a été développé selon une approche objet. Le choix de cette approche permet de rendre le système adaptatif dans le temps en facilitant les éventuelles opérations de mise à jour et de maintenance du SIAD. De plus, le prototype développé est facile d'utilisation et possède les principales caractéristiques recherchées dans un SIAD dont il a été fait mention au Chapitre 2 :

- il permet une recherche heuristique à l'aide de différentes approches de résolution de la base de modèle;
- il apporte une aide à différentes catégories d'utilisateurs en les regroupant en deux groupes;

- il laisse le contrôle à l'usager tout au long du processus de décision sans jamais se substituer à lui;
- il intègre différents modèles dans sa base de modèles ce qui permet au décideur d'explorer les problèmes sous différents aspects et ainsi d'élaborer différentes stratégies afin de faire face à de nouvelles conditions; et
- il possède une interface simple d'utilisation ce qui permet une prise en main plus rapide.

La combinaison de tous ces facteurs nous permet de penser que le SIAD proposé constitue un outil de résolution efficace pour un PMO.

CHAPITRE 6

CONCLUSION

Les SIAD ont pour principal objectif d'apporter un support à la prise de décision pour des problèmes complexes. Les SIAD sont des systèmes coopératifs permettant une répartition évolutive des compétences entre l'utilisateur et la machine et offrant une bonne intégration des deux entités (homme et machine) dans le processus de décision. Dans de nombreux secteurs de l'industrie, les décideurs sont confrontés à des problèmes complexes pour lesquels prendre une décision nécessite en général l'optimisation simultanée de plusieurs objectifs souvent contradictoires. En fait, la plupart des problèmes d'optimisation multi-objectifs sont des problèmes complexes. Les méthodes de résolution de problèmes multi-objectifs ne sont pas un sujet récent, toutefois ce domaine vit un renouveau depuis quelques années. En particulier, les algorithmes génétiques suscitent de plus en plus d'intérêt auprès des chercheurs et apparaissent maintenant comme une alternative intéressante aux méthodes classiques grâce à leur faculté à exploiter de vastes espaces de recherche et à générer plusieurs solutions de compromis en une seule étape d'optimisation. Il est donc intéressant de concevoir des SIAD permettant de résoudre des problèmes multi-objectifs à l'aide d'une approche basée sur des algorithmes génétiques. Cependant, la combinaison de ces différents domaines de recherche est encore à ses premiers balbutiements. Le travail réalisé dans ce mémoire a permis de mieux comprendre l'importance de la contribution de ce type d'outils dans la résolution de problèmes d'optimisation multi-objectifs.

Plus précisément, ce mémoire a permis d'apporter une certaine contribution dans les domaines des SIAD et de l'optimisation multi-objectifs par la réalisation des objectifs fixés au départ de cette recherche. Le premier objectif consistait à *développer un algorithme génétique multi-objectifs efficace permettant d'approximer l'ensemble Pareto optimal en*

une seule exécution. Cet objectif a été atteint grâce à l'élaboration et au développement de l'AG^{MOPI}, un algorithme génétique multi-objectifs hybride présenté au Chapitre 4. L'AG^{MOPI} combine une procédure génétique basée sur les concepts d'élitisme, de niche et de dominance Pareto avec des opérateurs de recherche locale. Même si cette approche se base sur des techniques établies, elle se distingue des autres algorithmes génétiques multi-objectifs de la littérature dans la manière où ces différentes techniques sont appliquées. Tout d'abord, l'AG^{MOPI} utilise un double niveau d'archivage afin d'assurer la conservation des individus Pareto optimaux tout au long du processus de recherche. De plus, une méthode de classement permet à l'algorithme d'assurer que les meilleurs individus rencontrés sont toujours conservés dans la population courante. Au niveau de la sélection, l'AG^{MOPI} utilise une sélection par tournoi favorisant les individus non dominés et situés dans des régions dépeuplées. Finalement, afin d'obtenir un bon compromis entre exploitation et exploration, deux procédures de recherche locales sont combinées dans cette approche de manière à intensifier la recherche. Les performances de l'AG^{MOPI} ont été illustrées à l'aide d'un problème d'ordonnancement industriel rencontré dans une entreprise de production d'aluminium décrit par Gravel *et al.* [2002]. Dans les expérimentations réalisées, il a été possible d'observer l'efficacité de l'AG^{MOPI} en le comparant avec ceux du NSGAII, un algorithme de l'état de l'art, et des ensembles de référence. En particulier, les résultats expérimentaux ont montré que l'AG^{MOPI} parvenait à bien couvrir l'espace des solutions et obtenait des résultats au moins identiques ou supérieurs à ceux du NSGAII sur tous les problèmes testés pour l'optimisation de deux ou de trois objectifs. De plus, les travaux ayant mené à l'élaboration de l'AG^{MOPI} ont permis d'abaisser les minimums connus

pour cinq des neufs problèmes testés sur l'objectif du retard tout en obtenant un résultat identique au minimum connu sur les deux autres objectifs. Cependant, les résultats expérimentaux laissent aussi entrevoir certaines limitations à cette approche en ce qui concerne le temps d'exécution de l'algorithme. On remarque ainsi que le temps d'exécution de l'AG^{MOPI} se dégrade plus rapidement que celui du NSGAII lorsque la taille du problème et le nombre d'objectifs augmentent. Les résultats obtenus justifient la pertinence de l'approche proposée et les limites identifiées suscitent un intérêt certain pour de futurs travaux de recherche et de développement.

Le deuxième objectif visait à *concevoir un SIAD multi-objectifs cohérent doté d'une interface graphique conviviale permettant de guider le décideur lors de son processus de prise de décision*. Pour cela, un prototype de SIAD a été proposé au Chapitre 5 pour la résolution du problème d'ordonnancement industriel. Ce prototype a été élaboré de manière à répondre aux principales caractéristiques communément recherchées dans les SIAD qui ont été présentées au Chapitre 2 :

- il permet une recherche heuristique à l'aide de différentes approches de résolution de la base de modèle;
- il apporte une aide à différentes catégories d'utilisateurs en les regroupant en deux groupes;
- il laisse le contrôle à l'usager tout au long du processus de décision sans jamais se substituer à lui;

- il intègre différents modèles dans sa base de modèles ce qui permet au décideur d'explorer les problèmes sous différents aspects et ainsi d'élaborer différentes stratégies afin de faire face à de nouvelles conditions et
- il possède une interface simple d'utilisation ce qui permet une prise en main plus rapide.

Pour satisfaire à ces différentes caractéristiques, le SIAD proposé est constitué des trois composants fondamentaux à tous SIAD : une base de modèles, une base d'informations et une interface homme-machine. La base de modèles est constituée de quatre approches de résolution heuristiques permettant au décideur de résoudre un problème de différentes manières en fonction de ses besoins. L'interface du SIAD est conviviale et supporte différents modes d'interaction ce qui permet de fournir une aide adaptée à différentes catégories d'utilisateurs. La base d'information du SIAD contribue de son côté à l'homogénéisation et à la centralisation des différentes informations relatives à l'environnement de production. La mise en place de ces différents mécanismes d'interaction et de résolution aide le décideur à simuler le processus de production et à anticiper les effets d'éventuels changements dans l'environnement qui influent sur le cycle de production tels que l'ajout de nouvelles commandes, l'augmentation de capacité de la machine à couler et autres. Cependant, il est important de noter que l'outil développé ne constitue qu'un support à la décision et que la décision finale ainsi que le contrôle du processus de décision restent du ressort du planificateur. De plus, de manière à faciliter d'éventuelles opérations de maintenance et de rendre le système adaptatif dans le temps, le prototype proposé a été élaboré à l'aide d'une approche objet.

Ce travail de recherche a donc non seulement atteint ses objectifs, mais a également ouvert la voie à plusieurs avenues de recherche autant dans le domaine de l'optimisation multi-objectifs que dans celui des SIAD en raison des nombreuses observations et interrogations soulevées ainsi que de certaines limites constatées.

Les expérimentations réalisées pour l'élaboration de l'AG^{MOPI} ont mis en évidence l'importance et surtout la complémentarité des phases d'intensification et de diversification pour résoudre un problème d'optimisation multi-objectifs. En effet, les algorithmes génétiques sont réputés pour effectuer naturellement des phases de diversification. C'est cette qualité qui les rend très utiles dans la résolution de PMO en leur permettant d'exploiter de vastes espaces de recherche et de générer plusieurs solutions de compromis en une seule étape d'optimisation. En ce qui concerne l'intensification, ce type d'algorithmes s'avère cependant moins efficace. Pour palier à cet inconvénient, nous avons combiné, dans l'AG^{MOPI}, un algorithme génétique avec des phases de recherche locale. Cependant, nous nous sommes limités à des méthodes d'amélioration locales simples. Même si l'approche proposée a apporté des résultats significatifs dans la résolution des problèmes tests, tout laisse à croire que sa performance pourrait être améliorée en y incorporant des stratégies d'intensification plus évoluées comme l'optimisation par colonie de fourmis ou la recherche avec tabous pour ne citer que ces deux approches. Il serait donc intéressant d'approfondir les mécanismes d'hybridation entre métaheuristiques afin d'obtenir une approche coopérative entre algorithmes de diversification et d'intensification.

La principale limite de l'AG^{MOPI} concerne la dégradation du temps d'exécution en fonction du nombre d'objectifs et de la taille du problème à traiter. En effet, lorsque la taille

du problème et le nombre d'objectifs augmentent, il est souvent nécessaire d'ajuster en conséquence la taille de la population ainsi que le nombre maximum de générations. Ces augmentations entraînant un accroissement important du temps de calcul de l'algorithme. De plus, les expérimentations ont montré que même en gardant identiques les différents paramètres de l'algorithme, les temps de calcul pouvaient augmenter de façon importante en fonction de la taille du problème ou du nombre d'objectifs. Dans ce travail, nous sommes limités à l'optimisation de problèmes d'ordonnancement de 10 à 140 commandes comportant deux ou trois objectifs. Pour traiter des problèmes de plus grandes tailles et comportant plus de trois objectifs, une solution envisageable à court terme serait de développer une version parallèle ou distribuée de l'AG^{MOPI}. En effet, certains mécanismes comme l'évaluation ou les améliorations locales ne s'appliquent qu'à un seul individu et peuvent donc être exécutés de manière indépendante. Une manière plus évoluée de paralléliser l'algorithme serait de subdiviser la population de l'AG^{MOPI} en plusieurs sous-populations réparties sur différents processeurs qui explorent simultanément l'espace des solutions. Les deux archives seraient alors partagées entre les différentes populations à travers une mémoire centrale.

Lors de l'élaboration du prototype de SIAD, nous avons aussi pu constater qu'il était difficile de représenter graphiquement les solutions pour les présenter au décideur lorsque le nombre d'objectifs du problème dépassait deux. Il est donc important d'approfondir les travaux existants afin d'apporter des modes de visualisation plus satisfaisants.

Dans ce travail, les composants optionnels des SIAD, comme la base de connaissances, ont été mis de côté et les efforts se sont concentrés sur les composants standards que sont

l'interface homme-machine, la base de modèles et la base d'informations. Le SIAD pourrait toutefois être étendu par l'ajout d'une base de connaissance utilisant des techniques d'analyse de données et d'apprentissage à partir d'observations du comportement des utilisateurs en phase de résolution de problèmes. De cette manière, il sera possible de détecter des régularités dans le comportement des utilisateurs et ainsi, d'une part, de mieux adapter la coopération entre l'homme et la machine et, d'autre part, de détecter les stratégies de résolution de ceux-ci dans des contextes particuliers.

Finalement, des travaux supplémentaires dans le but d'améliorer les temps d'exécution des différents algorithmes de la base de modèle, d'ajouter d'autres modèles de résolution ou de mettre en place de nouveaux modes d'interaction permettraient d'obtenir un SIAD encore plus performant. Même si l'outil proposé a été testé à partir d'un contexte réel, il serait intéressant de l'implanter dans une entreprise afin de recueillir la réaction des gestionnaires face à ce type d'outils et ainsi de mieux l'adapter à leurs besoins. De cette manière, l'approche proposée pourra être généralisée à d'autres secteurs de l'industrie.

BIBLIOGRAPHIE

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

Ben Hamida, S. (2001). Algorithmes Évolutionnaires: Prise en Compte des Contraintes et Application Réelle.

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

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

Bohm , B. W. (1976). Software Engineering. IEEE Transactions on computing, p. 1226-1241.

Booch, G. et J. Rumbaugh (1995). Unified Method for Object-Oriented Development, Rational Software Corporation.

Boukachour, H., C. Duvallet et A. Cardon (2000). Multiagent system to prevent technological risk. In Proceedings of ACIDCA'2000.

Chabbat, B. (1997). Modélisation multiparadigme de textes réglementaires, thèse de l'Insa de Lyon.

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

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

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.

Courbon, J.-C. et C. B. Stabell (1986). Artificial intelligence and the design of decision support systems. tutorial of the conference on economics and artificial intelligence, Aix-en-Provence.

Crossland , M. D. et B. E. Wynne (1995). "Spatial Decision Support Systems : an overview of technology and test of efficacy." Decision Support Systems **14**(3): p. 219-235.

Davis, G. B., M. H. Olson, J. Ajenstat et J.-L. Peaucelle (1986). Systèmes d'information pour le management. Volume 1 et 2, Montréal, édition G. Vermette.

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

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

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

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

Dorigo, M. (1992). Optimization, learning and natural algorithms. Italy, Politecnico di Milano.

Druzdzel, M. J. et R. R. Flynn (1999). Decision Support Systems, to appear in Encyclopedia of Library and Information Science, Allen Kent (ed.), Marcel Dekker, Inc.

Du, J. et J. Y. Leung (1990). "Minimizing total tardiness on one machine is NP-hard." Mathematics of Operations Research **15**: p. 483-494.

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

Eierman, M. A. et F. Niederman (1995). "DSS theory: a model of constructs and relationships." Decision Support Systems **14**(1): p. 1-26.

Farenc, C., P. Palanque et J. Vanderdonckt (1994). L'évaluation ergonomique de l'utilisabilité d'une application interactive, est-elle utilisable ? IHM'94. 6iemes journées sur l'ingénierie des interfaces homme-machine, Lille.

Finlay, P. (1994). Introducing Decision Support Systems, Oxford: Blackwell.

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, San Mateo, California, p. 416- 423,.

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

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.

Fox, M. (1981). Factory modelling, simulation and scheduling in the intelligent management system. In Proceedings of the 7th International Joint Conference on Artificial Intelligence.

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

Gachet, A. (2001). A Framework for Developing Distributed Cooperative Decision Support Systems - Inception Phase. Conference Proceedings, 2001 Informing Science Conference, June 19-22 Kraków, Poland.

Gagné, C. (2001). L'ordonnancement industriel : stratégies de résolution métaheuristiques et objectifs multiples.Thèse de doctorat. Faculté des sciences de l'administration, Université Laval.

Gagné, C., M. Gravel et W. L. Price (2002b). "Algorithme d'optimisation par colonie de fourmis avec matrices de visibilité multiples pour la résolution d'un problème d'ordonnancement industriel." Information Systems and Operational Research (INFOR) **40**(2): p. 259-276.

Gagné, C., M. Gravel et W. L. Price (2004a). "Using metaheuristic compromise programming for solution of multiple objective scheduling problems."

Gagné, C., M. Gravel et W. L. Price (2004b). "Optimisation multi-objectifs à l'aide d'un algorithme de colonie de fourmis." (Soumis pour publication).

Gagné, C., W. L. Price et M. Gravel (2002a). "Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence dependent setup times." Journal of the Operational Research Society **53**(8): p. 895-906.

Gannon, M. J. (1979). Organizational Behaviour: A Managerial and Organizational Perspective, Boston: Little, Brown.

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.

Garlatti, S. (1996). Tutorial: Multimédia et systèmes d'aide à la décision en situation complexe. in 43th meeting of the eurpean working group "Multicriteria Aid for Decisions", Brest.

Ghasemzadeh, F. et N. P. Archer (2000). "Project portfolio selection through decision support." Decision Support Systems and Electronic Commerce.

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

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

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.

Gorry, G. A. et S. Scott-Morton (1971). "A Framework for Management Information Systems." Sloan Management Review **13**(1).

Gravel, M., W. L. Price et C. Gagné (2002). "Scheduling continuous casting of aluminium using a multiple-objective ant colony optimization metaheuristic." European Journal of Operational Research **143**(1): p.218-229.

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

Hansen, J. V., R. D. Meservy et al. (1995). "Case-based reasoning application techniques for decision support." Intelligent systems in accounting, finance and management.

Harel, D., A. Pnueli, J. P. Schmidt et R. Sherman (1987). On the Formal Semantics of State Charts. Proc. 2nd Symp. on Logic in Computer Science (LICS 87), IEEE Computer Society Press,pp. 54-64.

Hättenschwiler, P. (1993). Computer Assisted Top Down Modeling, Modeling Tools for Decision Support, University of Fribourg: p. 101-131.

Hättenschwiler, P. (1999). Neue Konzepte der Entscheidungs-unterstützung, Working Paper 99-4, Institute of Informatics, University of Fribourg.

Hättenschwiler, P., M. Moresino et A. Schroff (1998). Rapid Prototyping of Decision Support System. conference proceedings of ICSC Symposium, Tenerife.

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

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

Holtzman, S. (1989). Intelligent Decision Systems, Addison Wesley.

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.

Huber, G. P. (1982). Group Decision Support Systems as Aids in the Use of Structured Group Management Techniques. DSS-82, Conference Proceedings.

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

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

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

Jacobson, I., M. Christerson, M. Jonsson et P. Overgaard (1992). OO Software Engineering,A Use Case Driven Approach. Addison-Wesley.

Johnson, D. S. et L. A. McGeoch (1997). The traveling salesman problem: a case study in local optimization, Local Search in Combinatorial Optimization, E.H.L. Aarts & J.K. Lenstra editor, John Wiley and Sons Ltd.

Keen, P. G. W., Ed. (1981). Decision Support Systems: A Research Perspective, in Decision Support Systems: Issues and Challenges, Pergamon Press.

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

Klein, D. A. (1988). Integrating Artificial Intelligence and Decision Theory to Forecast New Markets, IBM Research Division, Yorktown Heights.

Klein, M. et V. Tixier (1971). SCARABEE: a data and model bank for financial engineering and research. IFIP congress, North Holland.

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.

Kreie, J., T. P. Cronan , J. Pendley et J. S. Renwick (2000). "Applications development by end-users: can quality be improved?" Decision Support Systems.

Lévine, P. et J. Pomerol (1990). Systèmes interactifs d'aide à la décision et systèmes experts, Edition Hermès.

Liberatore, M. L. et G. J. Titus (1983). "The practice of management science in R&D project selection." Management Science **29**: p. 962-974.

Little, J. D. (1970). "Models and Managers: The Concept of a Decision Calculus." Management Science.

Mahfoud, S. W. (1995). Niching Methods for Genetic Algorithms, IlliGAL TR. n° 95001,University of Illinois at Urbana-Champaign, Urbana.

Myers, B. A. (1995). "User Interface Software Tools." ACM Transactions on Computer-Human Interaction. **2**(1): p. 64-103.

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.

Or, I. (1976). Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking, Ph.D. thesis, Northwestern University, Evanston, Illinois.

Osman, I. H. et G. Laporte (1996). "Metaheuristics: A bibliography." Annals of Operations Research(63): p.513-623.

Palanque, P., J. B. Long, J. C. Tarby, M. F. Barthet et K. Y. Lim (1994). Conception d'applications ergonomiques : une méthode pour informaticiens et une méthode pour ergonomes. Actes du congres ERGO-IA'94 (Ergonomie et Informatique Avancée), Biarritz, France.

Pareto, V. (1896). Cours d'économie politique. Lausanne, F. Rouge.

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.

Potvin, J.-Y. (1996). "Genetic Algorithms for the Traveling Salesman Problem." Annals of Operations Research **63**: p. 339-370.

Power, D. J. (1999). A Brief History of Decision Support Systems., DSS Resources, World Wide Web.

Probst, A. R. (1984). "Les systèmes d'aide à la décision: rôle, structure et évolution." Revue Gestion: p. 13-19.

Reix, R. (2000). Systèmes d'information et management des organisations. 3ième éd., Paris, Vuibert.

Rudnicka, A. et G. R. Madey (2001). A Framework for effective user interface design for web-based electronic commerce applications. Conference Proceedings, 2001 Informing Science Conference, June 19-22 Kraków, Poland.

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.

Sage, A. P. (1991). Decision Support System Engineering. New York, Wiley.

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.

Schroff, A. (1998). An Approach to User Oriented Decision Support Systems. Inaugural-Dissertation Nr. 1208, Druckerei Horn, Bruchsal.

Silverman, B., W. (1986). Density estimation for statistics and data analysis, Chapman and Hall, London.

Sprague, R. et H. Watson (1993). Decision Support Systems - Putting Theory into Practice, 3 rd Edition, Englewood Cliffs: Prentice Hall.

Srivinas, N. et K. Deb (1994). "Multiobjective Optimization using Non-dominated Sorting in Genetic Algorithms." Evolutionary Computation **2**(3): p. 221-248.

Sun, M. et R. E. Steuer (1996). "Quad-Trees and Linear Lists for Identifying Nondominated Criterion Vectors." Journal on Computing **8**(4): p367-375.

Talbi, E.-G. (1999). Métaheuristiques pour l'optimisation combinatoire multi-objectif: état de l'art, Rapport CNET(France Telecom).

Talbi, E.-G. (2000). Une taxinomie des métaheuristiques hybrides. ROADEF'2000.

Totton, K. A. E. et P. G. Flavin (1991). "An overview of Computer Aided Decision Support."

Turban, E. (1993). Decision Support and Expert Systems. New York, Macmillan.

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, D. A. (1999). Multiobjective, Evolutionary Algorithms : Classification, Analyses and New Innovation,. Air Force Institute of Technology, United States.

Whitley, D., T. Starkweather et D. Fuquay (1989). Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. Proc. of the 3rd Int'l. Conf. on GAs, Morgan Kaufmann.

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.

Zitzler, E. (2001). Evolutionary Algorithms for Multiobjective Optimization. EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems.

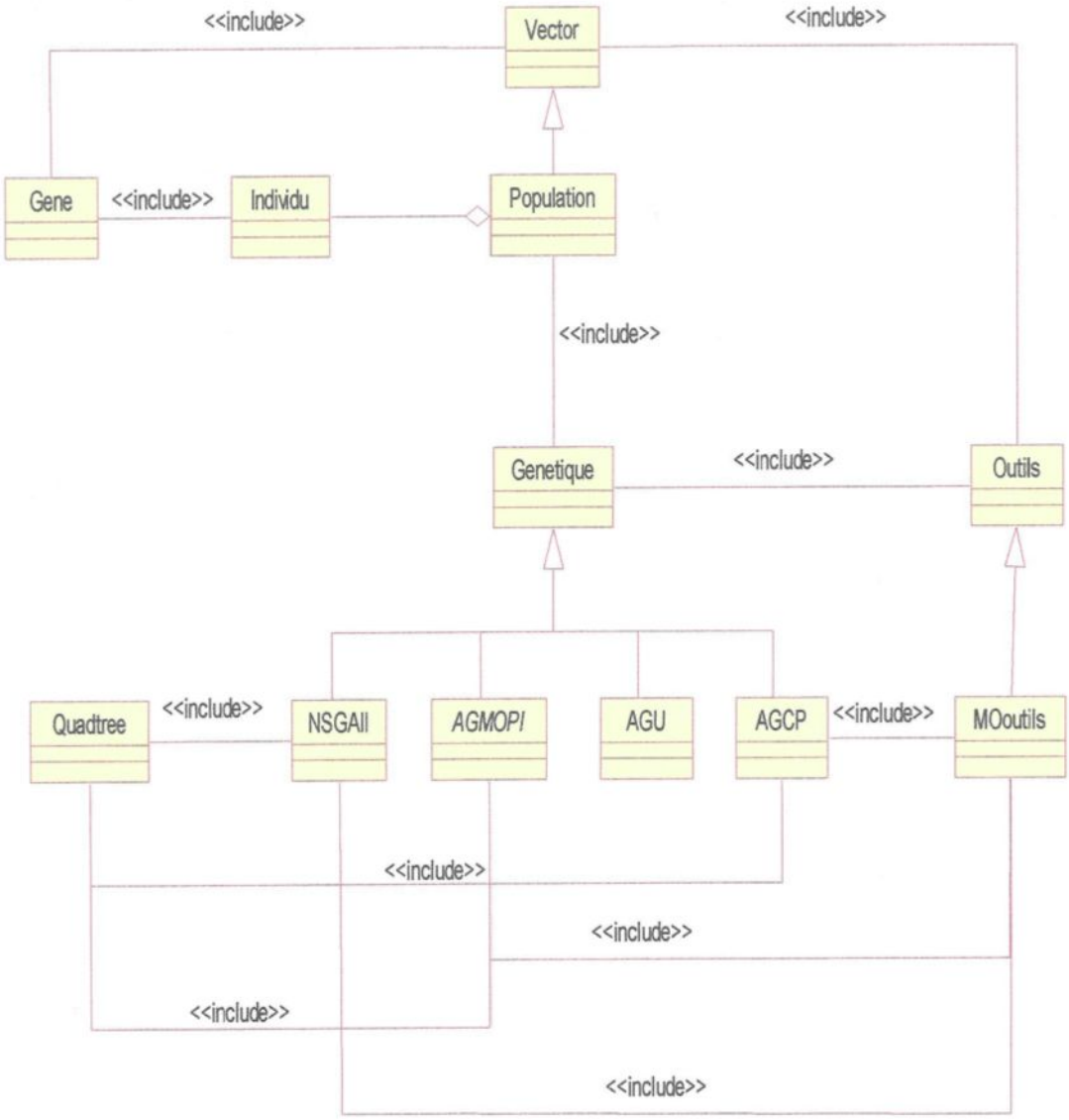
Zitzler, E., K. Deb et L. Thiele (1999). Comparison of Multiobjective Evolutionary Algorithms : Empirical Results,, Tech. Report n° 70, Swiss Federal Institute of Technology (ETH), Zurich, 1999.

Zitzler, E., M. Laumanns et L. Thiele (2001). SPEA2 : Improving the Strength Pareto Evolutionary Algorithm, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Zitzler, E. et L. Thiele (1998). An Evolutionary Algorithm for Multiobjective Optimization : The Strength Pareto Approach, TIK-Report n° 43,.

ANNEXE 1

HIÉRARCHIE DES CLASSES DE LA BASE DE MODÈLE DU SIAD


Vector
<<include>>
<<include>>
Gene
<<include>>
Individu
Population
<<include>>
Genetique
<<include>>
Outils
Quadtree
<<include>>
NSGAII
AGMOPI
AGU
AGCP
<<include>>
MOoutils
<<include>>
<<include>>
<<include>>
<<include>>