UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

RÉSOLUTION DE CONFLITS ET SÉQUENÇAGE D'AVIONS PAR ALGORITHMES ÉVOLUTIONNAIRES MULTIOBJECTIFS

Thèse de doctorat
Spécialité : génie électrique

Etienne LACHANCE

Jury : Charles-Antoine BRUNET (directeur)
François BOURDON
Philippe GOURNAY
Frédéric MAILHOT

Sherbrooke (Québec) Canada 27 mars 2014

À mes amours, Rachel, Audrey et Laurie.

RÉSUMÉ

L'augmentation grandissante du trafic aérien rend le travail des contrôleurs aériens de plus en plus ardu, spécialement en ce qui a trait aux tâches de résolution de conflits et de séquençage d'avions en arrivée. L'automatisation de la résolution de conflits et du séquençage restent toujours un problème ouvert aujourd'hui. L'automatisation de ces deux problèmes permettrait d'une part de mieux modéliser le comportement des contrôleurs aériens dans un simulateur de vol, ou d'améliorer les outils de gestion du traffic aérien.

Les caractéristiques combinatoires de ces problèmes conduisent à l'utilisation de techniques numériques stochastiques, plus spécifiquement des algorithmes évolutionnaires. De plus, les nombreux paramètres intervenant dans une situation de gestion de trafic aérien incitent à l'utilisation d'algorithmes multiobjectif. Dans un premier temps, un algorithme génétique multiobjectif (SPEA-MOD) et un algorithme de colonies de particules (PSO-MO) également multiobjectif ont été développés. Ces deux algorithmes ont été comparés à des problèmes multiobjectif contraints et non-contraints. Les résultats ont montré que SPEA-MOD et PSO-MO sont en général supérieurs à ce que l'on rapporte dans la littérature.

Dans un deuxième temps, les deux algorithmes ont résolu plusieurs situations conflictuelles de la phase de vol en route (régime de croisière). Les instructions fournies par les algorithmes peuvent être en deux ou en trois dimensions. Les objectifs et les contraintes représentent des paramètres tels que la minimisation d'instructions fournies aux avions et une séparation minimale entre les avions. De ces solutions numériques réalisées, l'algorithme SPEA-MOD s'est avéré particulièrement efficace à des problèmes fortement contraints.

Une modélisation novatrice de trajectoires complexes a permis de résoudre des problèmes de séquençage d'avions dans la phase d'arrivée. Le séquençage d'avions en arrivée par un algorithme évolutionnaire fut réalisé pour la première fois dans le cadre de cette recherche. Cette modélisation a également rendu possible la résolution de conflits de deux flux d'avions se croisant.

Mots-clés : Algorithme génétique, algorithme de colonies de particules, multiobjectif, résolution de conflits, contrôle aérien.

REMERCIEMENTS

Je voudrais d'abord remercier M. Charles-Antoine Brunet, mon directeur de recherche, pour toute l'aide apportée dans le cadre de ce projet ainsi que pour les conseils et remarques pertinentes qu'il m'a prodigués. Il a toujours crû au potentiel de mes travaux, malgré le fait que la majorité du travail s'est fait à temps partiel. Je lui suis également très reconnaissant pour le temps qu'il a su consacré à ce projet.

Je remercie François Bourdon, d'Adacel, qui en 2004 a accepté l'idée de partenariat de recherche avec l'université de Sherbrooke. Sans lui, je n'aurais pas découvert le monde fascinant du contrôle aérien.

Je remercie mes anciens collègues d'Adacel, Louis Beamier, Martin Brodeur, Daniel Michaud et Andrée Champagne pour les maintes discussions reliées à cette recherche que nous avons eues.

Je remercie également mes parents, Robert et Lyane, pour les maintes relectures de cet ouvrage et pour leur support.

Je voudrais également remercier les membres du jury pour avoir accepté d'évaluer cette thèse de doctorat.

Enfin, je suis particulièrement reconnaissant à ma conjointe Rachel. Sans son support moral et sa compréhension ce projet n'aurait pu voir le jour.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SYMBOLES

Symbole	Définition
deg	Degrés
km	Kilomètre
kts	Noeud (unité de vitesse)
nm	Mile nautique
sec	Second

LISTE DES ACRONYMES

Acronyme	Définition
ADS-B	Automatic Dependent Surveillance-Broadcast
AG	Algorithme Génétique
AGL	Above Ground Level
ASL	Above See Level
ATC	Air Traffic Control
ATCiB	Air Traffic Control in a Box
BADA	Base of Aircraft Data
CSP	Constraint Satisfaction Problem
CTAS	Center TRACON Automation System
DDL	Degrés De Liberté
ENAC	École Nationale d'Aviation Civile
ETA	Estimated Time of Arrival
GA	Genetic Algorithm
GPS	Global Positioning System
IFR	Instrument Flight Rules
MPL	Multi-Crew Pilot License
OACI	Organisation de l'Aviation Civile Internationale
PSO	Particle Swarm Optimization
PSO-MO	Algorithme PSO Multiobjectif
ROC	Rate of Climb
SI	Système International
SPEA	Strength Pareto Evolutionary Algorithm
SPEA-MOD	Algorithme SPEA Modifié
STAR	Standard Terminal Arrival Route
STCA	Short Term Conflict Alert
STL	Standard Template Librairy
TCAS	Traffic Alert Collision Avoidance System
TR	Trajectoire de Référence
UML	Unified Modelling Language
VFR	Visual Flight Rules

CHAPITRE 1

INTRODUCTION

Les travaux de recherche de cette thèse et les résultats qui en découlent, trouvent applications dans deux domaines très connexes de l'aviation, soit l'entraînement au pilotage et l'apport de nouveaux outils au contrôle aérien.

L'entraînement au pilotage d'aéronefs en simulateurs de vol est plus sécuritaire, plus efficace et surtout meilleur marché que l'entraînement dans un véritable aéronef. L'avancement technologique a permis d'accroître le niveau de fidélité et de réalisme des différents types de simulateurs de vol. Les différentes tâches d'un équipage peuvent se regrouper dans les trois catégories suivantes : faire voler l'aéronef, naviguer et communiquer. Il est possible de développer, de perfectionner et d'évaluer les habilités des deux premières catégories en simulateur, ce qui n'est pas le cas pour la dernière.

À ce jour, de par la règlementation, les manufacturiers de simulateurs de vol ont concentré leurs efforts sur la modélisation des aéronefs, comme les systèmes avioniques et la commande de vol, ainsi que sur l'aspect visuel des aéroports. L'environnement de trafic aérien ou d'ATC (*Air Traffic Control*) y est absent. L'absence de cet environnement rend la pratique de communication structurée ardue, spécialement pour les nouveaux pilotes. La plupart du temps, c'est l'instructeur qui joue le rôle des divers contrôleurs aériens en simulateur. La pratique des communications doit donc se parfaire dans un véritable aéronef, ce qui n'est pas souhaitable, à cause des coûts d'opérations élevés des avions.

Pour remédier à ce manque, l'OACI (Organisation de l'Aviation Civile Internationale) a proposé qu'il y ait un environnement ATC en simulateur pour la nouvelle licence MPL (*Multi-Crew Pilot License*). L'OACI précise également que le besoin d'un tel environnement est particulièrement important pour les phases de vol évoluant dans l'espace aérien terminal, c'est-à-dire pour les phases de vol de départ et d'arrivée. Le but de cette nouvelle licence est de mieux former les nouveaux pilotes à travailler en équipe. Pour ce faire, l'OACI favorise l'utilisation accrue de la simulation, d'où la nécessité d'avoir un meilleur environnement d'ATC pour accroître le réalisme.

Il est maintenant possible d'augmenter le réalisme par l'ajout d'environnement ATC en simulateur, car certaines solutions commerciales ont vu le jour. Les solutions les plus évoluées, comme celle appelée ATCiB (*Air Traffic Control in a Box*) de la compagnie

Adacel, offrent du trafic aérien automatisé ainsi qu'une reconnaissance de la parole permettant de traiter les requêtes ATC de l'équipage en formation. Il existe plusieurs défis technologiques dans la réalisation de ce genre d'environnement qui sont principalement reliés à la reconnaissance de la parole et à la modélisation de comportements humains (pilotes et contrôleurs virtuels). Des techniques de l'intelligence artificielle telles que les agents intelligents et la logique floue y sont employées.

Deux des tâches complexes des contrôleurs aériens sont la résolution de conflits et le séquençage d'avions en arrivée. La résolution de conflits consiste à maintenir une distance minimale entre tous les avions, tant pour des phases de vol d'arrivée que des phases de croisière (en route). Par exemple, dans une situation conflictuelle, le contrôleur pourrait momentanément dévier un avion de sa trajectoire. La tâche de séquençage quant à elle consiste à guider et ordonnancer les avions en arrivée vers la piste d'atterrissage tout en assurant une distance minimale entre les différents avions. Aucun des environnements ATC actuels ne modélisent ces deux comportements, ce qui a pour effet qu'à l'occasion certaines situations ne sont pas réalistes. Les travaux de recherches présentés dans ce document proposent de les réaliser.

Objectifs

À ce jour, plusieurs études traitant de la résolution de conflits dans la phase en route (partie du vol en régime de croisière) ont vu le jour. Dans certaines recherches, on propose des solutions théoriques ou expérimentales impliquant un nombre très limité d'avions. Les plus intéressantes et les plus prometteuses utilisent un algorithme d'optimisation globale tel que les algorithmes génétiques. Selon la revue de la littérature effectuée dans le cadre de cette recherche, il n'y a pas de solution pour le séquençage d'avions basé sur un algorithme d'optimisation globale. Un séquençage basé sur un algorithme d'optimisation permettrait de traiter un plus grand nombre d'avions, ce qui est nécessaire pour bien représenter un espace aérien à densité élevée de trafic.

La présente recherche a pour objectif principal de simuler les tâches de résolution de conflits et de séquençage d'avions pour ainsi améliorer les environnements ATC destinés à l'entraînement de pilotes. Cette thèse apporte une réponse à la question suivante : *Est-il possible de modéliser et d'implémenter la résolution de conflits et le séquençage d'avions de manière optimale afin de mieux modéliser le comportement de contrôleurs aériens ?*

Le deuxième objectif est de donner des outils d'aide à la décision pour les contrôleurs aériens ou améliorer ceux existants. Quelques outils existent sur le marché, mais aucun n'offre une aide à la résolution de conflits ni au séquençage.

Contributions

Cette étude apporte des solutions novatrices à la modélisation de la résolution de conflits dans la phase en route et du séquençage d'avions dans la phase approche. Deux algorithmes d'optimisation évolutionnaire multiobjectif, soit un algorithme génétique et un algorithme de colonies de particules, ont été développés pour résoudre les deux problèmes mentionnés. Selon la revue de la littérature, aucune recherche ne semble avoir utilisé un algorithme d'optimisation évolutionnaire pour résoudre un problème de séquençage d'avions. De plus, aucune recherche ne semble avoir employé un algorithme de colonies de particules dans la résolution des deux problèmes mentionnés.

La seconde contribution est le développement d'une architecture logicielle générique pour simuler les deux tâches mentionnées, rendant ainsi possible d'adapter la solution à des problèmes différents. De plus, la mise en oeuvre d'un prototype de cette architecture, réalisant les deux tâches, fonctionne en temps réel.

Plan du document

Ce document est divisé en deux grandes parties. La première partie, qui est essentiellement une revue de la littérature, est divisée en quatre chapitres.

Le chapitre 2 présente les principes généraux du contrôle aérien. On y présente également un bilan du domaine de la recherche sur les techniques de résolution de conflits et de séquençage d'avions. Les chapitres 3, 4 et 5 traitent des notions d'optimisation multiobjectif, des algorithmes génétiques et des algorithmes de colonies de particules respectivement.

La seconde partie quant à elle traite du modèle de la solution proposée ainsi que des résultats numériques. On y présente d'abord, au chapitre 6, deux nouveaux algorithmes évolutionnaires multiobjectifs et une comparaison de leurs performances sur des problèmes types. Les chapitres 7 et 9 décrivent le modèle de résolution de conflits de la phase en route, puis le modèle du séquençage d'avions en approche respectivement. Des résultats numériques à ces deux problèmes sont présentés au chapitre 8 et à la fin du chapitre 9.

Finalement, une conclusion de cette thèse de doctorat fait un retour sur les résultats obtenus et propose de prochaines avenues de recherche .

CHAPITRE 2
CONTRÔLE AÉRIEN

Ce chapitre présente une revue de la littérature reliée au contrôle aérien. Les sections 2.1 et 2.2 décrivent brièvement les grands enjeux du contrôle aérien ainsi que le projet américain de structure du contrôle aérien respectivement. La revue de la littérature sur la résolution de conflits dans la phase de vol en route (ou régime de croisière) et celle sur le séquençage automatisé du trafic dans la phase de vol arrivée sont traitées aux sections 2.3 et 2.4. Finalement, la section 2.5 fait une synthèse des forces et des faiblesses des études revues.

2.1 Description du contrôle du trafic aérien

La structure de l'espace aérien est divisée en plusieurs sous-espaces, appelés secteurs, et l'interaction entre ces derniers peut être assez complexe. Chaque secteur est desservi par un ou plusieurs contrôleurs aériens, selon l'intensité du flux de trafic. À l'étage supérieur, soit entre 23000 et 60000 pieds, se retrouvent les secteurs de centre dans lesquels évoluent les avions en vol de croisière. De façon générale, l'aéronautique emploie les unités du système d'unités international (SI). Par contre, certaines grandeurs telles que les altitudes, sont majoritairement exprimées dans le système impérial. Sous les secteurs de centre se trouvent les secteurs de région terminale. La région terminale a pour fonction de gérer le trafic en départ et en arrivée à un ou plusieurs aéroports. Finalement, la région terminale contient un secteur de tour par aéroport. De façon générale, un secteur de tour est un espace aérien cylindrique centré sur un aéroport dont le rayon est de 5 miles nautique (nm) et la hauteur de 2500 pieds. La coordination du trafic entre deux secteurs mitoyens est décrite dans une lettre d'entente entre les agences de contrôles.

Il y a également une ségrégation en deux modes de tous les vols d'aéronefs évoluant dans l'espace aérien d'un pays, soit les modes VFR (*Visual Flight Rules*) et IFR (*Instrument Flight Rules*). Dans le mode VFR, les aéronefs doivent assurer eux-mêmes leur séparation les uns par rapport aux autres par des moyens visuels ou par communication. En tout temps la navigation doit se faire en gardant une référence visuelle au sol, sauf pour les vols VFR au-dessus des nuages. Comparativement au vol IFR, ce mode est grandement affecté par les conditions météorologiques. Il est à noter que pour évoluer en IFR, le pilote

doit posséder une qualification de vol aux instruments et l'aéronef doit également être certifié pour le vol IFR. De par sa complexité d'opération, le mode IFR est soumis à une réglementation beaucoup plus stricte.

De plus, en mode IFR, il est obligatoire de déposer un plan de vol lorsqu'on évolue dans l'espace aérien contrôlé. Le plan de vol d'un aéronef contient plusieurs informations dont les aéroports de départ et d'arrivée, la route aérienne choisie et les heures de départ et d'arrivée. Une excellente description de la structure du système aérien actuel est présentée dans le livre de Nolan [Nolan, 1990]. On y décrit également différents aspects du contrôle aérien.

Un des rôles primordial du contrôleur aérien est d'assurer un espacement sécuritaire entre les différents aéronefs. Ce rôle peut se diviser en deux grands volets, soit la détection et la résolution de conflits. Le volet résolution est sans aucun doute le plus complexe. La résolution inclut les actions que le contrôleur doit poser pour maintenir une séparation respectant les normes entre les différents aéronefs. Par exemple, au Canada, ces normes sont décrites dans le manuel d'opération d'ATC (*Air Traffic Control*) [Canada, 1999]. Le contrôle dans les zones terminales ainsi que dans les zones d'approches sont parmi les secteurs les plus complexes et exigeants de l'ATC. La gestion du trafic dans ces zones inclut des tâches telles que déterminer la séquence d'avion en arrivée, donner des vecteurs radar pour éviter des collisions, faire attendre certains avions lorsque le secteur est congestionné, etc. Des vecteurs radar sont des instructions de virages sous forme de changement de direction (cap).

La majorité du temps les avions volent à des altitudes de croisière qui se situent à plus de 30000 pieds. Le trafic à ces altitudes évolue dans la structure en route, ce qui facilite grandement sa gestion. Les avions suivent des routes préétablies à différentes altitudes. Vers la fin de leurs vols, les avions entrent dans la zone terminale où l'ATC les guide de leur altitude de croisière vers les points d'entrée de la zone d'approche, qui se situent entre 5000 et 15000 pieds. Idéalement, les avions entrant dans la zone d'approche sont suffisamment espacés dans le temps. Les contrôleurs d'approche sont responsables de guider les avions vers leur piste d'atterrissage. Un autre rôle très important, qui est lié au contrôleur de la zone terminale, est de séquencer les aéronefs en vue de les faire atterrir. Cette opération peut devenir ardue dans des situations particulières, comme lors de la fermeture d'une piste ou lors d'un besoin de changer de piste qui est en opération.

La capacité d'aéronefs dans une zone donnée (nombre maximal d'aéronefs) dépend principalement des contrôleurs, donc de facteurs humains. Parmi ces facteurs humains il y

a le travail répétitif, les émotions, le stress et la fatigue [Kallus *et al.*, 1997]. La capacité d'aéronefs est également influencée par des facteurs comme la météo et les urgences, comme un orage ou un avion en détresse [Kuwata et Oohama, 1997]. Par exemple, dans de mauvaises conditions météorologiques, les contrôleurs doivent dérouter plus fréquemment les aéronefs vers des aéroports de déroutement. Deux facteurs additionnels rendent le contrôle aérien difficile [Lin et Hong, 1989]. Le premier est celui de l'augmentation incessante du trafic, se traduisant par un taux d'incidents plus élevés, alors que le second est la capacité limitée des routes aériennes et des aéroports. C'est pour ces raisons que plusieurs études ont été réalisées dans le but de pouvoir fournir de nouveaux outils (d'aide à la décision) d'analyse du trafic aérien [Isaacson et Robinson, 2001; Perry et Tekla, 1997]. Malheureusement, ces outils fournissent très peu d'information pour élaborer une solution automatisée de résolution de conflits.

2.2 *Next Gen* ou *Free Flight*

La congestion du trafic aérien est un inconvénient pour les voyageurs et un problème important pour les compagnies aériennes ainsi que pour les autorités de l'aviation. Le design et la taille d'un aéroport, les contrôleurs aériens, ainsi que les conditions météorologiques affectent grandement la capacité d'un aéroport. La congestion et les délais se produisent lorsque la fréquence d'arrivée d'aéronefs excède la capacité de l'aéroport. Les solutions traditionnelles consistaient à construire plus d'aéroports ou de restreindre le mouvement des aéronefs ; ceci demande un capital élevé et une limitation sur la flexibilité des horaires des compagnies aériennes. Les nouvelles avenues de solutions se concentrent principalement sur l'automatisation des différentes tâches d'un contrôleur aérien et sur l'optimisation de trajectoires. Le projet *Next Gen* de la FAA a pour but d'améliorer la situation actuelle.

Dans le projet *Next Gen*, qui était appelé *Free Flight* auparavant, les pilotes auraient plus de flexibilité pour choisir et modifier leurs routes aériennes de la phase en route. Ceci aurait pour effet de réduire les coûts d'opération et d'augmenter la capacité du système [Perry et Tekla, 1997]. Les deux principaux buts de ce nouveau concept sont de permettre une augmentation du trafic aérien en toute sécurité et l'économie d'essence et de temps associées à chaque vol. L'ATC serait impliqué dans la résolution de conflits en cas de nécessité seulement. Ainsi, la décentralisation du contrôle implique le besoin de pouvoir maintenir une séparation adéquate entre les avions. La détection et la résolution de conflits devient donc un facteur important dans *Next Gen*. Les pilotes pourraient voler où ils veulent en autant qu'ils ne pénètrent pas dans l'espace aérien protégé autour de chaque

aéronef (zone de protection). La transition vers le système *Next Gen* a déjà commencé et ce même si sa définition n'est pas encore définitive.

2.3 Détection et résolution de conflits

Dans la littérature on distingue principalement deux grandes catégories d'études de la détection et de la résolution de conflits, soit les approches opérationnelles et les approches dites plus théoriques. Les approches opérationnelles, initiées par les organismes du contrôle aérien, sont généralement influencées par le contexte opérationnel [Erzberger *et al.*, 1997; Prevôt *et al.*, 2003]. Le but principal de cette catégorie d'études est de fournir des aides à la décision qui pourront être employées par les contrôleurs aériens. La seconde catégorie, quant à elle, emploie généralement des techniques plus novatrices pour résoudre les mêmes problèmes. Cette dernière catégorie néglige souvent le contexte opérationnel dans les démarches choisies. Dans le cadre de cette recherche les méthodes de la première catégorie sont de peu d'utilité, car elles ne s'intéressent pas à l'automatisation de certaines tâches du contrôle aérien.

Dans l'ATC, on définit deux notions reliées à la distance entre deux avions, soit une perte de séparation et un conflit. Il y a perte de séparation lorsque deux avions se retrouvent à une distance inférieure à celle définie par la réglementation. Normalement la séparation est définie comme étant une zone de protection, autour d'un avion, ne devant en aucun moment être franchie par un autre avion. La séparation varie en fonction de l'espace aérien et est assujettie à une réglementation formelle. Par exemple, le critère de séparation en route à basse altitude est de 5 miles nautiques (nm) dans le plan horizontal et de 1000 pieds dans le plan vertical. Il s'agit donc d'un volume autour de chaque avion qui ne doit être traversé par aucun autre avion. La prédiction d'une perte de séparation dans un certain temps est définie comme étant un conflit. Il y a généralement deux types de conflits employés, soit ceux à court et long termes. Par exemple, il y aura perte de séparation dans 2 minutes si aucune action n'est prise. Par simplicité, la distinction entre conflit et perte de séparation sera omise dans cette recherche et le terme conflit sera généralement employé.

Le but de la détection de conflits est de prédire qu'une perte de séparation se produira et d'en aviser un opérateur tel qu'un contrôleur ou un pilote. La prédiction, dans un monde opérationnel, est obtenue par un modèle dynamique plus ou moins complexe, dépendant entre autres de la certitude des états, telles que la position et la vitesse. Plusieurs études se sont penchées sur le problème de la prédiction de conflits dans un environnement incertain [Durand et Alliot, 1997; Lecchini *et al.*, 2006]. Le groupe de recherche CTAS

(*Center-TRACON Automation System*) a développé un modèle de trajectoires à 4 dimensions (temps, latitude, longitude, altitude) permettant de prédire, avec une certaine précision, la position d'un avion [Erzberger *et al.*, 1997]. Cette précision est affectée par les systèmes de surveillance qui acheminent la position des avions aux systèmes du contrôle aérien. À titre d'information, les positions des avions dérivées par les radars sont obtenues à environ toutes les 10 secondes.

Par contre, il est raisonnable de prévoir que dans un avenir rapproché les avions évoluant dans les espaces aériens à densité de trafic élevé seront munis de systèmes de positionnement ayant une grande précision, tel que le système de positionnement global GPS (*Global Positioning System*) [Grewal *et al.*, 2007], ainsi qu'un système de transmission automatique de données tel que l'ADS-B (*Automatic Dependent Surveillance-Broadcast*) [RTCA Special Committe, 2007]. Ainsi, les trajectoires 4-D du groupe de recherche CTAS seront peut-être de moins en moins nécessaires.

Outre les méthodes analytiques ou théoriques, plusieurs recherches ont employé des algorithmes numériques dans la problématique de la résolution de conflits. Toutes ces méthodes se classifient en deux catégories, soit celles centralisées et celles autonomes. Les méthodes de séparation d'avions employées dans le système ATC sont basées sur une structure de routes contraintes ainsi qu'un ensemble de procédures. De par sa capacité d'analyse, le contrôleur aérien est un élément essentiel de ces procédures. Pour éviter des erreurs et augmenter la capacité du système, par exemple le projet américain Next Gen, plusieurs études de résolution de conflits automatisés ont vu le jour [Chian *et al.*, 1997; Durand, 1996; Durand et Alliot, 1997; Erzberger, 1995; Erzberger *et al.*, 1997; Hwang et Tomlin, 2002].

Kuchar et Yang ont fait une revue exhaustive de différents algorithmes de détection et de résolution de conflits [Kuchar et Yang, 1997, 2000]. La figure 2.1 présente la classification des différentes méthodes selon Kuchar et Yang. Ils ont établi cinq paramètres principaux de discrimination, soit la détection de conflits, la résolution, les manoeuvres permises, l'aspect de coopération entre les avions en conflits et les approches solutionnant des conflits à multiples avions. Généralement, une méthode donnée répond à plus d'un critère.

Détection de conflit

Ce critère détermine si la méthode emploie une détection explicite ou non explicite. Dans la cas explicite, la méthode détermine qu'il y a un conflit, ou perte de séparation et une résolution est nécessaire. Les aides à la décision de Erzberger [Erzberger *et al.*, 1997] sont

incluses dans la seconde catégorie, car il n'y a aucune action, même suggérée, lorsqu'un conflit est détecté.

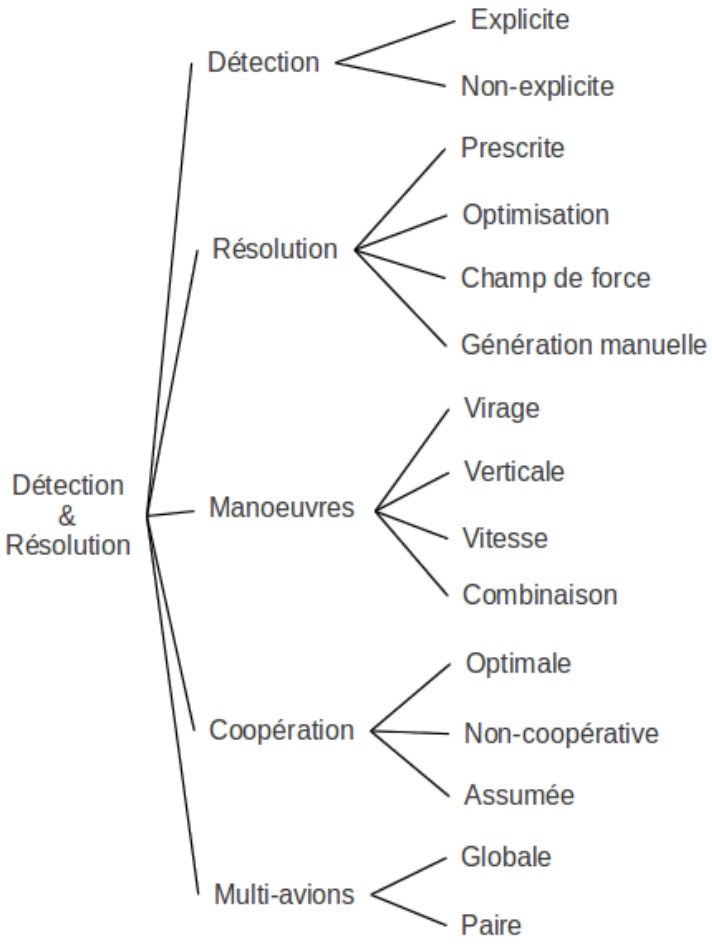


Figure 2.1 Classification des méthodes de détection et résolution de conflits selon Kuchar et Yang [Kuchar et Yang, 2000].

Les techniques décentralisées telles que le TCAS (*Traffic Collision Avoidance System*), la technique des jetons de Granger [Granger, 2002] et celle des forces répulsives de Duong et Zeghal [Duong et Zeghal, 1997] ne sont pas abordées car ce sont des algorithmes décentralisés. Dans ces techniques, les pilotes vont résoudre les conflits eux-mêmes, ce qui est contraire au fonctionnement normal de l'ATC.

La résolution de conflits se divise en deux problèmes importants, soit le moment pour initier une manoeuvre de résolution et la sélection de la manoeuvre à exécuter. Le choix du moment d'exécution est un compromis entre la nécessité et l'efficacité de la manoeuvre. Une manoeuvre initiée hâtivement est plus efficace sur le surplus de distance à effectuer (minimisation), par contre sa nécessité est moins certaine. D'un autre côté, lors d'une initi-

ation tardive, la certitude de la nécessité d'une manoeuvre est grande alors que l'efficacité de cette dernière risque d'être faible [Erzberger *et al.*, 1997]. Dans la majorité des travaux, tous sont d'accord sur le fait que les modifications de trajectoires doivent être simples à décrire par un contrôleur et à réaliser par un pilote. Il s'agit donc de manoeuvres discrètes, comme un changement de cap, et non des manoeuvres continues, comme le suivi d'une courbe.

Solutions analytiques

La résolution optimale de conflits s'intéresse à l'efficacité des manoeuvres à effectuer pour satisfaire un but précis, tel que l'économie d'essence, le confort des passagers ou le retard à l'arrivée. Il est assez difficile de modéliser les techniques utilisées par les contrôleurs de façon optimale. Il peut être également complexe d'ajouter de nouvelles préférences (sur les techniques) dans la fonction de coûts à optimiser lorsque la complexité de cette dernière augmente [Isaacson et Robinson, 2001]. La résolution basée sur le temps est la plus utilisée en pratique, malgré ses limitations importantes. Cette méthode applique la résolution de conflits à un point précis dans l'espace, c'est-à-dire que la séparation des aéronefs n'est pas considérée en tous points de leurs trajectoires [Erzberger, 1995]. Dans des situations complexes, la résolution basée sur le temps ne peut être appliquée, car il serait possible d'avoir des conflits à des temps intermédiaires.

Plusieurs études ont employé la théorie de la commande optimale, dont la variable de contrôle est soit le changement de direction [Krozel et Peters, 1997; Pallottino et Bicchi, 2000; Tomlin *et al.*, 1998] ou de vitesse [He *et al.*, 2009], pour la résolution de conflits entre deux avions. Le principal inconvénient de cette démarche est que les commandes sont impraticables du point de vue ATC, puisque la modification de trajectoires s'exprime de façon continue. De plus, comme toute approche théorique, la modélisation serait à refaire en entier si on devait résoudre un problème différent, par exemple la résolution dans le plan vertical et non horizontal.

Krozel et Peters ont utilisé une fonction pénalisant les retards (à l'heure prévue d'arrivée) ainsi que la consommation d'essence excédentaire nécessaire pour effectuer les manoeuvres [Krozel et Peters, 1997]. Cette fonction d'évaluation du coût d'opération CO étant

$$CO = C_{fuel}\Delta W_{fuel} + C_{time}\Delta T \tag{2.1}$$

où C_{fuel} est le coût du carburant, ΔW_{fuel} est la quantité d'essence excédentaire requise par la manoeuvre de résolution, C_{time} est le coût opérationnel relié au temps d'un avion et finalement ΔT est le temps additionnel requis par la manoeuvre. La quantité d'essence est

evaluée à l'aide de l'équation d'autonomie et de la distance de franchissement maximale de Breget [Roskam et Lan, 2003]. La fonction 2.1 est très intéressante, et ce même si elle requiert la connaissance de certains facteurs, tels que les coefficients aérodynamiques, qui ne sont pas faciles à obtenir.

Hwang et Tomlin ont proposé une méthode de résolution de conflits théorique, ou analytique, traitant plusieurs avions à la fois [Hwang et Tomlin, 2002]. Les instructions apportées par ATC sont des changements de direction uniquement. Tous les avions auront le même changement de direction, ce qui est irréaliste. De plus, le changement de cap d'un avion sera obligatoirement au milieu de leur trajectoire vu par l'algorithme. Ainsi, le parcours modifié aura la forme d'un triangle isocèle.

Solutions numériques

La résolution de conflits entre n aéronefs est un problème combinatoire exigeant en termes de quantités de calculs nécessaires pour trouver une réponse optimale. Cette problématique est similaire à la prédiction de trajectoires en présence d'obstacles mobiles [Reif et Sharir, 1994]. De façon générale, ces problèmes sont de type NP difficile (non déterministe en temps polynomial) [Chian *et al.*, 1997]. Durand a montré, dans sa thèse de doctorat, que si l'on refuse la notion de priorité et que l'on recherche l'optimum global du problème, les techniques classiques ont peu de chance d'aboutir [Durand, 1996]. Il a également montré que l'utilisation de la commande optimale pour résolution de conflits devient trop complexe lorsque le nombre d'avions est supérieur à deux.

Plusieurs travaux du Laboratoire d'Optimisation Globale de l'ENAC (École Nationale d'Aviation Civile) à Toulouse, plus particulièrement ceux de Durand, traitent d'optimisation de trajectoires en route. Le projet CATS (*Complete Air Traffic Simulator*) a vu le jour en 1996 suite à la thèse de doctorat de Nicolas Durand portant sur l'optimisation de trajectoires pour la résolution de conflits en route [Durand, 1996; Durand et Alliot, 1997]. Durand est un des pionniers de l'ENAC à avoir utilisé les algorithmes génétiques (mono-objectif) pour la résolution de conflits de trajectoires aériennes, principalement pour les phases de vol en route. Ses recherches ont servi de fondement pour la résolution de conflits dans la phase en route présentée au chapitre 7. Son algorithme possède un nombre fini de manoeuvres pouvant être exécutées par les aéronefs impliqués dans un conflit, et ce dans un plan horizontal. Toutes ces manoeuvres emploient des changements de direction à certains instants. Pour réduire l'espace de recherche, et ainsi diminuer le temps de calcul, un ensemble fini de changement de direction est permis. La section 7.4 reprend une de ces manoeuvres et l'explique en profondeur.

Kalaek et ses collaborateurs ont également employé un algorithme génétique pour la résolution de conflits dans la phase en route [Malaek et Alaeddini, 2009; Malaek *et al.*, 2011]. Une structure de chromosome similaire à celle définie par Durand a été choisie. De plus, une décomposition en quadrilatère de l'espace aérien dans lequel les avions évoluent, a permis de réduire l'espace de recherche et ainsi diminuer le temps de calcul. Les modifications de trajectoires, qui ont la forme de courbes continues, ne sont pas sous la forme d'instructions ATC traditionnelles, elles sont plutôt exprimées par le déplacement de plusieurs points de la trajectoire initiale. Malheureusement, on ne fournit pas assez de détails sur la façon dont les trajectoires sont modifiées. Cette façon de donner les instructions est incompatible avec la structure actuelle de l'ATC, car à ce jour il est impossible à un contrôleur aérien de fournir une instruction sous forme de trajectoire courbe à un pilote. Cette incompatibilité est principalement liée aux systèmes avioniques présentement en fonction.

Durand et Alliot ont employé un algorithme de colonies de fourmis [Durand et Alliot, 2009] pour résoudre des problèmes similaires à ceux présentés initialement par Durand [Durand, 1996]. Ils n'ont malheureusement pas fait de comparaison entre les techniques (algorithmes génétiques et colonies de fourmis). Ils ont par contre utilisé une astuce intéressante, soit celle de relaxer la contrainte de séparation dans les premières itérations, et ce pour favoriser l'apparition de solutions viables plus rapidement. Cette astuce semble utile dans des problèmes à densité élevée de trafic.

Treleaven et Mao ont étudié la résolution de conflits de flux d'avions se croisant dans la phase de vol en route [Huang *et al.*, 2012; Treleaven et Mao, 2008]. L'algorithme proposé se décompose en deux étapes. La première étape, qui est formé par des équations analytiques optimales, consiste aux déplacements latéraux optimaux des flux de trafic pour éviter la superposition des zones de conflits. Ainsi, chaque intersection possède sa propre zone de conflits. La seconde étape quant à elle consiste à résoudre les conflits à l'intérieur de chaque zone en utilisant une quelconque technique, par exemple un algorithme itératif, un algorithme génétique, etc.

2.4 Séquençage du trafic en approche

Les avions en arrivée évoluent dans l'espace aérien en route vers la zone terminale puis finalement dans la zone de contrôle de l'aéroport. La gestion du trafic dans une zone terminale est sans aucun doute une des régions ayant les tâches ATC les plus difficiles. Ces tâches sont par exemple : déterminer l'ordre d'atterrissage des avions, donner des

instructions de guidage, assurer la séparation et faire attendre certains avions dans des circuits d'attente lors de périodes de congestion de trafic.

La majorité des travaux du groupe de recherche CTAS se concentre sur le contrôle aérien à l'intérieur d'une zone terminale. Un des problèmes auquel le groupe s'est attaqué est le séquençage des vols en arrivée. Il s'agissait d'établir l'ordre dans lequel les avions atterriront et de les guider vers l'axe d'approche final.

À ce jour les méthodes sont basées sur le temps d'arrivée à un point fixe, comme par exemple le début d'une piste ou un point de navigation (virtuel ou réel) pour établir l'ordre de la séquence [Robinson *et al.*, 1997].

La figure 2.2 illustre plusieurs avions (représentés par les lettres majuscules) ainsi que leurs trajectoires vers l'atterrissage sur la piste. Les noms *Long*, *Vent arrière*, *Base* et *Finale* représentent des segments de trajectoires empruntés par les différents avions. Une séquence d'avions doit être établie, car la destination finale de chacun est la piste d'atterrissage. La figure 2.3 montre l'arbre de séquence qui décompose le problème original de séquençage en plusieurs petits problèmes de séquençage [Robinson et Isaacson, 2000; Robinson *et al.*, 1997]. La racine de l'arbre représente l'ordre selon lequel les avions atterriront. Les branches symbolisent des fusions de trajectoires dans le terminal. Ces branches se connectent ensuite à des noeuds, qui à leur tour représentent la séquence d'avions pour un segment de trajectoires données. Ce processus continue jusqu'à la racine. Par exemple, la deuxième ligne de l'arbre de la figure 2.3 illustre des avions se trouvant dans trois segments, soit *finale seulement*, *vent arrière* et *base*. Le noeud suivant, qui est la racine représente l'ordre d'atterrissage des avions de ces trois segments.

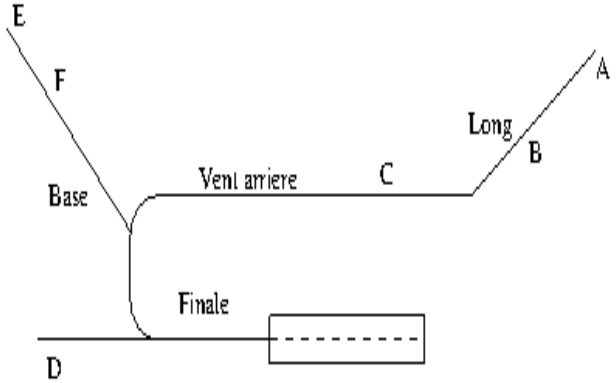


Figure 2.2 Trajectoires d'avions en vue d'un séquençage

Une approche intéressante, proposée par Robinson et coll., pour le séquençage est l'utilisation d'un modèle cognitif (base de données de règles floues) basé sur la prise de décision de contrôleurs réels [Robinson *et al.*, 1997]. Le modèle utilise les trajectoires 4D des différents

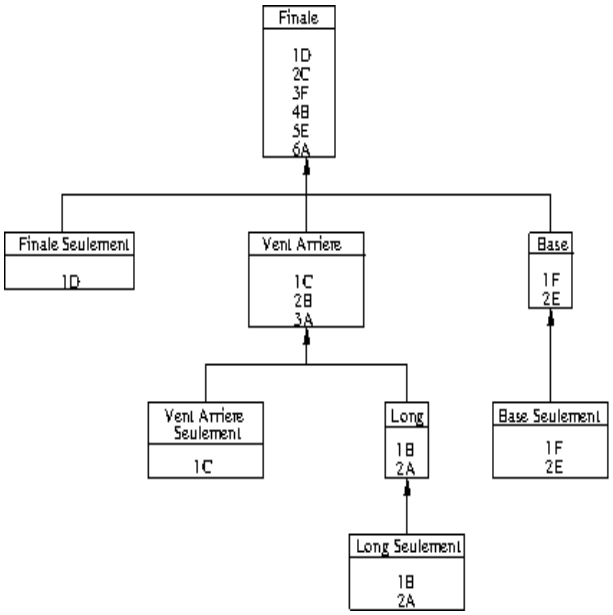


Figure 2.3 Représentation d'un séquençage par un arbre

avions impliqués dans la séquence, comparativement au séquençage basé sur le temps. Chaque trajectoire est définie par la position actuelle d'un avion et la piste où il prévoit atterrir. Notons que ces trajectoires sont générées en utilisant les équations de mouvement de point matériel en trois dimensions [Erzberger *et al.*, 1997]. Le modèle est finalement utilisé pour déterminer la séquence de chaque noeud.

Un séquençage optimal en approche, dans un environnement stochastique, basé sur une simulation de Monte Carlo avec Chaines de Markov a également été envisagé par Lecchini [Lecchini *et al.*, 2006]. Le modèle stochastique de simulation, qui est très gourmand en temps de calcul, est très évolué et tient compte de paramètres tels que le vent, la variation de masse et l'incertitude de position des avions. On y présente deux critères de performances employés dans différentes résolutions, soit obtenir un temps minimal entre deux avions consécutifs lors de leurs entrées dans la zone terminale et obtenir un temps minimal pour l'atterrissage de chaque avion. Cette méthode a été employée pour étudier l'effet de la variation de paramètre dynamique, comme la quantité d'essence, et les perturbations causées par le vent. Il aurait été intéressant de combiner ces deux critères dans une seule simulation. De plus, seulement le séquençage de deux avions a été démontré, et ce malgré le fait qu'une méthode numérique a été employée. Un temps de calcul important empêche pour l'instant l'utilisation de cette solution dans une application temps réel.

2.5 Sommaire

Rappelons que cette étude s'intéresse aux techniques de résolutions de conflits propres à l'ATC, c'est-à-dire les techniques dites centralisées. Dans ce chapitre, il a été mentionné que deux des principales tâches d'un contrôleur IFR en zone terminale sont d'assurer une séparation adéquate et d'ordonnancer le trafic. De nombreuses recherches ont étudié la résolution de conflits dans la phase de vol en route, alors que très peu ont aborbé l'ordonnancement dans la phase terminale (séquençage). De plus, il semble également qu'aucune étude n'ait abordé la problématique d'établir un ordonnancement de façon optimale et ce tout en respectant une séparation minimale. Les variables et contraintes d'optimisation des problèmes de résolution de conflits et d'ordonnancement sont traitées aux sections 7.6.1 et 7.6.2 respectivement.

Beaucoup d'études ont également abordé la problématique de résolution de conflits entre deux avions, mais très peu l'évaluent pour des situations à plus de deux avions. Il est vrai que dans la grande majorité des cas, les conflits se produisent uniquement entre deux avions. Cependant, il est également nécessaire de s'attaquer à des situations concernant plus de deux avions, car bon nombre de situations impliquent plus de deux avions. De plus, une augmentation de situations conflictuelles entre plus de deux avions est à prévoir, à cause de l'accroissement du trafic aérien. Les solutions analytiques ou théoriques à deux avions, bien qu'intéressantes, peuvent s'avérer inadéquates lors d'une densité de trafic élevée. La littérature présente très peu d'études portant sur le séquençage des avions en phase d'arrivée. Le séquençage en arrivée est une opération complexe du contrôle aérien et sa modélisation représente un défi important.

De ces études, certaines ont utilisé un algorithme d'optimisation global pour résoudre une situation conflictuelle de la phase en route, mais aucune d'entre elles ne semblent avoir traité le problème comme étant multiobjectif. Il semble qu'une approche employant un algorithme évolutionnaire multiobjectif est une avenue prometteuse, car elle permet de traiter plus facilement les différents critères de l'ATC. Par exemple, il serait souhaitable de minimiser la consommation d'essence des avions ainsi que les instructions fournies aux avions, et ce tout en respectant la séparation minimum.

La littérature présente une multitude de travaux donnant des résultats de résolution intéressants. Par contre, il semble qu'aucun d'entre eux ne s'est penché sur la qualité de la solution telle qu'interprétée par un humain. Est-il toujours possible à un contrôleur humain de donner les instructions (changement de direction, changement de vitesse, etc.) fournies par ces solutions ? Par exemple, dans les solutions présentées, il serait possible de

fournir des instructions à deux avions au même moment, ce qui est impossible en réalité pour un contrôleur aérien.

CHAPITRE 3

PROBLÈME D'OPTIMISATION

Des problèmes d'optimisation se retrouvent dans plusieurs domaines comme l'ingénierie et l'économie. Ce genre de problèmes est aussi commun dans la vie quotidienne, par exemple :

Monsieur ABC désire acheter un ordinateur. Il désire un modèle avec les meilleures spécifications mais à un faible coût.

Monsieur ABC est devant un problème d'optimisation multiobjectif comportant deux objectifs. Le premier étant d'obtenir un ordinateur avec les meilleures spécifications techniques possibles alors que le second est de dépenser le moins d'argent possible. Voici maintenant un exemple d'optimisation contraint inspiré du premier exemple :

Monsieur XYZ désire également acheter un ordinateur, mais il a un budget limité. Son but est d'acheter le meilleur ordinateur possible tout en respectant son budget. Comparativement au premier exemple, celui-ci est un problème d'optimisation contraint, car le respect d'un budget représente une contrainte.

Ce chapitre introduit la problématique de l'optimisation. La section 3.1 présente l'optimisation mono-objectif alors que la section 3.2 présente l'optimisation multiobjectif. La section 3.3, quant à elle, présente des techniques d'optimisation traitant de problèmes contraints. Finalement, la section 3.4 présente des techniques d'optimisation stochastiques. De par la nature du problème étudié dans cette recherche, les méthodes non-stochastiques n'ont pas été analysées. En effet, ce type de techniques semble peut efficace à résoudre les problèmes présentés aux chapitres 7 et 9. Le chapitre 4 traite des algorithmes génétiques pour la résolution de problèmes mono-objectif et multiobjectif.

3.1 Problème mono-objectif

Tout problème d'optimisation revient à définir une ou plusieurs fonctions de coûts $f(X)$, qu'il faut minimiser ou maximiser par rapport à certains paramètres. Ces paramètres, appelés variables d'optimisation, constituent les inconnus du problème à résoudre et correspondent aux composantes du vecteur X :

$$x = \left(x_1, x_2, \cdots, x_n\right) \in \mathbb{R}^n \tag{3.1}$$

où $x \in Q_{ab} \subseteq S$. La fonction de coût f est définie dans l'espace de recherche $S \subseteq \mathbb{R}^n$ et l'ensemble $Q_{ab} \subseteq S$ définit la région admissible de solutions. Généralement, l'espace de recherche est une hyper boîte de dimensions n, où n est le nombre de variables du domaine. Les domaines des variables sont définis par les bornes inférieures l et supérieures u

$$l(i) \leq x_i \leq u(i), \quad 1 \leq i \leq n \tag{3.2}$$

De son côté, l'espace de recherche S est restreint par m contraintes supplémentaires ($m \geq 0$) :

q contraintes d'inégalités

$$g_i(x) \leq 0, \quad \big(i = 1, \cdots, q\big) \tag{3.3}$$

et $m - q$ contraintes d'égalités

$$h_i(x) = 0, \quad \big(i = q + 1, \cdots, m\big) \tag{3.4}$$

Les solutions qui satisfont simultanément les contraintes des équations 3.2, 3.3 et 3.4 sont les solutions admissibles. Elles définissent l'espace de conception $Q_{ad} \subset \mathbb{R}^n$, également appelé espace admissible. Dans plusieurs situations, tel que présenté au chapitre 7, la satisfaction des contraintes est un problème en soi. Le problème n'a aucune solution lorsque toutes les contraintes ne peuvent être satisfaites à la fois.

La solution $x^* \in Q_{ad}$ qui minimise la fonction de coût est l'optimum global. À ce jour, il n'existe aucune technique traditionnelle permettant de déterminer l'optimum global au problème d'optimisation décrit ci-dessus par les équations 3.2 à 3.4. La solution analytique au problème existe seulement si les objectifs et les contraintes respectent certaines propriétés [Michalewicz, 1992].

Dans le reste de ce document, il sera question uniquement de minimisation, et ce sans perte de généralité. En effet, les problèmes de maximisation et de minimisation sont strictement équivalents et le passage de l'un à l'autre peut s'effectuer à l'aide de la relation suivante :

$$\max f(x) = -\min f(x)$$

3.2 Problème multiobjectif

L'optimisation de problèmes complexes est généralement composée de plusieurs critères (fonctions de coûts), qui peuvent être contradictoires. Ceci soulève la question suivante : comment différents objectifs doivent-ils être combinés pour obtenir une solution finale ? Également, comment faut-il chercher une solution à un problème donné ?

L'optimisation multiobjectif peut être définie de la façon suivante [Eschenauer *et al.*, 1990] :

> Déterminer un vecteur de variables de décisions satisfaisant les contraintes et optimisant une fonction vectorielle dont chaque élément représente une fonction objective du problème. Ces fonctions objectives sont généralement en conflit les unes par rapport aux autres. L'optimisation multiobjectif consiste donc à déterminer une solution rendant les valeurs de l'ensemble des fonctions objectives acceptables par l'usager.

De façon formelle, un problème multiobjectif consiste à définir des fonctions objectives

$$f(x) = [f_1(x), f_2(x), \ldots, f_k(x)]^T \tag{3.5}$$

qu'il faut minimiser par rapport à un vecteur de variable de décisions

$$x = [x_1, x_2, \ldots, x_n]^T \tag{3.6}$$

où $x \in Q_{ab} \subseteq S$. Les différentes composantes de $f(x)$ représentent les fonctions de coût (critères à optimiser). Notons que ces critères ne sont pas nécessairement exprimés dans les mêmes unités. Comme à la section 3.1, S et Q_{ab} représentent l'espace de recherche et l'espace admissible respectivement. L'espace admissible est défini par des contraintes de bornes, des contraintes d'inégalités et des contraintes d'égalités.

La solution optimale est représentée par x^*. L'optimalité est moins bien définie pour l'usager dans le contexte multiobjectif, car rarement il est possible d'obtenir un vecteur idéal x^* tel que

$$\forall i = 1, 2, \ldots, k \mid X \in S \wedge f_i(x^*) \leq f_i(x) \tag{3.7}$$

Il est généralement impossible d'avoir une situation dans laquelle toutes les $f_i(x)$ ont un minimum dans S en un point commun x^*. L'optimalité de Pareto, qui est présentée à la sous-section 3.2.1, permet de mieux définir une solution optimale pour un problème multiobjectif.

3.2.1 Optimalité de Pareto

Le concept de l'optimalité de Pareto, qui constitue l'origine de la recherche multiobjectif, a été formulé par Vilfredo Pareto au XIX siècle [Pareto, 1896]. Deux notions fondamentales doivent être définies, soit celle de *dominance* et d'*optimalité de Pareto*. Pour les définitions suivantes, considérons les deux vecteurs suivants [Deb, 2008] :

$$\begin{aligned} f^A &= [f_1^A, f_2^A, \cdots, f_n^A]^T \\ f^B &= [f_1^B, f_2^B, \cdots, f_n^B]^T \end{aligned}$$

Définition 1 ***Dominance de Pareto :*** *Le vecteur $f^A \in Q_{ad}$ domine le vecteur $f^B \in Q_{ad}$, $f^A \prec f^B$, si et seulement si f^A est partiellement inférieur à f^B :*

$$\forall i \in [1,n] : \quad f_i^A \leq f_i^B \ \wedge \ \exists i \in [1,n] : \ f_i^A < f_i^B$$

où Q_{ad} dénote l'espace de recherche admissible, ou tout simplement celui prescrit par le problème et $\wedge$ symbolise l'opérateur logique *et*.

Définition 2 ***Optimum de Pareto :*** *Une solution $x^* \in Q_{ad}$ est un optimum de Pareto si et seulement s'il n'existe pas d'autres solutions $x \in Q_{ad}$ pour laquelle $f(x)$ domine $f(x^*)$.*

Une solution x^* est optimale au sens de Pareto s'il n'existe pas d'autres solutions admissibles qui améliorent simultanément tous les objectifs. Une solution admissible offrant des valeurs plus faibles de certains objectifs, comparativement à x^*, doit nécessairement dégrader au moins un des autres objectifs.

Définition 3 ***Optimum local de Pareto :*** *Une solution $x^* \in Q_{ad}$ est un optimum local de Pareto si et seulement si, pour un $\delta > 0$ fixé : $\nexists x \in Q_{ad}, \ f(x) \in B\big(f(x^*), \delta\big) \ f(x) < f(x^*)$, où $B\big(f(x^*), \delta\big)$ représente une sphère centrée en $f(x^*)$ et de rayon δ.*

Définition 4 ***Ensemble des solutions non dominées :*** *Soit $\mathcal{F}$ l'image dans l'espace des objectifs de l'ensemble réalisable $\mathcal{X}$. L'ensemble des solutions non dominées de $\mathcal{X}$, est défini par l'ensemble $ND(\mathcal{X})$:*

$$ND(\mathcal{X}) = \{x \in \mathcal{X} \mid x \text{ est non dominé par rapport à } \mathcal{X}\}$$

Le front de Pareto $\mathcal{F}$ se définit de façon similaire :

Définition 5 ***Front de Pareto :*** *Soit $\mathcal{F}$ l'image dans l'espace des objectifs de l'ensemble réalisable $\mathcal{X}$. Le front Pareto $ND(\mathcal{F})$ de $\mathcal{F}$ est défini comme suit :*

$$ND(\mathcal{F}) = \{x \in \mathcal{F} \mid \nexists y \in \mathcal{F},\ y < x\}$$

Définition 6 ***Solution idéale :*** *Les coordonnées du point idéal correspondent aux meilleures valeurs de chaque objectif des points du front de Pareto. Les coordonnées de ce point correspondent aussi aux valeurs en optimisant chaque fonction objectif séparément.*

Le but de l'optimisation multiobjectif est d'optimiser simultanément plusieurs objectifs d'un problème, comparativement à un seul pour l'optimisation mono-objectif. L'optimum de l'optimisation multiobjectif est constitué d'un ensemble de solutions, qui est le front de Pareto. Ce but peut également se décomposer en deux autres buts, minimiser la distance des solutions par rapport au front de Pareto et maximiser la répartition des solutions au sein de l'approximation du front de Pareto. Ainsi, l'intérêt d'une optimisation multiobjectif est de trouver l'ensemble de solutions qui approxime l'optimal, tout en considérant que les objectifs ont la même importance. Suite à cette optimisation, l'usager peut choisir la solution au problème en utilisant de l'information supplémentaire. Cette procédure d'optimisation multiobjectif en deux étapes est illustrée à la figure 3.1. Sur cette figure le graphique de gauche montre que l'algorithme multiobjectif a trouvé plusieurs solutions (points) du front de Pareto (courbe). Le graphique de droite quant à lui illustre la solution choisie parmi celles trouvées par l'algorithme. Par inspection de cette procédure, on constate que l'optimisation mono-objectif est un cas particulier de l'optimisation multiobjectif.

Une méthode alternative, illustrée à la figure 3.2, consiste à transformer un problème multiobjectif original en un problème mono-objectif par le biais de pondération à chacun des objectifs [Deb *et al.*, 2000]. Le graphique de cette figure montre la solution obtenue par l'algorithme (point) sur le front de Pareto (courbe). L'objectif résultat sera la somme pondérée de ces objectifs. Elle est similaire à la méthode de pénalité de statique, présentée à la section 3.3, employée dans le traitement des contraintes. Bien que cette technique soit facile à mettre en oeuvre, elle est déconseillée car elle est très sensible au choix de ces pondérations. De plus, l'assignation de ces pondérations est subjective.

3.2.2 Exemple d'optimisation multiobjectif

Le problème de Schaffer est probablement la fonction de test multiobjectif la plus utilisée pour démontrer le fonctionnement d'un algorithme multiobjectif [Schaffer, 1984]. Ce prob-

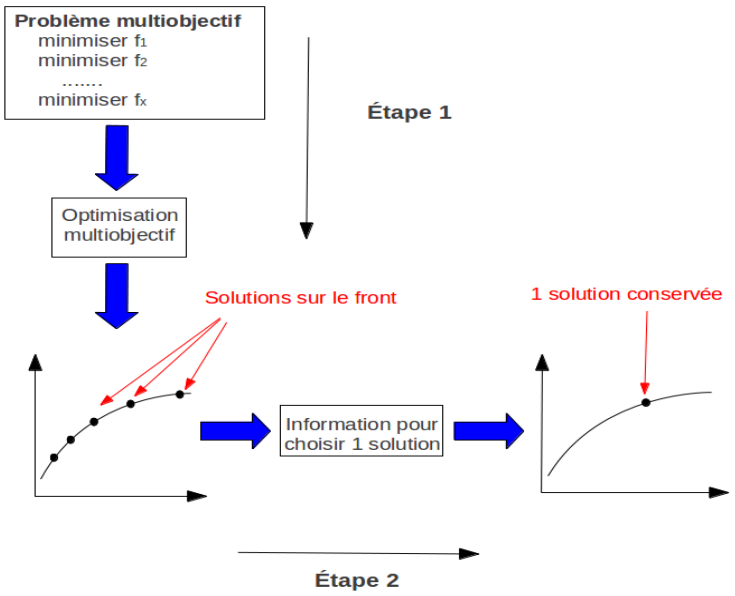


Figure 3.1 Choix de la solution à postériori.

lème possède deux objectifs et une seule variable d'optimisation, qui est employée par les deux objectifs. Il est décrit par les équations suivantes :

$$f_1(x) = x^2 \tag{3.8}$$
$$f_2(x) = (x-2)^2 \tag{3.9}$$
$$-A \leq x \leq A \tag{3.10}$$

On rapporte que des valeurs de A entre 10 et 10^5 ont été utilisées [Deb, 2008]. Par contre, plus la valeur de A augmente et plus il est difficile d'approcher le front de Pareto. Le front de Pareto de ce problème, qui est illustré à la figure 3.3, n'existe que pour les valeurs de x entre 0 et 2. Toutes les solutions optimales se situent sur cette courbe. Ainsi, un algorithme d'optimisation multiobjectif doit non seulement trouver des approximations de solutions optimales, il doit également favoriser une répartition homogène de ses solutions sur l'approximation du front de Pareto.

3.3 Problèmes contraints

La majorité des problèmes réels d'ingénierie d'optimisation ont des contraintes rendant la résolution beaucoup plus ardue. La difficulté de résolution peut avoir plusieurs causes,

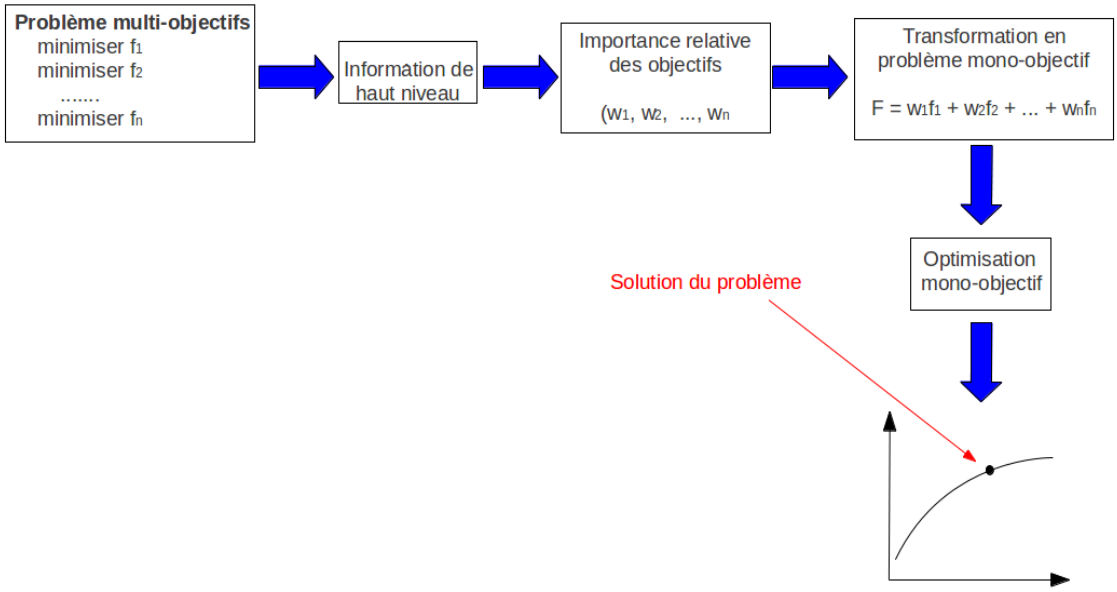


Figure 3.2 Choix des préférences à priori.

telles que les limites imposées aux variables de décision, l'interférence entre les contraintes et les relations entre les contraintes et les objectifs. La recherche sur les algorithmes d'optimisation évolutionnaires s'est d'abord concentrée sur la découverte de solution de l'optimum à des problèmes non-contraints. On remarque dans la littérature un intérêt grandissant sur l'étude de l'optimisation de problèmes contraints.

Le principal défi de l'optimisation sous contraintes est de simultanément traiter les contraintes et d'optimiser les différents objectifs. Deux obstacles surviennent souvent lors de l'optimisation de problèmes avec contraintes : les régions de recherches peuvent être non convexes et même disjointes. Les techniques de programmation linéaire et non-linéaire sont généralement inadéquates pour ce genre de problème [Kim et Myung, 1997]. L'obtention d'une solution viable [1] est beaucoup plus importante qu'optimiser les différents objectifs du problème. Il existe même des problèmes, appelés CSP (*Constraint Satisfaction Problem*), pour lesquels trouver une solution viable représente une tâche ardue. Ces problèmes sont traités comme étant des problèmes à satisfaction de contraintes.

À la base, les algorithmes évolutionnaires sont des techniques d'optimisation sans contrainte. Il est donc nécessaire de pouvoir y incorporer le traitement des contraintes. Selon la revue de la littérature, on classifie les techniques d'optimisation traitant les contraintes en deux grandes catégories [Coello Coello, 2000a; Deb, 2008; Michalewicz, 1992; Venkatra-

1. Une solution viable satisfait toutes les contraintes.

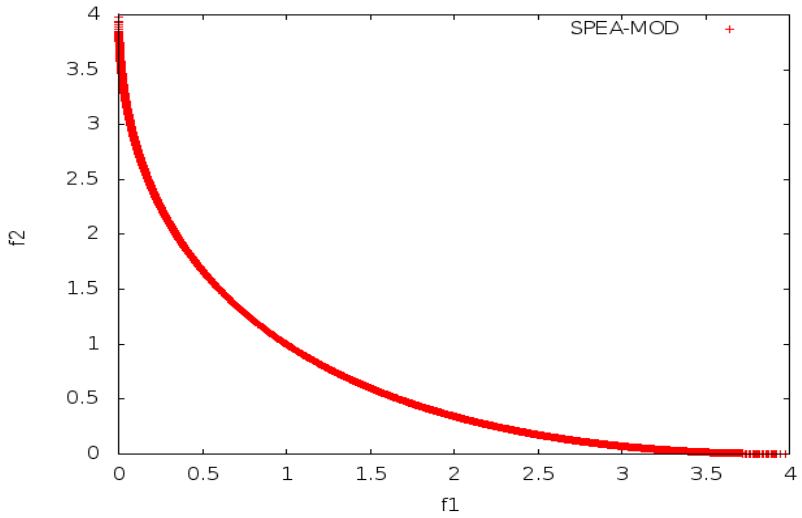


Figure 3.3 Front de Pareto du problème de Schaffer.

man et Yen, 2005]. Soit celles pénalisant les solutions non viables et celles basées sur une technique multiobjectif. Ces deux méthodes sont expliquées aux sections 3.3.1 et 3.3.2.

Pour un éventail plus élaboré de ces techniques, l'auteur recommande de consulter [Coello Coello, 2000a]. Les techniques impliquant des mécanismes de réparation [Michalewicz et Nazhiyath, 1995] ne sont pas traitées dans cette revue. Les algorithmes de réparation consistent à transformer un individu non-viable en un individu viable. Ces algorithmes sont intéressants lorsqu'il est facile de rendre viable une solution non-viable. Évidemment, ceci n'est pas toujours le cas, comme dans les problèmes de résolutions de conflits et de séquençage d'avions étudiés dans cette recherche [Coello Coello, 2000a].

3.3.1 Pénalisation des solutions non viables

Les techniques utilisant une fonction de pénalité étaient très utilisées dans les méthodes d'optimisation traditionnelle et furent parmi les premières employées dans les algorithmes évolutionnaires. Dans ces méthodes, les individus sont pénalisés selon leur degré de violation des contraintes et peuvent même à la limite être rejetés. On dénombre quelques variantes de cette méthode. La première étudiée, appelée la peine de mort, consiste à ignorer systématiquement les individus non-viables. L'inconvénient de cette approche est qu'elle n'exploite pas l'information des individus non-viables [Coello Coello, 2000a]. De plus, la population initiale doit être régénérée lorsqu'initialement elle contient uniquement des solutions non-viables. Cette méthode est très similaire à la conversion d'un problème multiobjectif en problème mono-objectif par l'agrégation des objectifs.

Une autre version est la méthode de pénalité statique qui est la somme pondérée de la violation des contraintes. Cette somme est ajoutée à l'objectif de la façon suivante :

$$F(x) = f(x) + \sum_{i=1}^{m} r_i c_i(x) \tag{3.11}$$

où $f(x)$ est la fonction objective initiale, $F(x)$ est la nouvelle fonction objective, r_i est le coefficient de pénalité de la contrainte i, c_i est le degré de violation de la contrainte i et m le nombre de contraintes. Le succès de cette méthode dépend du choix judicieux des coefficients de pénalités des contraintes.

Pour pallier cette difficulté, Joines et Houck ont proposé une alternative dynamique dans laquelle les coefficients de pénalités varient en fonction du nombre de générations [Joines et Houck, 1994]. Malheureusement, il est tout aussi difficile de dériver une fonction dynamique d'adaptation des coefficients que de choisir de bons coefficients statiques.

3.3.2 Techniques multiobjectif

L'idée principale de ces techniques est de transformer un problème mono-objectif contraint en un problème multiobjectif à $m + 1$ objectifs, ou m est le nombre de contraintes. Deb et al. ont défini un nouveau principe de domination de solutions pour des problèmes contraints [Deb *et al.*, 2001] :

Définition 1 ***Dominance de Pareto contrainte :*** *Une solution A domine la solution B si au moins une des conditions suivantes est vraie :*

1. *La solution A est valide et la solution B ne l'est pas.*
2. *Les solutions A et B sont invalides, mais la solution A a une plus faible violation des contraintes.*
3. *Les solutions A et B sont valides et la solution A domine la solution B.*

Une solution est dite valide si elle satisfait toutes les contraintes. Le deuxième critère peut être également interprété comme étant une domination de Pareto appliquée aux contraintes, les contraintes sont alors traitées comme des objectifs. Cette définition étant générique, elle peut s'intégrer dans n'importe quel algorithme évolutionnaire multiobjectif. Dans le présent document, aucune distinction ne sera faite entre la *Dominance de Pareto* et la *Dominance de Pareto contrainte*. La distinction est implicitement liée au type de problème, c'est-à-dire un problème contraint ou non-contraint. On utilisera la même notation que celle introduite à la définition 1, ($f^A \prec f^B$).

Venkatraman et Yen ont proposé un algorithme génétique traitant les problèmes contraints et ne nécessitant pas de connaissance à priori du problème [Venkatraman et Yen, 2005]. Dans leur algorithme, le premier but est porté sur la découverte de solutions réalisables. La population emploie un mécanisme d'élitisme pour s'assurer de ne pas perdre la meilleure solution réalisable. Le second but de cet algorithme consiste à explorer l'espace de recherche et ce, guidé par la satisfaction des contraintes et par l'optimisation de la fonction objective. L'algorithme est destiné aux problèmes mono-objectif mais il est possible de l'adapter aux problèmes multiobjectif, tel qu'il sera démontré au chapitre 6.

3.4 Méthodes stochastiques

Les méthodes stochastiques sont caractérisées par un processus de création aléatoire ou pseudo-aléatoire de points dans l'espace d'état en utilisant une heuristique qui permet de guider la convergence de l'algorithme. De plus, ces méthodes sont d'ordre zéro, c'est-à-dire qu'elles ne requièrent que les valeurs de la fonction objective et ne recourent pas au calcul de ses dérivées. Par conséquent, elles peuvent être appliquées à des fonctions discontinues, non dérivables, faisant intervenir des variables discrètes ou continues dans un espace d'état quelconque. Ces méthodes sont utilisées dans des problèmes où on ne connaît pas d'algorithme de résolution en temps polynomial et pour lesquels on espère trouver une solution près de l'optimum global.

Les sous-sections suivantes présentent quelques techniques d'optimisation stochastiques, dont les plus connues.

3.4.1 Monte-Carlo

Les méthodes de type Monte-Carlo recherchent l'optimum d'une fonction en générant une suite aléatoire de nombres en fonction d'une loi de distribution aléatoire uniforme [Fishman, 1997]. Il s'agit essentiellement d'une marche aléatoire dans l'espace de recherche. Il n'existe pas d'algorithme unique et le terme Monte-Carlo décrit un ensemble d'approches largement utilisées. Ces méthodes consistent à répéter des essais aléatoires, par exemple sous forme de simulations, plusieurs fois dans le but de d'obtenir une solution numérique. Ces approches tentent de suivre l'algorithme suivant :

a) Définir un domaine des variables du problème.

b) Générer aléatoirement des valeurs (variable) sur le domaine, selon une certaine distribution de probabilités.

c) Faire des calculs déterministes sur les variables (simulations).

d) Agréger les résultats

Il est à noter qu'il y a deux propriétés importantes des méthodes de Monte-Carlo. La première est que ces méthodes reposent sur une bonne génération de nombres aléatoires. La seconde est une convergence lente vers une meilleure approximation lorsque le nombre d'échantillons augmente.

3.4.2 Recuit simulé

Cette méthode d'optimisation globale est basée sur le processus de recuit des matériaux cristallins [Kirkpatrick *et al.*, 1983; Russel et Norvig, 1995]. Ce processus consiste à élever un solide à haute température et ensuite de l'amener à basse température. Chaque particule possède une très grande énergie et peut effectuer de grands déplacements aléatoires dans la matière lorsque la température du solide est élevée. Au fur et à mesure que la température baisse, les particules perdent de l'énergie et leur capacité à se déplacer diminue. Ainsi, un solide à énergie minimale représente la solution optimale.

Dans le recuit simulé, les déplacements aléatoires de chacun des points sont fonction d'une probabilité dépendante d'une variable, T, représentant la température du matériau. Voici l'algorithme :

a) Effectuer un déplacement aléatoire à partir d'un point initial x_0.

b) Si le déplacement mène à x_1 tel que $f(x_1) < f(x_0)$, alors x_1 est accepté. Sinon, il est accepté selon une probabilité $p = e^{\left(-|\Delta(f)|/kT\right)}$.

$\Delta(f)$ représente la distance de déplacement $x_1 - x_0$.

T est assimilé à une température décroissante au cours du temps.

k est une constante.

Au début, T est élevée ce qui se traduit par une grande capacité d'exploration, car même des points n'améliorant pas f sont acceptés. Au fur et à mesure que T diminue, cette capacité se réduit, car p diminue. Seuls les points améliorant f sont acceptés lorsque $T = 0$. Il est possible que l'algorithme se bloque à un minimum local d'une fonction convexe si l'énergie de départ n'est pas suffisante. Ainsi, une température de départ élevée fournit une bonne capacité d'exploration, alors qu'un refroidissement lent évite que la recherche s'arrête à un minimum local.

À ce jour, il semble y avoir très peu d'étude qui ont utilisé le recuit simulé pour des problèmes d'optimisation multiobjectif.

3.4.3 Recherche tabou

La recherche tabou est une méta-heuristique d'optimisation présentée par Fred Glover [Glover, 1986; Glover et Laguna, 1997]. Cette méthode est une méta-heuristique itérative qualifiée de recherche locale au sens large.

Cette méthode consiste, à partir d'une position donnée, à en explorer le voisinage et à choisir la position dans ce voisinage qui minimise la fonction objective. Il est essentiel de noter que cette opération peut conduire à augmenter la valeur de la fonction : c'est le cas lorsque tous les points du voisinage ont une valeur plus élevée. C'est à partir de ce mécanisme qu'il est possible de sortir d'un minimum local.

Le risque cependant est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper. C'est pourquoi il faut que l'heuristique ait de la mémoire : le mécanisme consiste à interdire (d'où le nom de tabou) de revenir sur les dernières positions explorées. Les positions déjà explorées sont conservées dans une liste (appelée souvent liste tabou) d'une taille donnée, qui est un paramètre ajustable de l'heuristique. Ce paramètre, qui est la taille de la liste, est appelé teneur tabou. La diversification cherche à diriger l'algorithme vers des régions de l'espace de recherche non encore explorées.

Les démonstrations de convergence, c'est-à-dire la capacité à trouver le minimum global si le temps disponible tend vers l'infini, pour la recherche tabou existent, mais suppose des conditions strictes, rarement présentes en pratique.

3.4.4 Colonies de fourmis

L'algorithme de colonies de fourmis est une méta-heuristique d'optimisation initialement proposée pour la recherche de chemins optimaux dans un graphe. Notons tout d'abord que cette méthode est différente de l'algorithme de colonies de particules. Elle est inspirée du comportement des fourmis recherchant un chemin de leur colonie vers leur nourriture. Plus généralement, les algorithmes de fourmis artificielles sont des méthodes qui s'inspirent des comportements de fourmis réelles.

Dans une vision simplifiée de la réalité, les fourmis commencent par se déplacer au hasard. Puis, lorsqu'elles ont trouvé de la nourriture, elles retournent vers leur colonie en balisant leur chemin à l'aide de traces. Si d'autres fourmis rencontrent ce chemin, il y a de fortes

chances qu'elles arrêtent leurs déplacements aléatoires et qu'elles rejoignent le chemin balisé, en renforçant le balisage à leur retour s'il mène bien vers de la nourriture.

Ainsi, dès lors qu'une fourmi découvre un chemin de la colonie vers de la nourriture, elle va le renforcer en y déposant une trace et les autres fourmis seront alors plus enclines à suivre ce chemin. Dans le même temps, le chemin le plus court sera plus parcouru, et donc plus renforcé et plus attractif. En considérant que la trace s'évapore, les chemins les moins renforcés finissent par disparaître, ce qui amène toutes les fourmis à suivre le chemin le plus court. L'idée de l'algorithme de colonies de fourmis, introduite par Marco Dorigo [Dorigo, 1992], est d'imiter ce comportement à l'aide de fourmis artificielles se déplaçant à travers le graphe qui représente le problème à résoudre.

3.4.5 Algorithmes évolutionnaires

Parmi les méthodes d'optimisation globale, les algorithmes évolutionnaires (AE) ont la particularité de requérir peu d'information sur la fonction objective et de pouvoir identifier simultanément plusieurs optimas locaux proches de l'optimum global. Les algorithmes évolutionnaires appartiennent à une famille d'algorithmes appelés méta-heuristiques dont le but est d'obtenir une solution approchée, en un temps correct, à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte pour le résoudre. Ce sont des algorithmes d'optimisation stochastique globale s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle, comme par exemple, la théorie darwinienne de la sélection naturelle [Fogel, 1964; Goldberg, 1989; Holland, 1975]. Les algorithmes génétiques, la programmation génétique, les stratégies d'évolution ainsi que certains types de classificateur font partie des algorithmes évolutionnaires. Une description approfondie des algorithmes génétiques est présentée au chapitre 4. De plus, quelques techniques d'optimisation multiobjectif utilisant des algorithmes génétiques, y sont présentées.

Les algorithmes évolutionnaires sont principalement caractérisés par :

1. une population contenant un ensemble de solutions possibles, ou individus, en compétition les unes avec les autres ;
2. des combinaisons et des altérations aléatoires d'individus dans le but de générer de meilleures solutions ;
3. des heuristiques de solutions qui ont pour but d'augmenter la proportion de meilleures solutions.

Les algorithmes évolutionnaires peuvent être utilisés dans la majorité des problèmes d'optimisation (continus ou non continus, bruités, mono-objectif ou multiobjectif, etc.). Ils

ont la capacité de trouver simultanément plusieurs optimas locaux proches de l'optimum global. En effet, il peut être intéressant pour l'usager d'obtenir non pas la solution optimale, mais un ensemble de solutions différentes proches de l'optimum. De plus, les AE n'ont pas besoin de la dérivée de la fonction objective, car ce sont des méthodes d'ordre zéro.

Malheureusement, les AE possèdent trois inconvénients majeurs : ce sont des méthodes nécessitant beaucoup de calculs, les résultats théoriques sont à l'occasion peu utiles en pratique et ils n'offrent aucune garantie de convergence vers l'optimum en un temps fini. Il est généralement conseillé de les utiliser lorsqu'aucune autre méthode ne fonctionne de manière satisfaisante.

Les stratégies d'évolution sont des algorithmes itératifs dans lesquels un parent génère un enfant. Le meilleur des deux individus survit et devient le parent de la génération suivante. Les algorithmes génétiques et les algorithmes de colonies de particules sont expliqués en détails au chapitre 4 et au chapitre 5 respectivement.

3.5 Sommaire

Ce chapitre a présenté les définitions fondamentales d'un problème d'optimisation mono-objectif et multiobjectif qui seront ultérieurement employées dans cette thèse. Les concepts reliés à l'optimisation de problèmes contraints ont également été revus.

Une revue de certaines techniques d'optimisation stochastique a également été présentée. Uniquement les techniques stochastiques ont fait partie de cette analyse, car la revue de la littérature du chapitre 2 a fait ressortir que l'optimisation de la résolution de conflits d'avions est un problème très complexe à résoudre par les techniques classiques. De ces techniques, les algorithmes évolutionnaires semblent être une avenue prometteuse pour cette recherche.

CHAPITRE 4

ALGORITHMES GÉNÉTIQUES

Les algorithmes évolutionnaires, tels que les algorithmes génétiques (AG), sont un champ d'études en pleine expansion, couvrant tous les aspects de la simulation évolutionnaire. Les AG sont apparus aux État-Unis dans les années 60. Ils font partie de la famille des algorithmes évolutionnaires, c'est-à-dire des algorithmes d'optimisation globale s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle, comme par exemple la théorie darwienne de la sélection naturelle. Leurs applications possibles sont très variées, comme par exemple, la minimisation de fonctions et la détermination de gains optimaux d'un contrôleur [Michalewicz, 1992].

D'un côté, la simulation de l'évolution naturelle fut utilisée par les biologistes pour étudier l'adaptation aux changements environnementaux dans le but de comprendre le processus d'évolution d'organismes trouvés sur la Terre. D'un autre côté, il a été montré à plusieurs reprises que ce genre d'algorithmes peut résoudre des problèmes d'optimisation complexes. À ce jour, il n'existe pas de démonstration formelle sur la convergence des algorithmes évolutionnaires, tels que les algorithmes génétiques, et c'est pourquoi on recommande de les employer lorsqu'aucune autre technique ne fonctionne de manière satisfaisante. Cerf a démontré la convergence des algorithmes génétiques mono-objectifs à codages binaires [Cerf, 1994]. Ces résultats se résument ainsi :

1. Il y a convergence si la taille de la population est assez grande, et ce, quel que soit l'état de la population initiale (dispersion des individus).
2. La limite de la convergence n'est pas à l'optimum si on ne converse pas le meilleur individu (absence de mécanisme d'élitiste).
3. La convergence est à l'optimum si l'élitisme est imposé.
4. La convergence est assurée par les mutations, et ce, même sans croisement. Le rôle du croisement est d'accélérer la convergence.

Malheureusement aucune démonstration de convergence n'existe pour la représentation codage réelle.

La discussion de la section 2.3 a mentionné que la résolution de conflits est un problème d'optimisation difficile à résoudre. Deux obstacles surviennent souvent lors de l'optimisation de problèmes avec contraintes : les régions de recherches peuvent être non con-

vexes et même disjointes, rendant les techniques de programmation linéaire et non-linéaire généralement inadéquates pour ce genre de problèmes [Kim et Myung, 1997].

Les AG ont la capacité de trouver simultanément plusieurs optima locaux proches de l'optimum global. En effet, il peut être souhaitable de rechercher non pas la solution optimale mais un ensemble de solutions différentes proches de l'optimum. Malheureusement, il y a deux inconvénients majeurs : ce sont des méthodes nécessitant beaucoup de calculs, et les résultats théoriques sont souvent peu utiles en pratique. Il est conseillé de les utiliser lorsqu'aucune autre méthode n'est appropriée, ce qui est le cas pour les problèmes de résolutions de conflits et de séquençage d'avions optimal étudiés dans la présente recherche [Chian *et al.*, 1997; Durand, 1996; Malaek et Alaeddini, 2009].

Ce chapitre traite des caractéristiques fondamentales des algorithmes génétiques utiles à cette thèse. La section 4.1 présente les concepts de base, alors que la section 4.2 traite de l'encodage des solutions (chromosomes) par des nombres réels. Finalement, la section 4.4 traite de l'aspect multiobjectif et plus particulièrement de l'algorithme SPEA (*Strength Pareto Evolutionary Algorithm*).

4.1 Description

Les AG sont basés sur les phénomènes biologiques de la génétique. Les organismes vivants sont constitués de cellules, dont les noyaux comportent des chromosomes qui sont des chaînes d'ADN. L'élément de base de ces chromosomes, le caractère de la chaîne d'ADN, est un gène. Ainsi, chacun de ces chromosomes est constitué d'une suite de gènes codant les fonctionnalités d'un organisme. On utilise aussi, dans les algorithmes génétiques, une analogie avec la théorie de l'évolution qui propose qu'au fil du temps, les gènes conservés au sein d'une population donnée sont ceux qui sont le plus adaptés aux besoins de l'espèce vis-à-vis son environnement.

Les AG agissent sur une population de solutions potentielles, selon le principe de la sélection naturelle, pour produire des individus de mieux en mieux adaptés dans le but d'obtenir une approximation d'une solution. À chaque génération, de nouveaux individus sont créés par la sélection des individus les mieux adaptés en utilisant des opérateurs empruntés à la génétique.

L'algorithme génétique est généralement résumé par les étapes 1 à 11 ci-dessous. La figure 4.1 présente un diagramme général de ce même algorithme. La sous-section 4.1 présente un exemple simple d'utilisation d'un AG.

1. Représenter une solution du problème par un chromosome. Choisir la taille N de la population, la probabilité de croisement p_c, la probabilité de mutation p_m et le critère d'arrêt. Le critère d'arrêt est généralement un nombre maximal de générations.
2. Définir une fonction d'aptitude (*fitness*) pour mesurer la performance d'un chromosome (solution). La fonction d'aptitude est à la base de la sélection des chromosomes en vue de la reproduction (opération de croisement).
3. Générer, de façon aléatoire, une population initiale de N chromosomes : $x_1, x_2, \ldots, x_N$.
4. Évaluer la performance, ou aptitude, des différents individus aux fonctions de coûts du problème $(f(x_1), f(x_2), \ldots, f(x_N))$. L'évaluation des contraintes (s'il y a lieu) se fait également à cette étape.
5. Choisir une paire de chromosomes, à l'intérieur de la population actuelle, pour la reproduction. Les parents sont choisis avec une probabilité en fonction de leurs aptitudes. Les chromosomes ayant une aptitude élevée ont plus de chance d'être choisis.
6. Deux enfants sont obtenus en appliquant l'opérateur de croisement, selon la probabilité p_c, sur les deux parents. Ces nouveaux enfants forment la nouvelle paire si le croisement a eu lieu, autrement la paire est constituée des chromosomes obtenus à l'étape précédente.
7. Modifier génétiquement la paire par l'opérateur de mutation, selon une probabilité p_m.
8. Ajouter la paire de chromosomes dans la nouvelle population.
9. Répéter les étapes 5 à 8 jusqu'à ce que la taille de la nouvelle population soit N. Une répétition de ces étapes est une génération.
10. Remplacer les chromosomes initiaux par ceux de la nouvelle population.
11. Retourner à l'étape 4 et répéter le processus jusqu'à ce que le critère d'arrêt soit satisfait.

La figure 4.2 illustre l'application des opérateurs génétiques. Tout d'abord, la sélection des parents se fait en fonction de leur adaptation. Plus l'adaptation d'un individu est élevée, plus la probabilité que ce dernier soit choisi par le processus de sélection est élevé. Il existe plusieurs méthodes de sélection, selon si le problème est mono-objectif ou multiobjectif [Deb, 2008; Goldberg, 1989]. Tout comme le processus de sélection, il existe plusieurs méthodes de croisement et de mutation. Des méthodes basées sur les nombres réels seront abordées à la section 4.2.

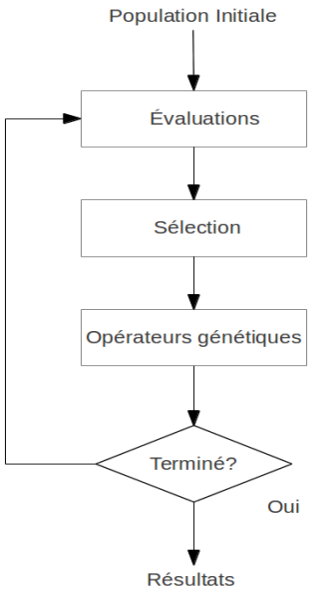


Figure 4.1 Diagramme général d'un algorithme génétique.

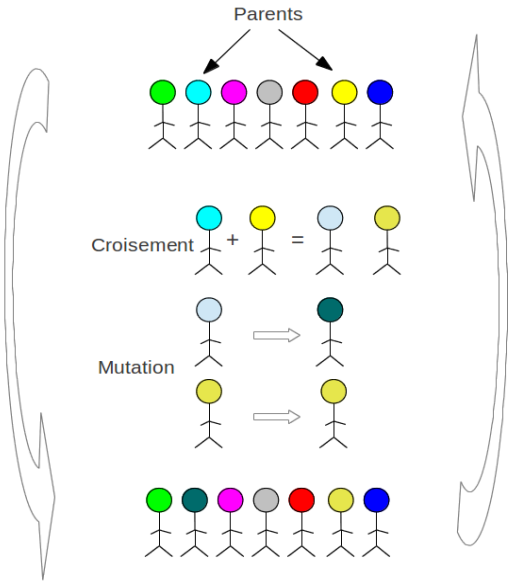


Figure 4.2 Illustration des opérateurs génétiques.

Population initiale

Une population initiale est nécessaire pour démarrer un algorithme génétique. Il n'y a aucune règle ou spécification guidant la création des individus de cette première population, on recommande par contre une initialisation aléatoire. Chaque individu doit simplement représenter une solution potentielle. Par exemple, si vous cherchez un chemin entre 2

points, les individus créés devront simplement posséder le bon point de départ et le bon point d'arrivée, peu importe par où ils passent.

Idéalement la population initiale doit être non homogène, c'est-à-dire que les individus sont répartis sur tout le domaine de recherche où l'on ne connait pas à priori d'information sur le problème à résoudre. Pour ce faire, une création des individus de façon aléatoire est généralement employée et c'est cette méthode qui est employée dans l'algorithme élaboré dans ce projet de recherche. De plus, il n'est pas nécessaire que ces individus initiaux soient des solutions viables, comme un individu ne respecte pas toutes les contraintes. Un des avantages d'une création aléatoire est qu'il apporte une certaine diversité génétique. Plus les individus de la population de départ seront différents les uns des autres, plus l'algorithme aura des chances de converger vers l'optimum global ou le front de Pareto.

La taille de la population est un autre paramètre influençant l'optimisation. Lorsque la taille de la population est élevée, l'algorithme a de meilleures chances de trouver l'optimum global, car l'espace de recherche est mieux parcouru. Il sera plus difficile de converger à l'optimum global si la taille est trop faible ; on risquera d'atteindre un optimum local. D'un autre côté, une taille trop élevée nécessitera un plus long temps de calcul. Pour l'instant, il ne semble pas y avoir de critère aidant à choisir la taille de la population.

Évaluation des individus

Cette phase consiste à évaluer les objectifs et contraintes (s'il y a lieu) de tous les individus de la population. L'étape d'évaluation peut être très exigeante du point de vue des opérations à exécuter pour des problèmes complexes, tels que ceux étudiés dans ce projet de recherche.

Opérateurs génétiques

Des opérateurs de diversification de la population, comme le croisement et la mutation sont utilisés. Généralement, l'opérateur de croisement, ou de reproduction, construit deux individus (enfants) à partir de deux autres (parents), alors que l'opérateur de mutation altère génétiquement un individu de la population. La mutation a pour mandat de favoriser une recherche sur tout le domaine, ce qui aidera à converger vers l'optimum global. Le croisement est utilisé pour améliorer la vitesse de convergence en combinant les attributs de deux individus parents. Règle générale, la probabilité de mutation est 10 fois plus petite que la probabilité de croisement. Il est à noter que les opérateurs de croisement et de mutation n'ont pas la même signification si l'on utilise un codage binaire ou réel des chromosomes [Michalewicz, 1992]. De plus, tous les chromosomes d'une population

donnée utilisent le même codage. Il est à noter qu'à l'origine des algorithmes génétiques, la représentation binaire était utilisée, car cette représentation est plus proche de la génétique.

Critères d'arrêt

Il n'y a pas consensus absolu dans le choix du critère d'arrêt, de la taille de la population, de la probabilité d'application des opérateurs de croisement et de mutation. Dans la littérature, on dénombre principalement deux critères d'arrêt. Le premier consiste à fixer le nombre de générations désirées. C'est ce que l'on doit faire lorsque l'on doit trouver la solution dans un certain temps. Le second critère est d'arrêter l'algorithme lorsque la population n'évolue plus ou plus assez rapidement.

Exemple d'utilisation

L'exemple, extrait de [Michalewicz, 1992], présenté ci-dessous illustre le fonctionnement typique d'un algorithme génétique mono-objectif. Soit la fonction de coût suivante

$$f(x) = x \cdot \sin(10\pi \cdot x) + 1.0$$

Le problème est de trouver la valeur de x comprise dans l'intervalle $[-1, 2]$ qui maximise $f(x)$. Cette fonction, qui possède une multitude d'optimums locaux, a son optimum global à $x = 1.85$. Chaque individu de la population contient une valeur de la variable x représenté en forme binaire (codage binaire). La représentation binaire était employée à l'origine des algorithmes génétiques. Sous cette représentation, les opérateurs génétiques sont similaires à ceux que l'on retrouve dans la nature.

Le domaine de x a une longueur de 3 et le requis de précision exige de diviser ce domaine en intervalles égaux de 3/1000000, ce qui veut dire que 22 bits sont nécessaires pour représenter x sous forme binaire : $2097152 = 2^{21} < 3000000 \leq 2^{22} = 4194304$. La conversion de x sous forme binaire à une forme réelle s'effectue en deux étapes

1. convertir la chaîne binaire $\langle b_{21} b_{20} \ldots b_0 \rangle$ en base 10 :

$$\langle b_{21} b_{20} \ldots b_0 \rangle_2 = \big(\sum_{i=0}^{21} b_i \cdot 2^i \big)_{10} = x^{'}$$

2. trouver une correspondance réelle pour le nombre x :

$$x = -1.0 + x^{'} \cdot \frac{3}{2^{22} - 1}$$

où -1 est la borne gauche de l'intervalle des valeurs admissibles de x.

Chaque chromosome de la population est représenté par une chaîne binaire de 22 bits. L'initialisation de la population initiale s'effectue en initialisant aléatoirement tous les bits de ses chromosomes.

Le processus d'évaluation d'un chromosome consiste à convertir sa chaîne binaire en valeur réelle x et d'évaluer la fonction de coût $f(x)$. Par exemple, les trois chromosomes suivants :

$$\begin{aligned} v_1 &= (1000101110110101000111) \\ v_2 &= (0000001110000000010000) \\ v_3 &= (1110000000111111000101) \end{aligned}$$

ont comme valeurs $x_1 = 0.637197$, $x_2 = -0.958973$ et $x_3 = 1.627888$ respectivement. Ces chromosomes ont comme adaptation : $f(x_1) = 1.586345$, $f(x_2) = 0.078878$ et $f(x_3) = 2.230650$.

Ainsi, le chromosome v_3 est le mieux adapté. Cet exemple emploie les deux opérateurs génétiques, soit la mutation et le croisement. À chaque génération, tous les individus sont mutés selon la probabilité de mutation. La mutation présentée consiste à modifier un bit d'un chromosome. Par exemple, si le cinquième but du chromosome v_3 est sélection pour la mutation, alors ce chromosome deviendrait

$$v_3^{'} = (1110100000111111000101)$$

et sa valeur réelle serait $x_3^{'} = 1.721638$. Ce chromosome muté est moins bien adapté que le chromosome original.

Illustrons l'opération de croisement des individus v_2 et v_3. Généralement uniquement les chromosomes les mieux adaptés sont choisis par le processus de sélection pour le croisement. Le point de croisement, qui est choisi aléatoirement, se situe après le cinquième bit.

$$\begin{aligned} v_2 &= (00000|01110000000010000) \\ v_3 &= (11100|00000111111000101) \end{aligned}$$

Les deux nouveaux individus obtenus sont

$$\begin{aligned} v_2^{'} &= (00000|00000111111000101) \\ v_3^{'} &= (11100|01110000000010000) \end{aligned}$$

Le nouveau chromosome $v_3^{'}$ est mieux adapté que ses deux parents, $f(v_3^{'}) = 2.459245$. Le processus décrit se répète à chaque génération, et ce jusqu'à un certain critère d'arrêt, qui est souvent un nombre de génération maximale. Le tableau 4.1 présente la valeur de $f(x)$ du meilleur chromosome de chaque génération.

Tableau 4.1 Résultats de l'optimisation après 145 générations.

Nombre de générations	$f(x)$
1	1.441942
6	2.250003
12	2.328077
40	2.345087
51	2.738930
99	2.849246
137	2.850217
145	2.850227

4.2 Codage réel

Le codage associe une structure de données à chacun des points de l'espace d'état. La qualité du codage des données conditionne le succès, ou la performance, d'un algorithme génétique. Les codages binaires ont été largement utilisés à l'origine. Maintenant, les codages réels sont très employés, surtout lorsque les variables du problème d'optimisation sont réelles [Michalewicz, 1992]. L'avantage principal de cette représentation est de permettre à l'algorithme génétique d'être plus près de l'espace d'état du problème. Cette représentation a la propriété que deux points côte à côte dans l'espace d'état de recherche sont également côte à côte dans l'espace d'état du problème, ce qui n'est pas le cas pour la représentation binaire.

Un autre désavantage de la représentation binaire est la taille requise, ou nombre de *bits*, pour avoir une certaine précision, tel que présenté à la section 4.1. Il est possible d'augmenter le nombre de *bits* pour avoir une précision comparable à la représentation réelle, par contre ceci augmente énormément le temps de calcul, car on doit employer des chaînes binaires ayant un nombre important de bits [Michalewicz, 1992]. Par exemple,

considérons la variable x, précise à 4 décimales, comprise dans l'intervalle $[-3, 12.1]$ que l'on veut représenter en nombre binaire. Compte tenu de la précision demandée, il faudrait avoir $15.1 \cdot 10000$ intervalles égaux, ce qui nécessiterait une représentation à 18 bits, car $2^{17} \leq 151000 \leq 2^{18}$.

Les opérateurs employés sont différents de ceux que l'on retrouve traditionnellement dans la représentation binaire, car ils fonctionnent dans l'espace réel. À cause des similitudes existantes, ces opérateurs sont également groupés en deux catégories, soit la mutation et le croisement. Michalewicz a proposé deux opérateurs génétiques non-uniformes ; leur action est fonction de l'âge de la population, ou tout simplement le nombre de générations [Michalewicz, 1992]. Les prochaines sous-sections décrivent certains de ces opérateurs.

4.2.1 Mutation réelle

Les opérations de mutation réelles uniforme et non-uniforme sont définies de façon similaire à leurs homologues binaires, c'est-à-dire qu'un gène du chromosome est modifié, tel qu'illustré à la figure 4.3.

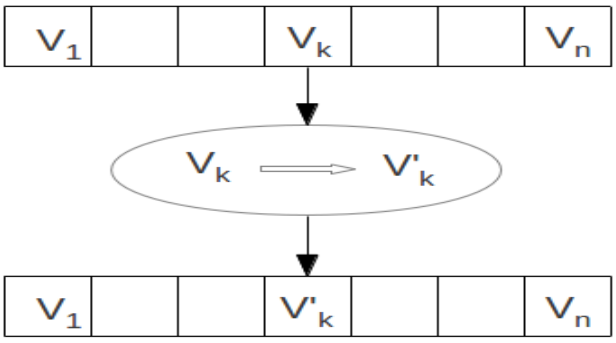


Figure 4.3 Mutation d'un gène.

Mutation uniforme

Si $v = (v_1, \cdots, v_n)$ est un chromosome, alors chaque élément v_k a la même probabilité d'être muté. Le résultat d'une seule opération sur ce chromosome sera
$v' = (v_1, \ldots, v'_k, \cdots, v_n)$, où $1 \leq k \leq n$ et v'_k est une valeur aléatoire dans la plage permise de ce paramètre ($v_k \in [v_{k_{min}}, v_{k_{max}}]$, où $v_{k_{min}}$ est la limite inférieure permise et $v_{k_{max}}$ est la limite supérieure permise.)

Mutation non-uniforme

Michalewicz a développé un nouvel opérateur de mutation, appelé mutation non-uniforme, pour les chromosomes à représentation réelle [Michalewicz, 1992]. Cet opérateur a pour but d'offrir une bonne exploration de l'espace de recherche tout en réduisant l'effet d'une recherche complètement aléatoire, ce qui nuit à la convergence. Le résultat de cet opérateur sur l'élément v_k du vecteur v se décrit comme étant

$$v_k^{'} = \begin{cases} v_k + \Gamma(g, v_{k_{max}} - v_k) & \text{si un nombre aléatoire est 0} \\ v_k - \Gamma(g, v_k - v_{k_{min}}) & \text{si un nombre aléatoire est 1} \end{cases} \tag{4.1}$$

où g est le nombre de générations. La fonction $\Gamma(g, x)$ retourne un nombre entre $[0, x]$, tel que la probabilité que $\Gamma(g, x)$ soit près de 0 augmente lorsque g augmente. Ainsi, la fonction a comme propriété initialement (g petit) de parcourir uniformément tout l'espace de recherche et de la parcourir plutôt localement lorsque g augmente, donc augmentant la probabilité d'obtenir un nombre près de son prédécesseur. $\Gamma(g, x)$ est définie par l'équation suivante :

$$\Gamma(g, x) = x \cdot \left(1 - r^{(1 - \frac{g}{gMax})^b}\right) \tag{4.2}$$

où r est un nombre aléatoire entre $[0, 1]$, $gMax$ est le nombre de générations maximal, b est un paramètre qui détermine le degré de dépendance sur l'itération g.

Jiménez et Verdegay ont également employé la mutation non-uniforme et le croisement non-uniforme de Michalewicz dans leur recherche [Jiménez et Verdegay, 1999].

4.2.2 Croisement arithmétique uniforme et non-uniforme

L'opérateur de croisement arithmétique est illustré à la figure 4.4. Cet opérateur est défini comme étant une combinaison linéaire de deux vecteurs [Michalewicz, 1992]. Le résultat du croisement de deux vecteurs parents, m et p produit deux vecteurs de solutions f et h :

$$\begin{aligned} h &= a \cdot m + (1-a) \cdot p \\ f &= a \cdot p + (1-a) \cdot m \end{aligned} \tag{4.3}$$

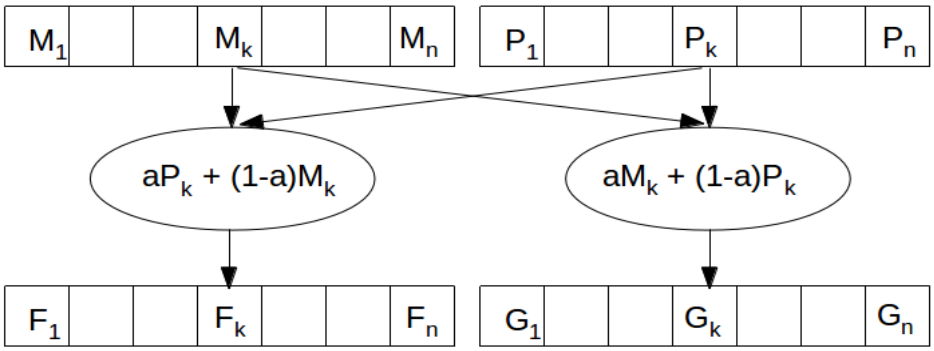


Figure 4.4 Croisement d'un individu.

Dans la définition originale de Michalewicz, le paramètre a est constant pour tous les gènes. De plus, sa valeur peut être constante (croisement uniforme) ou variable (croisement non-uniforme) en fonction du nombre de générations, tel que décrit par l'équation 4.2.

4.3 Élitisme

À cause d'effets aléatoires, il est possible que de bonnes solutions soient perdues au cours du processus d'optimisation. L'élitisme permet de conserver les meilleurs individus dans les générations futures. De Jong a observé que l'élitisme permet d'améliorer les performances d'un algorithme génétique mono-objectif [Jong, 1975]. Par contre, il peut causer une convergence prématurée lorsqu'employé dans des problèmes multiobjectifs. La complexité d'un algorithme élitiste dans le cadre de problèmes multiobjectifs est beaucoup plus élevée que dans le cadre de problèmes mono-objectifs. En effet, la meilleure solution, d'un problème multiobjectif est représentée par un ensemble d'individus et non par un seul.

Dans certaines mises en oeuvre d'algorithmes mono-objectifs, l'élitisme consiste simplement à s'assurer qu'un certain pourcentage des meilleurs individus de la génération actuelle se retrouvent dans la génération suivante. Par exemple, conserver 50% des meilleurs individus de la combinaison de la population des parents (population i) à celles de la population des enfants (population $i+1$).

Quant à lui, l'élitisme employé dans un algorithme multiobjectif est généralement réalisé de deux façons. Dans la première, il s'agit de combiner l'ancienne population et l'ensemble de reproduction (nouveaux candidats obtenus par reproduction) et d'utiliser un algorithme de sélection au lieu de remplacer directement des individus de l'ancienne population par les nouveaux candidats. La seconde stratégie consiste à utiliser une population secondaire, ou archive, dans laquelle les individus prometteurs y sont insérés [E. Zitzler et Thiele, 2000].

Comme la mémoire disponible n'est pas infinie, il faut trouver un critère afin de limiter le nombre de candidats dans les techniques présentées. La notion de dominance de Pareto est généralement utilisée. Si une population secondaire est employée, elle devrait contenir les individus qui approximent le front de Pareto ; les individus dominés y sont enlevés. Par contre, le critère de dominance n'est pas suffisant en soi. Par exemple, le front de Pareto d'un problème continu peut contenir une infinité de solutions. Encore une fois, des paramètres supplémentaires sont nécessaires pour y réduire le nombre de bonnes solutions. L'une des approches est l'utilisation du temps qu'un individu a passé dans la population secondaire [Rudolph et Agapie, 2000]. Une autre méthode est l'utilisation de la densité de la population [E. Zitzler et Thiele, 2000].

4.4 Techniques multiobjectifs de Pareto

Un des principaux avantages des algorithmes évolutionnaires est qu'ils permettent la mise en oeuvre de techniques dites de Pareto ainsi que d'autres types, car ils utilisent une population contenant plusieurs individus (solutions potentielles). Certains mécanismes de sélection implémentent la relation de dominance pour réaliser ainsi une sélection de Pareto.

Goldberg a suggéré d'utiliser un classement ainsi qu'une sélection utilisant la relation de dominance dans le but de déplacer la population vers le front de Pareto [Goldberg, 1989]. L'idée est d'assigner le rang le plus élevé à tous les individus non dominés de la population pour former le front de Pareto. De la même façon, on assigne le second rang à tous les individus non dominés de la population, à l'exception de ceux du premier rang qui sont ignorés. Le processus continue jusqu'à ce que toute la population soit ordonnée. L'adaptation d'un individu correspond à son rang dans la population. Contrairement aux sélections non Pareto, l'évaluation d'un individu ne dépend pas uniquement de lui-même, mais également de la population. Goldberg suggère également d'employer des techniques comme le partage (*sharing*) pour éviter une concentration trop forte d'individus en un seul point. Ce principe permet de ne pas hiérarchiser les objectifs entre eux. Par contre,

l'augmentation de la taille de l'espace de recherche en influence la performance. Il est possible que les rangs de tous les individus soient les mêmes, car le nombre de variables est élevé. Comment sélectionner certains individus dans une telle situation ? Ceci sera abordé dans les prochaines sous-sections.

Certains algorithmes fréquemment cités dans la littérature, tel que MOGA [Fonseca et Flemming, 1993], NPGA [Horn *et al.*, 1994], NSGAII [Deb *et al.*, 2000] et SPEA [Zitzler et Thiele, 1998], diffèrent d'un algorithme génétique traditionnel principalement par leur opérateur de sélection, alors que les opérateurs de croisement et de mutation restent inchangés.

SPEA

Zitler, Deb et Thiele ont employé des métriques pour comparer plusieurs algorithmes évolutionnaires multiobjectifs [E. Zitzler et Thiele, 2000]. La comparaison est faite avec des problèmes types dont les solutions analytiques sont connues. Certains de ces problèmes sont présentés à la section 6.4. De cette étude, il est ressorti que l'algorithme SPEA est un des algorithmes les plus performants parmi ceux étudiés. Depuis, il y a eu l'apparition de quelques autres techniques également performantes. Par contre, SPEA demeure une technique de choix étant donné sa performance et sa facilité de mise en oeuvre.

La méthode SPEA, qui est un algorithme évolutionnaire élitiste, fut développée par Zitzler et Thiele [Zitzler et Thiele, 1998]. L'élitisme est mis en oeuvre par le maintien d'une archive externe (population secondaire) contenant un certain nombre d'individus non dominés rencontrés durant la recherche. À chaque itération, ou génération, les nouveaux individus non dominés sont comparés à ceux existant dans l'archive et seul ceux non dominés sont conservés. Les solutions élites ne sont pas seulement conservées, elles sont employées, avec les solutions de la population, aux opérations génétiques dans le but de diriger la recherche vers de bonnes régions de l'espace de recherche. L'algorithme 1 présente les grandes étapes de SPEA.

Il est noté que $|\cdot|$ est l'opérateur de cardinalité. L'une des premières étapes de cet algorithme, consiste à créer une population P initiale, de la même façon que tout autre algorithme évolutionniste, et d'initialiser l'archive A à l'ensemble vide. Par la suite, l'archive est mise à jour régulièrement en fonction des individus non dominés de la population. Les opérateurs de sélection, de croisement et de mutation sont appliqués à chaque itération de l'algorithme. Dans l'algorithme original, il n'y a pas de spécification quant au codage employé ni sur les opérateurs génétiques. Il est à noter que l'archive et la population par-

Algorithme 1 SPEA

Initialisation compteur de générations $g = 1$
Génération maximale $gMax$
Initialisation populations aléatoires P_g et P_{g-1}
Initialisation archive à l'ensemble vide $A = \{0\}$
Taille de la population $N_p = |P_g|$

Tant que $g \leq gMax$ **Faire**
    Évaluation des individus
    Ensemble de reproduction $Repro = A + P_{g-1}$
    Évaluer l'adaptation des solutions de A (par l'équation 4.4)
    Évaluer l'adaptation des solutions de P_{g-1} (par l'équation 4.5)
    Mise à jour de l'archive A avec *Repro*
    Pour $i \leftarrow 1, N_p$ **Faire**
        père = sélection par tournoi (*Repro*)
        mère = sélection par tournoi (*Repro*)
        Si un nombre aléatoire $>$ probabilité de croisement **Alors**
            $enfant_1$ = croisement(père, mère)
            $enfant_2$ = croisement(mère, père)
        Sinon
            $enfant_1$ = père
            $enfant_2$ = mère
        Fin Si
        muter $enfant_1$ selon probabilité de mutation
        muter $enfant_2$ selon probabilité de mutation

        $P_g(i) = enfant_1$
        $P_g(i+1) = enfant_2$

        $g = g + 1$
    Fin Pour
Fin Tant que

ticipent à la sélection. Une sélection de type tournoi binaire fut utilisée dans la version originale de SPEA. Cette méthode de sélection consiste à choisir le meilleur individu parmi deux individus choisis aléatoirement dans l'ensemble de reproduction. Une étape de réduction de la taille de l'archive (*clustering*), détaillée à l'algorithme 2, permet de garder les meilleurs représentants, lorsque l'archive excède une certaine dimension.

La diversité employée est basée sur la notion de dominance de Pareto, contrairement à beaucoup d'autres algorithmes évolutionnistes multiobjectifs qui utilisent la méthode de partage de l'adaptation. Le but étant de distribuer uniformément la population le plus près possible du front de Pareto, telles que toutes les solutions de Pareto dominent le même nombre d'individus de la population. L'assignation de l'adaptation est réalisée en deux étapes. Dans la première, il faut assigner à chaque solution de l'archive une valeur réelle S ($S \in [0, 1]$), tel que décrit par l'équation 4.4. Il s'agit en fait du nombre d'individus qu'un élément $i \in A$ domine dans la population P.

$$S(i) = \frac{|\{j \in A \wedge f(i) \leq f(j)\}|}{|A| + 1} \tag{4.4}$$

où $f(i)$ représente l'adaptation de l'individu i. La seconde étape consiste à obtenir l'adaptation des autres individus de la population. L'adaptation de chacun consiste à additionner les forces de tous les individus de l'archive le dominant.

$$F(j) = 1 + \sum_{i \in A \wedge f(i) \leq f(j)} S(i) \tag{4.5}$$

Les nouveaux individus non dominés viennent s'ajouter à l'archive, alors que ceux dominés par ces nouveaux individus sont supprimés de l'archive. Les performances de l'algorithme, plus spécifiquement la vitesse de convergence, se dégradent significativement lorsque la taille de l'archive est grande. Dans la littérature, quelques méthodes ont été proposées pour réduire la taille d'un ensemble de Pareto. Une technique de réduction de taille (*clustering*) est donc utilisée pour remédier à ce problème. Cette technique est décrite par l'algorithme 2. Tout d'abord, chaque individu de l'ensemble de Pareto constitue son propre groupe. Par la suite, on fusionne deux à deux les groupes les plus proches en terme de leur distance. Cette étape se répète jusqu'à ce que le nombre désiré de groupe soit atteint. Finalement, il ne reste qu'à choisir un représentant par groupe, selon une certaine

méthode. La méthode employée dans SPEA est celle du centroïde, qui consiste à choisir l'individu ayant la plus petite distance moyenne entre tous les autres individus.

Le calcul de la distance entre deux individus dans l'algorithme 2, $D(C_i, C_j)$, peut être évalué dans l'espace objectif comme dans l'espace d'état. Dans le premier cas, le calcul se fait en utilisant les objectifs alors que dans le second, il s'effectue en employant les variables du problème d'optimisation. $D(C_i, C_j) = 0$ indique que les individus i et j sont identiques, alors que $D(C_i, C_j) = 1$ indique que ces deux individus sont totalement différents. Évidemment, le calcul est simple lorsqu'il n'y a qu'une seule variable d'état (évaluation dans l'espace d'état) ou un seul objectif (évaluation dans l'espace objectif). Autrement, il faut normaliser les différents éléments pour pouvoir faire le calcul de distance. Une distance normalisée dans l'espace d'état a été utilisée dans les deux algorithmes présentés au chapitre 6. La distance entre les sous-ensembles 1 et 2 est définie par l'équation 4.6.

Algorithme 2 Réduction de la taille de l'archive

1: Initialiser un sous-ensemble pour chaque solution, $C_i = \{i\}$, tel que $C = \{C_1, C_2, \ldots, C_N\}$
2: **Si** $|C| \leq N$ **Alors**
3: Alors allez à 7,
4: **Fin Si**
5: Pour chaque paire de sous-ensembles $\{C_i, C_j\}$, calculer la distance $D(C_i, C_j)$. Trouver la paire dont la distance est minimale.
6: Joindre les sous-ensembles C_i et C_j. Ceci réduit la taille de C de un. Allez à l'étape 2
7: Choisir une solution pour chaque sous-ensemble et en retirer les autres. La solution ayant la plus faible distance moyenne entre les autres solutions devrait être choisie.

$$D(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} d(i, j) \tag{4.6}$$

Un des désavantages de la réduction de la dimension de la taille de l'archive de l'algorithme SPEA est la possibilité d'avoir des oscillations dans l'approximation du front de Pareto. En effet, cette réduction peut à l'occasion supprimer certains individus qui domineront de nouveaux individus ajoutés à l'archive. Pour pallier ce problème, Fieldsend et ses collaborateurs ont employé une archive de taille illimitée [Fieldsend *et al.*, 2003]. Les individus de l'archive sont conservés dans une certaine structure pour en faciliter l'accès, telle que la comparaison. Malheureusement, dans la littérature, on retrouve peu d'études démontrant l'efficacité de cette technique.

4.5 Sommaire

La section 2.5 a abordé la problématique de la résolution de conflits et le séquençage d'avions comme pouvant être traité comme un problème d'optimisation et plus particulièrement de type multiobjectif. De par la nature de cette problématique, des techniques d'optimisation basées sur des algorithmes évolutionnaires semblent être une avenue intéressante. Cette stratégie sera abordée au chapitre 7 et 9.

Ce chapitre a d'abord traité des principes fondamentaux des algorithmes génétiques (AG). Un des éléments importants des AG est le codage, qui est en fait la représentation d'une solution potentielle. Les premiers AG employaient des codage binaires, mais ils ont été délaissés pour faire place à la représentation réelle, car cette dernière représentation est plus près de l'espace de recherche. Un exemple de représentation réelle est présentée à la section 7.4. La représentation réelle a permis de définir deux opérateurs génétiques de type non-uniforme, c'est-à-dire que leur comportement varie en fonction du nombre de générations. Ils favorisent une exploration de l'espace de recherche tout en réduisant l'effet d'une recherche uniquement aléatoire.

Finalement, une revue des principes des algorithmes génétiques multiobjectifs a été présentée. La technique SPEA a été traitée en détail, car la littérature a montré qu'elle est très performante. Cette technique sera le fondement de la nouvelle technique SPEA-MOD qui sera présentée au chapitre 7.

CHAPITRE 5

COLONIES DE PARTICULES

Les algorithmes d'optimisation de colonies de particules PSO(*Particle Swarm Optimization*), également appelés algorithmes d'essaim de particules, sont des méthodes d'optimisation stochastique s'inspirant du mouvement d'une colonies d'oiseaux à la recherche de nourriture [Eberhart et Kennedy, 1995]. Cette méthode a été inspirée par des résultats de simulations de vols groupés d'oiseaux et de bancs de poissons. Ces simulations ont révélé l'importance du mimétisme dans la compétition qui oppose les individus à la recherche de la nourriture. En effet, les individus sont à la recherche de nourriture qui est dispersée de façon aléatoire dans un espace de recherche, et dès lors qu'un individu localise une source de nourriture, les autres individus vont alors modifier les trajectoires suite à cette nouvelle information. Ce comportement social basé sur l'analyse de l'environnement et du voisinage constitue alors une méthode de recherche de l'optimum par l'observation des comportements des individus voisins. Chaque individu cherche à optimiser ses chances en suivant une tendance qu'il modère par ses propres expériences.

Comme première application des PSO, on dénote la détermination de poids dans un réseau de neurones [Ebarhart *et al.*, 1996]. Ils ont gagné en popularité par la suite comme technique d'optimisation, principalement dans des problèmes ayant des variables de décision réelles [Eberhart et Kennedy, 2001]. Leur simplicité ainsi que leur nombre réduit d'opération sont probablement les deux raisons majeures qui expliquent leur popularité grandissante.

Les algorithmes évolutionnaires (AE) sont fondés sur trois mécanismes : la représentation des individus, la sélection des individus, et le réglage de certains de leurs paramètres. Quant à eux, les PSO s'appuient sur deux de ces mécanismes, car il n'y a pas de fonction de sélection explicite. De plus, dans les PSO traditionnels il n'y a pas de génération de nouveaux individus comme dans les AE traditionnels. Une autre distinction majeure entre les AE traditionnels et les PSO est dans la façon dont les individus sont manipulés. Les PSO emploient un opérateur qui assigne la vitesse d'une particule dans une certaine direction, ce qui peut être vu comme un opérateur de mutation directionnel.

5.1 Description

L'algorithme de colonies de particules est une technique d'optimisation basée sur une population de particules, tout comme les algorithmes évolutionnistes. Une particule est essentiellement un individu d'une population, tel un chromosome l'est pour un algorithme génétique. Chaque particule possède une fonction de coût (ou objectif) devant être optimisée et dont les variables de décision sont représentées par le vecteur de position de la particule. Les PSO peuvent également traiter des problèmes multiobjectifs. Les particules se déplacent dans l'espace de recherche dans le but d'optimiser la fonction de coût. Leur déplacement est influencé par la meilleure position ayant été obtenue par une particule de leur voisinage et par leur expérience passée.

L'algorithme 3 présente les grandes étapes d'un PSO. Le principe de base réside dans la mise à jour du mouvement de chaque particule d'une population. La première étape consiste à initialiser aléatoirement les positions et les vitesses des particules de la population, et ce de façon similaire à ce qui est fait dans un algorithme évolutionniste. Subséquemment pour un nombre d'itérations donné, chaque particule de la population se déplace dans l'espace de recherche influencée par certains paramètres décrits à la section 5.2.

De façon similaire à tout autre algorithme évolutionniste, le critère d'arrêt est généralement un nombre maximal d'itérations ou un critère de convergence. Dans le cas mentionné, il s'agit du nombre maximal d'itérations : itération_{max}.

Algorithme 3 PSO mono-objectif

```
1: Initialisation de la population
2: itération = 1
3: Tant que itération ≤ itération_max Faire
4:     Pour particule ∈ population Faire
5:         Déterminer la meilleure position locale atteinte jusqu'à présent X_L
6:         Évaluer la position
7:         Évaluer la fonction de coût
8:     Fin Pour
9:     Déterminer la meilleure position du voisinage atteinte jusqu'à présent X_G
10:    itération := itération +1
11: Fin Tant que
```

L'évaluation de la meilleure position locale $\vec{X}_L$ atteinte consiste à vérifier si la fonction de coût possède une valeur inférieure à celle obtenue avec la meilleure position locale actuelle $\vec{X}$. Si oui, cette position locale actuelle devient la nouvelle meilleure position locale $\vec{X}_L$.

L'évaluation de la meilleure position globale consiste à déterminer la meilleure position dans le voisinage d'une particule.

5.2 Équations cinématiques

Avant de décrire les équations de déplacement, citons les paramètres influençant le déplacement d'une particule i à l'itération t :
- sa position dans l'espace de recherche, $\vec{X_i}(t)$;
- sa vitesse, $\vec{V_i}(t)$;
- la position de la meilleure solution par laquelle elle est passée, $\vec{X}_{Li}$;
- la position de la meilleure solution de son voisinage, $\vec{X}_{Gi}$;
- le vecteur de vitesse maximale d'une particule $\vec{V}_{MAXi}$;
- son coefficient d'inertie, W ;
- ses coefficients de confiance, ρ_1 et ρ_2.

Remarquons qu'une notation vectorielle est employée, car dans beaucoup de problèmes la dimension est supérieure à 1. Le calcul de vitesse d'une particule est fonction de sa vitesse actuelle $\vec{V_i}(t)$, de la position avec laquelle elle a obtenu la meilleure adaptation $\vec{X}_{Li}$ ainsi que la position ayant permis à une particule d'obtenir la meilleure adaptation de son voisinage $\vec{X}_{Gi}$. La notion de voisinage est couverte dans la sous-section 5.3. Notons que les positions X_L et X_G ne sont pas associées à une génération donnée. L'évaluation de la nouvelle position se fait à l'aide des équations 5.1 et 5.2 respectivement.

$$\vec{V_i}(t+1) = W \cdot \vec{V_i}(t) + \rho_1 \cdot (\vec{X}_{Li} - \vec{X_i}) + \rho_2 \cdot (\vec{X}_{Gi} - \vec{X_i}) \quad (5.1)$$

$$\vec{X_i}(t+1) = \vec{X_i}(t) + \vec{V_i}(t) \quad (5.2)$$

Le terme W est quant à lui le terme d'inertie pondérant l'influence de la vitesse actuelle. Les variables aléatoires de confiance pondèrent les tendances de la particule à vouloir suivre leur propre instinct ou à suivre leur voisinage. Elles sont définies de la façon suivante :

$$\rho_1 = r_1 \cdot c_1 \quad (5.3)$$

$$\rho_2 = r_2 \cdot c_2 \quad (5.4)$$

où r_1 et r_2 sont deux nombres aléatoires de distribution uniforme sur $[0, 1]$ obtenus de façon indépendante et c_1 et c_2 sont des constantes positives déterminées empiriquement

pondérant la contribution entre la meilleure position obtenue de la particule et la meilleure position obtenue par son voisinage. Par exemple, on accorderait plus d'importance à la meilleure position globale P_G qu'à la meilleure position locale P_L si $c_2 > c_1$.

Ainsi, à chaque itération, chaque particule se déplace selon un compromis entre les 3 tendances suivantes (voir la figure 5.1) :

- répéter son mouvement précédent ;
- se diriger vers sa meilleure position passée et ;
- se diriger vers la meilleure position (passée) de son voisinage.

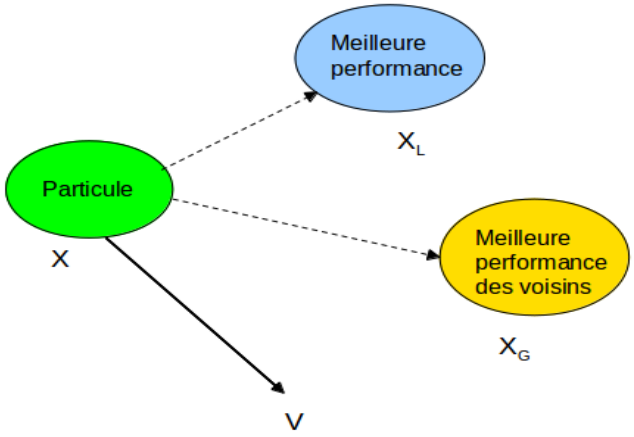
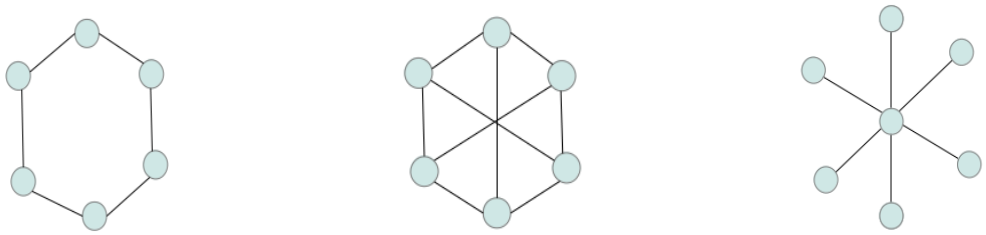


Figure 5.1 Influences d'une particule.

5.3 Topologie du voisinage

Le voisinage d'une particule décrit la façon dont elle est reliée à d'autres particules, caractérisant ainsi sa mise à jour de la meilleure solution du voisinage ($\vec{X}_G$). Il existe une multitude de combinaisons dont celles illustrées à la figure 5.2 sont les plus utilisées.

Malheureusement, il ne semble pas y avoir à ce jour de consensus sur l'utilisation d'une topologie donnée. L'étude détaillée de l'effet des différentes configurations de voisinages sur les résultats d'optimisation n'est pas traitée dans cet ouvrage. Il serait sans aucun doute intéressant d'analyser cet aspect, car la littérature actuelle ne semble pas présenter des résultats concluants.



(a) Anneau (b) Rayon (c) Étoile

Figure 5.2 Trois topologies de voisinage.

5.4 Techniques multiobjectifs

Des extensions à des problèmes multiobjectifs ont vu le jour dû au fait qu'il s'agisse d'une technique simple basée sur une population de solutions [Fieldsend, 2004; Reyes-Sierra et Coello Coello, 2006]. Certaines études ont montré que l'ajout d'un terme de turbulence a un effet positif sur le processus d'optimisation, car il favorise la diversification, évitant ainsi à l'algorithme de converger vers une solution unique [Fieldsend et Singh, 2002; Mostaghim et Teich, 2003]. La turbulence est équivalente à la mutation dans les algorithmes génétiques.

Une particule à la position $\vec{X}$ est poussée (attirée) vers son meilleur voisin $\vec{X}_G$, sa meilleure position passée $\vec{X}_L$ et en direction de son vecteur de vitesse $\vec{V}$. Ceci veut donc dire que le déplacement de cette particule est limité par l'hypercube formé par l'addition de ces trois composantes. La particule peut se déplacer vers n'importe quel point à l'intérieur de l'hypercube, mais pas à l'extérieur. Cette restriction sur le déplacement implique que tout optimum local peut être trouvé à l'intérieur de cette région, mais qu'il serait impossible de trouver un optimum global à l'extérieur de la région au cours de cette itération.

L'ajout d'un terme de turbulence a pour effet d'augmenter le volume de l'hypercube. La turbulence peut donc être vue comme étant un processus stochastique aidant à déloger les particules d'optimum locaux, favorisant ainsi une dispersion des solutions sur le front de Pareto. Par contre, un désavantage de la turbulence est l'addition d'un paramètre de configuration supplémentaire à l'algorithme. Il ne semble pas y avoir de consensus sur la valeur de la turbulence.

5.5 Sommaire

Ce chapitre a revu les grandes caractéristiques des algorithmes de colonies de particules (PSO). Récemment plusieurs études se sont intéressées à cette nouvelle technique d'op-

timisation basée sur le principe des algorithmes évolutionnaires. L'avantage principal de cette technique est sans aucun doute la simplicité de mise en oeuvre.

Les PSO ont d'abord été conçus pour résoudre des problèmes mono-objectif, et ce, de façon similaire aux algorithmes génétiques. Par la suite, des extensions supportant des problèmes multiobjectifs ont vu le jour.

Dans la littérature on retrouve peu d'études, comparativement aux algorithmes génétiques, ayant employés cette technique pour résoudre des problèmes réels complexes. De plus, on retrouve peu de recherches comparant les PSO à d'autres techniques d'optimisation. Dans les prochains chapitres une analyse comparative entre un algorithme génétique et un PSO est présentée.

CHAPITRE 6

CONCEPTION DE DEUX ALGORITHMES D'OPTIMISATION

Tant pour les algorithmes génétiques (AG) multiobjectifs que pour les algorithmes de colonies de particules (PSO) multiobjectifs, on retrouve une multitude de techniques proposées dans la littérature. Tel que mentionné antérieurement, la littérature des AG est très riche dû au fait que ce sont des algorithmes établis depuis plusieurs années. Par contre, bien que les PSO soient des techniques fleurissantes, la littérature y est moins exhaustive. Cette récente variété de techniques explique probablement pourquoi on ne retrouve pas d'étude comparative analysant toutes ces techniques. Ceci rend ardue la sélection d'un algorithme.

Le présent chapitre décrit deux nouveaux algorithmes d'optimisation évolutionnaire multiobjectif qui ont été mis au point au cours de cette recherche. Le premier étant un algorithme génétique alors que le second est un algorithme de colonies de particules. Tout au long de la thèse les deux nouveaux algorithmes seront comparés.

L'approche multiobjectif dans le traitement des contraintes, tel qu'introduit à la section 3.3, a été un facteur clé motivant l'utilisation d'algorithmes multiobjectifs. Cette méthode a été employée dans le cadre de cette recherche, car elle apporte, entre autres, une façon plus naturelle de traiter les objectifs et les contraintes du monde ATC. La section 7.6.1 apporte également des arguments justifiant l'utilisation d'une technique multiobjectif.

Un désavantage, ou défi supplémentaire, d'une approche multiobjectif est de déterminer l'individu qui sera la solution finale au problème d'optimisation. En effet, l'algorithme multiobjectif ne fournit pas qu'une seule solution, mais bien un ensemble de solutions, ce qui cause ainsi un problème dans l'application du résultat de l'optimisation. Pour pallier cet inconvénient, des techniques de sélection préférentielle peuvent être employées [Coello Coello, 2000b]. Ces techniques peuvent généralement être classées en quatre catégories, selon le moment pendant la procédure d'optimisation où l'utilisateur manifeste sa préférence pour les objectifs : jamais (non préférentielle), avant (a priori), pendant (interactive) ou après (posteriori) la procédure d'optimisation. Le but étant d'obtenir une seule solution du processus d'optimisation, et ce, basé selon certaines préférences. Le traitement des préférences

n'a pas été inclus dans ce projet de recherche. Le but de cette recherche est de modéliser le comportement de deux tâches du contrôle aérien de façon optimale. Le choix de la solution, parmi un ensemble de solutions, n'est pas considéré comme faisant partie de l'optimisation proprement dite.

Les deux nouveaux algorithmes sont présentés aux sections 6.1 et 6.2 respectivement. La section 6.3 quant à elle, traite de l'implantation de ces deux algorithmes. Finalement, les sections 6.4 et 6.6 présentent des problèmes multiobjectifs typiques ainsi que des résultats de résolutions de ces problèmes en utilisant les algorithmes décrits dans ce chapitre, et ce dans le but de valider les implémentations.

6.1 SPEA-MOD

Dans le monde actuel de l'ATC, plus particulièrement tout ce qui touche à la résolution de conflits, il est primordial d'obtenir des solutions réalisables, et ce même si ces solutions ne sont pas optimales. Ainsi, l'optimisation d'un problème donné est repoussée au second plan. Dans un premier temps, l'algorithme tente de satisfaire les contraintes du problème, alors que dans le second temps l'algorithme optimise le problème en utilisant les solutions qui satisfont les contraintes. C'est un des principes importants qui a inspiré l'élaboration des deux algorithmes employés dans le cadre de cette recherche. Les algorithmes sont basés en partie sur les études suivantes : [Jiménez et Verdegay, 1999; Venkatraman et Yen, 2005; Zitzler et Thiele, 1998]. De plus, l'auteur de cette présente recherche s'est intéressé à développer un algorithme génétique ne nécessitant pas de connaissance du problème à résoudre pour qu'il soit facilement adaptable à une gamme variée de problèmes.

L'algorithme génétique multiobjectif développé lors de cette étude fut inspiré de l'algorithme SPEA [Zitzler et Thiele, 1998] et de l'algorithme pour traiter les contraintes de Venkatraman et Yen [Venkatraman et Yen, 2005]. La technique proposée, appelée SPEA-MOD, est décrite par l'algorithme 4. Le nom SPEA-MOD signifie l'algorithme original (SPEA) modifié (MOD). Une description de haut niveau de certaines étapes est décrite dans la présente section. Cet algorithme diffère de l'algorithme SPEA original (page 46) en deux points importants, soit la méthode de sélection des individus du bassin de reproduction et la méthode d'évaluation des individus de la population et de l'archive. Dans l'algorithme 4, P_g est la population générée à la génération g, et $P_g(i)$ est l'individu i de la population P_g. *Aléa(probabilité)* est une valeur aléatoire, prenant la valeur vrai ou faux, évaluée en fonction d'une certaine probabilité. La pertinence et les améliorations de ces modifications seront abordées un peu plus loin dans ce chapitre.

Algorithme 4 SPEA-MOD

Initialisation compteur de générations $g = 1$
Génération maximale $gMax$
Initialisation populations aléatoires de P_g et P_{g-1}
Initialisation archive à l'ensemble vide $A = \{0\}$
Taille de la population $N_p = |P_g|$
Tant que $g \leq gMax$ **Faire**
    Évaluation des individus
    Ensemble de reproduction $Repro = A + P_{g-1}$
    Évaluer l'adaptation globale de $Repro$
    Mise à jour de l'archive A avec $Repro$
    Pour $i \leftarrow 1, N_p$ **Faire**
        père = sélection par rang($Repro$)
        mère = sélection par rang($Repro$)
        Si un nombre aléatoire $>$ probabilité de croisement **Alors**
            $enfant_1$ = croisement(père, mère)
            $enfant_2$ = croisement(mère, père)
        Sinon
            $enfant_1$ = père
            $enfant_2$ = mère
        Fin Si
        muter $enfant_1$ selon probabilité de mutation
        muter $enfant_2$ selon probabilité de mutation

        $P_g(i) = enfant_1$
        $P_g(i+1) = enfant_2$

        $g = g + 1$
    Fin Pour
Fin Tant que

Le terme d'initialisation de la population à la ligne 3 consiste à initialiser aléatoirement tous les individus de la population actuelle P_g et de la population précédente P_{g-1}. L'archive quant à elle est initialisée à l'ensemble vide. Les variables de décision du problème sont initialisées à des valeurs aléatoires comprises dans leurs plages admissibles respectives.

Le critère d'arrêt de la boucle principale de l'algorithme est un nombre de générations maximales $gMax$, mais un autre critère pourrait également être employé. La première étape de cette boucle consiste à former l'ensemble de reproduction, ou population intermédiaire. Cet ensemble est à son tour utilisé par l'algorithme de sélection.

Une mutation non-uniforme, telle que décrite par l'équation 4.2.1, a été employée. Cette mutation, comparativement à la mutation uniforme, favorise une meilleure convergence de l'algorithme, car son effet diminue avec les générations. Cet aspect sera abordé plus loin dans la section des résultats.

La sélection par tournoi n'a pas été employée dans SPEA-MOD, contrairement au modèle original SPEA qui est présenté à l'algorithme 1. Elle requiert un plus grand nombre d'opérations de calcul, augmentant ainsi le temps de résolution. Cette méthode est un peu plus complexe en multiobjectif qu'elle ne l'est en mono-objectif [Deb *et al.*, 2000; Horn *et al.*, 1994]. La sélection devient plus complexe lorsque les deux individus comparés sont non dominés, car d'un point de vue multiobjectif, ils sont équivalents.

La mise à jour de l'archive, qui est décrite par l'algorithme 5, consiste à placer les individus non dominés dans une population, appelée l'archive. Le but de cette archive est de conserver les bonnes solutions pour les prochaines générations. On y compare chaque individu de la population avec chaque individu de l'archive, dans le but de déterminer si des individus de la population pourraient être promus à l'archive. L'algorithme 2 de réduction de taille d'une population, présenté à la section 4.4 est utilisé pour contrôler la taille de cette archive.

Rappelons que $\prec$ est l'opérateur de dominance de Pareto, qui a été présenté à la définition 1 (page 22) et $==$ est l'opérateur de comparaison logique.

6.1.1 Traitement des contraintes

Le traitement des contraintes est particulièrement important dans la problématique de résolution de conflits et de séquençage d'avions. Une solution doit avant tout satisfaire toutes les contraintes pour être réalisable et être acceptable. Tel que mentionné à la section 3.3, il existe plusieurs alternatives pour traiter les contraintes dans un problème d'optimisation.

Algorithme 5 Mise à jour de l'archive

```
1: Données : A et P
2: individu ajouté = faux
3: Pour i ← 1, |P| Faire
4:     nombre dominés = 0
5:     Pour j ← 1, |A| Faire
6:         Si A(j) ≺ P(i) Alors
7:             nombre dominés = nombre dominés +1
8:         Fin Si
9:     Fin Pour
10:    Si nombre dominés == 0 Alors
11:        Ajouter P(i) à A
12:        individu ajouté = vrai
13:    Fin Si
14: Fin Pour
15: Si individu ajouté == vrai Alors
16:    enlever individus dominés de A
17:    enlever individus identiques de A
18:    réduction de taille A (algorithme 2, p. 48)
19: Fin Si
```

Plusieurs de ces techniques, comme la méthode de pénalités, reposent sur une certaine connaissance du problème en question, rendant quelque peu ardue l'adaptation de la technique à un autre problème.

Venkatraman et Yen ont développé un algorithme génétique mono-objectif traitant les contraintes qui ne nécessitent pas la connaissance du problème [Venkatraman et Yen, 2005]. Leur algorithme repose sur le principe que les contraintes sont en fait des objectifs. Ainsi, l'algorithme est divisé en deux phases, soit la satisfaction des contraintes et celle de l'optimisation proprement dite du problème. Dans la première phase, l'algorithme est à la recherche d'au moins une solution viable en traitant les contraintes comme étant des objectifs à minimiser. L'algorithme bascule ensuite en phase d'optimisation du problème contraint lorsque des solutions viables sont découvertes. C'est ce principe qui est à la base du traitement des contraintes dans les algorithmes présentés dans ce chapitre, par le biais de l'opérateur de dominance de Pareto contrainte.

La ligne 7 de l'algorithme 4 consiste à évaluer tous les individus de la population. Tout d'abord, les contraintes d'un individu sont évaluées et si cet individu s'avère viable, alors ses objectifs sont également évalués. Les fonctions objectives ne sont pas évaluées si l'individu est non-viable, car ces dernières n'interviendront pas dans le calcul de l'adaptation globale,

évitant des opérations de calculs inutiles. La définition 1 de la page 27 est employée comme méthode comparative entre deux individus dans l'adaptation globale (algorithme 6).

Ainsi, les quatre points suivants décrivent la relation entre des individus viables et non-viables dans l'algorithme SPEA-MOD :

1. Les populations, incluant l'archive, peuvent contenir des individus viables et non viables.
2. Les individus viables, guidés par les objectifs, évoluent vers l'optimum.
3. Les individus non-viables, guidés par les contraintes, évoluent vers la région réalisable de l'espace de recherche.
4. Les individus viables ont une plus grande probabilité d'être sélectionnés pour la mutation et le croisement que ceux non-viables.

6.1.2 Sélection d'individus

La sélection est basée sur l'adaptation globale présentée à l'algorithme 6 qui est similaire au calcul du rang employé dans l'algorithme NSGA [Deb, 2008]. Ainsi, les individus les plus performants, c'est-à-dire ceux ayant la plus petite adaptation, sont choisis pour les opérations génétiques de mutation et de croisement. Cette méthode de sélection, à partir d'une population intermédiaire, a pour but de favoriser la diversification lorsque l'archive est à sa taille maximale, évitant ainsi de converger vers un optimum local. Le processus de sélection choisit toujours un individu de l'archive lorsque cette dernière est de taille inférieure à la limite maximale permise. Par contre, à taille maximale, la sélection pourrait choisir un individu de l'archive ou de la population. Tous les individus non dominés ont le même rang, puisqu'ils sont d'aussi bonnes solutions les unes que les autres. S'il y a plus d'une solution de même rang, alors la sélection d'un individu se fait aléatoirement parmi l'ensemble non dominés.

6.2 PSO-MO

Cette section présente les modifications apportées à l'algorithme de colonies de particules (PSO) classique pour lui permettre de résoudre des problèmes multiobjectifs (MO), d'où le nom proposé PSO-MO.

Il y a principalement trois obstacles pour étendre l'utilisation des algorithmes PSO aux problèmes multiobjectifs [Reyes-Sierra et Coello Coello, 2006] :

Algorithme 6 Adaptation globale

```
1: individu ajouté = faux
2: Pour i ← 1, taille population Faire
3:     nombre dominés = 0
4:     Pour j ← 1, taille archive Faire
5:         Si archive(j) ≺ pop(i) Alors
6:             nombre dominés = nombre dominés + 1
7:         Fin Si
8:     Fin Pour
9:     Si nombre dominés == 0 Alors
10:        ajouter pop(i) à l'archive
11:        individu ajouté = vrai
12:    Fin Si
13: Fin Pour
```

1. Comment faire les sélections de particules devant être élues comme chef pour donner préséance aux solutions non dominées ?
2. Comment retenir les solutions non dominées trouvées durant la recherche ? De plus, comment s'assurer que ces dernières soient distribuées sur le front de Pareto ?
3. Comment maintenir la diversité pour éviter la convergence vers une seule solution ?

La technique proposée, appelée *PSO-MO*, est décrite par l'algorithme 7. L'algorithme emploie un terme de turbulence, tel que montré à la ligne 15. Cette turbulence est en fait une mutation du vecteur de position de chaque particule. Ainsi, comme dans la mutation présentée à la section 4.2.1, chaque élément du vecteur est affecté par la turbulence selon une certaine probabilité. Une mutation non-uniforme, telle que décrite à la section 4.2.1, a été employée. L'ajout de la turbulence augmente grandement la performance de l'algorithme, et ce point est traité à la section 6.6.

La ligne 13 consiste en l'évaluation de la nouvelle meilleure position locale P_L et à vérifier si la position actuelle de la particule domine cette position locale P_L.

La fonction de sélection, employée à la ligne 7, choisit un individu de l'archive et non de l'union de la population et de l'archive, tel que présenté précédemment dans l'algorithme génétique. Pour éviter de favoriser un individu donné, la sélection se fait de façon circulaire, c'est-à-dire que tous les individus de l'archive sont choisis à tour de rôle. Une sélection d'individus favorisant une distribution dans l'archive aurait également pu être employée. Par contre, bien que cette méthode aurait probablement aidé à répartir les solutions sur tout le front, elle aurait grandement affecté les performances en terme de temps de calcul. Cette méthode n'a pas été envisagée étant donné les bonnes performances de l'algorithme

Algorithme 7 PSO-MO

```
1: Initialisation population aléatoire Pop
2: Initialisation archive à l'ensemble vide A = {0}
3: Déterminer la meilleure position globale P_G
4: itération = 1
5: Tant que itération ≤ itération_max Faire
6:     Mise à jour de l'archive (Algorithme 5, page. 61)
7:     Sélection particule P de l'archive
8:     Si P domine P_G Alors
9:         P_G = P
10:    Fin Si
11:    i = 0
12:    Pour particule ∈ population Faire
13:        Déterminer nouvelle P_L (meilleur local)
14:        Évaluer position selon les équations 5.1 à 5.4
15:        Ajout de la turbulence(pTurb)
16:        Évaluer l'adaptation
17:    Fin Pour
18:    itération = itération +1
19: Fin Tant que
```

PSO-MO, et ce tel que présenté à la section 6.6. L'individu choisi par la méthode de sélection devient le meilleur individu uniquement s'il domine, au sens de Pareto, le meilleur individu utilisé présentement. Ceci a pour but d'éviter de changer le meilleur individu trop fréquemment et ainsi d'assurer une certaine stabilité.

Rappelons que les équations de déplacement d'une particule (équations 5.1 à 5.4) ont été présentées à la section 5.2. Ces équations sont également employées dans l'algorithme PSO-MO. À ce jour, il ne semble pas y avoir de consensus sur les valeurs à attribuer aux facteurs W, C_1, et C_2, qui sont employés dans ces équations. De plus, selon les divers essais effectués au cours de cette recherche, il semble que les choix de ces facteurs affectent grandement la qualité des solutions. Les auteurs [Hassan *et al.*, 2004] suggèrent les valeurs de 0.5, 1.5 et 1.5 pour W, C_1, et C_2 respectivement lors d'essais sur des problèmes mono-objectif. Selon les essais de l'auteur, les valeurs 6, 1.5 et 2.5 offraient de meilleurs résultats. De plus, comme pour les algorithmes génétiques, il ne semble pas y avoir de recommandation sur la taille de la colonie à utiliser. Plusieurs auteurs proposent un critère d'arrêt basé sur la convergence plutôt que sur un certain nombre de générations. À des fins de comparaison entre les deux techniques, SPEA-MOD et PSO-MO, un nombre de générations maximales (ou itérations) est utilisé comme critère d'arrêt.

6.3 Mise en oeuvre

Il existe quelques librairies d'algorithmes génétiques et de colonies de particules disponibles dans le domaine libre (*Open Source*), principalement pour l'optimisation mono-objectif. La majorité d'entre elles ont été conçues pour résoudre spécifiquement certains problèmes. Parmi la panoplie de librairies d'algorithmes génétiques et de colonies de particules disponibles, des plus intéressantes on retrouve GAUL (The Genetic Algorithm Utility Library) [Adcock, 2005], GAlib (*A C++ Library of Genetic Algorithm Components*) [Wall, 1999] et JGAP (Java Genetic Algorithm Package) [Rotstan, 2002].

De par la nature des algorithmes évolutionnaires, il peut devenir très fastidieux de résoudre un problème d'implémentation sans avoir recours à des tests unitaires. Un test unitaire est en fait une fonction permettant de tester l'implantation d'une fonction ou d'une méthode d'une classe, par exemple. Aucune des librairies consultées, mise à part JGAP, ne possédait une grande quantité de tests unitaires, ainsi il aurait été assez laborieux de valider l'implantation d'une d'entre elles.

Les librairies GAUL et GAlib ne supportent pas l'optimisation multiobjectif, qui est un critère de sélection fondamental. De plus, il ne semble plus y avoir d'activité depuis 2005 (mise à jour) sur le site de la librairie. La librairie JGAP représente maintenant un choix très intéressant, mais malheureusement au moment (2005-2006) où l'auteur devait choisir une librairie, cette dernière n'était pas disponible [Rotstan, 2002]. JGAP est écrit en JAVA et semble bien supporter les problèmes multiobjectifs. Elle possède un nombre impressionnant de tests unitaires et une communauté importante participe dans le développement et le maintien de la librairie. Si hypothétiquement, le projet de recherche était à refaire, l'utilisation de JGAP serait probablement le choix approprié.

L'architecture de GAlib est bien faite et il est facile de s'y retrouver. Par contre, son implantation utilisant la programmation structurée du langage C s'intègre plus difficilement avec les librairies modernes du langage C++ telles que STL (*Standard Template Librairy*) [Stroustrup, 1997] et BOOST [Boost, 2012]. L'utilisation de ces librairies aide à éviter certains problèmes de programmation, telles que des fuites de mémoire, par l'utilisation de pointeurs intelligents.

Pour remédier aux problèmes rencontrés dans les librairies disponibles, une nouvelle librairie a été conçue au cours de cette recherche. Son architecture a été grandement inspirée de celle de GAlib. L'auteur a choisi le langage de programmation C++ principalement pour sa rapidité d'exécution, mais le langage JAVA aurait également été un autre choix intéressant. Il existe cependant un inconvénient du C++, surtout pour des projets de recherche

nécessitant des prototypes, qui est la compilation. Pour accélérer le développement, certaines parties du projet ont été écrites avec le langage Python.

Python est un langage interprété et non-compilé dont la syntaxe est généralement simple. Il s'agit d'un langage approprié pour le prototypage rapide [Kuchling, 2013]. Pour ce faire, la librairie BOOST-Python a été employée, permettant ainsi de lier le langage C++ au langage Python [Boost, 2012]. Avec l'aide de cette librairie, il est possible d'écrire un programme Python qui emploie des fonctions et des classes écrites en C++. L'amalgame C++ et Python permet de combiner les points forts des deux langages, soit la rapidité d'exécution et la facilité de programmation. Au cours de cette recherche, cet amalgame a été employé dans certains programmes de tests et de simulation. La section 8.1 décrit comment cet amalgame a été utile dans le contexte de résolution de conflits de la phase de vol en route.

Les figures 6.1 et 6.2 illustrent deux diagrammes de classe UML (*Unified Modelling Language*) [Fowler, 2004] de haut niveau pour l'algorithme génétique SPEA-MOD et l'algorithme de colonies de particules PSO-MO respectivement. Plusieurs classes utilitaires de la librairie ne sont pas représentées sur ces deux diagrammes. Il est à noter que selon la représentation UML, un losange vide symbolise une agrégation, une ligne tiretée suivie d'un triangle symbolise l'implantation d'une interface ou d'une classe composante et une ligne pleine suivie d'un triangle symbolise la généralisation ou la dérivation. Les deux diagrammes UML reflètent la grande similarité entre les algorithmes génétiques et les algorithmes de colonies de particules. Par simplicité et clarté, les différents attributs et méthodes des classes ont été omis. On remarque sur cette figure que toutes les classes importantes (*Genome*, *Population* et *AlgoBase*) sont sous la forme de patrons de composants, ou communément appelées *template* [Stroustrup, 1997]. Ce concept permet de paramétrer, dans un composant (une fonction ou une classe), le type de certaines données manipulées. Les classes dérivées de ces composants peuvent être utilisées dans l'optimisation de problèmes à variables réelles tel que présenté dans ce chapitre. Pour des raisons de simplicité, ces classes ne sont pas utilisées dans les problèmes de résolutions de conflits et de séquençage. Ce point sera revu ultérieurement aux chapitres 7 et 9.

Ces deux diagrammes illustrent la très grande similitude entre l'implémentation des deux algorithmes développés au cours de ce projet. Ils identifient également deux différences dans leur mise en oeuvre. La première étant liée aux classes composantes PSO-MO et SPEA-MOD. Ces deux classes mettent en oeuvre les algorithmes 4 et 7 respectivement. La seconde différence est liée aux classes *GeneReel* et *ParticuleElementReel*, qui constituent les éléments de base servant à la création d'un individu ou d'une particule. Leur utilisation

dans la classe composante *Genome* est indiquée par un arc vert. Généralement, la mise en oeuvre de classes composantes est plus complexe que celles de classes traditionnelles. L'avantage d'avoir utilisé cette approche est la facilité avec laquelle il est possible de les adapter à différentes représentations (ou codage) par le biais d'un type paramétré.

Les AG et les PSO sont des algorithmes basés sur une population d'individus. Tel qu'illustré par les diagrammes UML, un individu est décrit par la classe *GenomeReel* pour le AG et par la classe *ParticuleReel* pour le PSO, qui tous deux sont des implémentations de la classe *Genome* ayant un type paramétré *GeneGAreel* et *GenePSOreel* respectivement. De la même façon, les classes *PopulationGAreel* et *PopulationPSOreel* sont des implémentations de la classe *Population* ayant les types paramétrés *GenomeReel* et *ParticuleReel* respectivement.

Finalement, plusieurs tests unitaires ont été développés pour s'assurer de la validité des calculs accomplis de la partie implémentée en C++. Il n'a pas été nécessaire de faire des tests unitaires pour la portion implantée en Python, car elle utilise, par l'entremise de BOOST-Python, l'implémentation C++. La mise en oeuvre de cette librairie a nécessité environ 12000 lignes de code en langage de programmation C++.

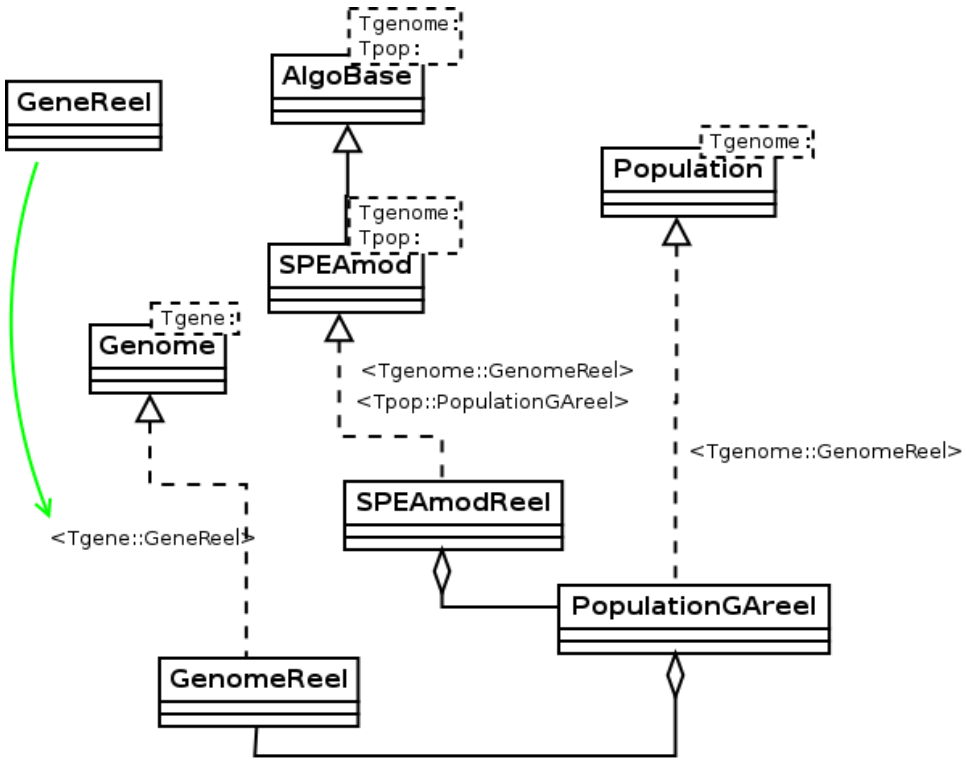


Figure 6.1 Diagramme UML de haut niveau de l'algorithme SPEA-MOD.

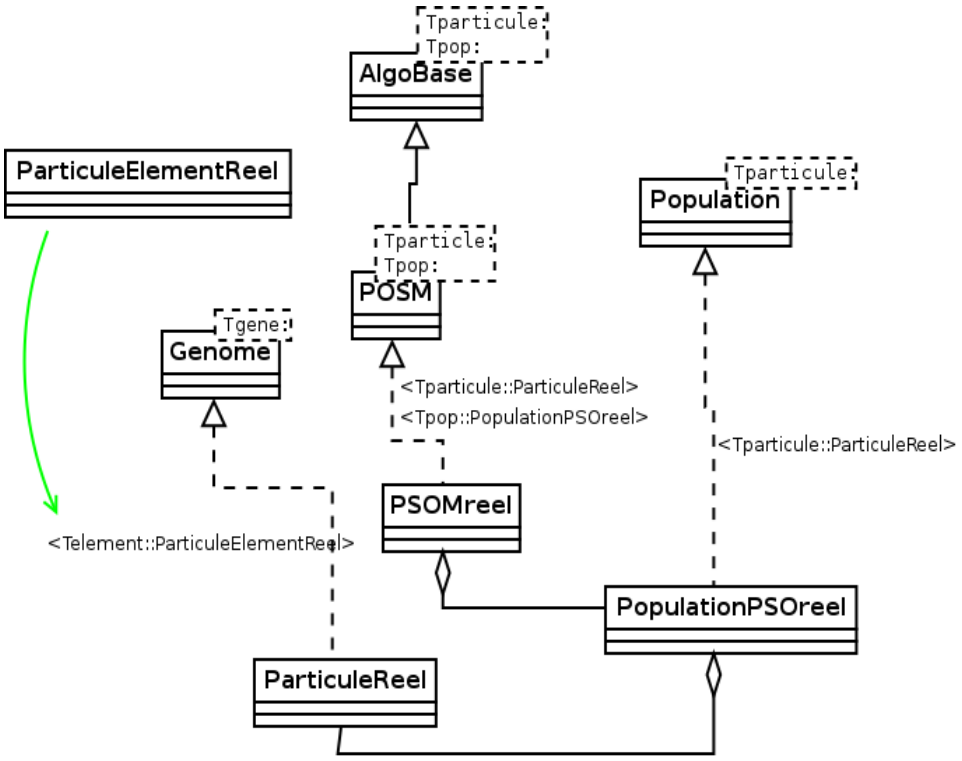


Figure 6.2 Diagramme UML de haut niveau de l'algorithme PSO-MO.

6.4 Comparaison à des problèmes typiques

Plusieurs problèmes tests ont vu le jour afin d'analyser la performance de différents algorithmes d'optimisation multiobjectif évolutionnaires. Ces problèmes sont intéressants pour étudier certains critères qui rendent l'optimisation multiobjectif difficile, par exemple, la convergence au front de Pareto et la répartition uniforme des solutions candidates sur le front de Pareto.

L'algorithme génétique décrit à la section 6.1 et l'algorithme de colonies de particules décrit à la section 6.2 ont été comparés à certains problèmes tests, présentés aux sous-sections ci-dessous, dans le but de valider sa performance. Le but étant de vérifier la validité des algorithmes proposés avant d'attaquer des problèmes de résolution de conflits. Les problèmes présentés sont uniquement à deux dimensions. Selon Zitzler et al., les problèmes à deux dimensions reflètent les différents aspects essentiels de l'optimisation multiobjectif [E. Zitzler et Thiele, 2000]. Les problèmes présentés ont été utilisés dans plusieurs études.

Quatre critères ont été utilisés dans la comparaison des deux algorithmes élaborés, soit :

1. la répartition des solutions sur le front de Pareto ;
2. la similitude des fronts obtenus comparativement à des résultats de d'autres études ;
3. la métrique C présentée à la section 6.5.1 ;
4. le temps de calcul.

6.4.1 Zitzler-Deb-Thiele

Zitzler, Deb et Thiele ont développé six problèmes typiques dans le but de comparer les algorithmes multiobjectifs entre eux [E. Zitzler et Thiele, 2000]. De ces tests, seul le test cinq n'a pas été employé, car il s'agit d'un problème de chaîne binaire. Dans ces problèmes, sauf le cinquième, le front de Pareto est formé par $g(x) = 1$. Ces problèmes de minimisation ont deux objectifs, $f_1(x)$ et $f_2(x)$. Le second objectif est relié au premier objectif par l'intermédiaire de deux fonctions, $g(x)$ et $h(x)$. $f_1(x)$ est traité comme étant une variable de la fonction $f_2(x)$. Les problèmes sont tous définis comme suit :

$$\begin{aligned} & f_1(x) \\ & f_2(x) \quad = \quad g(x)h(f_1(x), g(x)) \end{aligned}$$

ZDT1

Ce problème est formé par 30 variables de décision ($n = 30$) comprises entre $[0, 1]$ et possède un front de Pareto convexe. Ce problème est le plus simple de ceux proposés par ces auteurs [E. Zitzler et Thiele, 2000], car le front de Pareto est continu et il est formé par une distribution uniforme des solutions. Le nombre important de variables représente la difficulté pour un algorithme multiobjectif. La figure 6.3 présente le front de Pareto théorique de ce problème.

$$\begin{aligned} F_1(x) &= x_1 \\ G(x) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ H(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \tag{6.1}$$

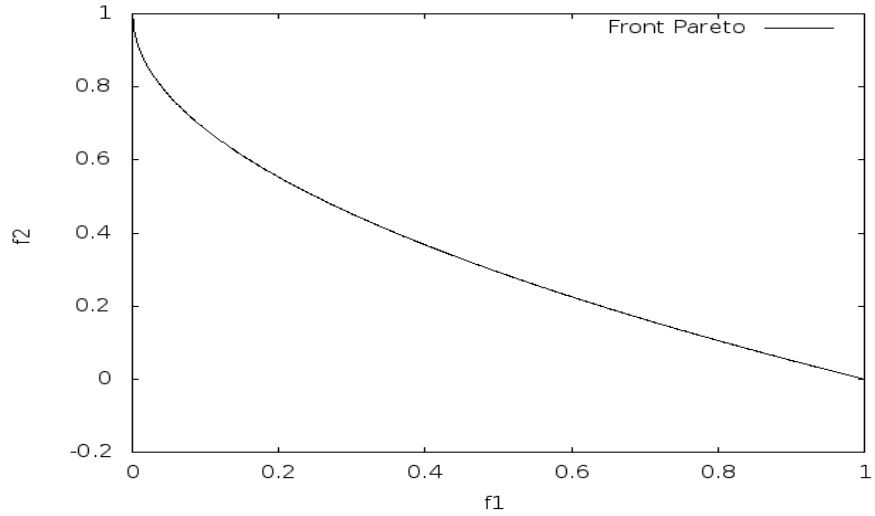


Figure 6.3 Front de Pareto du problème ZDT1.

ZDT2

Ce problème est également formé par 30 variables de décision ($n = 30$) comprises entre $[0, 1]$ et possède une distribution uniforme des solutions sur le front de Pareto. Nonobstant le nombre élevé de variables, l'unique difficulté est causée par le front de Pareto non convexe. La figure 6.4 présente le front de Pareto théorique de ce problème.

$$\begin{aligned} f_1(x) &= x_1 \\ g(x) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g) &= 1 - (f_1/g)^2 \end{aligned} \tag{6.2}$$

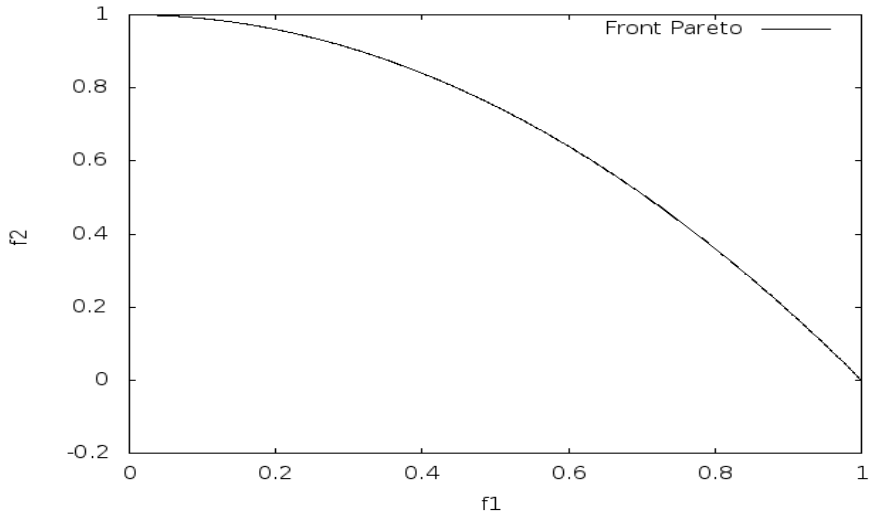


Figure 6.4 Front de Pareto du problème ZDT2.

ZDT3

Ce problème est également formé par 30 variables de décision ($n = 30$) comprises entre $[0, 1]$. La difficulté de ce problème est occasionnée par la discontinuité du front de Pareto. Ainsi, le défi pour un algorithme multiobjectif est de trouver toutes les régions continues et ce avec une répartition uniforme des solutions non dominées sur ces régions. La figure 6.5 présente le front de Pareto théorique de ce problème.

$$\begin{aligned} f_1(x) &= x_1 \\ g(x) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g) &= 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1) \end{aligned} \tag{6.3}$$

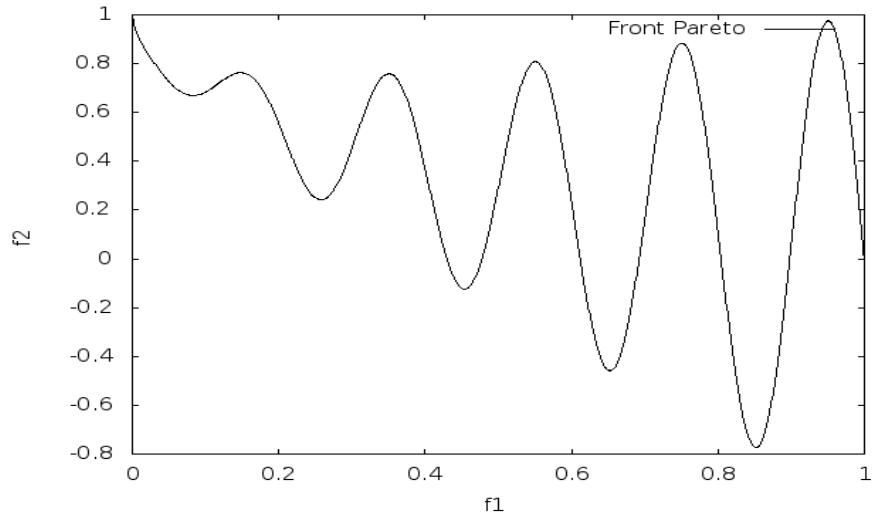


Figure 6.5 Front de Pareto du problème ZDT3.

ZDT4

Ce problème, de 10 variables ($n = 10$), possède un front de Pareto optimal convexe. La première variable, x_1, est comprise entre $[0, 1]$, alors que les autres sont comprises entre $[-5, 5]$. La figure 6.6 présente le front de Pareto théorique de ce problème.

$$\begin{aligned} f_1(x) &= x_1 \\ g(x) &= 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \tag{6.4}$$

Ce problème possède 21^9 solutions locales de Pareto. Le front optimal de Pareto correspond à $g(x) = 1$. Le second front de Pareto (optimum local) est pour $g(x) = 1.25$, le suivant pour $g(x) = 1.50$ et ainsi de suite jusqu'au dernier front local correspondant à $g(x) = 25.0$. Ainsi, le problème possède 100 fronts locaux de Pareto au total. La figure 6.6 montre le front de Pareto optimal ainsi que les deux premiers fronts locaux. La difficulté de ce problème, pour un algorithme multiobjectif, est le nombre de fronts locaux qui rend la convergence au front optimal de Pareto ardue.

ZDT6

Ce problème de 10 variables, comprises entre $[0, 1]$, possède un front de Pareto non-convexe. De plus, la densité de solutions dans la région immédiate du front de Pareto est non uniforme et la densité de solutions vers le front est mince. Le front de Pareto correspond

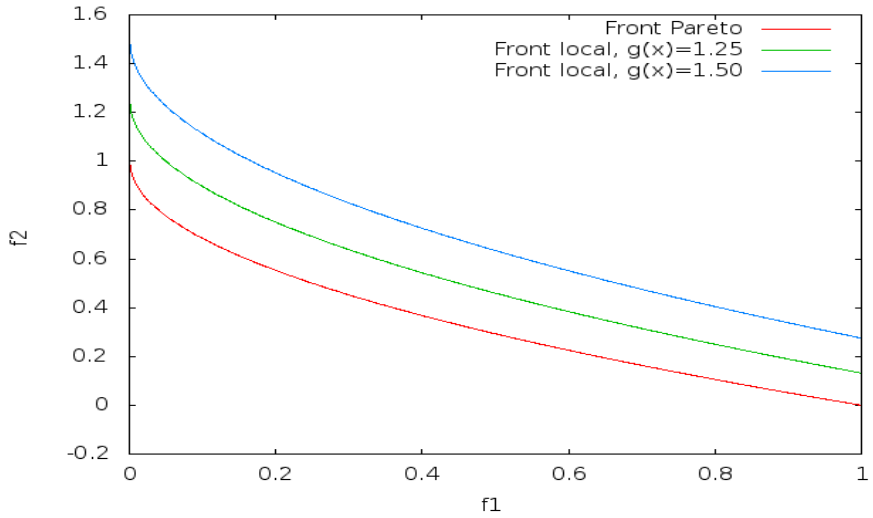


Figure 6.6 Fronts de Pareto locaux et global du problème ZDT4.

à $0 \leq x_1 \leq 1$ et $x_i = 0$ pour $i = 2, 3, \ldots, 10$. La figure 6.7 présente le front de Pareto théorique de ce problème.

$$\begin{aligned} f_1(x) &= 1 - e^{-4x_i} \sin^6(6\pi x_1) \\ g(x) &= 1 + 9\left[\left(\sum_{i=2}^{10} x_i\right)/9\right]^{0.25} \\ h(f_1, g) &= 1 - (f_1/g)^2 \end{aligned} \tag{6.5}$$

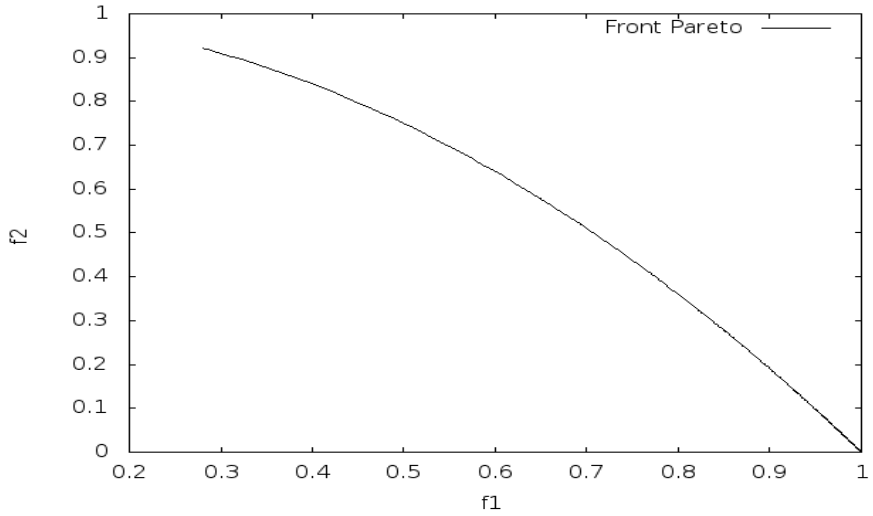


Figure 6.7 Front de Pareto du problème ZDT6.

6.4.2 BNH

Binh et Korn ont développé un problème de minimisation à deux variables, $x_1 \in [0, 5]$ et $x_2 \in [0, 3]$, ayant deux contraintes [Binh et Korn, 1997]. La figure 6.8 présente le front de Pareto théorique de ce problème.

$$\begin{aligned} f_1(x) &= 4x_1^2 + 4x_2^2 \\ f_2(x) &= (x_1 - 2)^2 + (x_2 - 5)^2 \\ c_1(x) &\equiv (x_1 - 5)^2 + x_2^2 \leq 25 \\ c_2(x) &\equiv (x_1 - 8)^2 + (x_2 + 3)^2 \geq 7.7 \end{aligned} \tag{6.6}$$

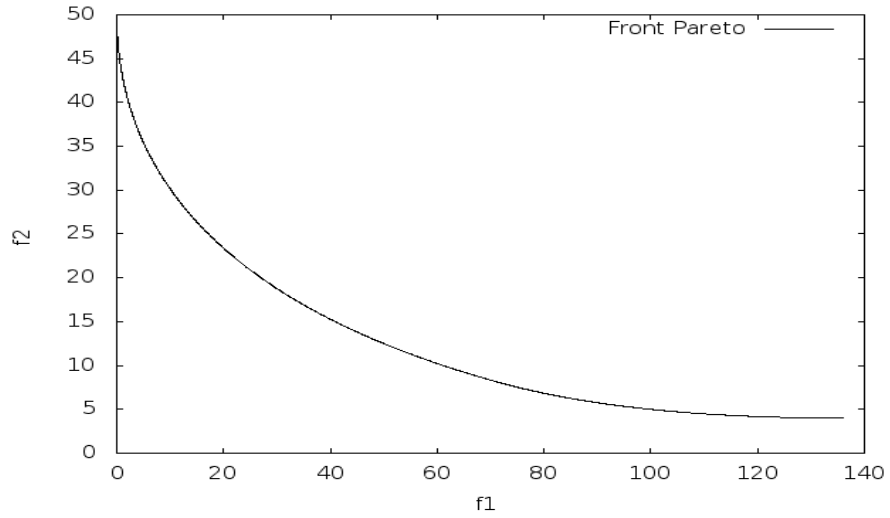


Figure 6.8 Front de Pareto du problème BNH.

6.4.3 OSY

Osyxzka et Kundu ont développé un problème de minimisation à six variables [Osyzka et Kundu, 1995]. Les deux fonctions objectives f_1 et f_2 du problème sont assujetties à six contraintes de la façon suivante :

$$
\begin{aligned}
f_1(x) &= -\big(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2\big) \\
f_2(x) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \\
c_1(x) &\equiv x_1 + x_2 \geq 0 \\
c_2(x) &\equiv 6 - x_1 - x_2 \geq 0 \\
c_3(x) &\equiv 2 - x_2 + x_1 \geq 0 \\
c_4(x) &\equiv 2 - x_1 + 3x_2 \geq 0 \\
c_5(x) &\equiv 3 - (x_3-3)^2 - x_4 \geq 0 \\
c_6(x) &\equiv (x_5-3)^2 + x_6 - 4 \geq 0
\end{aligned}
\tag{6.7}
$$

où $0 \leq x_1, x_2, x_6 \leq 10$, $1 \leq x_3, x_5 \leq 5$, $0 \leq x_4 \leq 6$. Le front de Pareto est la concaténation de cinq régions. Ce problème est difficile à résoudre, car il demande à un algorithme d'optimisation multiobjectif de maintenir les individus de sa population à différentes intersections des contraintes. La figure 6.9 présente le front de Pareto théorique de ce problème.

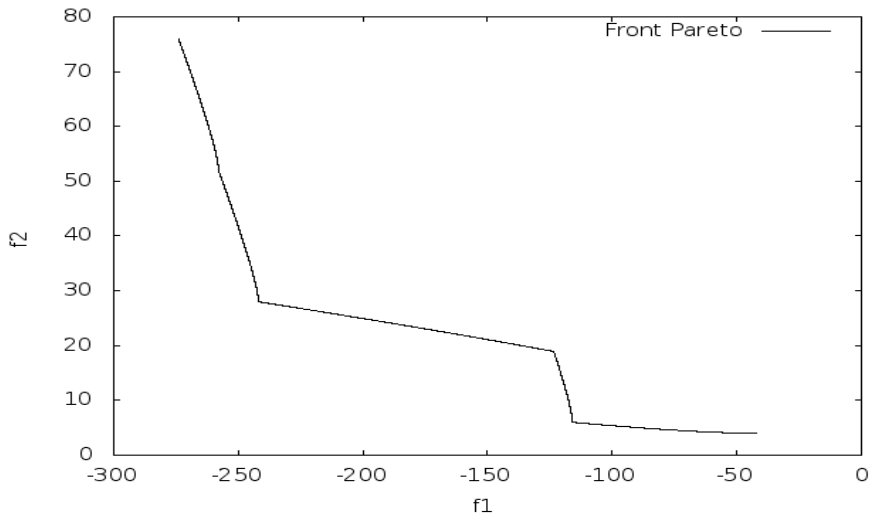


Figure 6.9 Front de Pareto du problème BNH.

6.5 Critères d'analyse

L'analyse des résultats d'algorithmes multiobjectifs est plus complexe que l'analyse de résultats d'algorithmes mono-objectif. De façon générale, les deux critères suivants doivent être considérés :

– minimiser la distance des solutions non dominées au front de Pareto, et

– obtenir une distribution uniforme des solutions sur le front.

Afin de comparer les deux algorithmes de façon objective, des métriques définies dans cette section sont employées [E. Zitzler et Thiele, 2000; Hu *et al.*, 2001; Tsou *et al.*, 2007]. Au lieu de déterminer si une solution de $X^{'}$ appartient au front de Pareto (P^{*}), les métriques ont pour but de comparer la qualité des fronts de Pareto approximés par les deux algorithmes.

6.5.1 La métrique C

La métrique C a été proposée par Zitzler et Thiele. [E. Zitzler et Thiele, 2000]. Soit $X^{'}, X^{''} \subseteq X$ deux ensembles de solutions, la fonction C assigne une valeur à la paire $(X^{'}, X^{''})$ dans l'intervalle $[0, 1]$:

$$C(X^{'}, X^{''}) = \frac{|a^{''} \in X^{''}; \exists a^{'} \in X^{'} : a^{'} \leq a^{''}|}{|X^{''}|} \tag{6.8}$$

La valeur $C(X^{'}, X^{''}) = 1$ indique que toutes les solutions de $X^{''}$ sont dominées ou égales à au moins une solution de $X^{'}$, alors que $C(X^{'}, X^{''}) = 0$ indique qu'aucune solution de $X^{''}$ n'est dominée par l'ensemble $X^{'}$. Notons également que $|X^{''}|$ représente le cardinal de $X^{''}$.

Un problème de cette métrique est qu'elle ne tient pas compte de la taille des ensembles de $X^{'}$ et de $X^{''}$. Par exemple, pour $C(X^{'}, X^{''}) = 1$, il serait possible qu'une solution de $X^{'}$ domine toutes les solutions de X''.

6.5.2 L'espacement

La métrique d'espacement est définie ci-dessous [Tsou *et al.*, 2007]. Soit $X^{'} \subseteq X$ un ensemble de solutions, et une fonction d'évaluation de la distance. La fonction S évalue l'espacement des solutions de $X^{'}$.

$$S = \sqrt{\frac{1}{|X'|} \sum_{i=1}^{|X'|} (d_i - \bar{d})^2} \tag{6.9}$$

où $d_i = \min\limits_{i,j \in X^{'}, \wedge j \neq i} \sum_{k=1}^{K} \left| f_k^i - f_k^j \right|$, f_k^i est la $k^{\text{ième}}$ adaptation de la solution i et $\bar{d}$ est la valeur moyenne de la distance calculée.

$$\bar{d} = \sum_{i=1}^{|X'|} \frac{d_i}{|X'|} \tag{6.10}$$

Il est à noter que cette mesure de distance est différente de la distance minimale Euclidienne. L'espacement mesure l'écart type des valeurs différentes de d_i. Une faible valeur d'espacement indique une répartition plus uniforme des solutions sur le front produit par un algorithme. L'espacement est une indication de la répartition des solutions. Par contre, elle ne fournit pas d'indication quant à la qualité de ces solutions par rapport au front Pareto optimal. Un algorithme qui obtient une valeur d'espacement plus faible est meilleur.

6.5.3 Distance moyenne

Cette métrique calcule une distance moyenne des solutions de X' à au front de Pareto P^* [Deb, 2008].

$$D = \frac{\left(\sum_{i=1}^{|X'|} d_i^p\right)^{1/p}}{|X'|} \tag{6.11}$$

Pour $p = 2$, le paramètre d_i est la distance Euclidienne, dans l'espace objectif, entre la solution $i \in X'$ et le membre le plus près de P^*.

6.6 Résultats des essais

Les deux algorithmes présentés aux sections 6.1 et 6.2 (SPEA-MOD et PSO-MO) ont été soumis aux problèmes de la section 6.4. Les conditions de ces tests, présentées au tableau 6.1, sont similaires à celles employées dans d'autres études, et ce dans le but de pouvoir comparer les résultats de SPEA-MOD et PSO-MO à ces études [E. Zitzler et Thiele, 2000; Hu *et al.*, 2001]. Les paramètres du tableau qui ont des valeurs spécifiées dans ces études sont indiqués par †. Les deux techniques présentées ont été comparées avec les tests de Zitzler (ZDT1-ZDT6), BNH et OSY, alors qu'uniquement les tests de Zitzler ont été employés dans les études mentionnées.

6.6.1 Analyse graphique

Pour chaque problème, 30 résolutions ont été faites et toutes les solutions non dominées des archives sont montrées dans un graphique pour chaque technique. Les figures 6.10

Tableau 6.1 Paramètres de résolution des algorithmes.

SPEA-MOD	
†Nombre d'essais	: 30
†Nombre de générations	: 250
†Taille de la population	: 100
†Taille maximale de l'archive	: 200
Taux de croisement	: 0.4
Taux de mutation	: 0.2

PSO-MO	
†Nombre d'essais	: 30
†Nombre de générations	: 250
†Taille de la population	: 100
†Taille maximale de l'archive	: 200
Taux de mutation	: 0.2

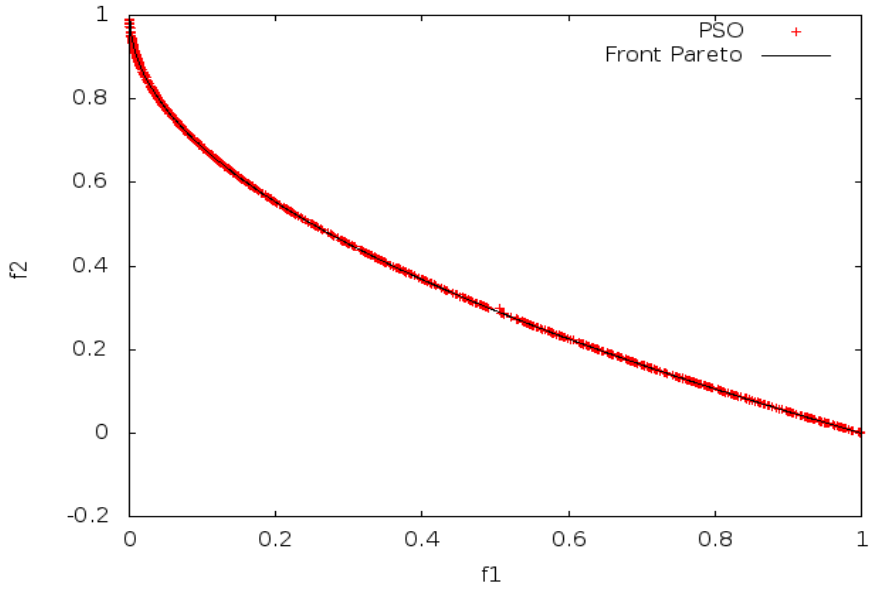
à 6.16 présentent les résultats d'optimisation des algorithmes SPEA-MOD et PSO-MO pour les problèmes ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, BNH, et OSY respectivement. À titre de comparaison le front de Pareto théorique associé à chaque problème y est également illustré.

Par inspection visuelle des résultats graphiques, les performances de SPEA-MOD et PSO-MO sont similaires. Les valeurs produites par les deux algorithmes sont représentées par des croix rouges, alors que les fronts de Pareto sont représentés par les courbes noires. On constate que les deux nouveaux algorithmes sont en mesure d'approximer les fronts car les croix sont superposées sur les courbes. De façon générale, PSO-MO produit une meilleure répartition des solutions sur le front pour tous les problèmes, sauf pour les problèmes ZDT3 et ZDT4. Par contre, PSO-MO semble incapable d'approximer un des fronts de ZDT4. Il est cependant difficile de conclure lequel de SPEA-MOD ou PSO-MO offre une meilleure performance.

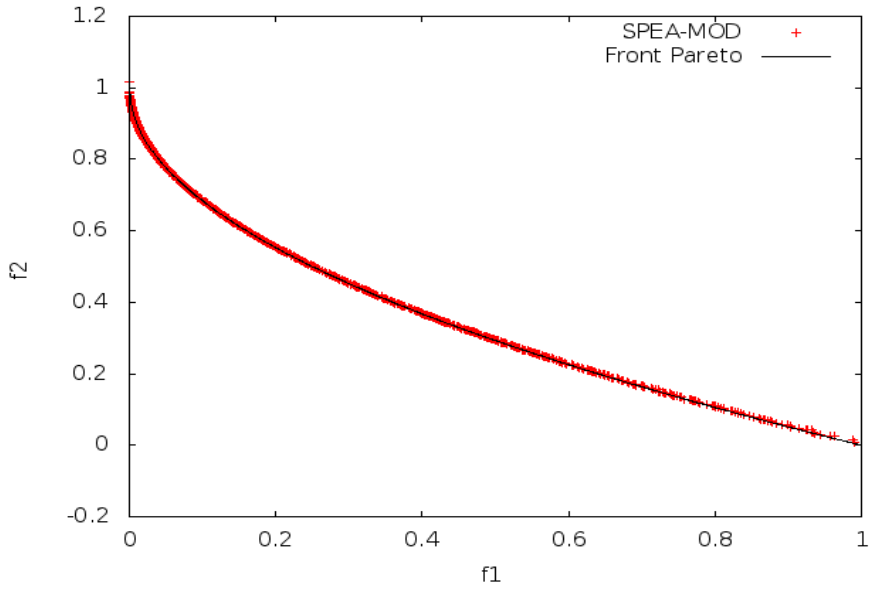
6.6.2 Analyse quantitative

Le tableau 6.2 de la page 86 présente le nombre de solutions non dominées obtenues pour les 30 résolutions consécutives aux problèmes mentionnés pour les deux algorithmes développés. Notons que tous les tableaux de résultats ont été générés automatiquement, éliminant les risques d'erreurs causés par la transcription.

Dans tous les problèmes, excepté ZDT3 et ZDT4, l'algorithme PSO-MO obtient un plus grand nombre de solutions non dominées que l'algorithme SPEA-MOD. Ces résultats indiquent, implicitement, une meilleure répartition des solutions sur le front de Pareto approximé. Les figures 6.10 à 6.15 confirment les résultats obtenus. Hu et ses collaborateurs ont obtenu des résultats similaires en comparant l'algorithme SPEA original à leurs algorithmes de colonies de particules multiobjectifs, DNPSO et m-DNPSO [Hu *et al.*, 2001].

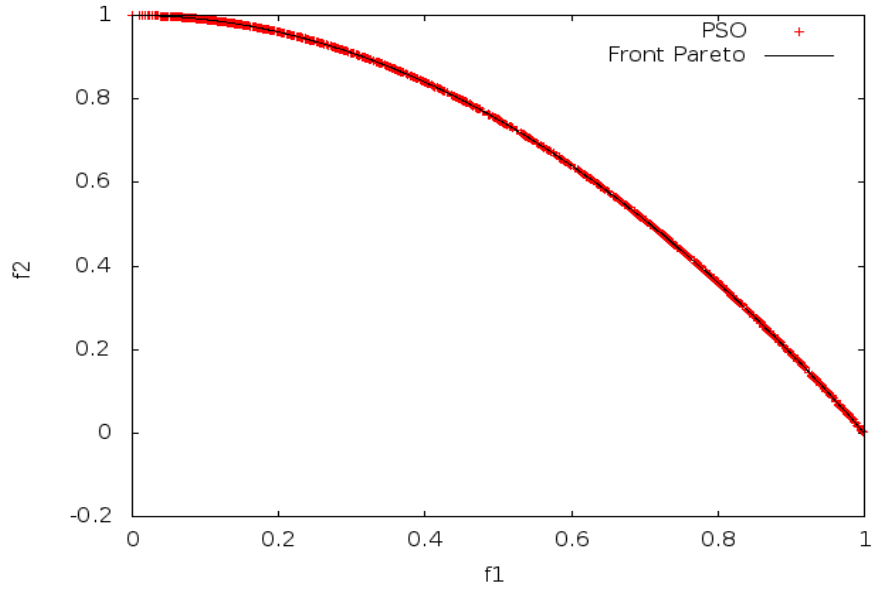


(a) PSO-MO

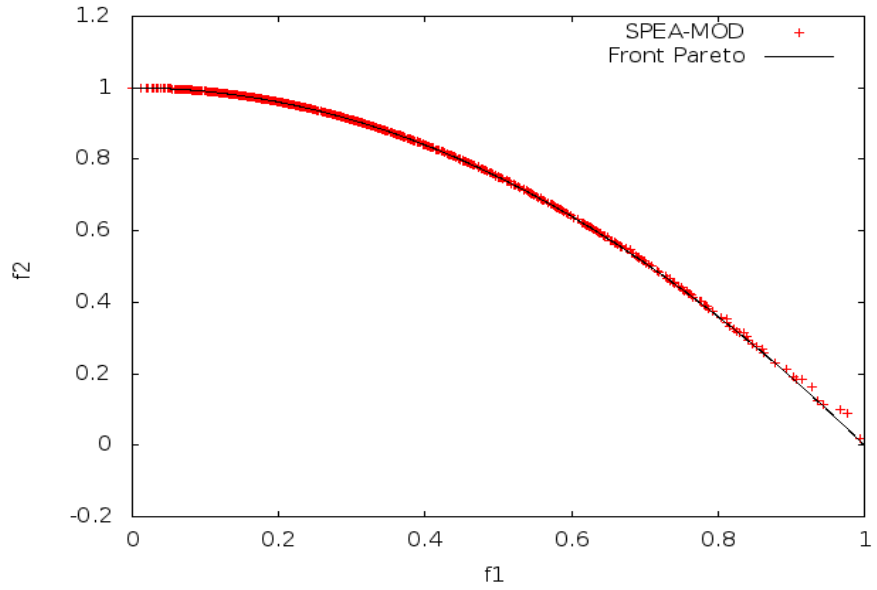


(b) SPEA-MOD

Figure 6.10 ZDT1, (Population = 100, Archive = 200)

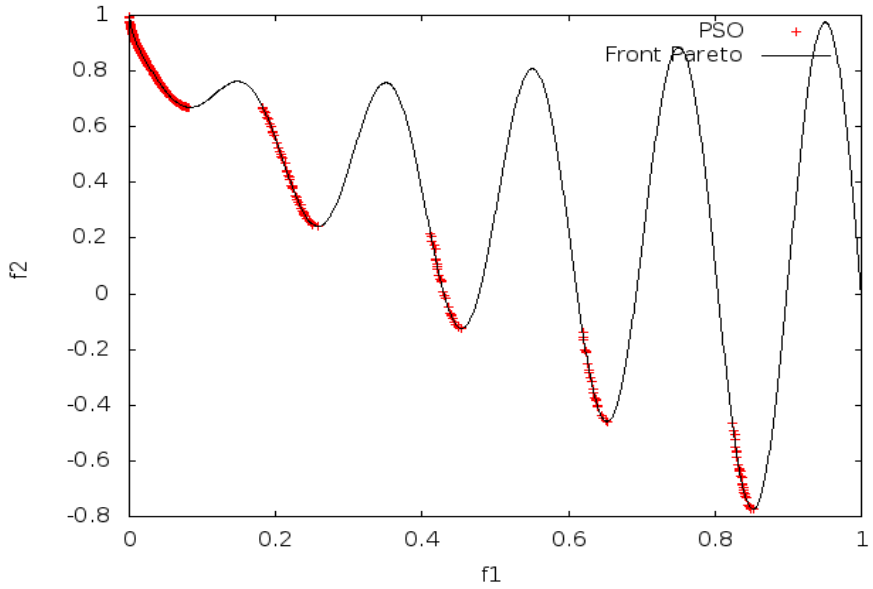


(a) PSO-MO

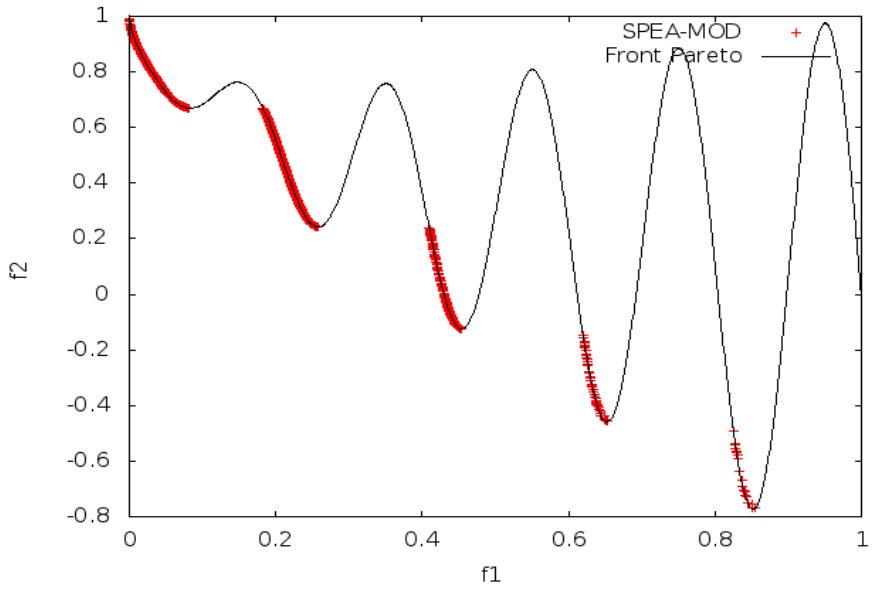


(b) SPEA-MOD

Figure 6.11 ZDT2, (Population = 100, Archive = 200)

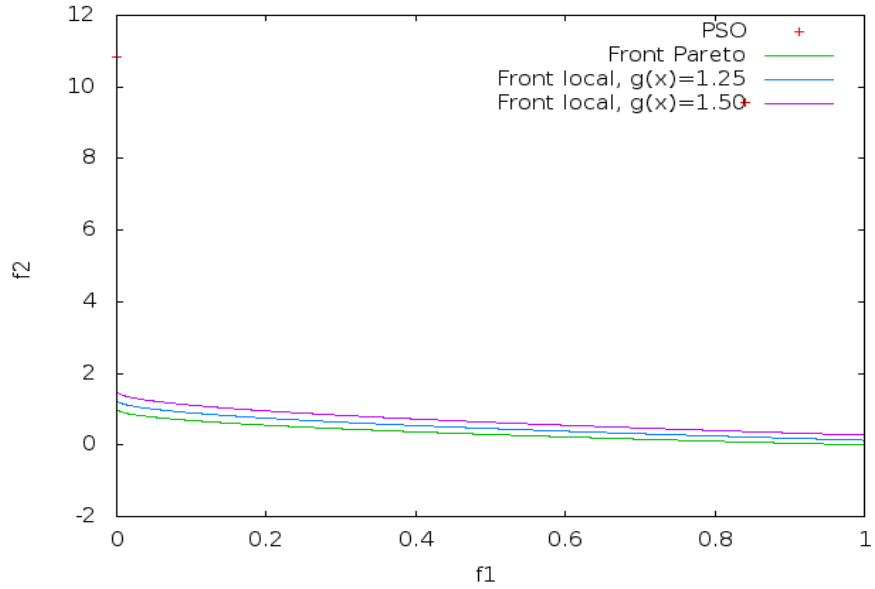


(a) PSO-MO

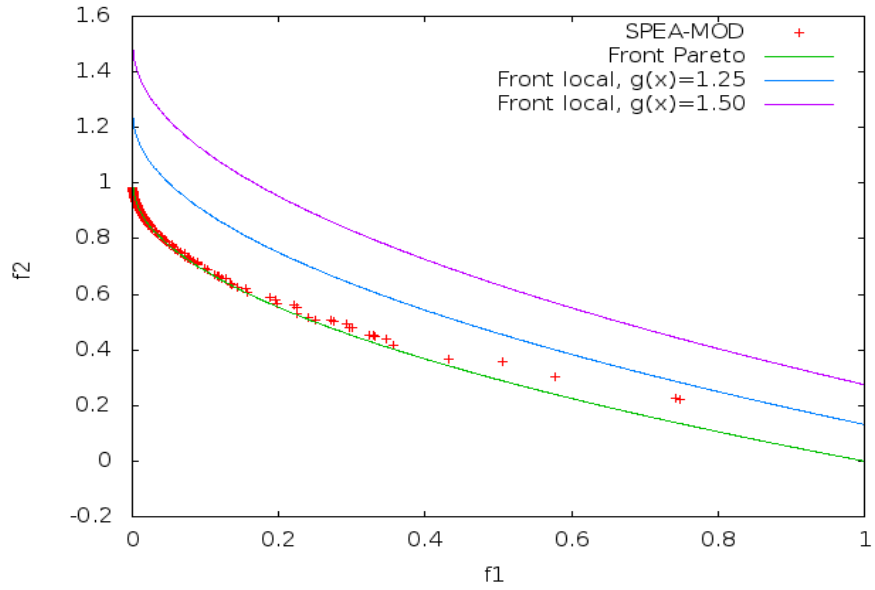


(b) SPEA-MOD

Figure 6.12 ZDT3, (Population = 100, Archive = 200)

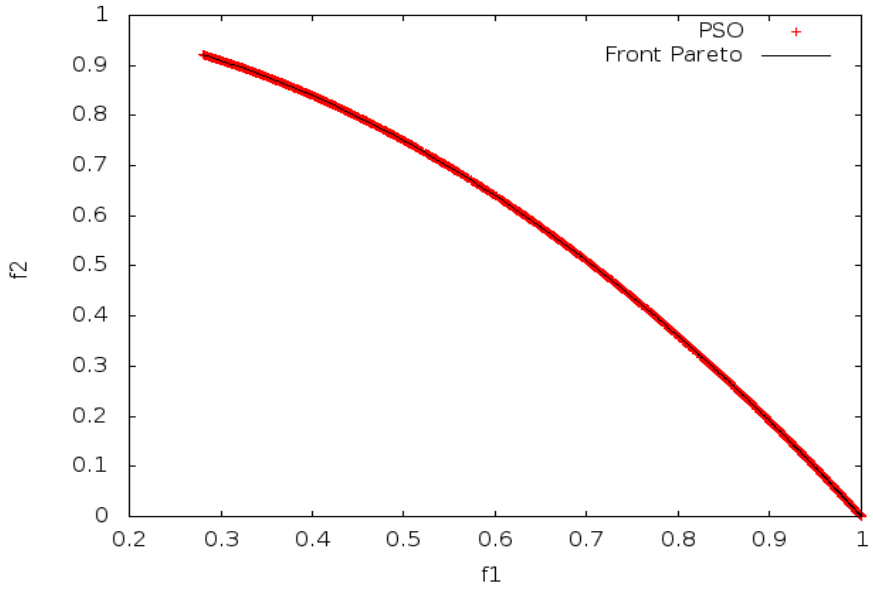


(a) PSO-MO

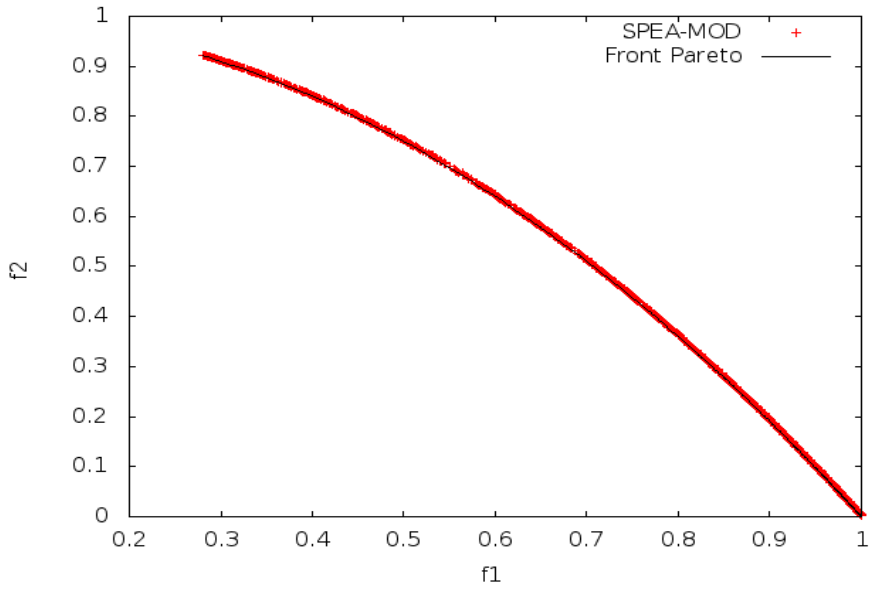


(b) SPEA-MOD

Figure 6.13 ZDT4, (Population = 100, Archive = 200)

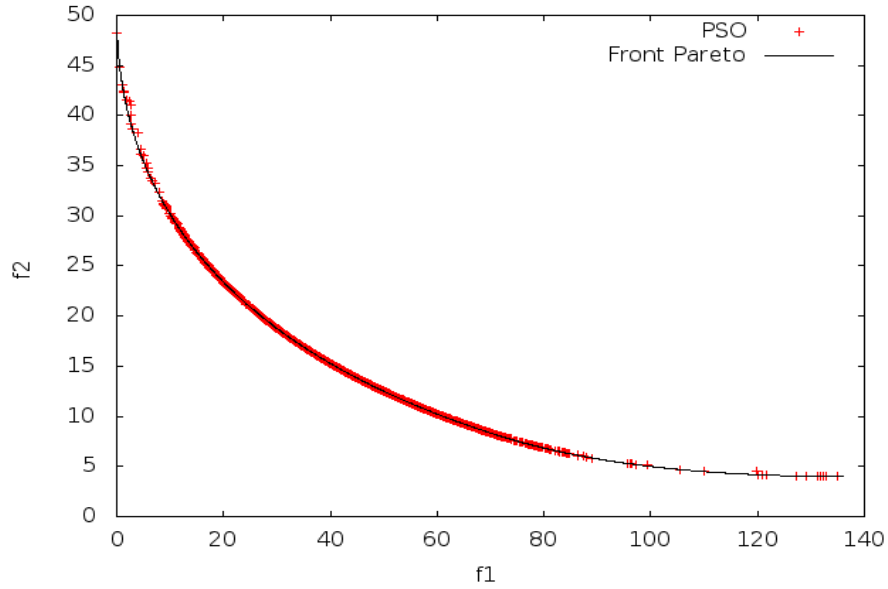


(a) PSO-MO

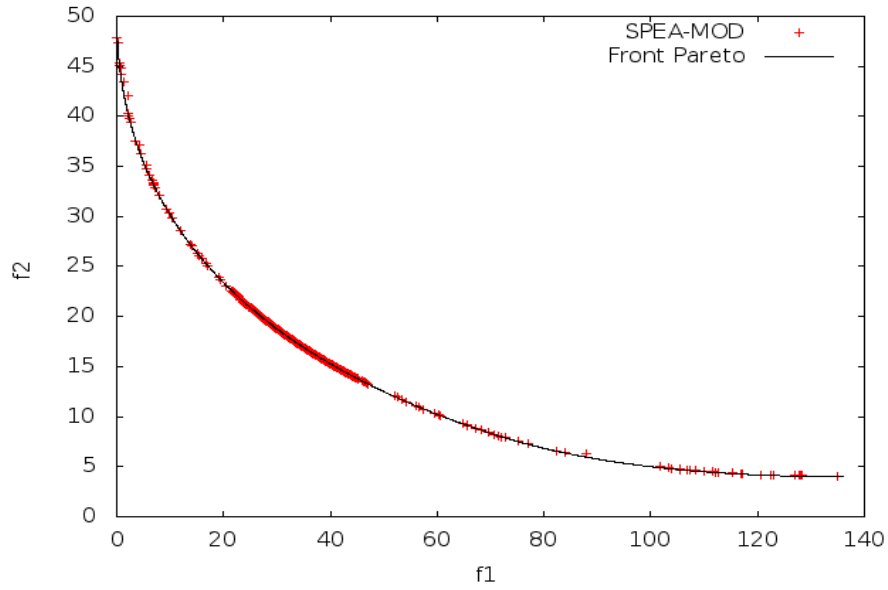


(b) SPEA-MOD

Figure 6.14 ZDT6, (Population = 100, Archive = 200)

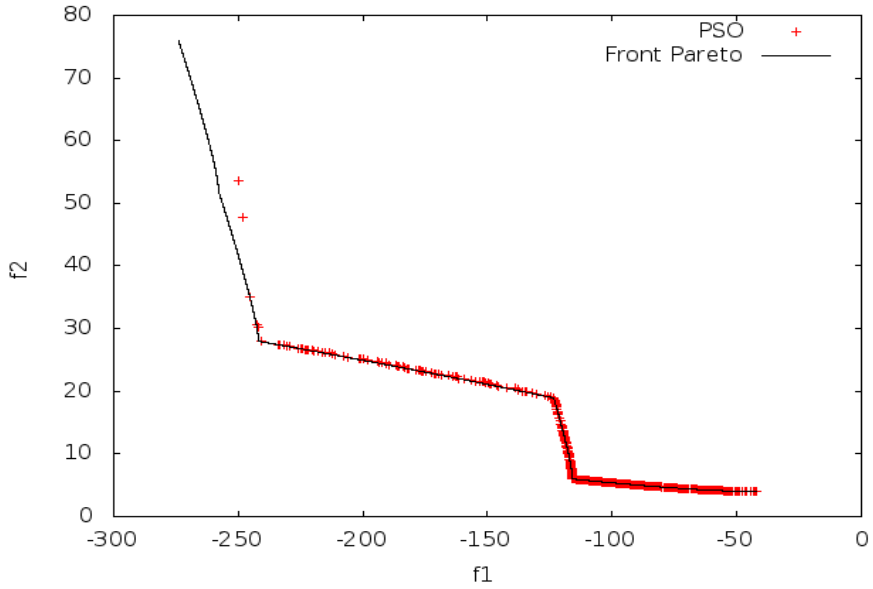


(a) PSO-MO

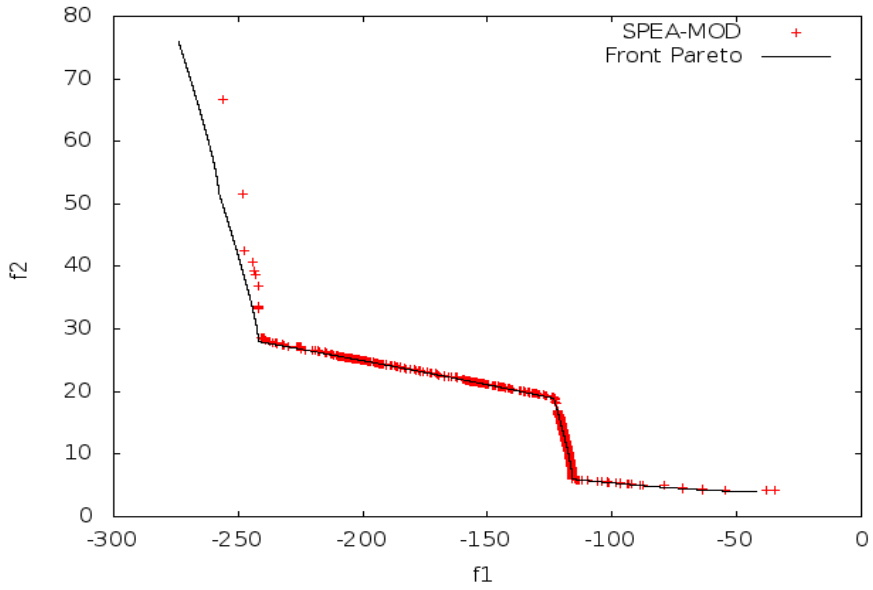


(b) SPEA-MOD

Figure 6.15 BNH, (Population = 100, Archive = 200)



(a) PSO-MO



(b) SPEA-MOD

Figure 6.16 OSY (Population = 100, Archive = 200)

Par contre, comme dans le cas de PSO-MO, l'algorithme de colonies de particules de [Hu *et al.*, 2001] a également été incapable d'approximer un des fronts du problème ZDT4.

Le tableau 6.3 quant à lui, présente les résultats aux tests ZDT1 à ZDT6 pour l'algorithme génétique SPEA et les algorithmes de colonies de particules DNPSO et m-DNPSO [Hu *et al.*, 2001]. SPEA-MOD et PSO-MO obtiennent beaucoup plus de solutions non dominées que SPEA. La démarcation est plus nuancée dans la comparaison de ces deux algorithmes à DNPSO et m-DNPSO. SPEA-MOD et SPEA sont les seuls algorithmes a obtenir plusieurs solutions non dominées pour le problème ZDT4.

Le nombre de solutions non dominées obtenues est une indication de la performance d'un algorithme par rapport à un autre. Par contre, il est tout à fait possible qu'un algorithme obtienne moins de solutions non dominées qu'un autre, mais que ces dernières dominent celles du second algorithme.

Tableau 6.2 Nombre de solutions non dominées. (Archive = 200)

Problème	SPEA-MOD	PSO-MO
ZDT1	784	812
ZDT2	618	925
ZDT3	774	384
ZDT4	141	2
ZDT6	861	3704
BNH	5123	2406
OSY	693	773

Tableau 6.3 Nombre de solutions non dominées de [Hu *et al.*, 2001].

Problème	SPEA	DNPSO	m-DNPSO
ZDT1	204	925	1293
ZDT2	112	424	515
ZDT3	202	436	825
ZDT4	156	6	2
ZDT6	22	1847	5998

Le tableau 6.4 présente les métriques d'*espacement* et de *distance moyenne*. Notons que les valeurs obtenues pour les tests ZDT1 à ZDT6 sont comprises en $[0, 1]$, car les fonctions objectives sur le front sont également comprises entre $[0, 1]$, ce qui n'est pas le cas pour les tests BNH et OSY. Les valeurs d'espacement du test ZDT4 pour l'algorithme PSO-MO sont très élevées, ce qui est également confirmé par la figure 6.10 de la page 79.

Les espacements des deux algorithmes sont similaires. Ce résultat semble en contradiction avec ce que l'on observe sur les figures 6.10 à 6.16. En effet, sur ces figures PSO-MO semble

Tableau 6.4 Résultats des tests, pour SPEA-MOD (S) et PSO-MO (P) Espacement (S), Distance moyenne (D).

Problème	S(S)	S(P)	D(S)	D(P)
ZDT1	0.393324	0.492148	0.001383	0.000588
ZDT2	0.419055	0.547771	0.000786	0.000346
ZDT3	0.347718	0.376865	0.001859	0.001436
ZDT4	0.201312	7.890074	0.006120	9.666297
ZDT6	0.744420	0.814437	0.003120	0.001570
BNH	33.165123	44.224417	0.026206	0.030108
OSY	156.984630	123.701895	0.198007	0.045763

produire une répartition plus égale des solutions. Ce résultat indique que PSO-MO produit des régions ayant une concentration élevée de solutions. La distance moyenne obtenue par PSO-MO est inférieure à celle obtenue par SPEA-MOD, sauf pour le problème ZDT4.

Il est également intéressant de constater que SPEA-MOD est plus performant que PSO-MO sur les problèmes difficiles, ZDT4 et BNH. Les résultats au test ZDT3 sont très similaires, mais SPEA-MOD est légèrement supérieur. Les métriques indiquent une meilleure performance de PSO-MO au problème OSY. Ce résultat est dû au fait que SPEA-MOD produit quelques solutions éloignées du front, tel qu'illustré à la figure 6.16 de la page 85.

Le tableau 6.5 présente les valeurs de la métrique C, entre les algorithmes SPEA-MOD et PSO-MO, aux problèmes présentés à la section 6.4. Selon ce tableau, PSO-MO produit des solutions de meilleure qualité que SPEA-MOD, en ce sens que majoritairement ces solutions dominent celles de SPEA-MOD. Les solutions de PSO-MO sont par contre dominées par celle de SPEA-MOD pour les problèmes ZDT4 et BNH.

Tableau 6.5 Métrique C de SPEA-MOD et PSO-MO.

Problème	C(SPEA-MOD, PSO-MO)	C(PSO-MO, SPEA-MOD)
ZDT1	0.012315	0.642857
ZDT2	0.001081	0.618123
ZDT3	0.052083	0.237726
ZDT4	1.000000	0.000000
ZDT6	0.000000	0.998839
BNH	0.264339	0.018153
OSY	0.058215	0.636364

Les distances moyennes ainsi que la métrique C semble indiquer que l'algorithme PSO-MO est plus performant. Encore une fois SPEA-MOD offre de meilleures solutions que PSO-MO aux problèmes difficiles ZDT4 et BNH. Malgré tous les résultats obtenus, il est difficile de conclure lequel de SPEA-MOD ou PSO-MO est supérieur. Sur certains aspects

SPEA-MOD est meilleur, alors que PSO-MO est meilleur sur d'autres. Par contre, étant donné que PSO-MO a été incapable d'approximer le front de ZDT4, SPEA-MOD semble être un meilleur choix d'algorithme pour les problèmes diffciles à résoudre.

6.6.3 Les temps de calcul

Un temps de calcul réduit est un facteur important dans le contexte de cette recherche, car la solution d'une résolution de conflits aériens doit pouvoir s'obtenir le plus rapidement possible. L'avantage principal d'un algorithme PSO, évoqué dans plusieurs études, est sans aucun doute le nombre réduit d'opérations, diminuant ainsi le temps de calcul. Par contre, cet avantage a été évoqué pour des algorithmes mono-objectifs.

Le tableau 6.6 présente les temps de calcul obtenus par SPEA-MOD et PSO-MO pour 30 résolutions consécutives des problèmes tests mentionnés.

De façon générale, SPEA-MOD nécessite moins de temps de calcul. Cette différence est expliquée par la taille de l'archive.

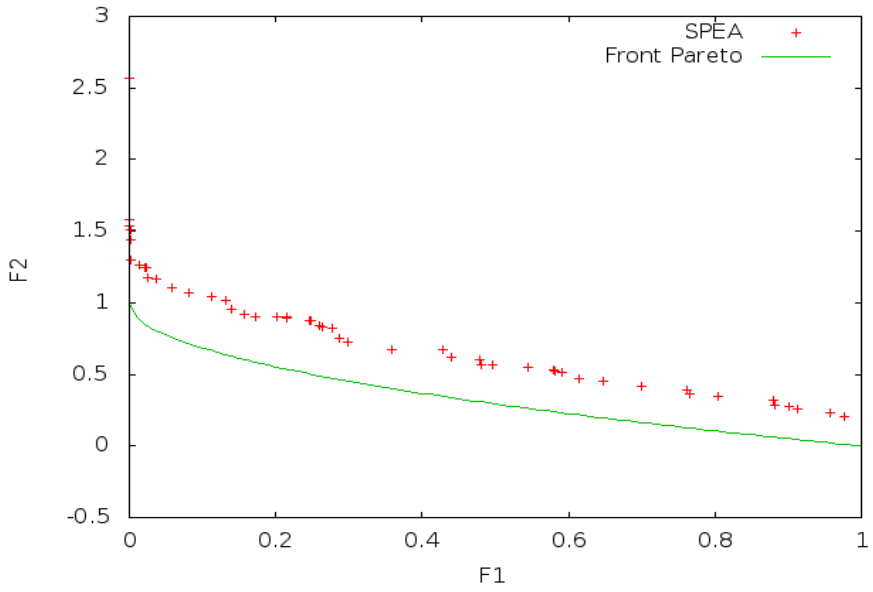
Tableau 6.6 Temps de calcul pour 30 essais (sec).

Problème	SPEA-MOD	PSO-MO
ZDT1	61	49
ZDT2	30	45
ZDT3	77	43
ZDT4	11	56
ZDT6	48	324
BNH	145	259
OSY	113	51

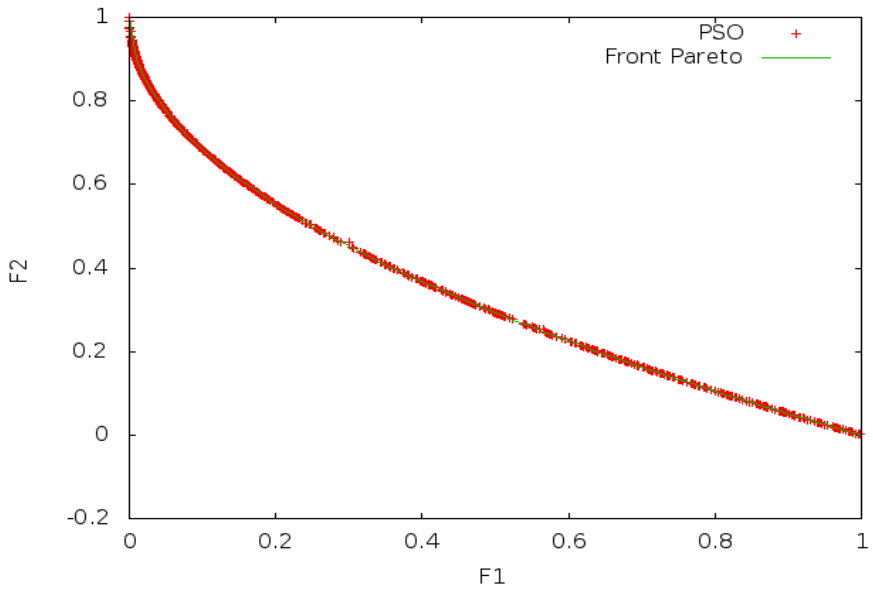
6.6.4 Mutation uniforme et non uniforme

La figure 6.17 illustre le résultat du test ZDT1 en utilisant une mutation uniforme au lieu d'une mutation non uniforme pour SPEA-MOD et PSO-MO. Rappelons que l'effet exploratoire de la mutation est diminué avec le nombre de générations par une mutation non uniforme, alors que l'effet reste inchangé pour une mutation uniforme. On emploie la mutation non uniforme pour favoriser une stabilité de l'algorithme évolutionnaire, c'est-à-dire pour favoriser la convergence en réduisant la recherche aléatoire. Cette réduction de recherche aléatoire se fait en fonction du nombre de générations.

Tel qu'illustré à la figure 6.17, la mutation uniforme détériore les performances de PSO-MO (de façon minime) et celles de SPEA-MOD.



(a) SPEA-MOD



(b) PSO-MO

Figure 6.17 ZDT1 avec mutation uniforme.

6.7 SPEA vs SPEA-MOD

Tel que mentionné précédemment, les algorithmes SPEA et SPEA-MOD diffèrent par la méthode de sélection des individus et par la façon d'évaluer les individus de la population et de l'archive. SPEA emploie une sélection en tournoi, mais les détails de la sélection sont laissés au libre choix de l'implémentation [Zitzler et Thiele, 1998]. L'algorithme SPEA

original a été implanté en utilisant une sélection en tournoi de Pareto [Horn *et al.*, 1994]. Un accroissement du temps de calcul a été remarqué en utilisant une sélection en tournoi comparativement à une sélection basée sur le rang et sans une augmentation remarquée de la performance. De plus, une sélection en tournoi requiert deux paramètres supplémentaires, soit le rayon de la niche et le pourcentage d'individus de l'ensemble de reproduction participant au tournoi augmentant la complexité de l'algorithme.

L'algorithme SPEA original a été implanté, à l'exception de la méthode de sélection de tournoi qui a été remplacée par le même que celui de SPEA-MOD (sélection basée sur le rang), dans le but de le comparer à SPEA-MOD aux problèmes types de la section 6.4. Le seul élément différenciateur entre les deux algorithmes durant la comparaison était la façon d'évaluer les individus. L'annexe A présente les résultats graphiques de SPEA à ces problèmes.

À l'exception du problème BNH et ZDT4, tous les résultats de SPEA-MOD sont meilleurs que ceux de SPEA. On remarque que les solutions de SPEA-MOD sont plus près et mieux dispersées sur les fronts de Pareto.

6.8 Sommaire

Ce chapitre a présenté deux nouveaux algorithmes évolutionnaires multiobjectifs, SPEA-MOD et PSO-MO, pouvant résoudre des problèmes multiobjectifs contraints, tels que ceux de résolution de conflits. SPEA-MOD est une version améliorée de l'algorithme génétique multiobjectif SPEA, alors que PSO-MO est un algorithme de colonies de particules multiobjectif ayant des similarités à l'algorithme SPEA-MOD. Les deux algorithmes ont été confrontés à six problèmes multiobjectifs types que l'on retrouve dans la littérature.

La mesure de performance de ces deux algorithmes s'est faite de façon visuelle par une analyse graphique et par une analyse numérique. Selon l'analyse visuelle des graphiques des page 79 à 85, l'algorithme PSO-MO offre de façon générale une meilleure répartition des solutions sur le front de Pareto. Par contre PSO-MO présente une piètre performance, comparativement à SPEA-MOD, lorsque confronté au problème ZDT4.

L'analyse numérique s'est faite en utilisant des métriques, soit le nombre de solutions non dominées, l'espacement, la distance moyenne et la métrique C. Selon les résultats des trois premières métriques présentés aux tableaux 6.3 et 6.4 (page 87) il est difficile de conclure sur la supériorité de SPEA-MOD ou PSO-MO. Sur certains points SPEA-MOD est supérieur, alors que sur d'autres PSO-MO est mieux. Par contre, selon les résultats de

la métrique C présentés au tableau 6.5 (page 87), les solutions PSO-MO dominent celles de SPEA-MOD dans la majorité des problèmes.

Au chapitre 8, les deux algorithmes seront soumis à des problèmes de résolution de conflits de la phase de vol en route. Il sera donc possible de conclure lequel de ces deux algorithmes est mieux adapté au problème de la résolution de conflits.

CHAPITRE 7

RÉSOLUTION DE CONFLITS, PHASE EN ROUTE

Ce chapitre présente l'algorithme conçu pour la résolution de conflits dans la phase en route. Cet algorithme sera également employé dans le séquençage du trafic dans la phase approche, tel que décrit au chapitre 9. Malgré le fait que cette étude s'intéresse à la modélisation de tâches de contrôleurs aériens, les facteurs humains, tels que la fatigue, le stress et le non respect des directives des contrôleurs ne sont pas abordés. Cette étude s'est attardée à modéliser ces comportements dans des situations idéales. Les système d'évitement de collisions, tel que le TCAS, ne sont pas traités, car ce ne sont pas des systèmes propres à l'ATC. De plus, une approche centralisée, contrairement à une méthode décentralisée, est employée, car c'est ce qui représente le mieux le système ATC actuel.

Rappelons également que les techniques d'optimisation classiques, comme la méthode du gradient, sont pratiquement inutiles pour ce genre de problème. La revue de la littérature de la section 2.3 a démontré que la résolution de conflits est un problème complexe qui peut être résolu par un algorithme d'optimisation global, tels que les algorithmes génétiques. C'est cette technique qui a été choisie dans cette recherche.

L'ATC doit fournir un service de contrôle aérien sécuritaire et efficace. La réglementation de l'ATC prescrit les paramètres à respecter, tels que les minimums de séparation, par les contrôleurs aériens. De façon générale, elle ne précise pas la technique préférentielle pour une situation donnée, ce choix étant laissé aux contrôleurs. Dans le cadre de ce projet de recherche, l'auteur a consulté plusieurs contrôleurs aériens d'expérience. Les manoeuvres d'évitement et les critères d'évaluation en ont été fortement inspirés. Pour certaines raisons, beaucoup de contrôleurs préfèrent résoudre les conflits sans effectuer de changements d'altitude, la résolution se fait donc dans un plan. D'autres préfèrent affecter fortement la trajectoire d'un avion plutôt que d'en affecter plusieurs avec des variations plus faibles. C'est cette liberté d'action qui a suscité l'intérêt pour une optimisation multi-objectif au lieu de mono-objectif. Dans la suite du présent document, le terme optimisation est employé pour désigner une minimisation.

La figure 7.1, qui est une reprise de la figure 2.1, illustre où se situent les techniques proposées dans cette recherche selon les critères de Kuchar et Yang [Kuchar et Yang, 1997, 2000]. Les cercles indiquent les critères employés dans cette étude.

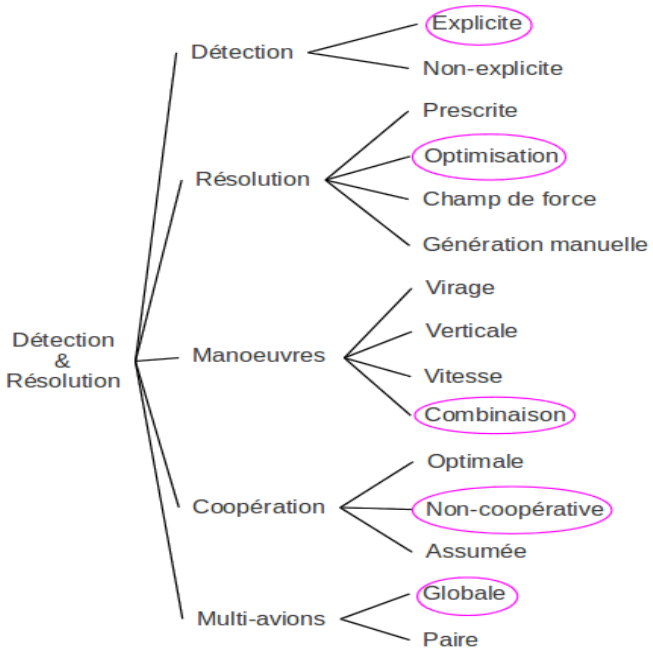


Figure 7.1 Critères de classification des techniques proposées dans cette étude.

Le présent chapitre décrit les différents algorithmes conçus pour la résolution de conflits d'avions de la phase en route. Plusieurs concepts et algorithmes définis sont également utilisés dans l'opération de séquençage présentée au chapitre 9. Quant au chapitre 8, il présente les résultats numériques de plusieurs simulations de résolution de conflits de la phase en route.

7.1 Processus de détection et de résolution de conflits

Tel que présenté au chapitre 2, la détection de conflits est un sujet complexe qui n'est pas abordé dans le cadre de ce projet de recherche. Rappelons que la détection de conflits consiste à déterminer les avions susceptibles d'être en conflit et de fournir cette information à la résolution de conflits. La résolution de conflits quant à elle consiste à fournir des manoeuvres d'évitement aux différents avions afin de respecter la séparation minimale. Ainsi, dans la réalité ce processus se fait en deux étape, soit la détection de conflits et par la suite si nécessaire, une résolution de conflits.

Dans tous les problèmes de résolution présentés, les avions apparaissant dans les différentes situations sont considérés comme étant en conflit, c'est-à-dire que l'on prend comme hypothèse que l'étape de détection de conflits a préalablement été effectuée. Le processus d'optimisation décidera des manoeuvres à ordonner aux avions et aucune instruction inutile ne sera fournie. Par conséquent, seulement les avions étant en situation réellement conflictuelle recevront des instructions d'évitement. À l'échelle du problème, c'est comme si le processus d'optimisation fait la détection et la résolution de conflits. Cette approche est acceptable en contexte de recherche, mais devrait probablement être revue dans un contexte d'application réelle. L'inconvénient de cette approche est le temps de calcul accru.

Une résolution de conflits basée sur une optimisation globale est utilisée. Ainsi, les différents conflits impliquant plusieurs avions sont traités simultanément. Cette approche a l'avantage qu'une solution trouvée pour un conflit est également valide pour tous les conflits traités. Donc, il n'est pas nécessaire de valider qu'une solution pour un conflit n'engendre pas de nouveaux conflits. Une résolution en paire d'avions aurait ce désavantage [Granger, 2002].

La figure 7.2 illustre les composantes majeures d'un système de détection et résolution de conflits. Les deux premières composantes, *Estimation des états* modélise l'obtention des positions des avions par des sources de surveillance comme des radars, alors que la composante *Modèle dynamique* simule les trajectoires d'avions. Ces trajectoires sont ensuite employées par la composante *Évaluation objectifs et contraintes* du problème d'optimisation. Finalement, les instructions ATC sont produites par la composante *Résolution de conflits* qui sont fournies au contrôleur aérien pour que ce dernier transmette les actions appropriées aux pilotes.

7.2 Hypothèses de base

Les techniques de résolution de conflits de ce chapitre et ceux du séquençage en arrivée du chapitre 9 prennent comme hypothèse que tous les aéronefs évoluent selon les règles du vol aux instruments (IFR). Cette hypothèse est tout à fait justifiée par le fait que la majorité du trafic aérien est IFR. Les avions IFR sont également soumis à une réglementation beaucoup plus stricte que les avions en vol à vue (VFR). Par exemple, tout changement de route doit nécessairement être coordonné par l'unité de contrôle aérien (ATC) appropriée. Ceci permet donc à l'ATC d'être en mesure de prédire les déplacements des avions pour ainsi y prévoir les conflits potentiels de trajectoires. Ceci justifie l'utilisation d'une méthode centralisée de résolution de conflits.

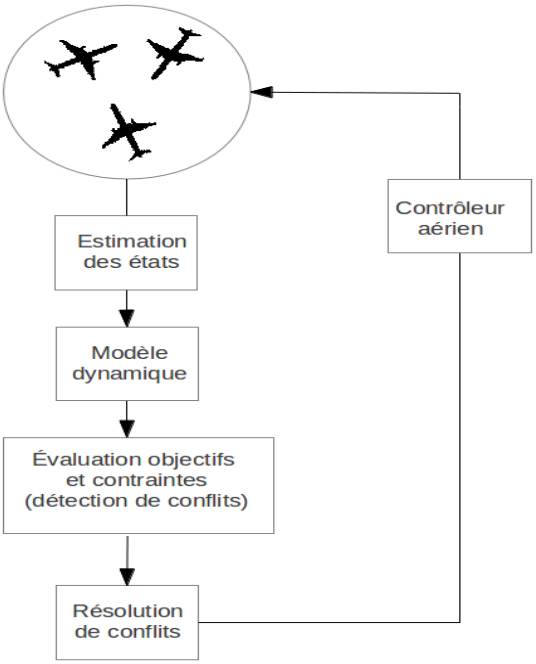


Figure 7.2 Composantes de la détection et de la résolution de conflits.

La présente étude ne s'est pas concentrée sur les problèmes d'incertitudes de positionnement puisque le but de l'algorithme est de rendre plus réaliste un environnement ATC virtuel. Ainsi, dans le monde virtuel utilisé, il n'y a pas d'incertitude sur la position et la cinématique des avions modélisés. C'est une hypothèse des plus réalistes, car rappelons que le but de ce projet de recherche est de fournir un environnement de trafic aérien réaliste pour des simulateurs de vol. De plus, tel que mentionné au chapitre 2, il est possible de croire qu'un jour cette hypothèse sera également valable pour un environnement opérationnel.

7.3 Modèle cinématique

La dynamique de vol des avions employée dans les simulations est décrite par un modèle simplifié, contrairement à ce que l'on retrouve dans la littérature. Plusieurs études ont employé des équations dynamiques, plus particulièrement des équations de point matériel, pour modéliser la dynamique de vol des avions. Ces équations nécessitent plusieurs paramètres, tels que des coefficients aérodynamiques et les forces de poussées des moteurs. Ces données sont fournies par le projet BADA (*Base of Aircraft Data*) de l'Eurocontrol [Eurocontrol, 2000]. Les projets de recherche de CTAS et de l'ENAC l'utilisent dans leurs algorithmes de génération de trajectoires. Malheureusement, la licence de BADA ne permet pas son utilisation dans une application commerciale. De plus, il est difficile d'en

justifier l'utilisation pour une modélisation de trafic aérien, comme celui de la présente recherche.

Un modèle cinématique du premier ordre dans le plan cartésien, décrit par les équations 7.1 à 7.7, est utilisé pour la modélisation du déplacement des avions. Les changements de vitesse et de directions se produisent de façon instantanée, alors que les changements d'altitude se produisent de façon linéaire et à un taux de montée, symbolisé par ROC (*Rate of Climb*), prédéfini à 2000 pieds/minute [1]. Cette hypothèse est réaliste lorsque la détection des conflits se fait au moins quelques minutes avant la perte de séparation [Bach *et al.*, 2007; Durand, 1996; Erzberger, 1995].

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{i+1} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{i} + \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}_{i} \Delta T \tag{7.1}$$

$$\Delta Z = Alt - Z \tag{7.2}$$

$$\gamma = \sin^{-1}(ROC/V) \quad (\gamma = 0 \text{ si } \Delta Z = 0) \tag{7.3}$$

$$V_{xy} = V \cdot \cos(\gamma) \tag{7.4}$$

$$V_x = V_{xy} \cdot \sin(\psi) \tag{7.5}$$

$$V_y = V_{xy} \cdot \cos(\psi) \tag{7.6}$$

$$V_z = ROC \tag{7.7}$$

Dans l'équation 7.1, $\begin{bmatrix} X & Y & Z \end{bmatrix}_{i+1}^{T}$ est le vecteur de position de l'avion au temps $i+1$ qui est obtenu par le vecteur de position au temps i auquel le produit du vecteur vitesse et du pas de calcul ΔT. Le terme γ symbolise l'angle de la trajectoire dans le plan vertical, mesuré entre le vecteur de vitesse et l'horizon. Les termes V, ψ et Alt sont des variables de contrôle dénotant la vitesse, la direction dans le plan horizontal et l'altitude désirée respectivement.

De par la simplicité du modèle, les perturbations de l'environnement tels que le vent et la pression ambiante ne sont pas considérés. Ces mêmes hypothèses ont également été utilisées dans d'autres projets de recherche [Durand, 1996; Inselberg, 2001]. Une implémentation plus rapide et plus simple sont les deux raisons pour lesquelles ce modèle est employé.

1. En aéronautique le taux de montée est généralement exprimé en pied par minute (pied/min).

7.4 Manoeuvres de résolution

Cette section décrit les manoeuvres utilisées dans la résolution de conflits de la phase en route. Bien qu'elles soient similaires, deux types ont été envisagés, soit les manoeuvres dans le plan horizontal et celle dans le plan horizontal et dans le plan vertical. La première manoeuvre est constituée de changements de direction ou de vitesse, alors que la seconde est constituée d'une combinaison d'instructions de changements de direction, de vitesse et d'altitude. Notons que ces manoeuvres sont semblables à celles proposées par Durand [Durand, 1996]. Une manoeuvre permettant de résoudre des problèmes de séquençage d'avions en arrivée et des problèmes de résolutions de conflits en route est présentée au chapitre 9.

Un requis implicite de la résolution de conflits est que chaque manoeuvre d'évitement respecte les critères de performance des avions en jeu. Pour ce faire, des bornes sur certains paramètres de simulation et de contrôle sont imposées. Par exemple, les changements de vitesse doivent être compris dans une plage de valeurs admissibles. Par exemple, les changements de direction assignés par les contrôleurs sont généralement des multiples de 10 degrés, alors que les changements d'altitude sont généralement par des multiples de 500 pieds. Les plages de valeurs admissibles employées sont spécifiées dans les exemples de résolution présentés dans le chapitre 8. Par simplicité, les avions ont les mêmes performances dans toutes les simulations présentées.

La figure 7.3 illustre la modélisation des étapes d'évitement employée dans la résolution de conflits associée à un avion donné. Tout d'abord, l'avion est dévié de sa trajectoire de référence (TR), ou route actuelle, et exécute une manoeuvre de résolution définie pour le problème en question. L'étape de l'exécution de la manoeuvre consiste en une déviation de la trajectoire initiale, alors que celle de la conclusion de la manoeuvre consiste à se rediriger vers la trajectoire de référence. Finalement, l'avion dévié revient sur sa trajectoire de référence. Ce modèle simple de résolution est non seulement valide pour des trajectoires nominales rectilignes, mais également pour des manoeuvres complexes telles que celles employées dans le séquençage en approche présentées au chapitre 9.

Point tournant

La figure 7.4 montre deux manoeuvres envisagées pour la résolution de conflits dans le plan horizontal. Sur cette figure les symboles t_0, t_1, t_2 et α représentent les paramètres de contrôle. Ils représentent respectivement : le début du virage d'éloignement, la fin du virage d'éloignement, le début du virage de retour, et l'angle de déviation par rapport à la trajectoire initiale. De son côté t_3, le temps de retour sur la trajectoire initiale, est déduit,

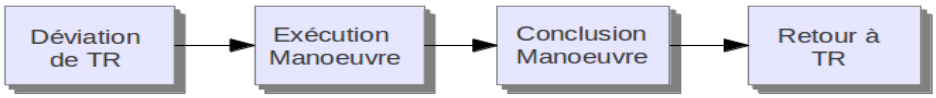
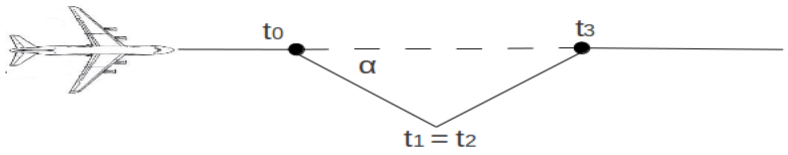
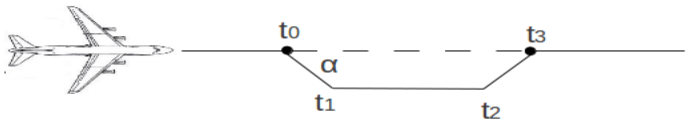


Figure 7.3 Étape d'une manoeuvre d'évitement.

car les côtés t_0t_1 et t_2t_3 sont égaux. Ces paramètres, qui seront modifiés par les opérateurs génétiques, constituent les variables du problème à optimiser.



(a) Point tournant



(b) Décalage

Figure 7.4 Manoeuvres dans le plan horizontal pour la résolution de conflits.

Dans le cadre de ce projet de recherche, seulement la manoeuvre du *point tournant* est utilisée (voir figure 7.4-a). La manoeuvre *décalage* aurait généré des résultats similaires, car la seule différence est que le temps t_2 est une variable à optimiser. Il ne faut pas voir les manoeuvres proposées comme étant la clé de la solution, elles doivent plutôt être adaptées aux problèmes de résolutions de conflits. Par exemple, les manoeuvres employées dans le séquençage en approche, présentées au chapitre 9, sont complètement différentes de celles proposées pour la résolution de la phase en route. Il est également plausible que pour un problème donné, les manoeuvres à utiliser puissent varier, dû à des facteurs environnementaux de l'espace aérien.

La manoeuvre originale du *point tournant* fut néanmoins modifiée par l'ajout d'une instruction de vitesse et d'altitude, et ce, dans le but de la rendre plus complète et de pouvoir ainsi résoudre des problèmes à densité de trafic plus élevé. Ainsi, au temps t_0 une combinaison des instructions de changement de direction (ΔCap), de changement de vitesse (ΔV) et de changement d'altitude (ΔAlt) sera donnée au lieu de donner uniquement une instruction de changement de direction. Par combinaison, il est bien entendu possible d'avoir une seule, deux, ou trois instruction(s) par avion. Le principal désavantage de cette manoeuvre est que le temps de résolution (ou de convergence) augmente, car il y a deux paramètres supplémentaires à optimiser.

Dans la réalité ATC, seulement un ensemble fini de possibilités est employé pour les paramètres ΔV, ΔCap et ΔAlt. Par exemple, il serait absurde d'émettre une instruction de virage vers le cap 243 degrés, car généralement les graduations de l'instrument de lecture de la direction à bord d'un avion sont des multiples de 5 degrés. Généralement, les contrôleurs radar n'emploient que des multiples de 10 degrés pour les instructions de virage, mais en cas de besoin, ils utilisent également des multiples de 5 degrés. De la même façon, les instructions de changement de vitesse sont généralement par multiple de 10 noeuds (kts). Par contre, les variations de vitesses acceptables dépendent des performances admissibles d'un avion. Les limites inférieure et supérieure de vitesse sont dictées par le décrochage et la vitesse à ne jamais excéder respectivement. La vitesse supérieure admissible peut également être limitée par la réglementation. Par exemple, au Canada, la limite de vitesse sous 3000 pieds AGL [2] est de 200 kts, alors qu'elle est de 250 kts sous 10000 pieds ASL [3]. Dans le même ordre d'idée, les assignations d'altitude se font majoritairement par multiple de 500'.

De plus, il est assumé que tous les avions possèdent les mêmes critères de performance et que les avions évoluent à des vitesses supérieures à 200 kts, sans toutefois dépasser les vitesses excessives. Enfin, dans le but de rendre le problème à optimiser plus réaliste, les paramètres à optimiser ne peuvent prendre qu'un ensemble fini de valeurs tel que décrit par les intervalles ci-dessous. Ces ensembles de valeurs rendent également l'optimisation plus simple, ce qui en fait un autre avantage.

$\Delta V \in [-60, 60]$ kts, par incrément de 10 kts
$\Delta Cap \in [-50, 50]$ deg, par incrément de 10 degrés
$\Delta Alt \in [-5000, 5000]$ pieds, par incrément de 500 pieds

2. *Above Ground Level* (altitude mesurée au-dessus du sol).
3. *Above See Level* (altitude mesurée par rapport au niveau moyen de la mer).

Les tableaux 7.1 et 7.2 montrent la représentation d'un individu pour une résolution horizontale et en trois dimensions. La première représentation n'est qu'un cas particulier de la seconde où $\Delta Alt = 0$. Rappelons qu'un individu est la représentation d'une solution employée par les deux algorithmes présentés au chapitre 6. Dans ces tableaux, chaque ligne représente un gène et l'ensemble des lignes constituent un chromosome ou individu. Il est à noter que le terme chromosome est souvent employé dans la littérature traditionnelle des algorithmes génétiques.

Tableau 7.1 Chromosome pour une résolution dans le plan horizontal.

Avion (i)	T_{0i}	T_{3i}	ΔCap_i	ΔV_i
1	T_{01}	T_{31}	ΔCap_1	ΔV_1
2	T_{02}	T_{32}	ΔCap_2	ΔV_2
⋮	⋮	⋮	⋮	⋮
n	T_{0n}	T_{3n}	ΔCap_n	ΔV_n

Tableau 7.2 Chromosome pour résolution en trois dimensions.

Avion (i)	T_{0i}	T_{3i}	ΔCap	ΔV	ΔAlt
1	T_{01}	$_{T31}$	ΔCap_1	ΔV_1	ΔAlt_1
2	T_{02}	T_{32}	ΔCap_2	ΔV_2	ΔAlt_2
⋮	⋮	⋮	⋮	⋮	⋮
n	T_{0n}	T_{3n}	ΔCap_n	ΔV_n	ΔAlt_n

Ainsi, dans le modèle adopté, un chromosome contient l'information des manoeuvres associées aux avions impliqués dans la résolution de conflits. Chaque ligne présente les valeurs d'optimisation pour un avion donné. Ces mêmes éléments sont représentés par un codage réel, car tel que mentionné à la section 4.2, ce type de codage est plus approprié que le codage binaire pour ce genre de problème.

7.5 Opérateurs génétiques

Deux opérateurs génétiques ont été employés, soit la mutation et le croisement pour la manoeuvre du *point tournant*. Les opérations de croisement des différents paramètres du *point tournant* sont décrites par les équations 7.8 à 7.12 pour le premier enfant et par les équations 7.13 à 7.17 pour le second enfant. Ces opérateurs sont en fait une combinaison linéaire des deux parents.

Premier enfant :

$$
\begin{aligned}
T_{0l} &= \Gamma(t,1) \cdot T0_i l + (1 - \Gamma(t,1)) \cdot T0_j l && (7.8)\\
T_{3l} &= \Gamma(t,1) \cdot T3_i l + (1 - \Gamma(t,1)) \cdot T3_j l && (7.9)\\
\Delta Cap_l &= \Gamma(t,1) \cdot \Delta Cap_{il} + (1 - \Gamma(t,1)) \cdot \Delta Cap_{jl} && (7.10)\\
\Delta V_l &= \Gamma(t,1) \cdot \Delta V_{il} + (1 - \Gamma(t,1)) \cdot \Delta V_{jl} && (7.11)\\
\Delta Alt_l &= \Gamma(t,1) \cdot \Delta Alt_{il} + (1 - \Gamma(t,1)) \cdot \Delta Alt_{jl} && (7.12)
\end{aligned}
$$

Deuxième enfant :

$$
\begin{aligned}
T_{0l} &= \Gamma(t,1) \cdot T0_j l + (1 - \Gamma(t,1)) \cdot T0_i l && (7.13)\\
T_{3l} &= \Gamma(t,1) \cdot T3_j l + (1 - \Gamma(t,1)) \cdot T3_i l && (7.14)\\
\Delta Cap_l &= \Gamma(t,1) \cdot \Delta Cap_{jl} + (1 - \Gamma(t,1)) \cdot \Delta Cap_{il} && (7.15)\\
\Delta V_l &= \Gamma(t,1) \cdot \Delta V_{jl} + (1 - \Gamma(t,1)) \cdot \Delta V_{il} && (7.16)\\
\Delta Alt_l &= \Gamma(t,1) \cdot \Delta Alt_{jl} + (1 - \Gamma(t,1)) \cdot \Delta Alt_{il} && (7.17)
\end{aligned}
$$

où $\Gamma(t,1)$ est défini par l'équation 4.2 (page 42). Un indice supplémentaire a été ajouté aux paramètres à optimiser, comparativement à ce qui est présenté aux tableaux 7.1 et 7.2, pour différencier les individus. Les indices i et j représentent respectivement la mère et le père, alors que l'indice l représente le $l^{\text{ième}}$ gène d'un individu. La différence entre le premier et le deuxième enfant se situe au niveau de la permutation des indices i et j. Les deux nouveaux individus sont égaux lorsque $\Gamma(t,1) = 0.5$

Les séquences de mutation sont détaillées par l'algorithme 8. Chaque gène est muté selon la probabilité de mutation de l'algorithme. La mutation de chacun des gènes se fait par la fonction *muter*, qui est définie par l'équation 4.1 (page 42). Notons que les paramètres mutés sont toujours compris dans les plages de valeurs admissibles du problème. La fonction *nbAléatoire* génère un nombre aléatoire à chaque fois qu'elle est employée.

7.6 Fonctions de coûts

Les trois principes fondamentaux enseignés aux contrôleurs aériens sont la *sécurité*, l'*ordre* ainsi que l'*efficacité*. Le premier est sans aucun doute le plus simple à implanter en ce qui concerne ce projet. Il s'agit simplement de ne pas compromettre la sécurité des avions

Algorithme 8 Algorithme des séquences de mutation d'un chromosome

```
1: Probabilité de mutation : mut
2: Fonction de calcul d'un nombre aléatoire ∈ [0, 1], nbAleatoire
3: Pour Tout Gène ∈ Chromosome Faire
4:     Si nbAleatoire ≥ mut Alors
5:         muter T_1
6:     Fin Si
7:     Si nbAleatoire ≥ mut Alors
8:         muter T_2
9:     Fin Si
10:    Si nbAleatoire ≥ mut Alors
11:        muter ΔCap
12:    Fin Si
13:    Si nbAleatoire ≥ mut Alors
14:        muter ΔV
15:    Fin Si
16:    Si nbAleatoire ≥ mut Alors
17:        muter ΔAlt
18:    Fin Si
19: Fin Pour
```

sous sa juridiction, ce qui ce traduit par un maintien d'une séparation minimale. Le second principe quant à lui est relié à l'ordre des services que le contrôleur doit fournir. Il doit, dans la mesure du possible, répondre aux avions dans l'ordre dans lequel il a été contacté (premier arrivé premier servi). L'ATC doit essayer de respecter cet ordre sauf si cela contrevient aux deux autres principes. Le principe de l'efficacité quant à lui consiste à fournir des instructions de façon à réduire les délais et la consommation d'essence des avions en jeux.

Cette section présente les fonctions objectives et les contraintes définies pour le processus d'optimisation. Elles sont inspirées des principes fondamentaux de l'ATC. Le choix des objectifs et des contraintes peut varier d'un problème à l'autre, selon les besoins recherchés, tel qu'illustré aux chapitres 8 et 9.

7.6.1 Objectifs d'optimisation

Cette section décrit les fonctions objectives *nombre d'instructions* et *Écart du temps prévu d'arrivée* pouvant être employées dans la résolution.

Écart du temps prévu d'arrivée

Cet objectif représente la somme totale de la différence de temps prévu d'arrivée des avions impliqués dans la résolution. L'OACI divise les coûts d'opération et d'horaire, des avions en deux catégories, soit ceux reliés à la consommation d'essence et les autres. La raison d'être de cette fonction est de prendre en compte les coûts d'opérations des avions autre que celui de l'essence. Le tableau 7.3 présente des coûts d'opération typiques pour trois types d'avions [OACI, s. d.]. Dans le cadre de cette recherche, ces coûts d'opération sont intéressants uniquement si les problèmes à résoudre comportent des avions de types différents.

Tableau 7.3 Coûts d'opérations typiques par heure de vol.

Type d'avion	Consommation d'essence (gallon US)	Autres Coûts ($US)	Coûts d'essence ($US)	Total ($US)
ATR-42	200	1006	140	1146
A320	886	1520	753	2273
A340-600	2174	2654	2196	4850

Pour un avion donné, le calcul de l'écart du temps prévu d'arrivée ΔT_{PA} est en fait la différence de temps en absolu entre le temps d'arrivée initial T_{pa} et le temps révisé d'arrivée T_{ra}, pénalisant à la fois les retards et les avances. Cette fonction objective additionne les variations prévues d'arrivée de tous les N avions (gènes) d'un individu, tel qu'illustré par l'équation 7.18.

$$\Delta T_{pa} = \sum_{i=1}^{N} |T_{pa} - T_{ra}| \tag{7.18}$$

où T_{pa} et T_{ra} représente respectivement le temps prévu d'arrivée sans résolution et le temps prévu d'arrivée avec résolution. Dans le reste du texte cette fonction objective est appelée différence dans l'ETA (*Estimated Time of Arrival*).

Les variations d'altitude, de direction et de vitesse sont les paramètres influençant ΔT_{pa}. Une façon logique de relier ces paramètres dans une fonction objective est de les relier aux coûts d'opération (CO), tel que proposé dans [Krozel et Peters, 1997]. Une fonction d'évaluation de l'accroissement de la consommation d'essence des N avions d'un individu peut être est définie de la façon suivante :

$$CO = \sum_{i=1}^{N} C_{fuel} \Delta W_{fuel} + C_{time} \Delta T$$

où C_{fuel} est le coût du carburant, ΔW_{fuel} est la quantité d'essence excédentaire requise par la manoeuvre de résolution, C_{time} est le coût opérationnel relié au temps d'un avion et finalement ΔT est le temps additionnel requis par la manoeuvre. Seul ΔT_{PA} est employé dans les analyses présentées aux chapitres subséquents, car les avions modélisés sont de même type.

Nombre d'instructions

Cette fonction objective consiste à minimiser le nombre total d'instructions en premier lieu et à minimiser le nombre d'avions affectés par ces instructions. Ainsi, selon ces deux critères, il est préférable de dévier fortement moins d'avions qu'en dévier faiblement un plus grand nombre. C'est également ce que préfèrent les contrôleurs aériens, car cela réduit leurs interactions avec les pilotes, diminuant donc les risques d'erreurs de part et d'autre.

La valeur calculée représente le nombres d'instructions pour un individu donné. Cette valeur s'évalue tel que décrit par l'algorithme 9.

Algorithme 9 Algorithme de l'objectif du nombre d'instructions d'un individu

1: Individu (chromosome) I
2: Avion (gène) A
3: Nombre d'instructions totals, nbInsTotal
4: **Pour Tout** $A \in I$ **Faire**
5: nbInsTotal = nbInsTotal + *Nombre d'instructions de A*
6: **Fin Pour**

7.6.2 Contraintes d'optimisation

Cette section décrit les contraintes suivantes :

- séparation minimale,
- respect de l'espace aérien,
- avion non-affecté,
- temps minimal entre deux commandes.

Toutes ces contraintes, à l'exception de *respect de l'espace aérien*, ont été employées dans les essais de résolutions présentés.

Séparation minimale

Cette contrainte est sans aucun doute la plus importante de tous les paramètres d'un problème de résolution de conflits.

L'ATC prend pour hypothèse que chaque avion est entouré par une enveloppe de protection de forme cylindrique, tel qu'illustré à la figure 7.5. Cette enveloppe est modélisée par un cylindre placé au centre de gravité de l'avion. Les dimensions de ce cylindre varient selon l'espace aérien dans lequel l'avion se situe. Par exemple, dans la phase en route, le rayon R normalement employé est de 5 nm (mile nautique) et la hauteur de 2000 pieds ($\Delta Alt = \pm 1000$), alors que le rayon d'une phase terminale est généralement de 3 nm. Ainsi, deux avions sont en conflit si un avion pénètre l'enveloppe de protection de l'autre.

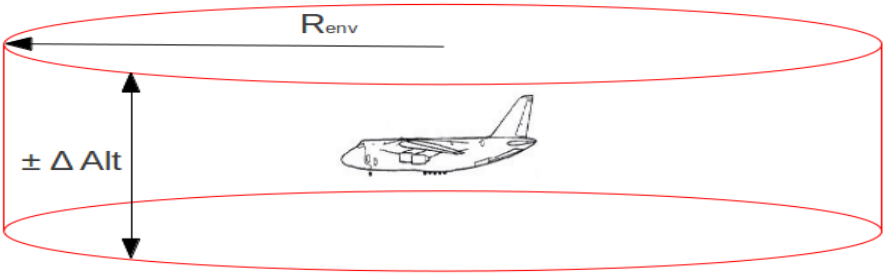


Figure 7.5 Enveloppe de protection autour d'un avion.

Dans cette représentation, aucune distinction n'est faite entre la notion de perte de séparation et celle d'alerte de conflits à court terme (STCA)[4]. Le STCA est en fait une alarme indiquant une perte de séparation imminente. Par exemple, une perte de séparation pourrait avoir un rayon de $3nm$, alors que celui d'une STCA pourrait être de $6nm$.

La position d'un avion est décrite dans un espace cartésien en trois dimensions. L'algorithme employé est très simple : il s'agit pour un temps t donné de vérifier qu'aucun point de toutes les trajectoires ne soit à distance inférieure de la limite horizontale (rayon) et verticale. Cette évaluation est répétée pour chaque pas de calcul. Notons qu'un pas de calcul trop grand ne pourrait peut-être pas garantir une séparation dans une situation ayant une densité de trafique élevée. D'un autre côté, un pas trop faible accroît le temps de calcul pour la résolution.

Soit les avions A et B dont les positions cartésiennes, au temps t, sont définies par $\vec{S_a} = \left(S_{a_x}, S_{a_y}, S_{a_z}\right)$ et $\vec{S_b} = \left(S_{b_x}, S_{b_y}, S_{b_z}\right)$ respectivement. La zone de protection autour de chaque avion est respectée si les inégalités suivantes sont satisfaites, en sachant que $\Delta X = S_{b_x} - S_{a_x}$, $\Delta Y = S_{b_y} - S_{a_y}$ et $\Delta Z = S_{b_z} - S_{a_z}$:

4. STCA : *Short Term Conflict Alert*

$$\Delta X^2 + \Delta Y^2 > R^2 \tag{7.19}$$

ou

$$\Delta X^2 + \Delta Y^2 \leq R^2 \quad \text{et} \quad |\Delta Z| > \Delta Alt \tag{7.20}$$

L'équation 7.21 décrit la fonction de coût de cette contrainte pour un individu composé de N avions.

$$\Delta R_{env} = \sum_{i=0}^{N} \begin{cases} \sqrt{\Delta X_i^2 + \Delta Y_i^2} & \text{si } \Delta Z < \Delta Alt \\ 0 & \text{sinon} \end{cases} \tag{7.21}$$

où R_{env} et ΔR_{env} symbolise respectivement le rayon de l'enveloppe de protection de l'avion et la variation de ce rayon qui a été enfreint.

La figure 7.6 présente graphiquement un conflit entre deux avions pour un rayon de protection de 5 nm. Notons que les cercles illustrés autour des deux avions ont des rayons de 2.5 nm. Ainsi, le début d'un conflit se produit lorsque les deux cercles se touchent. Cette astuce a également été employée dans d'autres études [Geser *et al.*, 2002].

Respect de l'espace aérien

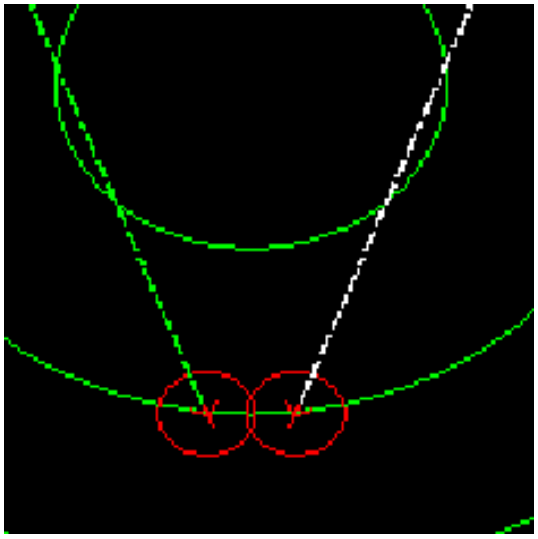
La contrainte *Respect de l'espace aérien*[5] est une variante de la contrainte précédente et ne sera pas abordée dans les essais en simulation. Cette contrainte peut être traitée de façon similaire à la contrainte de *Séparation minimale*.

Il y a un seul espace aérien dans les essais qui seront présentés. Dans la réalité, le ciel est divisé en plusieurs espaces aériens, ou volumes, ayant des propriétés différentes qui sont généralement propres au contrôle aérien. Le respect de l'espace aérien consiste à s'assurer que les avions ne traverseront pas un espace aérien donné. Cette contrainte pourrait être employée pour éviter qu'un avion pénètre un espace aérien interdit, tel qu'une zone d'entrainement militaire.

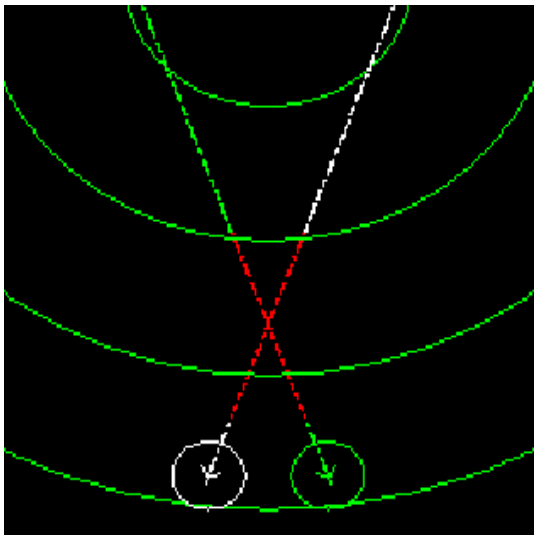
Avion non-affecté

Cette contrainte vise à s'assurer qu'un avion désigné ne reçoive pas d'instructions (ou d'instructions supplémentaires) de la part de l'ATC. C'est une situation qui se produit occasionnellement pour des appareils ambulances (*MEDEVAC*). Ces avions ont toujours

5. En anglais le terme *airspace infringement* est employé.



(a) Début du conflit.



(b) Conflit terminé.

Figure 7.6 Représentation graphique d'un conflit entre deux avions en 2 dimensions.

la priorité de la part de l'ATC tant et aussi longtemps que cette priorité ne compromet pas la sécurité. Les appareils ayant déclaré une urgence sont également traités de façon prioritaire. Dans les exemples de résolutions de conflits de ce projet, la priorité se traduit par l'omission d'instruction ATC à certains avions.

Temps minimal entre deux commandes

Cette contrainte a pour but de s'assurer qu'il y ait un délai minimum entre les diverses instructions transmises aux avions. Dans le monde actuel, les instructions destinées aux avions sont généralement acheminées par le biais de communications entre contrôleurs et

pilotes sur des fréquences radio appropriées. De plus, une instruction ATC doit normalement être suivie par une relecture de la part du pilote, assurant l'ATC que ce dernier a bien compris ce qui lui est demandé. Selon la revue de la littérature effectuée par l'auteur, il semble qu'aucune étude de résolution de trafic automatisé n'ait traité cet aspect. Voici un exemple d'un échange entre un contrôleur et un pilote :

ATC (instruction) : Air Canada 123, virez à gauche cap 150 maintenez 6000 pieds
Pilote (relecture) : Air Canada 123 gauche cap 150 maintient 6000 pieds

Le délai associé à cette contrainte modélise le temps de communication nécessaire pour échanger une instruction et une relecture. Il serait irréaliste d'assigner une valeur trop faible, car les communications ne pourraient pas se faire de façon appropriée. D'un autre côté, une valeur trop grande serait trop conservative et affecterait beaucoup les résultats d'optimisation. Un échange typique (instruction et relecture) prend environ de 5 à 10 secondes. Par contre, plus de temps est nécessaire lorsque la relecture du pilote est incorrecte. Pour répondre à la deuxième situation, un délai de 10 secondes a été employé. Notons que plus ce délai est élevé et plus il est difficile de respecter cette contrainte, rendant par le fait même plus ardue la découverte d'une solution viable. Voici un exemple d'un échange dans lequel le pilote a fait une mauvaise relecture :

ATC (instruction 1) : Air Canada 123, virez à gauche cap 150 maintenez 6000 pieds
Pilote (relecture 1) : Air Canada 123 gauche cap 170 maintient 6000 pieds
ATC (instruction 2) : Air Canada 123, négatif virez à gauche cap 150
Pilote (relecture 2) : Air Canada 123 gauche cap 150

7.7 Algorithme de résolution

L'algorithme de résolution de conflits emploie l'algorithme génétique et l'algorithme de colonies de particules décrits aux sections 6.1 et 6.2 respectivement. Comme dans tout algorithme évolutionnaire, les individus sont évalués (contraintes et objectifs) à chaque génération. Dans le cas de la résolution de conflits, l'évaluation de certaines contraintes, tel que le respect de la séparation, est assez coûteux, du fait du grand nombre d'opérations de calcul.

La difficulté dans l'évaluation des objectifs et des contraintes repose sur le fait que leur évaluation est liée aux trajectoires de tous les avions composant un individu. Les objectifs et les contraintes sont évalués de façon indépendante, et ce, comme tout algorithme évolutionnaire. Pour réduire le nombre d'opérations nécessaires, une fonction de pré-évaluation est utilisée. L'évaluation d'un individu se décompose en trois étapes, soit l'évaluation de

la fonction de pré-évaluation, l'évaluation des contraintes et l'évaluation des objectifs. Les objectifs sont évalués uniquement si toutes les contraintes sont satisfaites.

Dans le cadre de cette recherche, un individu représente une solution à un problème de résolution de conflits. La fonction de pré-évaluation consiste à faire évoluer tous les avions d'un individu pour la période de résolution définie. Ainsi, à chaque génération, tous les déplacements des avions seront évalués (ou simulés). L'avancement dans le temps sera fonction du pas de calcul employé. Tout au long de l'évaluation, la fonction enregistre des paramètres qui seront par la suite utilisés par les fonctions objectives et de contraintes.

L'algorithme 10 présente l'algorithme de pré-évaluation des objectifs et des contraintes. Il est parcouru pour chaque individu de la population à chaque génération de l'algorithme génétique. L'évaluation de la fonction de pré-évaluation est très coûteuse en terme d'opération arithmétique, et c'est pour cette raison qu'un modèle cinématique simple a été employé. L'étape de *remise à zéro* consiste à replacer chaque avion dans sa position initiale, tel que défini dans le problème. Le temps de résolution T_{max} correspond à la durée maximale de simulation employée durant la résolution, alors que ΔT est l'incrément de temps dans la simulation des mouvements d'avions, décrit à la ligne 9.

Algorithme 10 Algorithme de la fonction de pré-évaluation d'un individu

1: Individu : I
2: Avion i de I : A_i
3: Temps maximum de résolution : T_{max}
4: Initialisation de la perte de séparation totale de I : $TotalDistSep = 0$
5: **Pour Tout** $A_i \in I$ **Faire**
6: Retour aux conditions initiales de A
7: **Fin Pour**
8: **Tant que** $temps < T_{max}$ **Faire**
9: Évoluer A pendant ΔT
10: **Pour Tout** $A_i \in I$ **Faire**
11: **Pour Tout** $A_j \in I$, $(i \neq j)$ **Faire**
12: $distance =$ Séparation(A_i, A_j)
13: $TotalDistSep = TotalDistSep + distance$
14: **Fin Pour**
15: **Fin Pour**
16: $temps = temps + \Delta T$
17: **Fin Tant que**

Les étapes suivantes sont liées à l'évaluation de la séparation minimale. Malgré l'utilisation d'une cinématique très simple, telle que définie à la section 7.3, l'analyse de la séparation se fait de façon numérique et non analytique. Une évaluation analytique, quoique complexe,

aurait probablement permis de réduire le nombre d'opérations et ainsi diminuer le temps de résolution. Le grand désavantage est qu'une nouvelle méthode doit être définie pour chaque type de manoeuvre employée dans la résolution.

La fonction *Séparation* de la ligne 12, évalue la séparation minimale entre les avions A_i et A_j. Lorsqu'il y a perte de séparation, elle retourne une valeur correspondante à la variation de la distance de violation. Par exemple, la fonction retournera 1 nm si la distance entre les deux avions est de 4 nm alors que la séparation minimale est de 5 nm. La fonction retourne 0 si le critère de distance minimale est respectée. La distance de violation totale, *TotalDistSep*, est utilisée par la contrainte *Séparation minimale*.

La fonction d'évaluation globale d'un individu est détaillée par l'algorithme 11.

Algorithme 11 Algorithme d'évaluation d'un individu

```
1: Individu : I
2: Pré-évaluation de I
3: Évaluation des contraintes de I
4: Si I est réalisable Alors
5:     Évaluation des objectifs de I
6: Fin Si
```

7.8 Implantation

La figure 7.7 illustre le diagramme UML de classes de l'algorithme SPEA-MOD adapté à la résolution de conflits. Ce diagramme diffère du diagramme 6.1 principalement par la classe composante *Avion* et la classe *Cmd*. Ces deux classes définissent un gène du problème de résolution. La classe *Avion* modélise la cinématique d'un avion, alors que les manoeuvres d'évitement et les opérateurs génétiques sont définis par la classe *Cmd*. La mise en oeuvre des algorithmes décrit au chapitre 6 et de ce chapitre ont nécessité environ 15000 lignes de programmation dans le language C++.

Il y a une classe de manoeuvre de résolution pour une résolution en 2D et une différente pour une résolution en 3D. Bien entendu, ces deux classes portent des noms différents, mais par simplicité elles s'appellent *Cmd*. L'algorithme de séquençage présenté au chapitre 9 emploie les mêmes classes à l'exception de la classe Cmd qui sera différente pour ce problème.

Un autre point de l'implémentation devant être éclairci est la façon dont les résultats d'optimisation sont obtenus puis rejoués en simulation. De par la nature de l'algorithme de détection de conflits, les temps t_1 et t_2 doivent être des multiples du pas de calcul

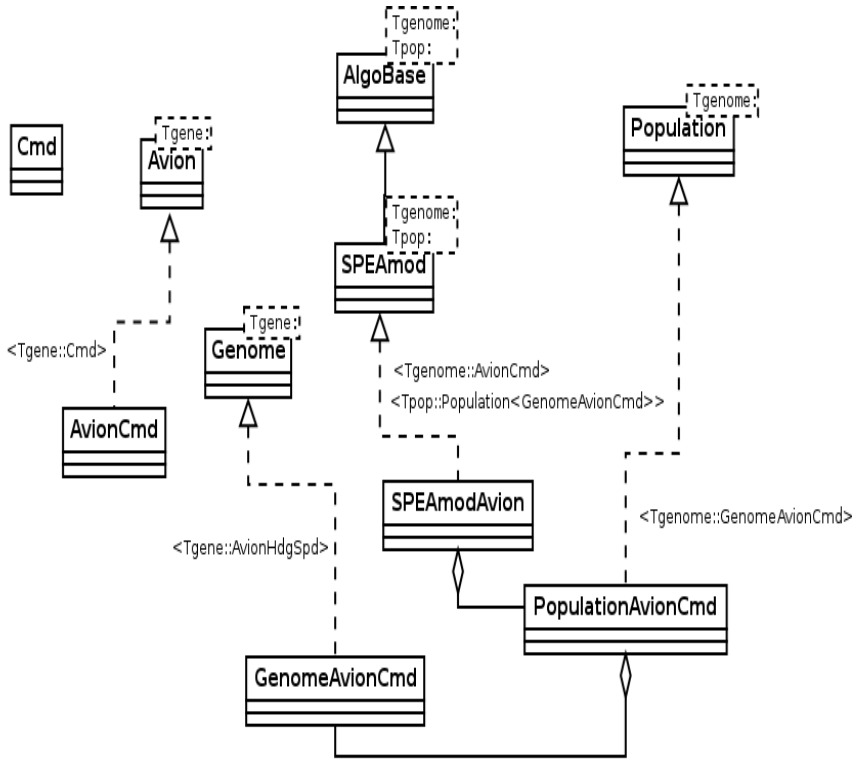


Figure 7.7 Diagramme UML de l'algorithme SPEA-MOD pour la résolution de conflits.

ΔT. Par exemple, t_1 et t_2 sont des multiples de 10 sec lorsque le pas de calcul est de 10 secondes. Ainsi, de cette façon, même en augmentant le pas de calcul pour diminuer le temps de résolution, les résultats obtenus par ce nouveau pas seraient valides même en rejouant les résultats à un pas de calcul inférieur.

CHAPITRE 8

RÉSOLUTION NUMÉRIQUE, PHASE EN ROUTE

Ce chapitre présente des situations de résolutions de conflits dans la phase de vol en route pour les algorithmes SPEA-MOD et PSO-MO. La conception de ces algorithmes à la résolution de conflits a été définie précédemment au chapitre 7. Une comparaison de performance entre ces deux algorithmes a été étudiée au chapitre 6, sans vraiment parvenir à déterminer lequel des deux est le plus performant. Ces deux algorithme seront à nouveau comparés, afin de déterminer lequel est le plus approprié pour résoudre des problèmes de résolutions de conflits.

La section 8.1 présente un survol des outils de simulation et d'analyses des résultats qui ont été nécessaires à la réalisation de ce projet. La section 8.2 traite d'une résolution de conflits entre deux avions volant en sens contraire. Ce premier problème a pour but de valider les algorithmes conçus. La section 8.3, quant à elle, présente des situations conflictuelles impliquant plusieurs avions situés sur un cercle et se dirigeant vers le centre de ce cercle. Les déviations de trajectoires de ces situations sont dans le plan horizontal uniquement. Même si ces exemples semblent académiques, ils démontrent néanmoins la puissance de la solution, car elle résout des situations plus complexes que ce qu'il pourrait se produire dans la réalité. Finalement, la section 8.4 traite également de situations conflictuelles autour d'un cercle, mais cette fois-ci elles sont résolues en trois dimensions, car les déviations de trajectoires sont dans l'espace.

8.1 Outils de simulation

Deux outils ont été développés pour l'analyse des résultats. Le premier consiste à générer des tableaux de résultats de façon automatique, alors que le second outil est constitué de deux programmes Python, un simulateur et un écran de visualisation.

Après chaque résolution complétée, l'algorithme écrit des données dans des fichiers. Par la suite, des scripts écrits en langage Python parcourent ces fichiers et produisent des tableaux de résultats en langage LaTeX, qui à leur tour sont inclus automatiquement dans cette thèse. Cet outil a permis d'éviter des erreurs de transcription pour produire les

tableaux de résultats. Les tableaux des chapitres 6.4 et 9.3 ont également été été produits de cette façon. Les programmes en langage Python totalisent environ 1500 lignes de code.

Le simulateur utilise les données de résolutions pour simuler les trajectoires des avions d'une situation donnée. L'écran de visualisation, qui ressemble à un écran radar simplifié, reçoit les positions d'avions produites par le simulateur et affiche les positions actuelles ainsi que les positions passées des avions. De plus, un symbole d'avion et un cercle représentant la séparation minimale sont affichés à la position actuelle. À des périodes de temps définies, l'écran de visualisation calcule la séparation entre les différents avions de la situation et montre en rouge les avions dont la séparation minimale n'est pas respectée. Le symbole d'avion reprend sa couleur originale lorsque la séparation est respectée. Cet outil fut essentiel à la validation des résultats de situations ayant plusieurs avions. Le simulateur et l'écran de visualisation totalisent environ 6000 lignes de code de programmation en langage Python.

8.2 Collision à deux avions

Ce premier exemple de résolution de conflit implique deux avions volant à 200 kts sur la même trajectoire mais dans des directions opposées, tel qu'illustré à la figure 8.1. Notons que l'image illustre une situation dans laquelle le conflit a été résolu. Initialement les avions sont à 600 secondes (10 minutes) d'une collision. Cet exemple simple a permis de valider les algorithmes PSO-MO et SPEA-MOD au problème de la résolution de conflits.

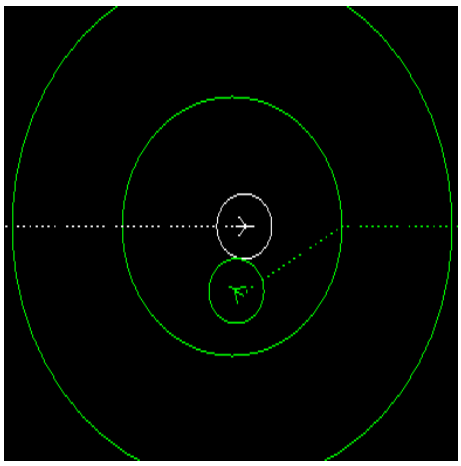


Figure 8.1 Résolution de conflit entre deux avions en 2 dimensions.

Le tableau 8.1 présente les paramètres de résolution employés.

Notons que les paramètres de la manoeuvre point tournant (T_0, T_3 , ΔV et ΔCap) sont des multiples des incréments spécifiés. Les tableaux 8.2 et 8.3 présentent une moyenne des instructions fournies pour 100 résolutions de ce problème par les algorithmes SPEA-MOD et PSO-MO respectivement. La deuxième colonne de ces deux tableaux présente la durée moyenne d'une instruction donnée, alors que la troisième colonne présente l'utilisation moyenne d'une commande par solution. De par la nature des algorithmes évolutionnaires, les résultats peuvent différer d'un essai à l'autre. La comparaison sur 100 simulations permet de comparer les performances de ces deux algorithmes sur une moyenne d'essais. Ce nombre d'essais semble suffisant, car plusieurs répétions de 100 essais ont montré que les résultats étaient répétables. Les changements de vitesse ne sont pas d'une grande utilité lors d'une collision face à face puisqu'ils ne modifient pas les trajectoires des avions et ont donc été omis. La majorité des algorithmes de résolutions de 2 avions analytiques, tel que celui proposé par Back et ses collaborateurs [Bach *et al.*, 2007], fournissent uniquement des changements de direction. Les résultats présentés vont dans le même sens puisqu'il n'y a aucun changement de vitesse.

Tableau 8.1 Paramètres de résolution pour deux avions face à face.

Nombre d'avions en conflit	2
Taille de la population	25
Taille de l'archive	15
Taux de croisement (SPEA-MOD)	0.2
Taux de mutation	0.1
Nombre maximal de générations	300
Temps minimal pour débuter la résolution	60 sec
Temps total de résolution	3000 sec
Incrément de temps	10 sec
Plage de ΔCap	$[-70, 70]$ deg
Incrément de ΔCap	10 deg
Plage de ΔV	$[-30, 30]$ kts
Incrément de ΔV	10 kts
Objectifs à minimiser :	
Différence dans l'ETA	
Nombre total de commandes	
Contraintes :	
Perte de séparation (voir 7.5, page 106)	R=5 nm
Temps minimal entre deux commandes	10 sec

Ce problème est relativement simple à résoudre, tel qu'indiqué par les résultats. Les deux algorithmes obtiennent une solution viable dès la première génération et une seule commande est nécessaire. Pour un essai donné, il a été remarqué que tous les points de l'archive convergent vers la même solution. Par contre, on remarque un problème de convergence

vers un optimal global, pour tous les essais, puisque la distribution des instructions ne tend pas vers une solution unique. Les algorithmes génétiques sont efficaces pour s'approcher rapidement de l'optimum global pour des problèmes combinatoires de grandes tailles, mais cette efficacité est moindre pour une convergence sur l'optimum. Cet aspect a également été remarqué avec l'algorithme PSO-MO. La combinaison de l'algorithme avec une autre technique de recherche locale aurait sans doute amélioré la convergence.

Tableau 8.2 Statistiques : 100 essais, 2 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	696	0.03
$\Delta Cap = 20$	511	0.31
$\Delta Cap = 30$	327	0.14
$\Delta Cap = 40$	239	0.52
$\Delta Cap = 50$	270	0.01
Génération moyenne pour une solution réalisable : 1		
Nombre moyen de commandes par solution : 1.01		
Nombre de solutions réalisables : 100/100		

Tableau 8.3 Statistiques : 100 essais, 2 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1050	0.10
$\Delta Cap = 20$	530	0.63
$\Delta Cap = 30$	371	0.23
$\Delta Cap = 40$	290	0.04
$\Delta Cap = 50$	0	0.00
Génération moyenne pour une solution réalisable : 1		
Nombre moyen de commandes par solution : 1		
Nombre de solutions réalisables : 100/100		

Le tableau 8.4 présente la solution du problème correspondant à la figure 8.1. Rappelons que T_0, T_3, ΔCap_i, ΔV_i sont les variables d'optimisation et que ces variables correspondent aux gènes d'une solution, tel que présenté à la figure 7.4 (page 99) et au tableau 7.1 (page 101).

Tableau 8.4 Un exemple de solution en 2 dimensions au problème.

Avion	T_0	T_3	ΔCap	ΔV
1	0	0	0	0
2	70	1120	-10	0

Les temps T_0 et T_3, obtenus par l'optimisation, représentent les moments de début et de fin de manoeuvre respectivement. La manoeuvre du point tournant étant un triangle isocèle, les temps $T_1 = T_2$ s'obtiennent donc par calcul. Ces résultats sont compatibles avec le fonctionnement du contrôle aérien actuel. Les résultats de cette situation pourraient se traduire en instructions ATC de la façon suivante pour un avion ayant un cap initial de 270 degrés :

À 70 secondes (T_0)
ATC (instruction) : Air Canada 123, virez à gauche cap 260
Pilote (relecture) : Air Canada 123 gauche cap 260

À 595 secondes (T_1, T_2)
ATC (instruction) : Air Canada 123, virez à droite cap 280
Pilote (relecture) : Air Canada 123 droite cap 280

À 1120 secondes (T_3)
ATC (instruction) : Air Canada 123, virez à gauche cap 270
Pilote (relecture) : Air Canada 123 droite cap 270

Le tableau 8.5 présente une analyse comparative des deux algorithmes par le biais de la *Métrique C*, telle que définie à la section 6.5.1. L'accumulation des solutions non dominées pour 100 résolutions a servi à calculer cette métrique. Selon cette métrique, les deux algorithmes produisent des solutions équivalentes. Ce résultat n'est pas surprenant étant donné que ce problème est simple à résoudre.

Tableau 8.5 Métrique C la situation face à face.

Tests	C(SPEA-MOD, PSO-MO)	C(PSO-MO, SPEA-MOD)
face à face	1.000000	1.000000

Comparaison à une méthode analytique

De façon similaire à ce qui a été fait au chapitre précédent, le modèle de résolution planaire à deux avions en face à face a été validé avec un algorithme analytique. Bach et ses collaborateurs ont développé un algorithme de résolution de conflits planaire entre deux avions [Bach *et al.*, 2007]. Le principe de l'algorithme est de dévier un seul avion, par une instruction de virage, pour respecter la séparation. L'avion dévié passera sur une tangente de l'enveloppe protectrice (en forme de cercle) de l'autre avion. Bien qu'aucune démonstration d'optimalité de cet algorithme ne fut démontrée, il semble tout de même optimal pour une situation à deux avions. L'algorithme est résumé en annexe C.

Le tableau 8.6 présente des résultats comparatifs entre l'algorithme géométrique de Bach et SPEA-MOD. Dans cette situation les avions volent en sens inverse à une vitesse de 200 kts. Les temps T_1 et T_2 sont ceux de la manoeuvre du point tournant, les distances représentent les longueurs des segments d'évitement. ψ_a et ΔCap sont les changements de direction obtenus par les algorithmes de Bach et SPEA-MOD respectivement.

Les valeurs spécifiées par l'algorithme analytique sont représentées par des nombres réels, arrondis au dixième, et non des nombres entiers comme pour l'algorithme SPEA-MOD. Afin de mieux comparer les deux algorithmes, l'incrément en changement de direction ΔCap permis est de 1 degré, comparativement aux 10 degrés normalement utilisés (voir le tableau 8.1).

Tableau 8.6 Comparaison entre SPEA-MOD et méthode analytique.

$T_1(sec)$ (sec)	T_2 (sec) (sec)	Distance (km)	ψ_a (deg) Bach	ΔCap (deg) SPEA-MOD
120	1070	97.7	10.9	11
230	980	77.2	13.9	14
350	860	53.5	20.3	21
380	840	47.3	23.0	24

Le tableau 8.6 montre que les ΔCap fournis par SPEA-MOD sont très près de l'algorithme de Bach, démontrant que les résultats sont près de la solution analytique. Ceci peut également être confirmé par inspection visuelle de la figure 8.1.

L'algorithme de Bach ne peut pas être utilisé directement pour des situations ayant plus de deux avions en conflits. En le combinant à une technique itérative, comme celle du *jeton* de Granger, il pourrait sans aucun doute supporter des situations conflictuelles très complexes [Granger, 2002]. Par contre, cet algorithme ne serait pas optimal.

8.3 Avions convergeant vers le centre d'un cercle - 2D

Cette section illustre des situations de résolutions de conflits que l'auteur qualifie de théorique, car il est très improbable qu'une situation similaire à celles présentées puisse se produire dans un environnement réel. Ces situations sont donc beaucoup plus complexes que la réalité. Néanmoins, ces deux exemples mettent en valeur la validité de la solution proposée à la résolution de conflits dans la phase en route. Ces deux situations sont constituées d'avions évoluant à la même vitesse et à la même altitude convergeant vers le centre d'un cercle. Nonobstant quelques détails de mise en oeuvre, les deux exemples diffèrent

par la méthode de résolution choisie. Ce problème est très difficile à résoudre, car chaque avion est en conflit avec tous les autres avions.

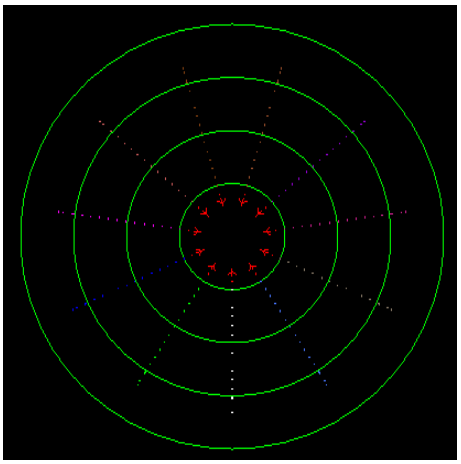


Figure 8.2 11 avions convergeant vers le centre d'un cercle, sans résolution de conflit.

Initialement les avions évoluent à la même altitude et à une vitesse de 200 kts. Tous les avions sont répartis uniformément sur la circonférence d'un cercle fictif et se dirigent vers le centre de ce dernier. Finalement, tous les avions sont à 10 minutes du centre du cercle. La figure 8.2 illustre les trajectoires d'avions sans qu'il y ait de résolution de conflits. Le tableau 8.7 quant à lui expose les paramètres d'optimisation employés. Ce problème autour d'un cercle a été analysé pour sept configurations d'avions, 5, 7, 9, 11, 13, 15 et 17 avions. Pour alléger le texte, uniquement les résultats de la situation à 11 avions sont présentés dans cette section, les autres résultats se trouvent à l'annexe B.1.

Le paramètre *temps minimal pour débuter la résolution* modélise une durée pendant laquelle il ne peut y avoir d'instructions aux avions. Ainsi, les avions sont donc en fait à 9 minutes du centre du cercle lorsque les premières commandes sont données. Ce paramètre a été utilisé uniquement pour rendre la visualisation plus claire. Il est à noter que la perte de séparation se produit avant que les avions aient atteint le centre du cercle, tel qu'illustré à la figure 8.2. Le paramètre *temps total de résolution* quant à lui modélise la durée totale de la simulation employée dans la résolution. Dans l'exemple présenté dans cette section, les avions sont à 600 secondes du centre du cercle et une valeur de 3000 secondes pour ce paramètre est employée. La résolution se fait donc sur une période de 3000 secondes et pendant cette période toutes les contraintes doivent être respectées pour qu'une solution soit valide. Le but de cette période de résolution, est de s'assurer que les manoeuvres sont terminées, afin d'éviter qu'un conflit se produise après cette période. Malheureusement, une certaine heuristique est employée dans l'assignation de ce paramètre. Les temps de fin

Tableau 8.7 Paramètres de résolution en deux dimensions.

Nombre d'avions en conflit	17, 15, 13, 11, 9, 7, 5
Taille de la population	25
Taille de l'archive	15
Taux de croisement (SPEA-MOD)	0.2
Taux de mutation	0.1
Nombre maximal de générations	300
Temps minimal pour débuter la résolution	60 sec
Temps total de résolution	3000 sec
Incrément de temps	10 sec
Plage de ΔCap	$[-70, 70]$ deg
Incrément de ΔCap	10 deg
Plage de ΔV	$[-40, 40]$ kts
Incrément de ΔV	10 kts
Objectifs à minimiser :	
Différence dans l'ETA	
Nombre total de commandes	
Contraintes :	
Perte de séparation	$R = 5$ nm
Temps minimal entre deux commandes	10 sec

de résolution T_3 doivent être inférieurs au temps total de résolution, ce qui nécessite une valeur de total élevée. Par contre, une valeur trop élevée ralentit le temps de résolution inutilement.

La figure 8.3 illustre l'état d'une résolution après 500 secondes et 800 secondes de simulation, alors que la figure 8.4 illustre l'état de la même résolution à 1000 et 1200 secondes respectivement. Rappelons que le critère d'arrêt de l'optimisation est le nombre de générations qui est fixé à 300. Quoique cet exemple ne représente pas une situation réaliste, car trop complexe, il permet néanmoins de démontrer la puissance de l'algorithme. Il aurait été difficile, voir même presque impossible, à un contrôleur aérien d'égaler les performances en utilisant la même méthode de résolution.

Le tableau 8.8 témoigne de la complexité du problème en illustrant une solution réalisable. Les deux dernières colonnes présentent les valeurs des objectifs *Différence dans l'ETA* (ΔETA) et *Nombre total de commandes* (Nb Cmd) par avion. Pour les avions 2 et 10 les valeurs de ΔETA sont négatives, indiquant l'arrivée de l'avion plus tôt que prévue.

Les tableaux 8.9 et 8.10 présentent des résultats de 100 simulations au moyen de SPEA-MOD et PSO-MO respectivement. En comparant les résultats de ces deux tableaux, on remarque que SPEA-MOD est plus performant que PSO-MO. SPEA-MOD obtient un taux

Tableau 8.8 Un exemple de solution en 2 dimensions au problème.

Avion	T_0	T_3	ΔCap	ΔV	ΔETA (sec)	Nb Cmd
1	100	1390	40	-30	1096	2
2	200	1490	30	20	-45	2
3	160	690	40	-10	237	2
4	150	1200	30	-30	668	2
5	1820	2910	0	0	0	0
6	110	1360	-50	0	695	1
7	250	1170	40	30	8	2
8	300	970	30	-10	188	2
9	510	1050	-50	30	109	2
10	140	890	-10	30	-162	2
11	220	950	30	-10	205	2

de solutions réalisables de 85% comparativement à PSO-MO qui en obtient uniquement 56%. Dans ce contexte, une solution réalisable veut dire que l'algorithme a réussi à obtenir au moins une solution réalisable à un essai donné. Ainsi, un taux de solution réalisable de 85% veut dire que l'algorithme a réussi à obtenir des solutions viables à 85 des 100 essais. D'un point de vue du contrôle aérien, ce paramètre est le plus important, car il indique que l'algorithme a réussi à trouver au moins une solution sécuritaire, c'est-à-dire en respectant la séparation minimum requise.

Si la taille de la population de PSO-MO est de 45, au lieu de 25, alors le taux de solutions réalisables devient à 79%. Ce nouveau résultat est toujours inférieur à celui obtenu par SPEA-MOD. De plus, SPEA-MOD requiert moins de temps, en terme du nombre de générations (ou itérations) pour obtenir une première solution réalisable, 93 générations versus 196 générations pour PSO-MO. Un second point intéressant pour le contrôle aérien est que SPEA-MOD requiert en moyenne 17 instructions comparativement à 19 pour PSO-MO.

Le tableau 8.11 présente une analyse comparative des deux algorithmes par le biais de la métrique C, pour plusieurs configurations d'avions. Uniquement les solutions non dominées des 100 essais sont prises en compte dans le calcul de cette métrique car on veut comparer les meilleures solutions produites par les deux algorithmes. Les solutions de PSO-MO dominent celles de SPEA-MOD uniquement pour la situation à 5 avions. Pour un problème fortement contraint, comme celui à 11 avions, SPEA-MOD semble mieux performer sur tous les aspects. Ainsi, SPEA-MOD est mieux adapté que PSO-MO pour des problèmes fortement contraints. De plus, selon les résultats de l'annexe B.1, l'algorithme SPEA-MOD

Tableau 8.9 Statistiques : 100 essais, 11 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1045	0.52
$\Delta Cap = 20$	1054	1.20
$\Delta Cap = 30$	955	1.87
$\Delta Cap = 40$	1033	3.18
$\Delta Cap = 50$	1164	3.13
$\Delta V = 10$	1150	1.74
$\Delta V = 20$	1026	3.19
$\Delta V = 30$	1016	2.19
Génération moyenne pour une solution réalisable : 93		
Nombre moyen de commandes par solution : 17.02		
Nombre de solutions réalisables : 85/100		

Tableau 8.10 Statistiques : 100 essais, 11 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1326	0.83
$\Delta Cap = 20$	1387	1.41
$\Delta Cap = 30$	1520	1.99
$\Delta Cap = 40$	1573	2.83
$\Delta Cap = 50$	1568	3.35
$\Delta V = 10$	1477	3.08
$\Delta V = 20$	1466	3.36
$\Delta V = 30$	1508	2.18
Génération moyenne pour une solution réalisable : 196		
Nombre moyen de commandes par solution : 19.03		
Nombre de solutions réalisables : 56/100		

a été incapable d'obtenir une solution viable pour l'essai à 17 avions, alors que PSO-MO a été incapable pour les essais à 15 et 17 avions.

Tableau 8.11 Comparaison de couverture pour un certain nombre d'avions, cercle-2D.

Tests	C(SPEA-MOD, PSO-MO)	C(PSO-MO, SPEA-MOD)
5 avions	0.447570	0.876623
7 avions	0.872521	0.846715
9 avions	0.994398	0.900000
11 avions	1.000000	0.733871
13 avions	1.000000	0.928177
15 avions	1.000000	0.994764
17 avions	1.000000	1.000000

Ces résultats semblent légèrement en contradiction avec ceux obtenus aux problèmes typiques de la section 6.6. La différence de performance entre les deux algorithmes s'explique par la nature des problèmes étudiés. La mauvaise performance de PSO-MO est attribuable au fait qu'il a plus de difficulté à favoriser la diversité comparativement à SPEA-MOD. Ceux de la section 6.6 étaient soit non-contraints ou légèrement contraints en comparaison à celui présenté dans cette section. Ainsi, les résultats semblent indiquer que l'algorithme PSO-MO est moins approprié que SPEA-MOD lorsque les problèmes sont fortement contraints.

Ces résultats sont similaires à ceux obtenus par Hassan et ses collaborateurs, à l'exception que leurs résultats étaient obtenus pour des problèmes mono-objectif [Hassan *et al.*, 2004]. Ils ont remarqué que l'algorithme PSO est plus efficace qu'un algorithme génétique dans la résolution de problèmes non-linéaires et non-contraints à variables de décision continues (codage réel) et moins efficace pour des problèmes non-linéaires et contraints avec des variables de décision continues. Même si les deux études semblent obtenir des conclusions similaires, il serait souhaitable de poursuivre la recherche dans ce domaine pour y confirmer les résultats.

Tel que présenté à la section 6.2, PSO-MO emploie un terme de turbulence, équivalent à la mutation dans les algorithmes génétiques, pour favoriser l'exploration de l'espace de recherche. Notons que les résultats sont moins bons lorsque ce terme est omis.

La figure 8.5 illustre le nombre de solutions irréalisables par génération produites par l'algorithme SPEA-MOD pour un essai donné. Ces solutions irréalisables se retrouvent dans la population et dans l'archive. Ce nombre est légèrement supérieur à la taille de la population au début, car il n'y a aucune solution réalisable. Le nombre diminue légèrement lorsque l'algorithme découvre une première solution réalisable, qui dans cet exemple est produite à la génération 62. Après la génération 62, l'archive contient uniquement des solutions réalisables, alors la population contient environ 23 solutions irréalisables, et ce jusqu'à la dernière génération. Ce nombre élevé s'explique principalement par le fait qu'il s'agit d'un problème fortement contraint et également par les opérateurs génétiques, plus particulièrement la mutation. Vers la $250^{\text{ième}}$ génération, le nombre de solutions irréalisables diminue pour atteindre 0 à la $294^{\text{ième}}$ génération. Cette diminution est due à la mutation non-uniforme qui ralentit la diversification de la recherche à mesure que le nombre de générations approche le nombre de générations maximales. Des résultats similaires ont été obtenus par PSO-MO. Il est à noter que le nombre de solutions irréalisables demeure constant dans le cas où l'algorithme n'a obtenu aucune solution réalisable.

Les opérateurs de croisement et de mutation (turbulence pour PSO-MO) se basent sur la génération de nombres aléatoires non uniformes, tel que défini à la section 4.2. Des problèmes de convergence de l'algorithme ont été observés lors de l'utilisation d'une génération uniforme dans les opérateurs, plus particulièrement pour la mutation. Ces problèmes sont absents lorsqu'on emploie une mutation non uniforme.

8.4 Avions convergeant vers le centre d'un cercle - 3D

Cette section reprend le même problème qu'à la section précédente, mais cette fois-ci en trois dimensions, car une instruction d'altitude est ajoutée. La manoeuvre d'évitement employée est celle du point tournant en trois dimensions, telle que définie à la section 7.4. Selon la littérature, il semble qu'aucune recherche n'ait abordé la résolution de conflits en 3D de façon optimale. Le tableau 8.12 présente les paramètres de résolutions, alors que les tableaux 8.13 et 8.14 présentent les résultats d'optimisation pour une situation impliquant 17 avions.

Tableau 8.12 Paramètres de résolution en trois dimensions.

Taille de la population	25
Taille de l'archive	15
Taux de croisement	0.2
Taux de mutation	0.1
Nombre maximal de générations	300
Temps minimal pour débuter la résolution	60 sec
Temps total de résolution	3000 sec
Plage de ΔCap	$[-70, 70]$ deg
Plage de ΔV	$[-40, 40]$ kts
Plage de ΔAlt	$[-2000, 2000]$ m
Objectifs à minimiser :	
Variation de l'ETA	
Variation de la trajectoire dans le plan horizontal	
Nombre total de commandes	
Contraintes :	
Perte de séparation	$R = 5$ nm
	$\Delta Alt = \pm 1000$ pieds
Temps minimal entre deux commandes	10 sec

Les résultats des tableaux 8.13 et 8.14 montrent clairement la supériorité de l'algorithme SPEA-MOD. Tout d'abord, 90 des 100 résolutions de SPEA-MOD ont trouvé des solutions réalisables, comparativement à seulement 24 pour PSO-MO. De plus, SPEA-MOD

nécessite environ 226 générations par résolution pour obtenir une solution viable, comparativement à 263 pour PSO-MO.

Tableau 8.13 Statistiques : 100 essais, 17 avions, 3D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1396	2.69
$\Delta Cap = 20$	1417	3.19
$\Delta Cap = 30$	1384	3.78
$\Delta Cap = 40$	1400	3.28
$\Delta Cap = 50$	1415	2.23
$\Delta V = 10$	1426	5.90
$\Delta V = 20$	1368	5.15
$\Delta V = 30$	1393	2.39
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	1357	4.50
$\Delta Alt = 1000$	1394	5.10
$\Delta Alt = 1500$	1439	3.09
$\Delta Alt = 2000$	1475	1.48
Génération moyenne pour une solution réalisable : 226		
Nombre moyen de commandes par solution : 42.78		
Nombre de solutions réalisables : 90/100		

Tableau 8.14 Statistiques : 100 essais, 17 avions, 3D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1360	2.95
$\Delta Cap = 20$	1545	3.27
$\Delta Cap = 30$	1644	3.40
$\Delta Cap = 40$	1634	3.55
$\Delta Cap = 50$	1658	2.57
$\Delta V = 10$	1548	6.19
$\Delta V = 20$	1562	4.96
$\Delta V = 30$	1560	2.55
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	1516	4.74
$\Delta Alt = 1000$	1585	5.04
$\Delta Alt = 1500$	1548	3.10
$\Delta Alt = 2000$	1642	1.91
Génération moyenne pour une solution réalisable : 263		
Nombre moyen de commandes par solution : 44.23		
Nombre de solutions réalisables : 24/100		

L'annexe B.2 présente des résultats similaires, mais pour un nombre d'avions différents. PSO-MO a réussi à obtenir des 100% des essais réalisables pour des situations jusqu'à 13 avions.

Malgré le fait que des instructions d'altitude peuvent être fournies, il semble que dans cette configuration, une situation ayant un nombre d'avions légèrement supérieur à 17 soit une limite supérieure pouvant être traitée par SPEA-MOD. La contrainte de temps minimum entre les commandes est l'élément limitant l'algorithme.

Le tableau 8.15 présente l'analyse comparative des algorithmes SPEA-MOD et PSO-MO par la métrique C. Les résultats montrent encore une fois la supériorité de l'algorithme SPEA-MOD pour des problèmes fortement contraints.

Tableau 8.15 Comparaison de couverture pour un certain nombre d'avions.

Tests	C(SPEA-MOD, PSO-MO)	C(PSO-MO, SPEA-MOD)
9 avions	1.000000	0.608059
11 avions	1.000000	0.225080
15 avions	1.000000	0.171429
17 avions	1.000000	0.468750

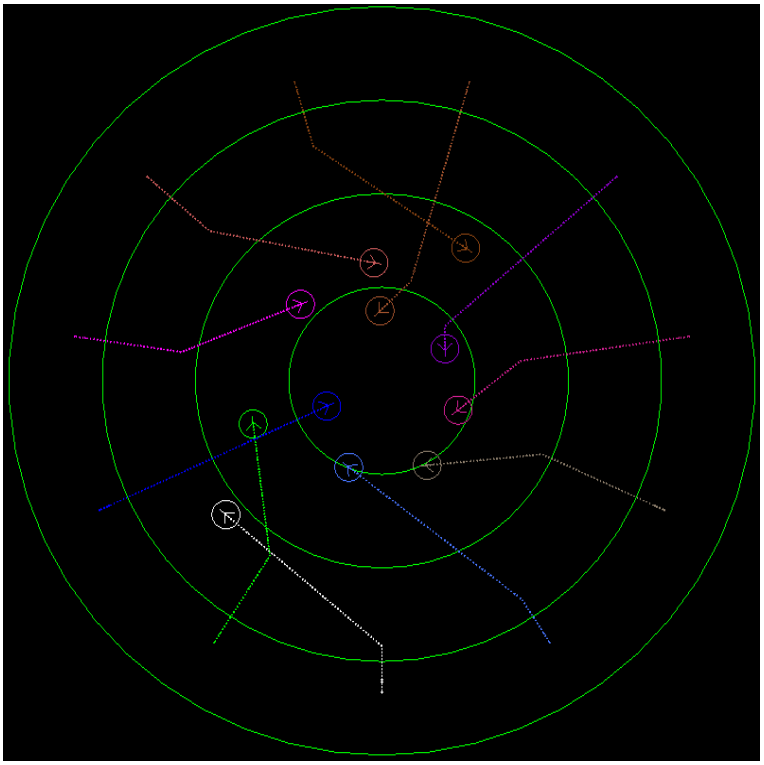
8.5 Sommaire

L'approche multiobjectif ajoute une certaine complexité à l'algorithme qui est reliée au choix de la solution finale parmi l'ensemble des solutions non dominées. Notons que ce problème n'a pas vraiment été abordé dans la présente recherche. Par contre, un avantage important, présent dans les deux algorithmes développés, est l'absence d'heuristique dans le traitement des objectifs et des contraintes. De par l'approche multiobjectif, aucune relation entre les objectifs et les contraintes n'est nécessaire. Cela a rendu possible le traitement de différents critères propres au domaine du contrôle aérien, et ce d'une manière élégante. Les seuls paramètres devant être ajustés de façon manuelle sont le nombre de générations, le taux de mutation et le taux de croisement.

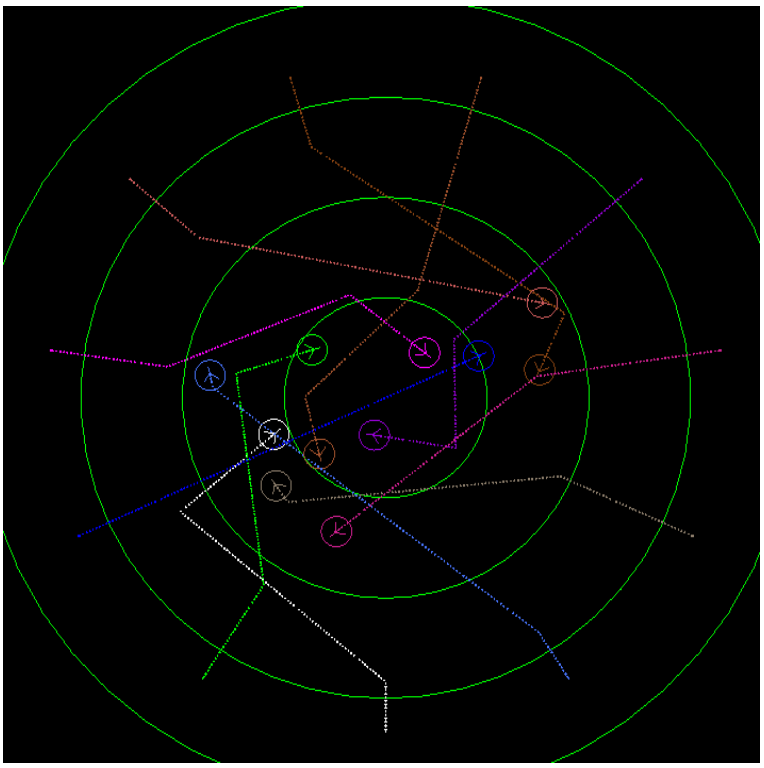
Dans tous les exemples numériques du présent chapitre et celui du chapitre 9.3, un nombre maximal de générations a été utilisé. Dans de futures recherches il serait intéressant, dans le but de raccourcir le temps de calcul, de faire varier le nombre de générations maximal en fonction d'un certain taux de changement de l'archive. De cette façon, il serait plus facile d'appliquer les algorithmes évolutionnaires à des situations réelles de trafic aérien.

La supériorité de l'algorithme SPEA-MOD a été démontrée à plusieurs reprises au cours de ce chapitre. Les problèmes de séquençage en approche, présentés au chapitre 9, utiliseront donc uniquement l'algorithme SPEA-MOD.

Un désavantage du modèle de solution présenté est qu'un seul ensemble d'instructions peut être transmis à un avion pour une résolution donnée. Il est facile d'imaginer une situation dans laquelle cet inconvénient représente un obstacle majeur ; par exemple deux flux de trafic se croisant. L'algorithme présenté au prochain chapitre permet de résoudre ce genre de situation de problème.

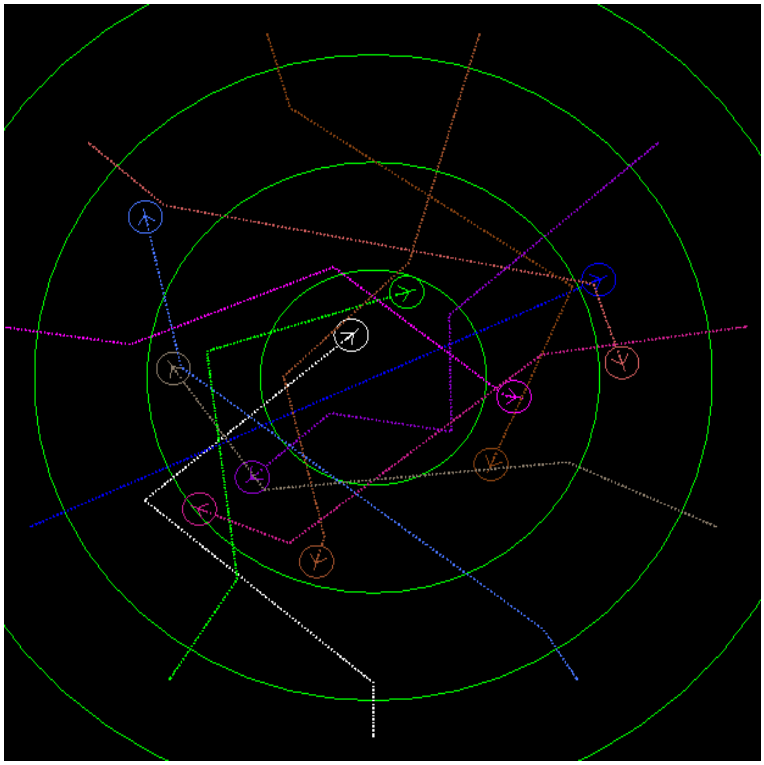


(a) 500 sec.

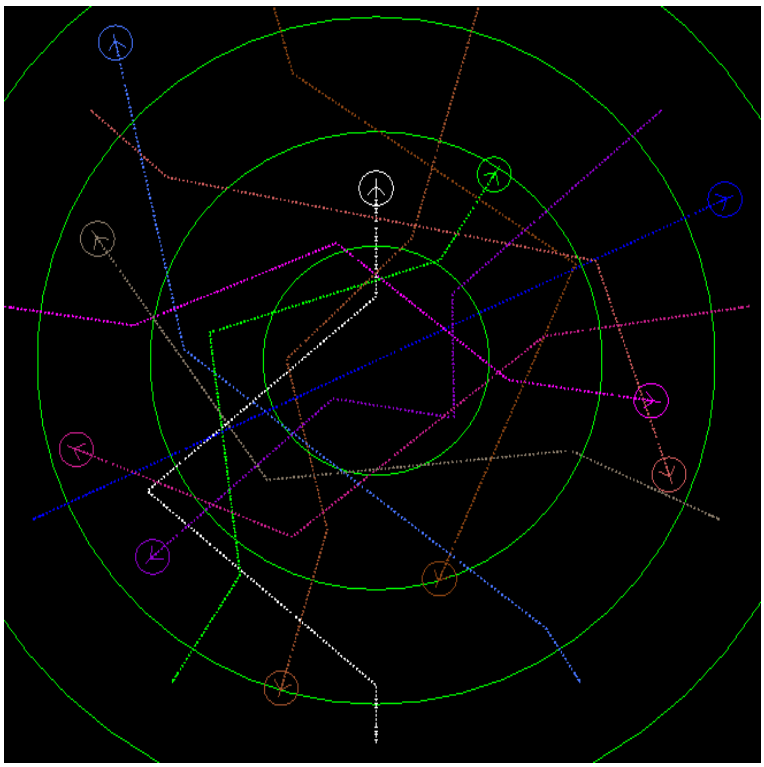


(b) 800 sec.

Figure 8.3 État de la résolution à 500 et 800 secondes.



(a) 1000 sec.



(b) 1200 sec.

Figure 8.4 État de la résolution à 1200 et 2000 secondes.

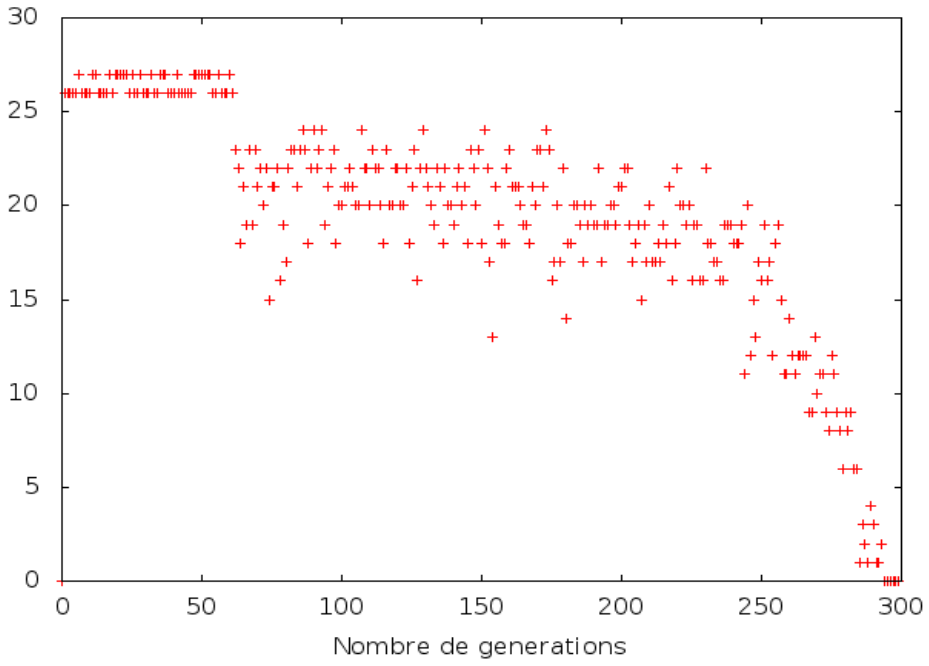


Figure 8.5 Nombres de solutions irréalisables par génération pour SPEA-MOD.

CHAPITRE 9

SÉQUENÇAGE DE LA PHASE APPROCHE

Ce chapitre présente une modélisation de la tâche de séquençage d'avions en arrivée de façon automatisée. Cette tâche consiste à aligner les avions avec l'axe de la piste en vue de l'atterrissage. Le séquençage a également comme résultat le maintien de la séparation. Ainsi, il ne faut pas voir cette tâche comme étant une cause de la résolution de conflits. Comme il s'agit également d'un problème NP difficile, il a été résolu par un algorithme génétique et un algorithme de colonies de particules, et ce de façon similaire à ce qui a été présenté aux chapitres 7 et 8. À ce jour, aucune recherche n'a résolu un problème de séquençage d'avions de façon optimale.

La section 9.1 présente la problématique du séquençage d'avions en arrivée alors que la section 9.2 traite de la solution développée pour résoudre ce problème complexe. Le chapitre 9.3 présente des résultats numériques à certains problèmes.

9.1 Introduction

Le contrôleur radar d'une région terminale, également appelé contrôleur d'approche, gère principalement le trafic en départ et en arrivée d'un ou plusieurs aéroports. Ces régions sont séparées en secteurs dont la division peut se faire dans le plan horizontal et vertical, par exemple, un secteur nord et un secteur sud. Lorsque la densité de trafic est élevée, le flux d'avions est divisé en deux, soit les vols de départs et les vols d'arrivées, nécessitant ainsi deux contrôleurs au lieu d'un seul.

Une région terminale est généralement un espace aérien complexe, tel que montré par la figure 9.1 qui représente la région de Montréal [Nav Canada, 2008a]. Cette figure présente 4 aéroports (Mirabel, Pierre-Elliot Trudeau, St-Hubert et St-Jean), des balises de navigation ainsi que des voies aériennes standards de la région terminale de Montréal. Les lignes de couleurs sont des représentations partielles des routes aériennes standards d'arrivées, ou communément appelées STAR (*Standard Terminal Arrival Route*) de l'aéroport Pierre-Elliot Trudeau pour la piste 24 gauche [1]. Les STAR sont les routes d'arrivées prédéter-

1. Les pistes sont nommées par un nombre entier entre entre 01 et 36, qui est généralement un 1/10 du cap magnétique de la piste. Une lettre est ajoutée s'il y a plus d'une piste pointant dans la même direction.

minées par le système d'ATC. Ainsi, les plans de vol des routes d'arrivées sont souvent des STAR.

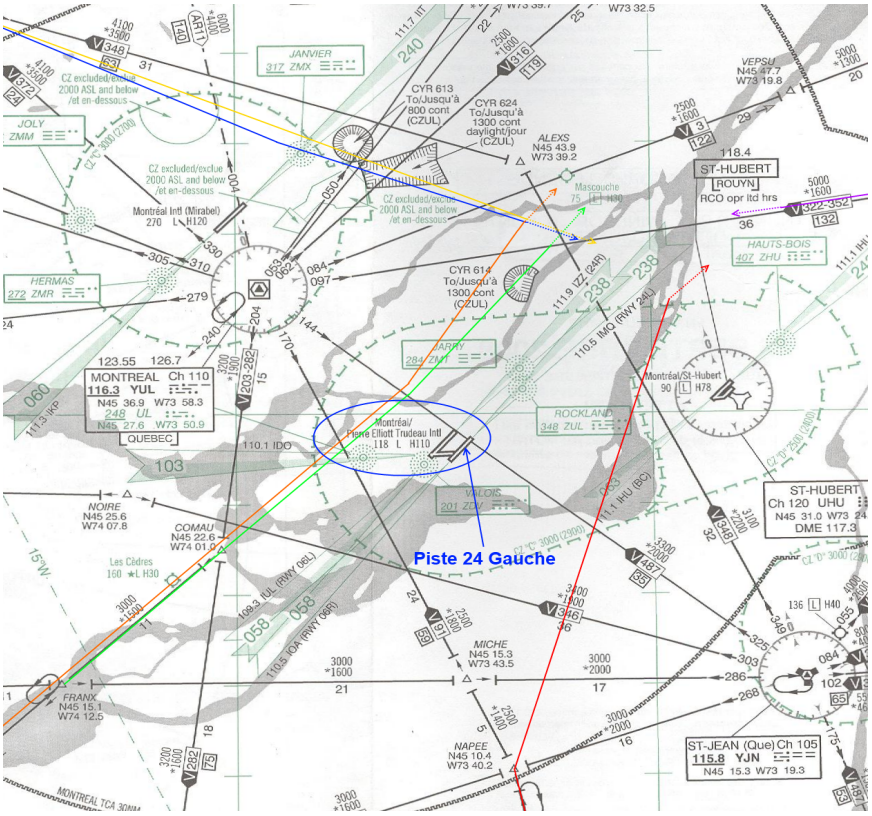


Figure 9.1 Carte IFR du terminal de Montréal.

Dans la figure de la région de Montréal, les STAR servent à aiguiller plus facilement le trafic provenant du sud, de l'est et de l'ouest de l'aéroport, facilitant ainsi le travail des contrôleurs d'arrivées. À Montréal durant les périodes de la journée où l'affluence est faible, comme la nuit, les avions peuvent voler en suivant les routes d'arrivées et ensuite se guider vers l'axe de piste en vue de l'atterrissage, et ce sans intervention des contrôleurs. Par contre, lorsque l'intensité du trafic est plus élevée les contrôleurs doivent aiguiller le trafic vers l'aéroport. Les contrôleurs font face à un problème d'entonnoir : des avions provenant de plusieurs directions qui désirent atterrir au même aéroport. Le contrôleur d'approche guide les avions en arrivée et détermine l'ordre dans lequel ils atterriront, d'où l'utilisation du terme séquençage. Cette intervention humaine est requise dans bien des circonstances, et ce malgré le fait que les avions modernes sont munis de systèmes avioniques sophistiqués permettant une navigation autonome jusqu'à l'interception de l'axe de piste en vue de l'atterrissage. Ainsi, les contrôleurs aériens vont fournir des instructions de virages, communément appelé guidage radar, aux avions pour les guider à intercepter l'axe

de piste. Le guidage radar effectué par le contrôleur en phase d'approche, ou séquençage du trafic en arrivée, est l'une des opérations radar la plus complexe à exécuter.

Un exemple de trajectoires pouvant être obtenues par guidage radar pour la région de Montréal est présenté à la figure 9.2. Le travail du guidage radar en approche consiste essentiellement à guider les avions pour qu'ils interceptent l'axe de la piste d'arrivée entre 8nm et 14nm du seuil de la piste tout en respectant la séparation minimale. Les lignes jaune, verte, bleue, rouge et mauve représentent les mêmes routes d'arrivées (STAR) que celle de la figure 9.1, alors que les lignes magenta illustrent des trajectoires de guidage radar. Les avions suivent les routes d'arrivées pour être ensuite guidés par les contrôleurs. La séparation minimale est généralement de 3nm en région terminale comparativement à 5nm pour un espace aérien de la phase de vol en route. Notons également que la distance normale d'interception est d'environ 12nm. En l'absence de trafic, un avion provenant du sud, de l'est, de l'ouest, interceptera l'axe de piste à une distance d'environ 12nm. Évidemment, le point d'interception variera pour chaque avion selon les conditions de trafic.

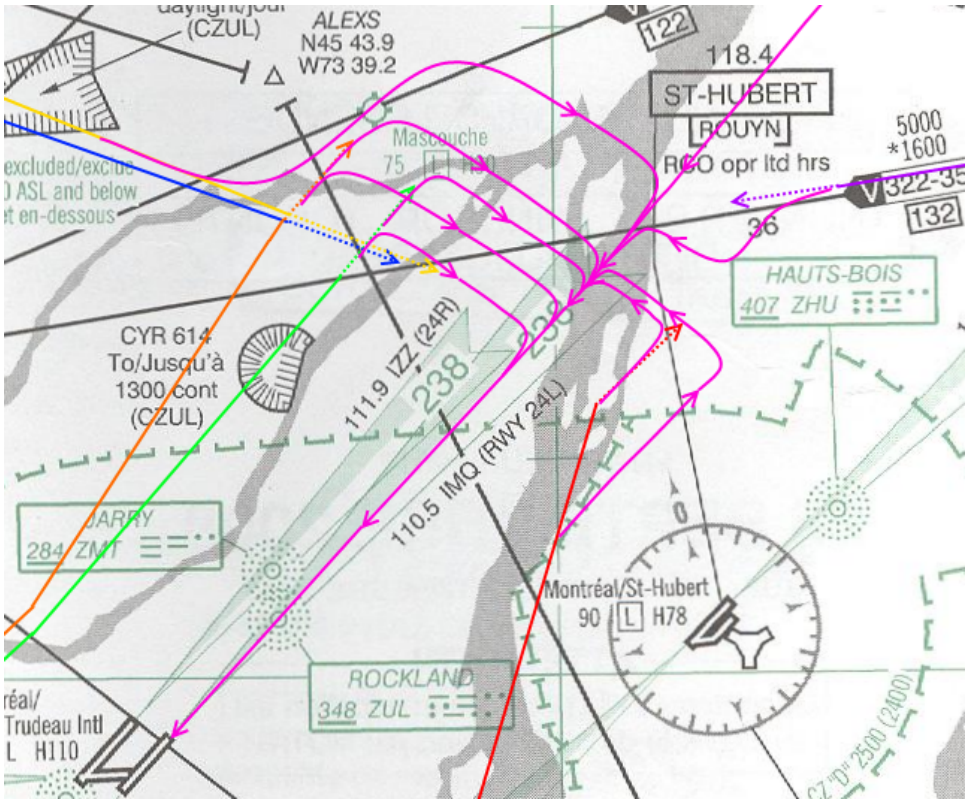


Figure 9.2 Exemple de guidage radar.

Une modélisation novatrice du guidage radar, pouvant être faite par un algorithme évolutionnaire ou stochastique, est présentée à la prochaine section.

9.2 Modélisation

L'auteur a fait plusieurs entrevues avec d'anciens contrôleurs aériens pour comprendre le fonctionnement du guidage radar. La modélisation de l'aiguillage du trafic IFR (vol aux instruments) en arrivée a été inspirée par la façon dont les vols VFR (vol à vue) en arrivée sont traités par les contrôleurs et par la façon dont le guidage radar réel est fait. La solution proposée est très proche de la façon dont les contrôleurs d'approche effectuent cet aiguillage.

Les avions VFR en arrivée (désirant atterrir) doivent intégrer le circuit de la piste d'arrivée. Le circuit est en fait un ensemble de segments devant être suivis pour intercepter l'axe final de la piste en vue d'atterrir. La figure 9.3 illustre un circuit VFR gauche où tous les virages doivent se faire par la gauche [Transport Canada, 2009]. Selon cette figure, un pilote dans un circuit VFR suivra les segments *vent arrière*, *base* et *finale*. Un pilote seul dans le circuit tournera en *vent de travers* lorsque son altitude sera de 1000 pieds au-dessus de l'aérodrome, en *vent arrière* à une distance d'environ 1nm (évalué à vue) et en *base* lorsque l'avion croisera une ligne fictive de 45 degrés (voir la figure 9.3).

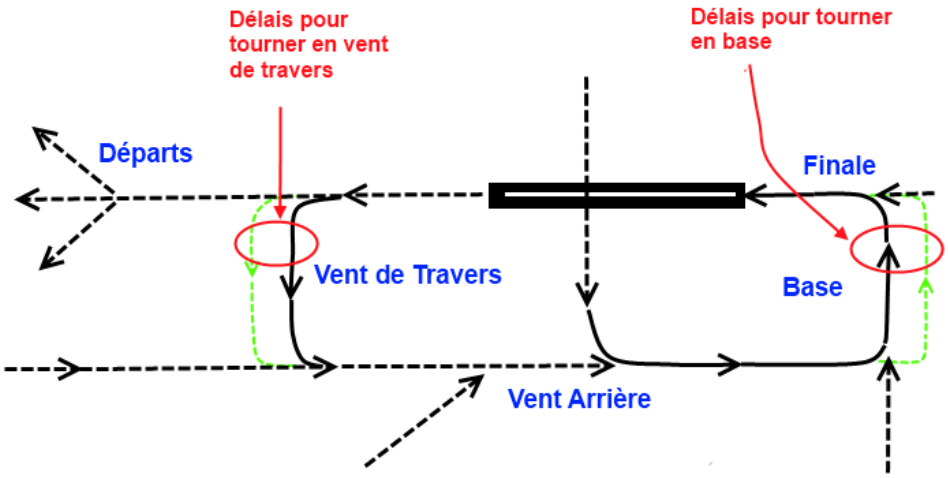


Figure 9.3 Circuit VFR

Contrairement au guidage radar IFR, c'est au pilote qu'incombe la responsabilité de tourner au moment opportun dans un circuit VFR. Par contre, lorsqu'il y a plusieurs avions à proximité de l'aéroport, un pilote peut recevoir des instructions de la part d'un contrôleur aérien sur le moment pour joindre le prochain segment. Par exemple, un avion

en *départ* suivant l'axe de piste pourrait se faire demander d'attendre avant de tourner en vent traversier pour permettre à un autre avion de joindre le circuit dans le segment *vent arrière*. De la même façon, un avion en vent arrière pourrait devoir allonger son vent arrière pour permettre à un avion de décoller. Les lignes vertes tiretées représentent l'effet de l'allongement des segments de départ et *vent arrière*. Notons une similarité avec le changement de la distance d'interception de l'axe de piste dans le guidage radar.

Manoeuvre de séquençage

Un nouveau modèle de manoeuvre a été développé pour la problématique du séquençage d'avions en arrivée, plus spécifiquement dans la phase d'approche. Il consiste à modifier une trajectoire de référence, et ce de façon similaire à la manoeuvre du point tournant, présentée à la section 7.4. Par contre, contrairement au point tournant les trajectoires de références sont composées de segments de droite, permettant ainsi de créer des trajectoires plus complexes, tel qu'illustré à la figure 9.4. Elle illustre, de manière similaire à la figure 2.2 (page 14), des avions en arrivée ainsi qu'une série de trajectoires nominales. Ces trajectoires illustrent les parcours nominaux que suivront les avions en arrivée pour le problème de guidage radar en phase d'approche. Il s'agit d'une représentation simplifiée d'une image d'un radar d'une zone terminale.

Les trajectoires d'arrivées modélisées sont composées de segments, dont les principaux sont : vent arrière gauche et vent arrière droit, base gauche et base droite et finale. Par exemple, un avion situé du côté droit de la piste se déplacera en vent arrière gauche, puis base gauche et en finale. Notons que les segments *de gauche* se situent sur le côté droit de la figure alors que les segments *de droite* se situent du côté gauche. Dans les segments de gauche l'avion tourne vers la gauche et vers la droite pour les segments de droite. Par exemple, un avion en vent arrière gauche tournera en base gauche toujours au même endroit. On suppose également que tous ces avions doivent atterrir sur la même piste. Ces trajectoires, qui sont fragmentées en quelques segments, décrivent les parcours employés par les avions sans tenir compte des autres avions. Par exemple, un avion initialement en vent arrière gauche allongé, évoluera en vent arrière gauche, puis en base gauche et terminera en finale.

Sur la figure 9.4 les cercles bleus pâles sont des points de contrôle servant à adapter les trajectoires alors que les cercles noirs sont des points fixes. Les points de contrôle doivent être placés à des endroits stratégiques appropriés au problème traité. Dans la présente situation, ces points servent à changer la longueur des segments *vent arrière* et des segments *base* pour un avion donné. Par exemple, il pourrait y avoir une perte de

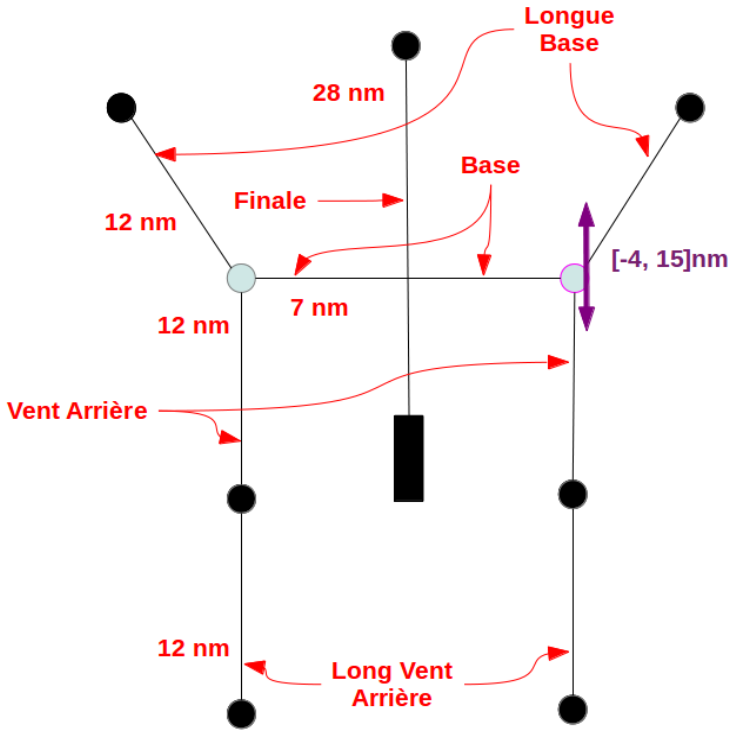


Figure 9.4 Trajectoires d'arrivées simplifiées avec 2 points de contrôle.

séparation (ou collision face à face) dans une situation ayant deux avions en *vent arrière gauche et droit* à une même distance des segments *base*. Une solution potentielle serait d'allonger le segment *vent arrière* d'un des deux avions, évitant ainsi une situation de face à face. Notons que les trajectoires de cette figure ont été inspirées des procédures actuellement employées par les contrôleurs d'approche.

Cette nouvelle modélisation a été développée tout d'abord pour résoudre le problème de séquençage des avions en arrivée. Bien que ce modèle soit différent de celui détaillé au chapitre 7, il peut également être employé dans la résolution de problèmes en route, comme présenté à la section 9.3.3.

La figure 9.5 illustre un exemple simple d'une trajectoire altérée (ligne tiretée) par le déplacement de son point de contrôle selon une variation en x et en y. Cette simple trajectoire aurait pu par exemple être une manoeuvre alternative au problème de collision face à face présenté à la section 8.2.

Dans ce modèle, les trajectoires sont modifiées par le déplacement des différents points de contrôle. Chaque point peut se déplacer selon l'axe x ou l'axe y, selon les degrés de liberté (DDL) associés à chacun de ces points. La définition des DDL est propre à un problème donné et elle fait partie de l'algorithme de résolution. Théoriquement, il n'y a

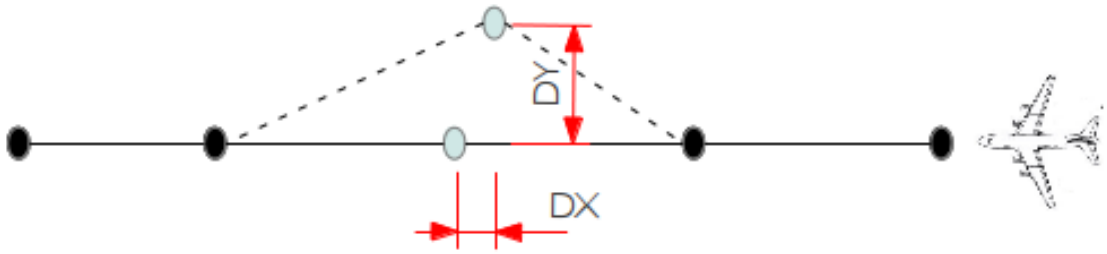


Figure 9.5 Déplacement d'un point de contrôle.

pas de limite sur le nombre de points de contrôle par trajectoire. Cependant, ce nombre affecte grandement la performance de l'algorithme d'optimisation, tant pour le temps de calcul par itération (génération) que pour le temps de convergence. Un choix judicieux est donc important.

Contrairement au point tournant, les variables de contrôle ne sont pas une indication de virage (ou de cap) mais le déplacement d'un point de contrôle. Malheureusement le déplacement d'un point d'une route ne constitue pas une instruction ATC valide. Il serait par contre possible de traduire le déplacement d'un point de contrôle en indication de virage. Cette traduction n'a pas été étudiée, car elle n'influe en rien la validité et la pertinence du modèle et des résultats présentés. Il serait par contre intéressant de l'analyser dans une étude ultérieure.

Selon le modèle ATC actuel, le déplacement d'un point sur route ne constitue pas une instruction fournie par un contrôleur aérien en guidage radar. Bien entendu, le déplacement d'un point pourrait également se convertir à une ou plusieurs instructions de virage similaire à celle définie au chapitre 7, précisant le moment du virage ainsi que la direction et l'amplitude.

9.3 Résultats numériques de séquençage

Le guidage radar dans la phase approche peut comporter des instructions de changement de direction, de vitesse et d'altitude. Les changements d'altitude et de vitesse servent principalement à faire descendre les avions et ralentir les avions en vue de l'atterrissage. Le guidage proprement dit s'effectue à l'aide d'instructions de changement de direction. Notons que c'est ce guidage qui est relativement complexe. Tous les exemples présentés sont en deux dimensions, c'est-à-dire que les avions évoluent dans le même plan XY pour deux raisons. La première étant pour employer des instructions similaires à ce qui est

fourni par les contrôleurs et la seconde pour montrer la pertinence de la manoeuvre à guider les avions.

Les sous-section 9.3.1 et 9.3.2 présentent des situations de séquençage à 8 et 12 avions respectivement. Ces situations sont représentatives d'aéroports ayant une densité de trafique moyennement élevée.

9.3.1 Séquençage de 8 avions en approche

Contrairement à la résolution de conflits, le séquençage consiste à guider les avions pour les aligner vers l'axe de la piste d'arrivée dans le but de les faire atterrir. Ce guidage doit nécessairement se faire en évitant toute perte de séparation, ce qui rend le problème beaucoup plus complexe. Une situation de complexité moyenne comportant huit avions a été analysée comme premier exemple de séquençage, tel qu'illustré à la figure 9.6. Les longueurs des segments *base*, *vent arrière* et *finale*, que parcourent les avions, sont telles qu'indiquées à la figure 9.4. Sur cette figure, les lignes rouges représentent les trajectoires nominales que suivront les avions, alors que les nombres identifient les différents avions. Bien que cette situation comporte uniquement huit avions, elle demeure tout de même assez complexe à résoudre, car les avions doivent converger vers le même endroit, qui est le segment final.

Le tableau 9.1 présente les spécifications du problème. Ce problème est traité en deux dimensions (2D), car tous les avions sont à la même altitude. De plus, par simplicité tous les avions évoluent à la même vitesse. Cette seconde hypothèse est conforme à ce qui est employé par les contrôleurs aériens. Tel que mentionné à la section 7.4, des restrictions de vitesses de 250 kts et 200 kts sont imposées aux avions sous 10000 pieds et sous 3000 pieds respectivement. Le traitement de cette situation en deux dimensions est conforme à ce qui est généralement fait par les contrôleurs aériens. En effet, dans une région d'approche, ou terminale, les avions sont guidés à la même vitesse et à la même altitude, et ce sans égard aux types d'avions [Robinson *et al.*, 1997].

Deux contraintes sont présentées dans le tableau de paramètres. La première étant le respect d'une séparation minimale de 3 nm, alors que la seconde indique que l'avion en finale ne peut pas recevoir d'instruction de changement de trajectoires. Cette seconde contrainte est causée par la manoeuvre de résolution pour ce problème. Les changements de trajectoires se font uniquement en changeant la position des points de virage en base. L'avion qui est en finale ne peut donc pas être affecté par un de ces points.

Dans le tableau de paramètres, les plages de ΔX et ΔY indiquent les variations possibles de la position des deux points de contrôle. Notons que ΔX est toujours 0, car les points de contrôle de cet exemple peuvent bouger uniquement dans la direction Y, c'est-à-dire dans l'axe du segment final. Une valeur négative de ΔY indique un déplacement d'un point de contrôle vers la piste, alors qu'une valeur position indique un déplacement en éloignement de la piste. Par exemple pour $\Delta Y = -4$, un avion tournerait en base à 8 nm de la piste contrairement à 12 nm lorsqu'il n'y a aucune variation. Un avion ne peut pas être tourné en base à une distance inférieure de 8 nm de la piste. Ces plages de valeurs sont conformes à ce qui est employé par les contrôleurs aériens.

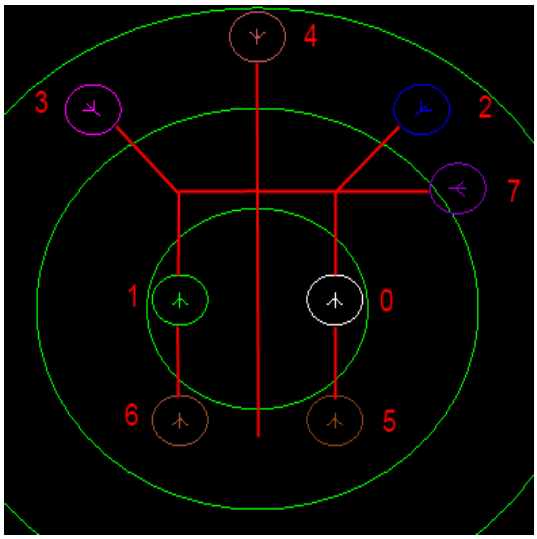


Figure 9.6 Situation initiale du séquençage de huit avions.

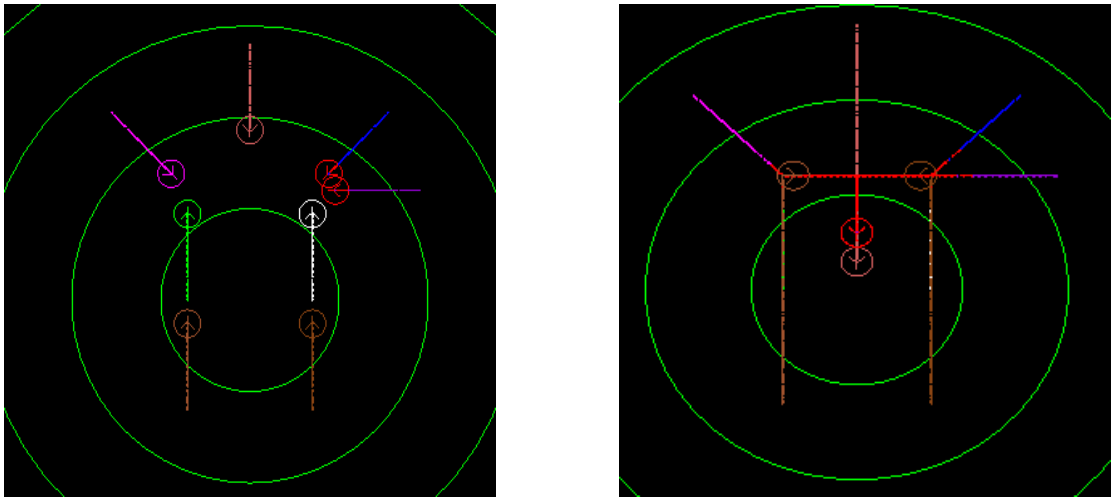
La figure 9.7 (a et b) illustre le déplacement des avions à 215 et 560 secondes respectivement sans déplacement des points de contrôle (sans instruction ATC) de la situation initiale présentée à la figure 9.6. Ainsi, chaque avion suivra sa trajectoire nominale, ce qui engendrera nécessairement des pertes de séparation. La perte de séparation entre deux avions est indiquée par le changement de couleur au rouge des enveloppes de protection et des trajectoires pointillées. Les cercles verts [2] sont employés à titre de référence visuelle permettant d'obtenir une certaine interprétation de la distance. Le plus petit cercle est de 10 nm et la différence de rayon entre deux cercles consécutifs est également de 10 nm.

Les figures 9.8 a) à d) illustrent l'état d'une résolution à différents temps dans la résolution. Il est intéressant de voir comment les avions sont déviés de leurs trajectoires initiales dans l'opération de séquençage. Les nombres sur ces figures servent à identifier les avions, tel qu'il a été présenté à la figure 9.6. La figure 9.8 d) illustre les avions une fois le séquençage

2. Appelés *Range Rings* dans le jargon ATC.

Tableau 9.1 Paramètres de résolution pour le séquençage de 8 avions en approche.

Taille de la population	25
Taille de l'archive	15
Taux de croisement	0.2
Taux de mutation	0.1
Nombre maximal de générations	300
Temps total de résolution	2000 sec
Vitesse des avions	160 kts
Plage de ΔX	[0] nm
Plage de ΔY	$[-4, 15]$ nm
Objectifs :	
Variation de l'ETA	
Variation de la trajectoire dans le plan horizontal	
Nombre total de commandes	
Contraintes :	
Perte de séparation	$R = 3$ nm
Avion en finale ne peut être affecté	

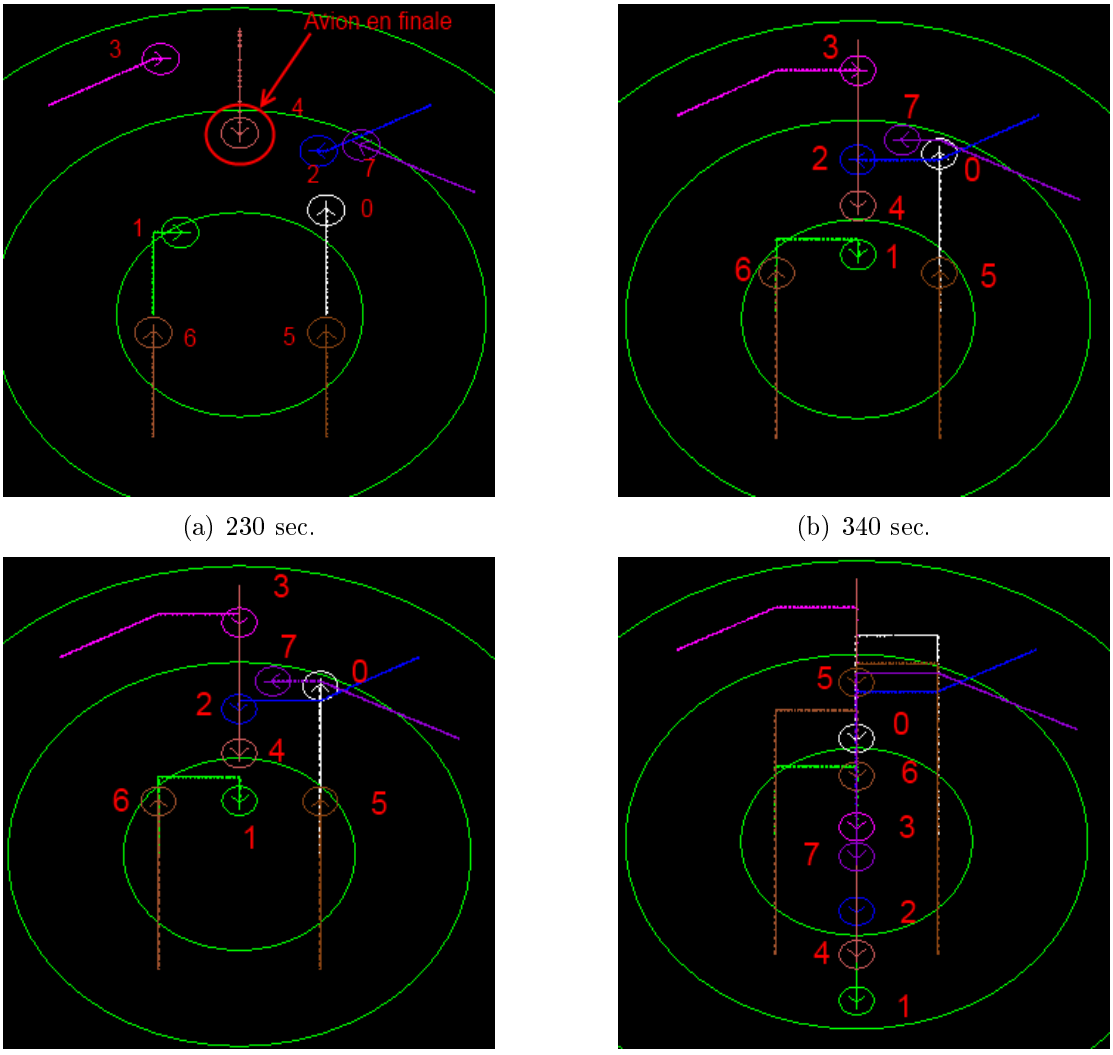


(a) 215 sec. (b) 560 sec.

Figure 9.7 Huit avions en approche, sans résolution de séquençage.

terminé. On remarque que tous les avions sont alignés avec l'axe de piste, et ce tout en respectant la séparation minimale. L'avion 4, qui est déjà aligné correctement avec l'axe de la piste ne subit aucun changement de trajectoire.

Le tableau 9.2 (page 142) présente le déplacement des points de contrôle pour chaque avion de la situation. Les nombres de la première colonne du tableau identifient les avions, tel que montré à la figure 9.6. La colonne *Point* indique le point de contrôle qui a été déplacé pour chaque avion. Le nombre 1 étant le point bleu gauche de la figure 9.4 alors que le



(a) 230 sec.

(b) 340 sec.

(c) 420 sec.

(d) 850 sec.

Figure 9.8 Séquençage de 8 avions en approche, à différents temps dans la résolution.

Tableau 9.2 Solution au problème de séquençage.

Avion	Point	$\Delta X(nm)$	$\Delta Y(nm)$
0	2	0	10
1	1	0	-4
2	2	0	4
3	1	0	13
4	–	–	–
5	1	0	7
6	1	0	2
7	2	0	6

Tableau 9.3 Statistiques : 100 essais, 8 avions, 2D, (SPEA-MOD).

ΔX moyen :	0.00 nm
ΔY moyen :	5.79 nm
Nombre de commandes moyennes par solution :	6.77
Génération moyenne pour une solution réalisable :	45
Nombre de solutions réalisables :	98/100

nombre 2 est le point bleu droit. Finalement, les colonnes ΔX et ΔY sont les variations de positions des différents points de contrôle. On remarque que les valeurs de ΔY sont conformes à la plage de valeurs spécifiées au tableau 9.1.

Le tableau 9.3 (page 142) montre les résultats moyens réalisés pour 100 essais. L'algorithme a obtenu des solutions réalisables à 98 des 100 essais réalisés.

9.3.2 Séquençage de 12 avions en approche

Cette sous-section présente un second exemple de la résolution d'un problème de séquençage en arrivée. Cet exemple est constitué de douze avions, comparativement à huit avions pour l'exemple présenté à la section 9.3.1.

Nonobstant le plus grand nombre d'avions traité dans cet exemple, sa particularité est liée au nombre de points de contrôle par trajectoire. La figure 9.4 (page 136) comportait un maximum d'un seul point de contrôle par trajectoire. La figure 9.9 illustre les trajectoires modélisées de cet exemple. Comparativement à la figure 9.4, on retrouve deux points de contrôle et quatre segments supplémentaires.

Le tableau 9.4 présente une solution de ce problème de séquençage à 12 avions. Il est à remarquer que les avions 9 et 11 ont deux points de contrôle dans chacune de leurs trajectoires respectives. Sans ces deux points additionnels, l'algorithme est incapable de

trouver une solution viable au problème à cause de la complexité accrue de cet exemple. La figure 9.10 illustre la résolution à différents temps.

L'ajout de points de contrôle à des endroits spécifiques permet de traiter une gamme variée de problèmes du contrôle aérien, tant pour la phase d'arrivée que pour la phase en route.

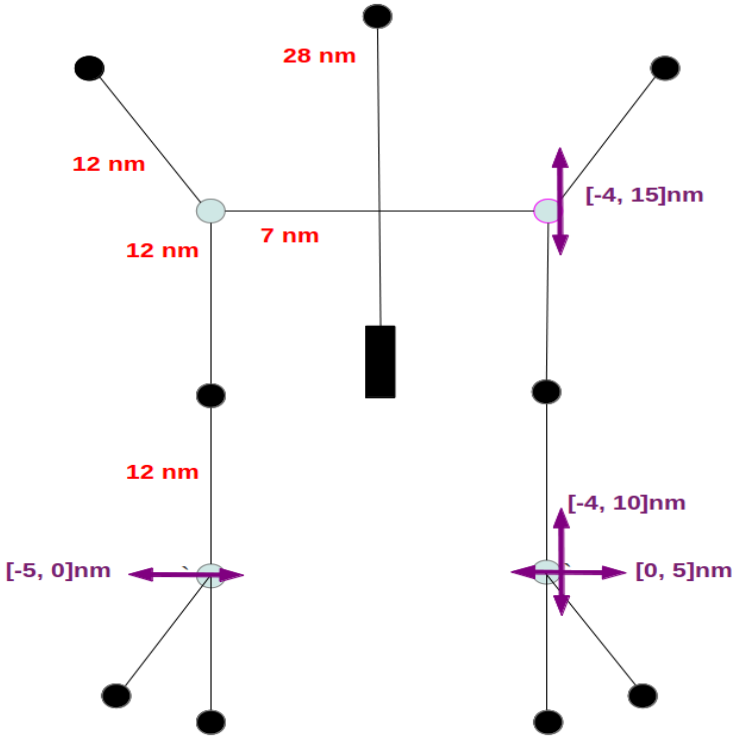


Figure 9.9 Trajectoires d'arrivées simplifiées avec 4 points de contrôle.

9.3.3 Trajectoires croisées

Bien que les manoeuvres présentées au chapitre 7 sont efficaces, elles comportent un désavantage important lorsque les routes ne sont pas qu'un seul segment de droite. La figure 9.11 présente une sous-section de la carte *En Route Niveau Inférieur de l'Atlantique Nord* [Nav Canada, 2008b]. On y retrouve plusieurs routes standards, appelées des routes *Victor*, par exemple entre Saint-Jean et Québec il y a la *Victor* 98 (V98) et entre Saint-Jean et la Beauce, il y a la *Victor* 346 (V346). Ces deux routes sont composées uniquement d'un segment. Par contre, la route *Victor* 98 entre Burlington et Québec est formée de deux segments, ce qui rend impossible d'y utiliser le modèle de manoeuvre développé.

Sur des routes à plus d'un segment, une manoeuvre en forme de triangle isocèle ne serait pas toujours possible.

Tableau 9.4 Solution au problème de séquençage (12 avions).

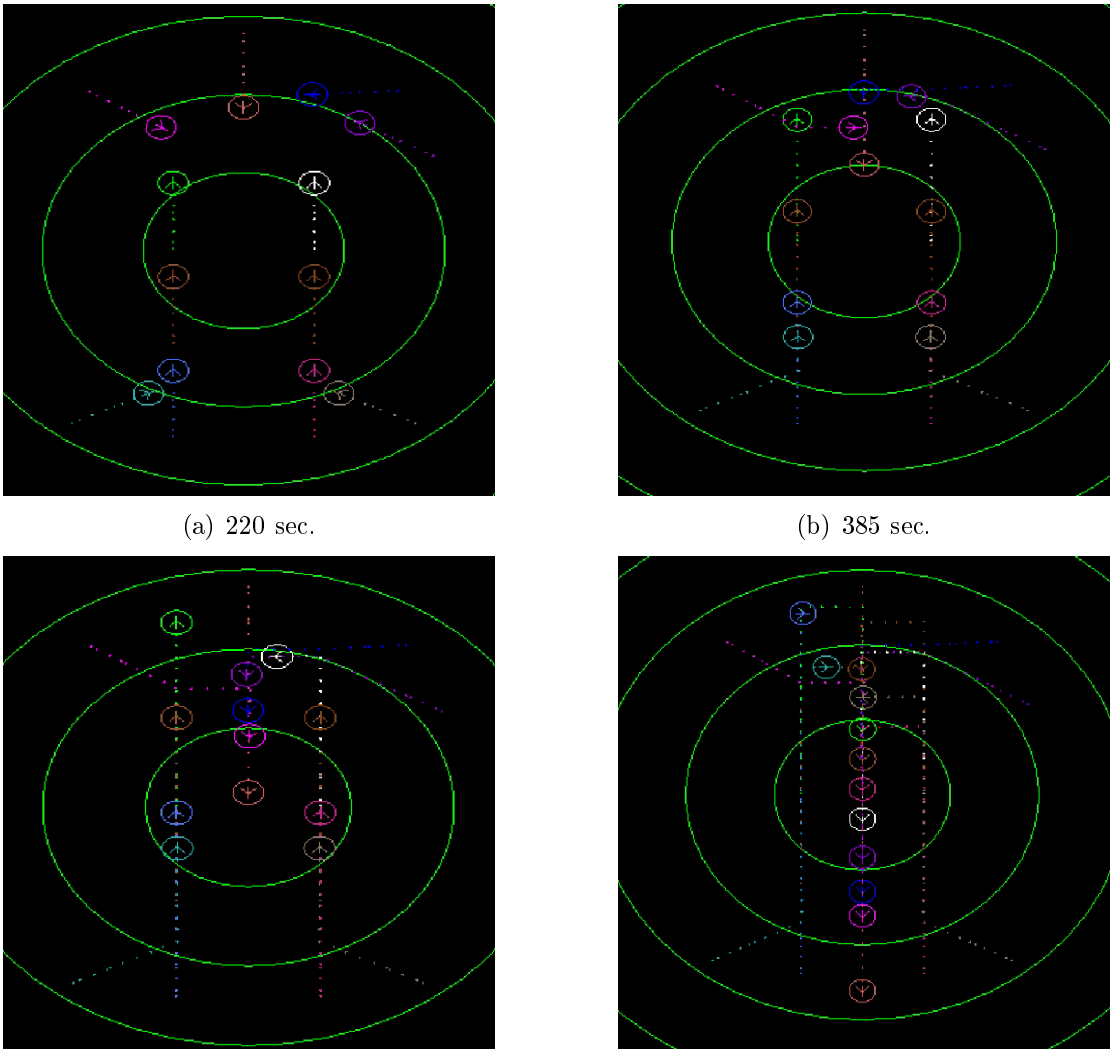
Avion	Point	$\Delta X(nm)$	$\Delta Y(nm)$
0	2	0	-1
1	1	0	12
2	2	0	12
3	1	0	3
4	–	–	–
5	1	0	-3
6	1	0	4
7	2	0	10
8	2	0	10
9	2	0	-4
	2	0	1
10	2	0	7
11	2	0	4
	2	0	3

Un problème de résolution de conflits de deux flux se croisant a été étudié pour démontrer que les manoeuvres proposées dans le présent chapitre peuvent également s'appliquer à des problèmes autre que le séquençage en arrivée.

Le problème de trajectoires se croisant a été étudié par Mao et ses collaborateurs [Huang *et al.*, 2012; Treleaven et Mao, 2008], tel que présenté à la section 2.3. L'adaptabilité de l'algorithme proposé dans ce chapitre est démontrée par la résolution d'un problème impliquant deux trajectoires croisées.

Dans l'espace en route, bon nombre de trajectoires entre 2 avions et plus se croisent. Présentement, la technique employée par les contrôleurs pour assurer une séparation est d'assigner des altitudes différentes selon la direction (cap) des flux de trafic. Normalement, pour les caps entre 360 et 179 degrés une altitude paire est donnée (ex : 4000', 6000', etc.) alors qu'une altitude impaire (ex : 3000', 5000', etc.) est assignée pour les caps compris en 180 et 359 degrés [Canada, 1999]. Cette assignation d'altitude permet d'assurer une séparation uniquement pour des avions volant dans des directions opposées. Généralement les changements de direction sont plus économiques que les changements de d'altitude.

La figure 9.12 illustre deux trajectoires se croisant à un angle de 45 degrés dont l'intersection des deux trajectoires se situe au centre des cercles. Selon la règle d'assignation d'altitude de l'ATC, ces avions seraient tous à la même altitude. Sans instruction de changement de direction, aucun avion ne respecte la séparation minimale. Ce problème a été résolu uniquement par un algorithme génétique SPEA-MO, car le but était de démon-



(a) 220 sec. (b) 385 sec.

(c) 525 sec. (d) 1115 sec.

Figure 9.10 Séquençage de 12 avions en approche, à différents temps dans la résolution.

trer l'adaptabilité de la manoeuvre de résolution présentée dans le présent chapitre. De plus, aucun changement d'altitude n'a été émis, augmentant ainsi la difficulté du problème. Le tableau 9.5 illustre les paramètres de résolution employés.

Les figures 9.13 a) et b) illustrent une résolution à 1600 et 2800 secondes respectivement.

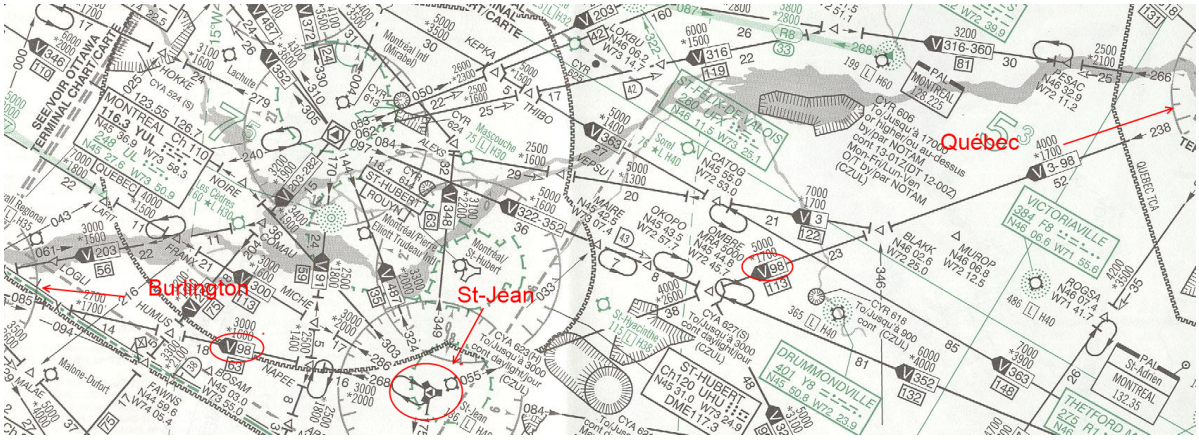


Figure 9.11 Carte en route de niveau inférieur.

Tableau 9.5 Paramètres de résolution pour deux flux d'avions croisés.

Taille de la population	25
Taille de l'archive	15
Taux de croisement	0.2
Taux de mutation	0.1
Nombre maximal de générations	300
Temps total de résolution	6000 sec
Vitesse des avions	250 kts
Distance entre avions	20 nm
Plage de ΔX	$[-20, 20]$ nm
Plage de ΔY	$[-20, 20]$ nm
Objectifs :	
Variation de l'ETA	
Variation de la trajectoire dans le plan horizontal	
Nombre total de commandes	
Contraintes :	
Perte de séparation	$R = 5$ nm

9.4 Considérations futures

Il serait intéressant d'aborder le respect des procédures d'atténuation de bruit dans une recherche ultérieure[3] [Canada, 1999]. Ces procédures obligatoires, spécifiques à chaque aéroport, fournissent des instructions aux pilotes pour minimiser l'émission de bruit des avions au-dessus des zones urbaines. Le respect de ces procédures n'est pas abordé dans le cadre de cette recherche puisqu'elles sont principalement utilisées pour les vols en départ. Les restrictions imposées sont, par exemple, un taux de montée maximal, l'interdiction

3. Noise Abatement Procedure

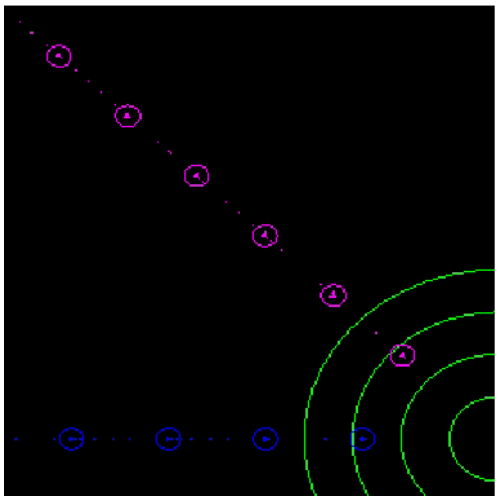
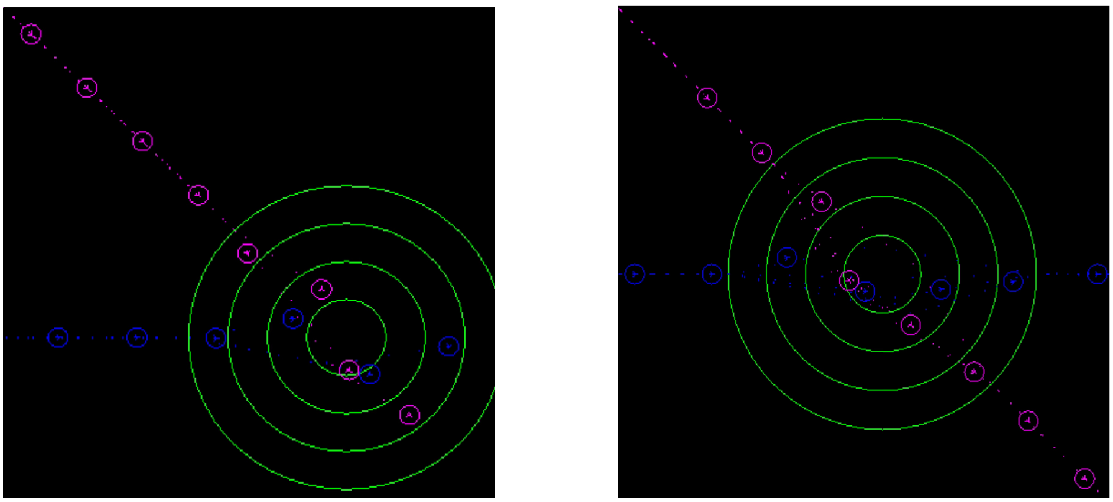


Figure 9.12 Croisement de deux trajectoires d'avions, sans résolution.



(a) 1600 sec. (b) 2800 sec.

Figure 9.13 Croisement de deux trajectoires d'avions, à 1600 et 2800 secondes.

de virage avant qu'une certaine altitude soit atteinte. etc. Ces procédures pourraient être modélisées par le type de manoeuvre définie dans le présent chapitre.

Dans le cadre de ce présent projet de recherche, l'espace aérien était composé d'un seul secteur géré par un contrôleur, éliminant ainsi la complexité liée à la coordination entre les différents secteurs. Cette simplification n'affecte pas la validité de la solution proposée. Par contre, il serait intéressant d'étudier l'effet de multiples secteurs sur la résolution. Dans un environnement ayant plusieurs secteurs, l'auteur suggère d'utiliser un moteur de résolution de conflits par secteur. La coordination entre les différents secteurs se traduirait comme des conditions initiales et des contraintes supplémentaires.

9.5 Sommaire

Une nouvelle technique novatrice permettant de séquencer des avions en approche de façon optimale a été présentée. Cette technique emploie l'algorithme génétique multiobjectif SPEA-MOD qui a été présenté au chapitre 6 (page 57) ainsi qu'un nouveau type de chromosome permettant de modéliser des trajectoires et des manoeuvres complexes d'avions. Rappelons qu'un chromosome, ou individu, représente une solution potentielle d'un problème résolu par un algorithme évolutionnaire.

Les trajectoires, définies pour un problème donné, sont composées d'une suite de points qui seront suivis par un avion donné. Il y a deux types de points possibles, fixes et de contrôle. Les points fixes ne peuvent pas bouger, alors que les points de contrôle peuvent selon une certaine plage de valeurs. Ainsi, les variables du problème d'optimisation sont le déplacement de ces points de contrôle.

La résolution de deux problème de séquençage d'avions en arrivée, à huit et douze avions respectivement, a été présentée. Le premier exemple comportait au maximum un point de contôle par trajectoire d'avion, alors que dans le second certaines trajectoires en comportaient deux. Le nombre supplémentaire de points de contrôle du deuxième exemple a permis de résoudre une situation de séquençage ayant plus d'avions que dans le premier. Ces deux exemples ont montré l'adaptabilité de l'algorithme proposé.

Cette modélisation qui a tout d'abord été développée pour le séquençage d'avions en arrivée, peut également résoudre des situations conflictuelles de la phase de vol en route, et ce tel qu'il a été démontré par la résolution d'une situation ayant deux flux de trafic se croisant.

CHAPITRE 10

CONCLUSION

Ces travaux de recherches se sont se penchés sur la problématique de la résolution de conflits dans le séquençage des avions dans les phases de vol en route et d'arrivée, respectivement. Deux domaines connexes de l'aviation bénéficient des contributions de cette recherche, soit l'entrainement de pilotes et la gestion du trafic aérien (ATC).

Une avenue prometteuse dans la formation de pilotes est l'utilisation d'un environnement aéroportuaire simulé, permettant ainsi d'avoir un environnement de formation plus près de la réalité. Une des facettes de cet environnement est la simulation de contrôleurs aériens, qui fournissent des instructions ATC (*Air Traffic Control*) aux pilotes virtuels et réels. À ce jour, aucun environnement ATC virtuel ne simule les tâches de résolution de conflits et de séquençage. Quant aux outils d'aide à la décision, ils facilitent le travail des contrôleurs en les assistant dans diverses tâches complexes, telles que la résolution de conflits et le séquençage en arrivée. Rappelons que le but de ces travaux n'est pas de changer la structure de fonctionnement de l'ATC, mais bien d'en modéliser le comportement.

La résolution de conflits et le séquençage d'avions sont des problèmes combinatoires très complexes. Plusieurs études ont employé des techniques analytiques pour résoudre ces problèmes. Bien que les méthodes analytiques soient intéressantes, elles ne permettent de résoudre qu'un nombre très restreint de problèmes spécifiques. L'auteur a choisi l'utilisation de techniques d'optimisation multiobjectif basées sur des algorithmes évolutionnaires, et ce dans le but de résoudre une plus grande variété de problèmes complexes.

Trois contributions originales sont le résultat de cette thèse. La première contribution de cette thèse est la conception de deux nouveaux algorithmes évolutionaires multiobjectifs pouvant traiter des contraintes. Le premier est un algorithme génétique nommé SPEA-MOD, alors que le second est un algorithme de colonies de particules nommé PSO-MO. Il a été démontré que ces deux algorithmes obtiennent de très bons résultats sur des problèmes classiques d'analyse. Les résultats de cette comparaison indiquent que PSO-MO produit une meilleure approximation du front de Pareto pour des problèmes faiblement contraints, alors que SPEA-MOD performe mieux sur des problèmes fortement contraints. Ce résultat a également été confirmé lors de l'analyse de résolution de conflits.

La seconde contribution de cette thèse est l'automatisation de la résolution de conflits, plus spécifiquement de la phase en route, par un algorithme évolutionnaire multiobjectif. Selon la revue de la littérature, il semble que ce soit la première fois que la problématique de la résolution de conflits soit solutionnée par un algorithme d'optimisation multiobjectif. Les trois principes fondamentaux du contrôle aérien, soit la *sécurité*, l'*ordre* ainsi que l'*efficacité*, ont été modélisés par des objectifs et des contraintes d'un problème d'optimisation. Cette modélisation permet d'inclure les différents critères d'opération ATC dans la résolution d'une situation conflictuelle. De plus, la résolution peut s'effectuer en deux ou en trois dimensions, selon ce qui est spécifié.

L'algorithme SPEA-MOD a également mieux performé que l'algorithme PSO-MO dans les problèmes de résolution de conflits relativement complexes. Les résultats ont montré que SPEA-MOD permet d'obtenir un plus grand nombre de solutions réalisables que PSO-MO. Par exemple, dans une situation de la phase en route ayant 11 avions, SPEA-MOD a obtenu des solutions réalisables à 85 des 100 essais effectués, comparativement à 56 pour l'algorithme PSO-MO. De plus, un plus grand nombre de solutions de SPEA-MOD dominent les solutions de PSO-MO, en faisant une meilleure technique pour les problèmes de résolution de conflits.

La troisième contribution est la modélisation du séquençage d'avions en arrivée par un algorithme génétique multiobjectif. Aucune étude ne semble avoir abordé cette problématique de façon optimale. Il est à noter que le séquençage est l'une des tâches les plus difficiles du contrôle aérien. Pour ce faire, une nouvelle modélisation de manoeuvres a été mise au point. Cette manoeuvre consiste à déplacer au moins un point d'une trajectoire de référence, et ce pour chaque avion impliqué dans le séquençage. Il a été montré que la manoeuvre proposée peut également être employée pour des situations de résolution de conflits complexes.

Rappelons qu'un autre avantage des algorithmes présentés est qu'aucune modification de l'avionique des avions ou des procédures de contrôle aérien n'est requise, ce qui est important pour une mise en oeuvre réelle.

Les deux objectifs fondamentaux de cette thèse ont été atteints. Le premier était de simuler les tâches de résolution de conflits d'avions dans la phase de vol en route et du séquençage d'avions en arrivée. Le second objectif quant à lui était la conception d'une architecture logicielle générique pour simuler les deux tâches. L'architecture développée peut s'adapter facilement à des problèmes différents de résolution de conflits et de séquençage d'avions.

Finalement, cette thèse apporte réponse à la question mentionnée au chapitre 1 : *Est-il possible de modéliser et d'implémenter la résolution de conflits et le séquençage d'avions de manière optimale afin de mieux modéliser le comportement de contrôleurs aériens ?* Tel que présenté aux chapitres 7 et 9, les algorithmes de résolutions de conflits et de séquençage d'avions sont basés sur des algorithmes évolutionnaires multiobjectifs, permettant de prendre en compte des aspects du contrôle aérien. Plusieurs solutions à des problèmes complexes ont été présentées aux chapitre 8 et 9, démontrant ainsi qu'il est possible de d'automatiser ces tâches de manières optimales.

Ouverture pour de prochaines recherches

Les techniques développées pourraient apporter des améliorations aux outils d'aide à la décision employés par les contrôleurs aériens. Une question qui serait intéressante lors de prochaines recherches est la suivante : *Comment devrait-on présenter les résultats d'un problème de résolution de conflits ou de séquençage ?* Une étude ergonomique de ces différents paramètres apporterait sans aucun doute des solutions prometteuses.

Dans un projet ultérieur, il serait intéressant d'analyser l'impact d'un modèle cinématique plus complet. Par exemple, les changements de direction ne devraient plus se faire de façon instantanée. Quel en serait l'effet sur le temps de calcul et la complexité de l'algorithme ?

Un autre point d'analyse touchant le réalisme, serait l'intégration des algorithmes développés dans un simulateur ATC et d'en étudier le réalisme des solutions de résolution de conflits et de séquençage dans un environnment plus évolué.

ANNEXE A
RÉSULTATS D'ESSAIS SPEA ORIGINAL

Cette annexe présente les résultats de l'algorithme SPEA original sur les problèmes types présentés à la section 6.4. Tous les essais ont été effectués avec caractéristiques mentionnées à la section 6.6.

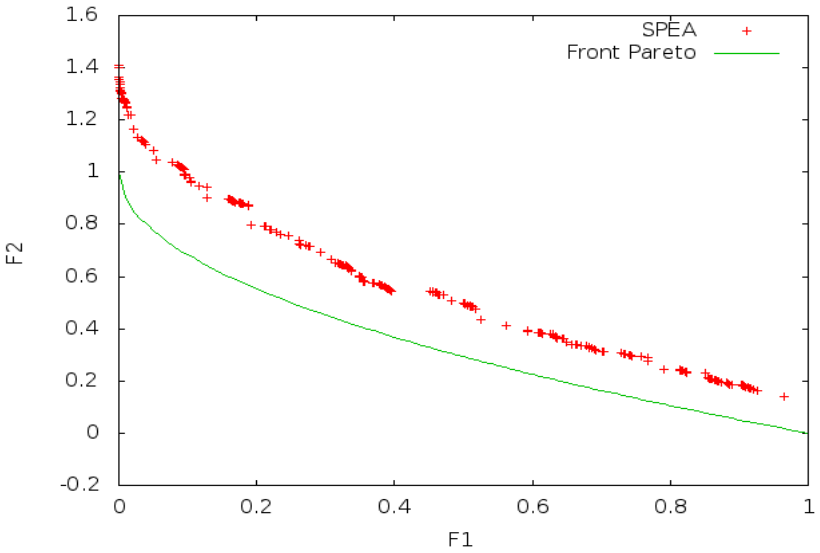


Figure A.1 SPEA - ZDT1

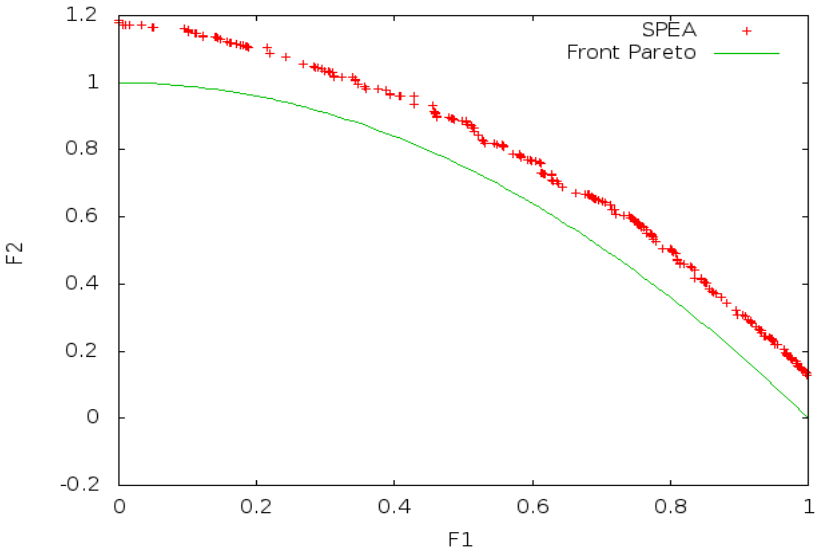


Figure A.2 SPEA - ZDT2

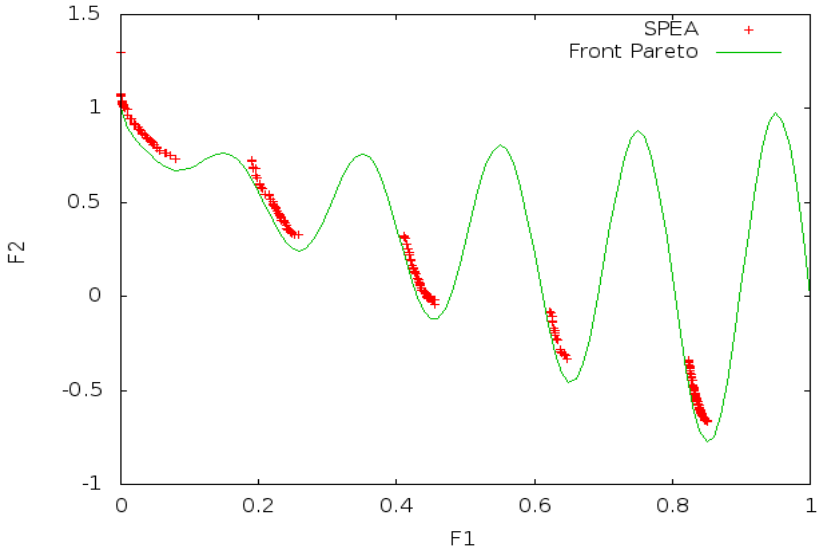


Figure A.3 SPEA - ZDT3

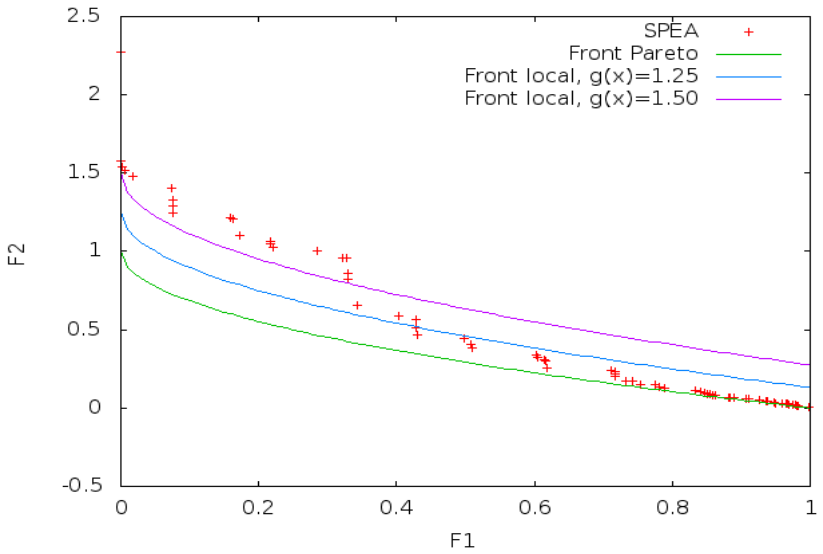


Figure A.4 SPEA - ZDT4

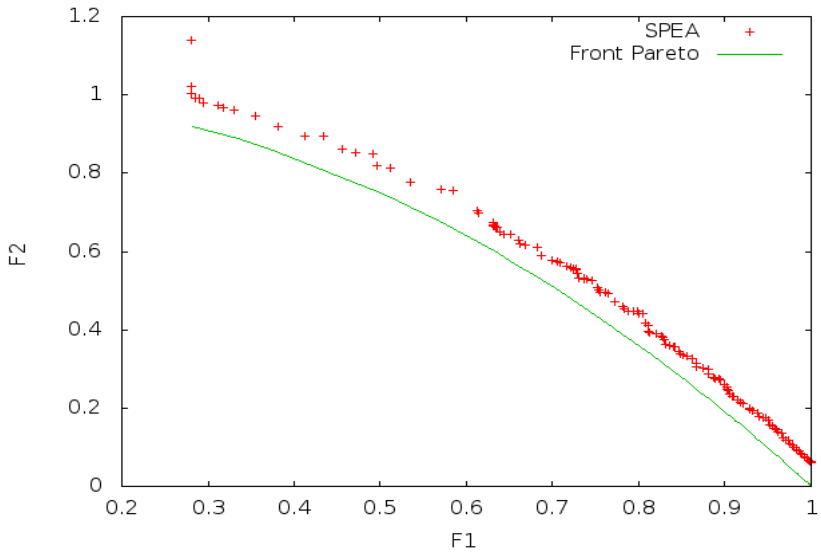


Figure A.5 SPEA - ZDT6

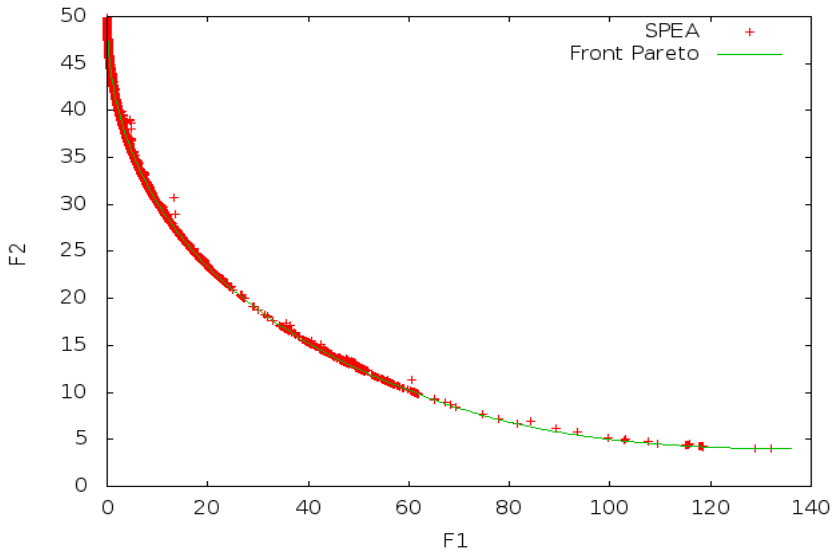


Figure A.6 SPEA - BNH

ANNEXE B

RÉSOLUTION DE CONFLITS, PHASE EN ROUTE

B.1 Convergent vers le centre d'un cercle - 2D

B.1.1 5 avions

Tableau B.1 Statistiques : 100 essais, 5 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	885	0.09
$\Delta Cap = 20$	700	0.80
$\Delta Cap = 30$	516	0.94
$\Delta Cap = 40$	442	1.88
$\Delta Cap = 50$	386	0.18
$\Delta V = 10$	630	0.27
$\Delta V = 20$	511	0.56
$\Delta V = 30$	428	1.79
Génération moyenne pour une solution réalisable : 8		
Nombre moyen de commandes par solution : 6.51		
Nombre de solutions réalisables : 100/100		

Tableau B.2 Statistiques : 100 essais, 5 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	856	0.06
$\Delta Cap = 20$	747	1.93
$\Delta Cap = 30$	640	0.95
$\Delta Cap = 40$	563	0.60
$\Delta Cap = 50$	438	0.17
$\Delta V = 10$	747	0.04
$\Delta V = 20$	613	0.15
$\Delta V = 30$	456	0.47
Génération moyenne pour une solution réalisable : 10		
Nombre moyen de commandes par solution : 4.37		
Nombre de solutions réalisables : 100/100		

B.1.2 7 avions

Tableau B.3 Statistiques : 100 essais, 7 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	813	0.19
$\Delta Cap = 20$	733	1.17
$\Delta Cap = 30$	677	1.38
$\Delta Cap = 40$	590	2.29
$\Delta Cap = 50$	515	0.61
$\Delta V = 10$	726	0.57
$\Delta V = 20$	608	0.93
$\Delta V = 30$	545	2.33
Génération moyenne pour une solution réalisable : 19		
Nombre moyen de commandes par solution : 9.47		
Nombre de solutions réalisables : 100/100		

Tableau B.4 Statistiques : 100 essais, 7 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	960	0.19
$\Delta Cap = 20$	839	1.49
$\Delta Cap = 30$	790	1.76
$\Delta Cap = 40$	828	1.47
$\Delta Cap = 50$	652	0.74
$\Delta V = 10$	769	0.39
$\Delta V = 20$	724	0.84
$\Delta V = 30$	788	1.07
Génération moyenne pour une solution réalisable : 30		
Nombre moyen de commandes par solution : 7.95		
Nombre de solutions réalisables : 100/100		

B.1.3 9 avions

Tableau B.5 Statistiques : 100 essais, 9 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	772	0.34
$\Delta Cap = 20$	830	0.99
$\Delta Cap = 30$	806	1.82
$\Delta Cap = 40$	790	2.71
$\Delta Cap = 50$	754	1.95
$\Delta V = 10$	883	1.03
$\Delta V = 20$	760	1.54
$\Delta V = 30$	704	2.31
Génération moyenne pour une solution réalisable : 41		
Nombre moyen de commandes par solution : 12.69		
Nombre de solutions réalisables : 98/100		

Tableau B.6 Statistiques : 100 essais, 9 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1193	0.38
$\Delta Cap = 20$	1240	1.33
$\Delta Cap = 30$	1170	2.00
$\Delta Cap = 40$	1170	2.28
$\Delta Cap = 50$	1140	1.92
$\Delta V = 10$	1246	1.46
$\Delta V = 20$	1173	1.83
$\Delta V = 30$	1173	1.44
Génération moyenne pour une solution réalisable : 111		
Nombre moyen de commandes par solution : 12.64		
Nombre de solutions réalisables : 93/100		

B.1.4 13 avions

Tableau B.7 Statistiques : 100 essais, 13 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1451	0.86
$\Delta Cap = 20$	1419	1.29
$\Delta Cap = 30$	1488	1.92
$\Delta Cap = 40$	1673	3.38
$\Delta Cap = 50$	1746	4.97
$\Delta V = 10$	1700	3.89
$\Delta V = 20$	1577	3.78
$\Delta V = 30$	1599	3.32
Génération moyenne pour une solution réalisable : 168		
Nombre moyen de commandes par solution : 23.41		
Nombre de solutions réalisables : 17/100		

Tableau B.8 Statistiques : 100 essais, 13 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1285	1.21
$\Delta Cap = 20$	1328	1.54
$\Delta Cap = 30$	1591	1.84
$\Delta Cap = 40$	1640	3.49
$\Delta Cap = 50$	1752	4.34
$\Delta V = 10$	1542	4.34
$\Delta V = 20$	1597	3.75
$\Delta V = 30$	1578	2.79
Génération moyenne pour une solution réalisable : 254		
Nombre moyen de commandes par solution : 23.30		
Nombre de solutions réalisables : 8/100		

B.1.5 15 avions

Tableau B.9 Statistiques : 100 essais, 15 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1368	1.21
$\Delta Cap = 20$	1454	1.42
$\Delta Cap = 30$	1612	2.24
$\Delta Cap = 40$	1741	3.72
$\Delta Cap = 50$	1815	5.85
$\Delta V = 10$	1666	4.67
$\Delta V = 20$	1659	4.77
$\Delta V = 30$	1640	3.36
Génération moyenne pour une solution réalisable : 240		
Nombre moyen de commandes par solution : 27.24		
Nombre de solutions réalisables : 1/100		

Tableau B.10 Statistiques : 100 essais, 15 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1234	1.21
$\Delta Cap = 20$	1403	1.85
$\Delta Cap = 30$	1589	2.22
$\Delta Cap = 40$	1695	3.96
$\Delta Cap = 50$	1767	5.09
$\Delta V = 10$	1634	4.79
$\Delta V = 20$	1592	4.81
$\Delta V = 30$	1611	3.12
Génération moyenne pour une solution réalisable : —		
Nombre moyen de commandes par solution : 27.05		
Nombre de solutions réalisables : 0/100		

B.1.6 17 avions

Tableau B.11 Statistiques : 100 essais, 17 avions, 2D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1370	1.12
$\Delta Cap = 20$	1444	1.59
$\Delta Cap = 30$	1576	2.63
$\Delta Cap = 40$	1658	4.32
$\Delta Cap = 50$	1810	6.55
$\Delta V = 10$	1683	5.24
$\Delta V = 20$	1635	5.61
$\Delta V = 30$	1590	3.57
Génération moyenne pour une solution réalisable : —		
Nombre moyen de commandes par solution : 30.63		
Nombre de solutions réalisables : 0/100		

Tableau B.12 Statistiques : 100 essais, 17 avions, 2D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1253	1.37
$\Delta Cap = 20$	1451	2.06
$\Delta Cap = 30$	1493	2.64
$\Delta Cap = 40$	1724	4.29
$\Delta Cap = 50$	1809	5.74
$\Delta V = 10$	1599	5.30
$\Delta V = 20$	1612	5.29
$\Delta V = 30$	1665	3.84
Génération moyenne pour une solution réalisable : —		
Nombre moyen de commandes par solution : 30.53		
Nombre de solutions réalisables : 0/100		

B.2 Convergent vers le centre d'un cercle - 3D

B.2.1 9 avions

Tableau B.13 Statistiques : 100 essais, 9 avions, 3D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	654	0.58
$\Delta Cap = 20$	605	1.47
$\Delta Cap = 30$	456	1.05
$\Delta Cap = 40$	434	1.12
$\Delta Cap = 50$	316	0.20
$\Delta V = 10$	505	0.88
$\Delta V = 20$	449	1.10
$\Delta V = 30$	421	1.12
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	817	2.30
$\Delta Alt = 1000$	823	2.80
$\Delta Alt = 1500$	761	0.78
$\Delta Alt = 2000$	926	0.58
Génération moyenne pour une solution réalisable : 18		
Nombre moyen de commandes par solution : 13.98		
Nombre de solutions réalisables : 100/100		

Tableau B.14 Statistiques : 100 essais, 9 avions, 3D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	827	0.91
$\Delta Cap = 20$	759	2.05
$\Delta Cap = 30$	700	1.51
$\Delta Cap = 40$	653	0.77
$\Delta Cap = 50$	379	0.18
$\Delta V = 10$	716	1.98
$\Delta V = 20$	620	1.56
$\Delta V = 30$	674	0.62
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	825	2.70
$\Delta Alt = 1000$	878	2.64
$\Delta Alt = 1500$	785	1.13
$\Delta Alt = 2000$	836	0.44
Génération moyenne pour une solution réalisable : 29		
Nombre moyen de commandes par solution : 16.49		
Nombre de solutions réalisables : 100/100		

B.2.2 11 avions

Tableau B.15 Statistiques : 100 essais, 11 avions, 3D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	721	1.00
$\Delta Cap = 20$	616	2.33
$\Delta Cap = 30$	544	1.92
$\Delta Cap = 40$	481	1.30
$\Delta Cap = 50$	384	0.32
$\Delta V = 10$	544	1.42
$\Delta V = 20$	515	2.04
$\Delta V = 30$	459	1.35
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	725	2.64
$\Delta Alt = 1000$	776	3.31
$\Delta Alt = 1500$	676	1.56
$\Delta Alt = 2000$	707	0.69
Génération moyenne pour une solution réalisable : 31		
Nombre moyen de commandes par solution : 19.88		
Nombre de solutions réalisables : 100/100		

Tableau B.16 Statistiques : 100 essais, 11 avions, 3D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1101	1.49
$\Delta Cap = 20$	1038	2.66
$\Delta Cap = 30$	963	2.35
$\Delta Cap = 40$	843	0.95
$\Delta Cap = 50$	778	0.38
$\Delta V = 10$	937	3.06
$\Delta V = 20$	948	2.25
$\Delta V = 30$	843	0.74
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	1015	3.40
$\Delta Alt = 1000$	1028	3.13
$\Delta Alt = 1500$	972	1.65
$\Delta Alt = 2000$	931	0.60
Génération moyenne pour une solution réalisable : 69		
Nombre moyen de commandes par solution : 22.66		
Nombre de solutions réalisables : 100/100		

B.2.3 13 avions

Tableau B.17 Statistiques : 100 essais, 11 avions, 3D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	721	1.00
$\Delta Cap = 20$	616	2.33
$\Delta Cap = 30$	544	1.92
$\Delta Cap = 40$	481	1.30
$\Delta Cap = 50$	384	0.32
$\Delta V = 10$	544	1.42
$\Delta V = 20$	515	2.04
$\Delta V = 30$	459	1.35
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	725	2.64
$\Delta Alt = 1000$	776	3.31
$\Delta Alt = 1500$	676	1.56
$\Delta Alt = 2000$	707	0.69
Génération moyenne pour une solution réalisable : 31		
Nombre moyen de commandes par solution : 19.88		
Nombre de solutions réalisables : 100/100		

Tableau B.18 Statistiques : 100 essais, 11 avions, 3D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1101	1.49
$\Delta Cap = 20$	1038	2.66
$\Delta Cap = 30$	963	2.35
$\Delta Cap = 40$	843	0.95
$\Delta Cap = 50$	778	0.38
$\Delta V = 10$	937	3.06
$\Delta V = 20$	948	2.25
$\Delta V = 30$	843	0.74
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	1015	3.40
$\Delta Alt = 1000$	1028	3.13
$\Delta Alt = 1500$	972	1.65
$\Delta Alt = 2000$	931	0.60
Génération moyenne pour une solution réalisable : 69		
Nombre moyen de commandes par solution : 22.66		
Nombre de solutions réalisables : 100/100		

B.2.4 15 avions

Tableau B.19 Statistiques : 100 essais, 15 avions, 3D, (SPEA-MOD).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1071	2.07
$\Delta Cap = 20$	1061	3.83
$\Delta Cap = 30$	1039	3.06
$\Delta Cap = 40$	960	2.02
$\Delta Cap = 50$	837	0.81
$\Delta V = 10$	1018	4.08
$\Delta V = 20$	922	3.29
$\Delta V = 30$	880	1.25
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	1041	4.25
$\Delta Alt = 1000$	1015	4.62
$\Delta Alt = 1500$	1055	2.11
$\Delta Alt = 2000$	1061	0.81
Génération moyenne pour une solution réalisable : 149		
Nombre moyen de commandes par solution : 32.20		
Nombre de solutions réalisables : 100/100		

Tableau B.20 Statistiques : 100 essais, 15 avions, 3D, (PSO-MO).

Manoeuvres	ΔT moyen (sec)	Nb commandes moyen
$\Delta Cap = 10$	1413	2.31
$\Delta Cap = 20$	1495	3.12
$\Delta Cap = 30$	1509	3.37
$\Delta Cap = 40$	1513	3.07
$\Delta Cap = 50$	1567	1.79
$\Delta V = 10$	1524	5.12
$\Delta V = 20$	1430	4.54
$\Delta V = 30$	1455	1.96
$\Delta Alt = 0$	0	0.00
$\Delta Alt = 500$	1471	4.23
$\Delta Alt = 1000$	1479	4.49
$\Delta Alt = 1500$	1504	2.93
$\Delta Alt = 2000$	1528	1.51
Génération moyenne pour une solution réalisable : 224		
Nombre moyen de commandes par solution : 38.44		
Nombre de solutions réalisables : 75/100		

ANNEXE C

RÉSOLUTION DE CONFLITS THÉORIQUES DANS LE PLAN

Cette annexe décrit l'algorithme proposé par Bach et ses collègues, pour la résolution de conflits dans le plan [Bach *et al.*, 2007].

Considérons deux avions, A et B, ayant des positions connues, volant à la même altitude et à des vitesses constantes. Il est noté que les effets atmosphériques, tel que le vent, sont ignorés. Les vecteurs de position des avions A et B sont r_a et r_b respectivement. Les vecteurs de position relative, entre A et B, ainsi que de vitesse relative sont donnés par

$$\begin{aligned} S &= r_a - r_b \\ v_r &= v_a - v_b \end{aligned}$$

où v_a et v_b sont les vecteurs vitesse de A et B. La figure C.1 illustre la situation.

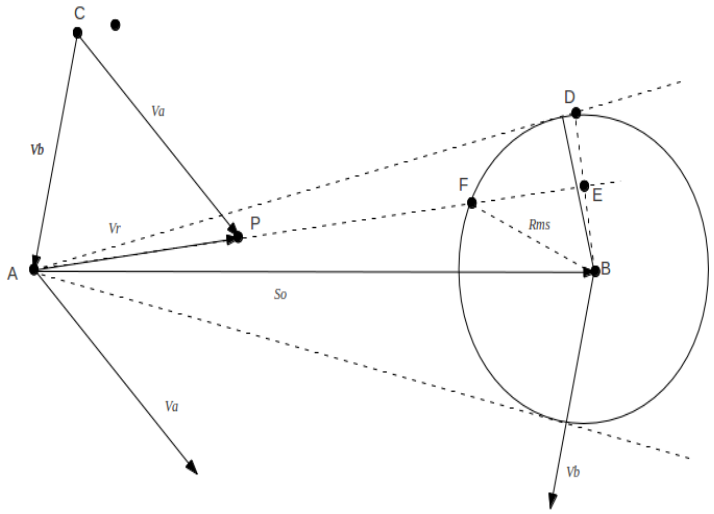


Figure C.1 Avions convergeant vers le centre d'un cercle. Sans résolution.

$$\begin{aligned} S &= r_a - r_b \\ v_r &= v_a - v_b \\ \beta &= \pm \sin^{-1}(R_{ms}/S_0) \\ S_0 &= \left(\Delta x^2 + \Delta y^2\right)^{1/2} \\ \psi_0 &= \tan^{-1}(\Delta y/\Delta x) \end{aligned}$$

où $\Delta x = x_b - x_a$ et $\Delta y = y_b - y_a$.

Les caps sont positifs dans le sens des aiguilles d'une montre à partir du nord. La norme de la variation de la vitesse relative $|v_r|$ ainsi que le cap relatif peuvent également être obtenus en appliquant la loi des sinus au triangle de vitesse ACP de la figure C.1.

$$\frac{|v_a|}{\sin(\psi_b - \psi_r)} = \frac{|v_b|}{\sin(\psi_a - \psi_r)}$$

Par l'utilisation d'identités trigonométriques, on obtient

$$\begin{aligned} \psi_r &= \tan^{-1}(N/D) \\ v_r &= (N^2 + D^2)^{1/2} \end{aligned}$$

où

$$\begin{aligned} N &= |v_a| \sin(\psi_a) - |v_b| \sin(\psi_b) \\ D &= |v_a| \cos(\psi_a) - |v_b| \cos(\psi_b) \end{aligned}$$

La solution proposée est d'ordonner un changement de direction à un seul avion, soit le plus rapide. Il est possible de visualiser le changement de cap nécessaire à l'avion A pour respecter la séparation avec l'avion B. Il suffit simplement de faire tourner le vecteur v_a autour du point C en sens horaire ou anti-horaire pour faire en sorte que le vecteur de vitesse relative vers le point c ou e.

Le changement de cap, pour l'avion le plus rapide, est obtenu par la loi des sinus

$$\begin{aligned} \psi_a^* &= \psi_r^* + \sin^{-1}\left(\sigma_v \sin(\psi_b - \psi_r^*)\right) \\ \sigma_v &= |v_b|/|v_a| \end{aligned}$$

où ψ_a^* est le nouveau cap de l'avion A (vers Cc ou Ce) et ψ_r^* est le cap résultant du vecteur de vitesse relative (vers Ac ou Ae).

LISTE DES RÉFÉRENCES

Adcock, S. (2005). *GAUL, The Genetic Algorithm Utility Library.* `http://gaul.sourceforge.net/` (page consultée le 2013-05-02).

Bach, R., Farrell, C. et Erzberger, H. (2007). *An Algorithm for Level-Aircraft Conflict Resolution* (Rapport technique 94035). NASA Ames Research Center, Moffett Field CA, 12 p.

Binh, T. et Korn, U. (1997). Mobes : A multiobjective evolution strategy for constrained optimisation problems. Dans *Third International Conference on Genetic Algorithms (Mendel 97)*. p. 176–182.

Boost (2012). *Boost C++ Librairies.* `http://www.boost.org` (page consultée le 2013-02-15).

Canada, N. (1999). *Air Traffic Control Manual of Operation (ATC MANOPS).* Nav Canada, 511 p.

Cerf, R. (1994). *Une théorie asymptotique des algorithmes génétiques.* Thèse de doctorat, Université de Montpellier II, France.

Chian, Y., J.T.Klosowski, Lee, C. et Mitchell, J. (1997). Geometric algorithms for conflict detection/resolution in air traffic management. Dans *Proceedings of the IEEE Conference on Decision and Control.* Volume 2. IEEE, p. 1835–1840.

Coello Coello, C. (2000a). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, Volume 17, p. 319–346.

Coello Coello, C. (2000b). Handling preferences in evolutionary multiobjective optimization : A survey. Dans *2000 Congress on Evolutionary Computation.* IEEE, p. 30–37.

Deb, K. (2008). *Multi-Objective Optimization using Evolutionary Algorithmes*, première édition. Wiley, 516 p.

Deb, K., Agrawal, S. et Pratap, A. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation : NSGA-II. Dans *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature.* PPSN VI. Springer-Verlag, London, UK, UK, p. 849–858.

Deb, K., Amrit, A. et Meyarivan, T. (2001). Constrained test problems for multi-objective evolutionary optimization. Dans *First International Conference On Evolutionary Multi-Criterion Optimization.* Volume 1993/2001. Springer Verlag, p. 284–298.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms.* Thèse de doctorat, Politecnico di Milano, Italy.

Duong, V. N. et Zeghal, K. (1997). Conflict resolution advisory for autonomous airborne separation in low-density airspace. Dans *Proceedings of the 36th Conference on Decision and Control*. Volume 3. IEEE, p. 2429–2434.

Durand, N. (1996). *Optimisation de trajectoires pour la résolution de conflits*. Thèse de doctorat, ENAC : École Nationale d'Aviation Civile, Toulouse, France, 200 p.

Durand, N. et Alliot, J. (2009). Ant colony optimization for air traffic conflict resolution. Dans *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)*. ATM - Seminar, p. 1–6.

Durand, N. et Alliot, J.-M. (1997). Optimal resolution of en route conflicts. Dans *Séminaire Europe/USA, Saclay*. p. 12.

E. Zitzler, K. D. et Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms : Empirical results. *Evolutionary Computation*, Volume 8, numéro 2, p. 173–195.

Ebarhart, R., Dobbins, R. et Simpson, P. (1996). *Computational Intelligence PC Tools*. Morgan Kaufmann Publishers, 464 p.

Eberhart, R. et Kennedy, J. (1995). A new optimizer using particle swarm theory. Dans *Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan*. Volume 1. IEEE, p. 39–43.

Eberhart, R. et Kennedy, Y. S. J. (2001). *Swarm Intelligence*, première édition. W, 512 p.

Erzberger, H. (1995). *Design Principles and Algorithms for Automated Air Traffic Management* (Rapport technique MS 210-9). NASA AMES, 31 p.

Erzberger, H., Paielli, R., Isaacson, D. et Eshowl, M. (1997). Conflict detection and resolution in the presence of prediction error. Dans *1st USA/Europe Air Traffic Management R&D Seminar*. ATM - seminar, p. 1–19.

Eschenauer, H., Koski, J. et Osyczka, A. (1990). *Mutlicriteria Design Optimization*. Springer Verlag, 482 p.

Eurocontrol (2000). *Base of Aircraft Data (BADA)*. `http://www.eurocontrol.int/services/bada`.

Fieldsend, J. (2004). *Multi-Objective Particle Swarm Optimisation Methods* (Rapport technique 419). Department of Computer Science, University of Exeter, Exeter, UK, 20 p.

Fieldsend, J., Everson, R. et Singh, S. (2003). Using unconstrained elite archives for multiobjective optimization. *IEEE Transaction on Evolutionary Computation*, Volume 7, numéro 3, p. 305–323.

Fieldsend, J. et Singh, S. (2002). A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. Dans *Proceedings of UK Workshop on Computational Intelligence (UKCI'02)*. p. 37–44.

Fishman, G. (1997). *Monte-Carlo : Concepts, Algorithm and Applications.* Springer-Verlag, 732 p.

Fogel (1964). *An Analysis of the behaviour of a class of genetic adaptive systems.* Thèse de doctorat, University of Michigan, Lansing, MI, USA, 253 p.

Fonseca, C. et Flemming, P. (1993). Genetic algorithm for multiobjective optimization : Formulation, discussion, and generalization. Dans *Proceedings of the Fifth Internional Conference on Genetic Algorithms.* Volume 1. p. 416–423.

Fowler, M. (2004). *UML Distilled : A Brief Guide to the Standard Object Modeling Language*, troisième édition. Addison-Wesley, 175 p.

Geser, A., noz, C. M., Dowek, G. et Kirchner, F. (2002). *Air Traffic Conflict Resolution and Recovery* (Rapport technique NASA/CR-2002-211637, ICASE No. 2002-12). NASA, 24 p.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Volume 13, numéro 5, p. 533–549.

Glover, F. et Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publishers, 412 p.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley Publishing Company, 412 p.

Granger, G. (2002). *Détection et résolution de conflits aériens : modélisations et analyse.* Thèse de doctorat, ENAC : École Nationale d'Aviation Civile, Toulouse, France, 152 p.

Grewal, M., Weill, L. et Andrews, A. (2007). *Global Positioning System, Inertial Navigation and Integration*, deuxième édition. Wiley, 525 p.

Hassan, R., Cohanim, B., Weck, Q. et Venter, G. (2004). A comparison of particle swarm optimization and the genetic algorithm. *American Institute of Aeronautics and Astronautics*, p. 1–13.

He, X., Liao, M. et Chen, W. (2009). A conflict detection and resolution scheme using dynamic flight model. Dans *Proceedings of the 8th International Conference on Machine Learning and Cybernetics.* Volume 1. IEEE, p. 3194–3197.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* Thèse de doctorat, University of Michigan, Ann Arbor.

Horn, J., Nafploitis, N. et Goldberg, D. (1994). A niched Pareto genetic algorithm for multi-objective optimization. Dans *Proceedings of the First IEEE Conference on Evolutionary Computation.* Volume 1. IEEE, p. 82–87.

Hu, X., Eberhart, R. et Shi, Y. (2001). Particle swarm with extended memory for multi-objective optimization. Dans *Proceedings of the 2003 IEEE Swarm Intelligence Symposium.* Volume 1. IEEE, p. 193–197.

Huang, S., Feron, E. et Mao, Z.-H. (2012). Optimal configuration for intersecting flows of aircraft. Dans *15th Internantional IEEE Conference on Intelligent Transportation Systems*. IEEE, p. 1447–1452.

Hwang, I. et Tomlin, C. (2002). *Protocol-based Conflict Resolution for Air Traffic Control* (Rapport technique). Departement of Aeronautics and Astronautics, Standford University, 49 p.

Inselberg, A. (2001). Conflict detection and planar resolution for air traffic control. Dans *Proceedings of the IEEE Intelligent Transportation Systems Conference*. Volume 1. IEEE, p. 1200–1205.

Isaacson, D. et Robinson, J. (2001). A knowledge-based conflict resolution algorithm for terminal air traffic control advisory generation. Dans *Proceedings of the AIAA Guidance, Navigation and Control Conference*. Volume 2001-4116. American Institute of Aeronautics and Astronautics, AIAA, p. 1–11.

Jiménez, F. et Verdegay, J. (1999). Evolutionary techniques for constrained optimization problems. Dans *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT99), Aachen, Germany*. Volume 1. Springer-Verlag, p. 1–7.

Joines, J. et Houck, C. (1994). On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with gas. Dans *Proceedings of the First IEEE Conference on World Congress on Computational Intelligence*. Volume 2. Evolutionary Computation IEEE, p. 579–584.

Jong, K. D. (1975). *An Analysis of the behaviour of a class of genetic adaptive systems*. Thèse de doctorat, University of Michigan, Lansing, MI, USA, 253 p.

Kallus, K., Barbarino, M. et Damme, D. V. (1997). *Model of the Cognitive Aspects of Air Traffic Control* (Rapport technique HUM.ET1.ST01.1000-DEL02). European Organisation for the Safety of Air Navigation, 62 p.

Kim, J.-H. et Myung, H. (1997). Evolutionary programming techniques for constrained optimization problems. *IEEE Transaction on Evolutionary Computation*, Volume 1, numéro 2, p. 129–140.

Kirkpatrick, S., Gelatt, C. et Vecchi, M. (1983). Optimization by simulated annealing. *Science*, , numéro 220, p. 671–680.

Krozel, J. et Peters, M. (1997). Strategic conflict detection and resolution for free flight. Dans *Proceedings of the 36th Conference on Decision and Control*. Volume 1. IEEE, p. 1822–1828.

Kuchar, J. et Yang, L. (1997). Survey of conflict detection and resolution modelling methods. Dans *Proceedings of the AIAA Guidance, Navigation and Control Conference*. Numéro 97-3732. American Institute of Aeronautics and Astronautics, AIAA, p. 1388–1397.

Kuchar, J. et Yang, L. (2000). A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, Volume 1, numéro 4, p. 179–189.

Kuchling, A. (2013). *Python Advocacy HOWTO*. `http://docs.python.org/2/howto/advocacy.html`.

Kuwata, Y. et Oohama, H. (1997). A case study of a real-time problem solving strategy in an air traffic control problem. *Expert Systems with Applications*, Volume 12, numéro 1, p. 71–79.

Lecchini, A., Glover, W., Lygeros, J. et Maciejowski, J. (2006). Monte carlo optimization for conflict resolution in air traffic control. *IEEE Transactions on Intelligent Transportation Systems*, Volume 7, numéro 4, p. 470–482.

Lin, C. et Hong, M. (1989). A knowledge-based en route monitor for air traffic control. *IEEE Transaction on Aerospace and Electronic Systems*, Volume 25, numéro 3, p. 392–400.

Malaek, M. et Alaeddini, A. (2009). Conflict resolution maneuvers based on genetic algorithm modified webs. Dans *Aerospace conference*. IEEE, p. 1–8.

Malaek, M., Alaeddini, S. et Gerren, D. (2011). Optimal maneuvers for aircraft conflict resolution based on efficient genetic webs. *IEEE Transactions on Aerospace and Electronic System*, Volume 47, numéro 4, p. 2457–2472.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structres = Evolution Programs*, troisième édition. Springer Verlag, 387 p.

Michalewicz, Z. et Nazhiyath, G. (1995). GENOCOP III : A coevolutionary algorithm for numerical optimization problems with nonlinear constraints. Dans *International Conference on Evolutionary Computation*. Volume 2. IEEE, p. 647–651.

Mostaghim, S. et Teich, J. (2003). Strategies for finding good local guides in multi-objective particule swarm optimization (MOPSO). Dans *2003 Swarm Intelligence Symposium*. IEEE, p. 26–33.

Nav Canada (2008a). *Cartes de régions terminales*. `http://www.navcanada.ca/NavCanada.asp?Language=fr&Content=ContentDefinitionFiles\Publications\AeronauticalInfoProducts\Charts\TAC\default.xml` (page consultée le 2008-09-25).

Nav Canada (2008b). *Carte en route niveau inférieur*. `http://www.navcanada.ca/NavCanada.asp?Language=FR&Content=ContentDefinitionFiles\Publications\AeronauticalInfoProducts\Charts\EnrouteCharts\default.xml` (page consultée le 2008-09-25).

Nolan, M. (1990). *Fundamentals of Air Traffic Control*, première édition. Wadsworth Publishing Company, 560 p.

OACI (s. d.). *Fourth Meeting of the ALLPIRG/Advisory Group (ALLPIRG/4)*. `http://legacy.icao.int/icao/en/ro/allpirg/allpirg4.htm` (page consultée le 2013-05-15).

Osyzka, A. et Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, Volume 10, numéro 2, p. 94–99.

Pallottino, L. et Bicchi, A. (2000). On the optimal conflict resolution for air traffic control. Dans *Proceedings of Intelligent Transportation Systems*. IEEE, p. 167–172.

Pareto, V. (1896). *Cours d'Économie Politique, Volume I et II*. F. Rouge, Lausanne.

Perry et Tekla, S. (1997). In search of the future of air traffic control. *IEEE Spectrum*, Volume 34, numéro 8, p. 18–35.

Prevôt, T., Battiste, V., Palmer, E. et Shelden, S. (2003). Air traffic concept utilizing 4D trajectories and airborne separation assistance. Dans *Proceedings of the AIAA Guidance, Navigation, and Control Conference, AIAA-2003-5770*. American Institute of Aeronautics and Astronautics, AIAA, p. 1–11.

Reif, J. et Sharir, M. (1994). Motion planning in the presence of moving obstacles. *Journal of the Association for Computing Machinery*, Volume 41, numéro 4, p. 764–790.

Reyes-Sierra, M. et Coello Coello, C. (2006). Multi-objective particle swarm optimizers : A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, Volume 2, numéro 3, p. 287–308.

Robinson, J. et Isaacson, D. (2000). A concurrent sequencing and deconflicting algorithm for terminal area air traffic control. Dans *Proceedings of the AIAA Guidance, Navigation and Control Conference*. American Institute of Aeronautics and Astronautics, AIAA, p. 1–11.

Robinson, J., Thomas, T. et Isaacson, D. (1997). Fuzzy reasoning-based sequencing of arrival aircraft in the terminal area. Dans *Proceedings of the AIAA Guidance, Navigation and Control Conference*. American Institute of Aeronautics and Astronautics, AIAA, p. 1–11.

Roskam, J. et Lan, G.-T. E. (2003). *Airplane Aerodynamics and Performance*. Design, Analysis and Research Corporation (DAR corporation), 699 p.

Rotstan, N. (2002). *JGAP - Java Genetic Algorithm Package*. `http://jgap.sourceforge.net/` (page consultée le 2012-07-02).

RTCA Special Committe (2007). *Minimum aviation system performance standards for automatic dependent surveillance broadcast (ADS-B)* (Rapport technique DO-242B). 204 p.

Rudolph, G. et Agapie, A. (2000). Convergence properties of some multi-objective evolutionary algorithms. Dans Zalzala, A. et Eberhart, R., *Congress on Evolutionary Computation (CEC 2000)*. Volume 2. Piscataway, NJ, IEEE press, p. 1010–1016.

Russel, S. et Norvig, P. (1995). *Artificial Intelligence, A Modern Approach*, première édition. Prentice Hall, 932 p.

Schaffer, J. (1984). *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*. Thèse de doctorat, Université Vanderbilt, Nashville, TN, USA, 166 p.

Stroustrup, B. (1997). *The C++ Programming Language*, troisième édition. Addison-Wesley, 175 p.

Tomlin, C., Pappas, G. et Sastry, S. (1998). Conflict resolution for air traffic management : A study in multiagent hybrid systems. *IEEE Transactions on Automation Control*, Volume 43, numéro 4, p. 509–521.

Transport Canada (2009). *Manuel d'information aéronautique*. `www.tc.gc.ca/AviationCivile/publications/tp14371/menu.htm` (page consultée le 2009-10-20).

Treleaven, K. et Mao, Z.-H. (2008). Conflict resolution and traffic complexity of multiple intersection flows of aircraft. *IEEE Transactions on Intelligent Transportation Systems*, Volume 9, numéro 4, p. 633–643.

Tsou, C., Chang, S.-C. et Lai, P.-W. (2007). Using crowding distance to improve multi-objective pso with local search. Dans Chan, F. T. S. et Manoj, *in Swarm Intelligence : Focus on Ant and Particle Swarm Optimization*. Itech Education and Publishing, p. 532.

Venkatraman, S. et Yen, G. (2005). A generic framework for constrained optimization using genetic algorithms. *IEEE Transaction on Evolutionary Computation*, Volume 9, numéro 4, p. 424–435.

Wall, M. (1999). *GAlib, A C++ Library of Genetic Algorithm Components*. `http://lancet.mit.edu/ga/` (page consultée le 2009-01-20).

Zitzler, E. et Thiele, L. (1998). *An Evolutionary Algorithm for Multiobjective Optimization : The Strength Pareto Approach* (Rapport technique 43). Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 43 p.