ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE MÉMOIRE
PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAITRISE EN GÉNIE
M. Ing.

PAR
Romain BRISEMEUR

APPLICATION DES ALGORITHMES GÉNÉTIQUES À LA GESTION DES CONFLITS AÉRIENS EN CROISIÈRE

MONTRÉAL, LE 14 JUIN 2010

PRÉSENTATION DU JURY

CE RAPPORT DE MÉMOIRE A ETE ÉVALUÉ

PAR UN JURY COMPOSÉ DE

Mme Botez, directrice de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M Guy Gauthier, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M Franck Guilbert, examinateur externe
Bombardier Aéronautique

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 9 JUIN 2010

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je souhaiterais avant tout remercier ma directrice de thèse Mme Ruxandra Botez pour m'avoir proposé de prolonger mes études suite à mon stage au sein de son laboratoire LARCASE (Laboratoire de Recherche en Commande Active, Avionique et Aéroservolélasticité) ainsi que pour son support durant ces études. Je voudrais également remercier Mr François Stephan, la direction de l'ESTACA (École Supérieure des Techniques Aéronautiques et de Construction Automobile), Mr Eric Doré et l'équipe du BRIRE (Bureau des Relations Internationales et des Relations Extérieures) de l'ETS pour avoir rendu possible l'échange entre ces deux écoles.

J'exprime ma reconnaissance à la région Pays de la Loire (France) et son programme ENVOLEO ainsi qu'au Bureau des Cycles Supérieurs de l'ETS pour leurs programmes de bourses qui m'ont permis de financer mes études et sans qui cette expérience n'aurait pas été possible.

Je tiens également à adresser mes remerciements aux étudiants et également amis avec qui j'ai pu travailler au sein du laboratoire LARCASE, où la diversité et la bonne humeur règne au quotidien, et tout particulièrement Julien Fays, Gabriel Kouba et Nicolas Boëly qui m'ont accompagnés durant toute ma recherche. C'est en partie cette ambiance de travail qui permet l'excellence des travaux de ce laboratoire.

Enfin je remercie mes amis et ma famille qui ont fait de ces trois années passés au Canada une expérience enrichissante aussi bien sur le plan professionnel que humain.

APPLICATION DES ALGORITHMES GÉNÉTIQUES À LA GESTION DES CONFLITS AÉRIENS EN CROISIÈRE

Romain BRISEMEUR

RÉSUMÉ

L'augmentation constante du trafic aérien depuis ces dernières années entraîne une surcharge de l'espace aérien qui ne facilite pas le travail du contrôle aérien. Les phénomènes de conflits entre différents avions actuellement résolus par le contrôleur au sol ou bien, en dernier recours, par un système embarqué, ne cesseront de s'amplifier.

Afin de résoudre ces problèmes de gestion de conflits, dans ce mémoire nous proposons une solution de gestion automatique des conflits en croisière afin de décharger la quantité de travail du contrôleur au sol.

La résolution proprement dite utilise une méthode d'algorithme génétique appliquée au problème afin d'optimiser les trajectoires proposées suivant des facteurs préalablement définis: tous les conflits doivent être résolus, les trajectoires proposées doivent être les plus courtes possibles et le nombre de manœuvres à effectuer par le ou les pilote(s) doit être réduit au maximum afin de faciliter son (leur) travail. Des manœuvres simples d'exécution et de compréhension sont également utilisées et la solution est volontairement proposée dans le plan horizontal.

Les résultats concluants démontrent à quel point cet outil pourrait faciliter le travail des contrôleurs aériens et permettrait de faire un pas vers l'automatisation de la gestion du contrôle vers laquelle le transport aérien semble se diriger.

Mots clés : algorithme génétique, contrôle aérien, gestion des conflits, gestion du trafic aérien.

GENETIC ALGORITHM METHOD APPLIED TO THE CONFLICT MANAGEMENT DOMAIN IN CRUISE

Romain BRISEMEUR

ABSTRACT

In recent years, an increase of the air traffic has caused an overload in the airspace making the air traffic controller work more complex than it used to be. Indeed, the conflicts between aircrafts which are currently resolved by the air traffic controller from the ground or in the worst case, by an embedded system on board, are bounding to grow.

To solve these conflict management problems, this report aims to propose an automatic solution to unload the amount of work of the air traffic controllers.

The proposed solution itself uses a genetic algorithm method. It has been applied to this problem to optimized proposed trajectories through several factors: all the conflicts must be solved, the proposed trajectories must be as short as possible; the number of operations done by one or several pilots must be reduced at most to facilitate his (their) work. Simple operations of execution and understanding are also used and the solution is voluntarily suggested only on a horizontal path.

The concluding results of that study illustrate how much this tool could facilitate the work of the air traffic controllers on a daily basis and could allow making a step towards an automation of the air traffic control.

Keywords: genetic algorithm, air traffic control, conflict management.

TABLE DES MATIERES

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACAS	Airborne Collisions Avoidance System.
AERA	Automated En-Route Air Traffic Control.
ATC	Air Traffic Control.
BRIRE	Bureau des Relations Internationales et des Relations Extérieures
CENA	Centre d'Étude de la Navigation Aérienne.
D	Dimension.
ESTACA	École Supérieure des Techniques Aéronautiques et de Construction Automobile.
ETS	École de Technologie Supérieure.
FAA	Federal Aviation Administration.
FL	Flight Level.
FMS	Flight Management System.
FREER	Free Route Experimental Encounter Resolution.
LARCASE	Laboratoire de Recherche en Commande Active, Avionique et Aéroservolélasticité.
OACI	Organisation de l'Aviation Civile Internationale.
RTA	Required Time of Arrival.
TCAS	Traffic Collision Avoidance System.

LISTE DES SYMBOLES ET UNITÉS DE MESURE

°	degré.
cm	centimètre.
ft	pieds (*feet*).
h	heure.
kg	kilogramme.
km	kilomètre.
Nm	mille nautique.
Pa	Pascal.
s	seconde.

INTRODUCTION

La navigation aérienne s'est développée de pair avec l'augmentation du trafic aérien ; d'une navigation à vue, les pilotes ont vu naître des équipements embarqués de plus en plus précis, puis l'arrivée d'un système de contrôle aérien centralisé au sol.

L'arrivée des avions à réaction dans les années 1950 va imposer une évolution du contrôle aérien, alors dépassé pour la gestion de tels engins. Il laissera place au radar comme outil performant de surveillance pour les contrôleurs aériens avant de devenir un outil de contrôle avec l'augmentation du trafic aérien. Les radars permettront par la suite d'affiner les espaces entre les aéronefs afin d'augmenter la capacité de l'espace aérien.

L'ère de l'automatisation se voit peu à peu prendre place avec la mise en fonction de programmes automatisés de coordination du contrôle de l'espace aérien. Dans cette optique, nous avons souhaité développer une méthode de résolution de conflits aériens afin d'optimiser l'espace aérien et diminuer la charge de travail du contrôleur aérien durant la phase de croisière. Pour cela nous avons choisi d'utiliser un algorithme génétique.

Ne possédant pas de données réelles issues de la navigation ou bien d'un simulateur de trafic aérien, l'objectif de ce mémoire est de montrer la viabilité d'une telle méthode qui devra par la suite être testée et validée à l'aide de données réelles.

Une mise en contexte du trafic aérien et de sa gestion fait l'objet d'une première partie, avant qu'une revue de littérature sur l'évitement aérien soit développée dans une seconde partie. Le choix de la méthode par algorithme génétique sera ensuite expliqué ainsi que détaillé. Nous y aborderons le fonctionnement d'un tel algorithme ainsi que les ajustements apportés pour que ces derniers résolvent les problèmes présentés. Enfin, dans une dernière partie, l'influence des différents paramètres ajustables dans la méthode des algorithmes génétiques sera exposée, avant d'afficher les résultats finaux obtenus et les limites de cet outil.

CHAPITRE 1

CONTEXTE DU TRAFIC AÉRIEN ET DE SA GESTION

1.1 Contexte

1.1.1 Le trafic aérien

Depuis environ trente ans désormais, le trafic aérien n'a cessé d'augmenter, connaissant une progression constante au fil des années (environ 5% d'augmentation du trafic aérien par an). Comme nous l'avons vu précédemment, la gestion du contrôle aérien se doit d'évoluer en même temps, parallèlement à la gestion du trafic aérien. L'Europe de l'Ouest et l'Est de l'Amérique sont les plus touchés par cette expansion rendant leur ciel très dense, cette contrainte ne se limitant pas seulement en approche des zones aéroportuaires.

Le nombre d'avions augmentent et ainsi le nombre d'informations à traiter également. Pour faire face à cette évolution, différentes théories s'opposent maintenant depuis quelques années. Il s'agit de deux politiques d'approches radicalement opposées qui s'affrontent pour déterminer l'avenir du contrôle aérien mondial :

- Il y a tout d'abord la politique de « *tout-automatisation* » et le concept de « *free-flight* [1] » qui s'échappe de cette approche. Cependant cette conception est pour le moment rendue irréalisable pour des raisons essentiellement politiques, mais également humaines et parfois techniques. En effet les avancées technologiques se heurtent parfois à des obstacles imprévus : le facteur humain. Il a fallu de nombreuses années pour introduire dans les mœurs le concept de métro sans conducteur, et aujourd'hui encore il est difficile d'imaginer un train à grande vitesse (pouvant atteindre les 320 km/h) sans conducteur. Pourtant cela constitue bien l'avenir du transport.

[1] Concept Nord Américain qui permettrait automatiquement aux avions d'emprunter des routes de leur choix dans des espaces de faible densité de trafic.

À quand donc un transport aérien entièrement automatisé ? Cela arrivera sans doute plus vite que nous ne l'imaginons, mais il est bien raisonnable de penser à ce concept tout d'abord pour la gestion du contrôle aérien et c'est dans cette voie d'automatisation que ce dernier doit se diriger.

- À cela s'oppose les réfractaires à une automatisation préférant laisser ces opérations à la charge humaine. Cependant l'augmentation visible du trafic aérien entraine l'apparition de nombreuses informations à traiter pour le contrôleur qui voit ainsi sa charge de travail accroître dangereusement. Cette solution n'a pour le moment pas d'avenir si l'on se fie aux courbes de prévisions du transport aérien.

Convaincu que l'avenir du contrôle aérien passera par une automatisation de sa gestion (du moins en partie), le laboratoire LARCASE (***La**boratoire de **R**echerche en **C**ommande **A**ctive, Avionique et Aéro**ser**volélasticité*) situé à l'École de Technologie Supérieure de Montréal à mis en place différents projets traitant de cette automatisation. Nous y reviendront dans la suite de ce rapport.

1.1.2 Les conflits aériens

Avant de s'intéresser à la mise en place du contrôle aérien, il est nécessaire, dans un souci de compréhension, d'expliquer la notion de conflit aérien. Un conflit aérien se produit lorsqu'au moins deux avions pénètrent dans un même espace restreint :

- 5 milles nautiques[2] (Nm) dans le plan horizontal ;
- 1000 pieds[3] (feet ou ft) dans le plan vertical au dessous du *Flight Level*[4] 290 (FL 290) et 2000 ft au dessus du FL 290.

---

[2] Unité de distance utilisée en navigation aérienne ou nautique; un mille nautique vaut la longueur initiale d'une minute d'arc terrestre et équivaut à 1852 m.

[3] Unité de mesure qui équivaut à 32,48 cm.

[4] Un Flight Level est un niveau de vol exprimé en centaine de pieds au dessus de la surface isobare 1 013,25 hPa. FL 290 correspond à une altitude de 29 000 ft au dessus de l'isobare 1 013,25 hPa.

On appelle aussi *cluster* d'avions, un conflit à *n* avions où *n* représente le nombre d'avions impliqués dans le *cluster*. Si l'avion A est en conflit avec l'avion B lui-même en conflit avec l'avion C (alors que l'avion A et C ne sont pas en conflit), les avions A, B et C appartiendront au même *cluster* comme nous le montre la figure 1.1. Nous verrons par la suite que nous résoudrons des *clusters* et non uniquement des conflits deux à deux.

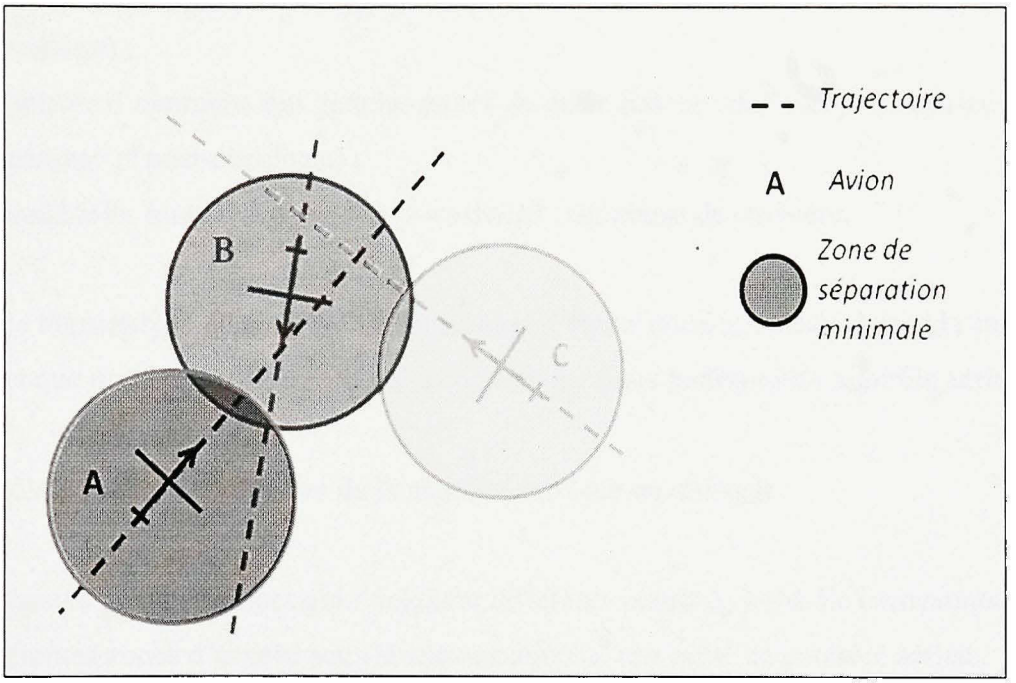

Figure 1.1 Cluster à 3 avions à une même altitude.

Ces conflits sont actuellement majoritairement gérés par le contrôle aérien et nous pouvons définir deux types de zones où ces conflits peuvent survenir :

- en approche des zones aéroportuaires où le trafic est très dense ;
- moins communément, durant la phase de croisière. C'est durant cette phase de vol que nous allons porter notre attention à une résolution automatique des conflits aériens.

1.1.3 Le contrôle aérien

Le contrôle aérien (ou *Air Traffic Control*) a pour but de réguler les flux de trafic aérien pour assurer la sécurité des vols. On distingue trois types de contrôle aérien :

- le contrôle des aérodromes qui gère le trafic au sol (phases de roulage, décollage et atterrissage) ;
- le contrôle d'approche qui gère les zones de trafic proches des aéroports (phases de pré-atterrissage et post-décollage) ;
- le contrôle en route qui gère les avions durant leur phase de croisière.

C'est à ce dernier type de contrôle aérien que nous allons nous intéresser durant la suite de ce rapport et que nous ferons désormais allusion lorsque nous parlerons du contrôle aérien.

Le contrôle en route est structuré de la manière suivante en utilisant :

- des routes prédéterminées qui rejoignent différents points de contrôle (*waypoints*) ;
- différentes zones d'espace sous la responsabilité d'une unité de contrôle aérien.

De nos jours, le trafic aérien est beaucoup trop dense pour n'être géré que par un seul contrôleur aérien. La charge de travail est ainsi répartie et un découpage de l'espace, appelé sectorisation du trafic aérien, est alors effectué pour permettre d'alléger la charge de travail de cet opérateur au sol.

Plusieurs méthodes d'optimisation de sectorisation d'espace aérien ont été étudiées [3] [8] et mises en place, mais l'augmentation du trafic va indéniablement rendre ces optimisations compliquées pour les contrôleurs aériens et finira par saturer leur travail. La découpe de l'espace ne peut pas être indéfinie, car nécessite parallèlement une lourde gestion du transfert de zone (une communication est mise en place lorsqu'un avion passe d'un secteur à un

autre). Ainsi une automatisation du contrôle aérien, tout d'abord à l'aide de quelques fonctions, sera indéniablement nécessaire afin d'alléger le travail de l'opérateur.

Cependant le contrôleur aérien n'est pas le seul à avoir la charge d'éviter les collisions entre les avions et fluidifier le trafic ; il fait partie de la chaîne du contrôle aérien qui se définit par les filtres suivants :

- **Le filtre stratégique** qui opère une organisation grossière du trafic à moyen et long terme (six mois et plus). Les flux de trafic aérien sont déjà identifiés et connus.
- **Le filtre de pré-régulation** qui opère une organisation à court terme (la veille ou l'avant-veille) du trafic. Les plans de vol, la capacité de contrôle de chaque unité de contrôle ainsi que la capacité de chaque secteur sont évalués en se fiant aux statistiques des années précédentes. Les différentes plages horaires de décollage des avions (créneaux temporels durant lesquels l'avion est autorisé à décoller) sont déterminées durant cette phase de pré-régulation.
- **Le filtre de régulation en temps réel** qui ajuste l'organisation du trafic en fonction des dernières informations disponibles comme les facteurs météorologiques et/ou les incidents techniques.
- **Le filtre tactique**, dernier filtre au sol de la chaîne du contrôle aérien. C'est à cette étape que le contrôleur aérien va identifier et gérer les potentiels conflits intervenant sur son secteur.
- **Le filtre d'urgence** qui prend le relais lorsqu'il y a eu une défaillance et qu'un conflit subsiste à l'issu du filtre tactique. Il existe deux filtres d'urgence distincts :

    - Le *filet de sauvegarde* qui prédit les trajectoires des avions sur une courte fenêtre temporelle (de l'ordre de quelques minutes) et déclenche une alarme sonore en cas de conflit imminent. Cependant il est à noter que ce système au sol ne propose pas de solution, et restera donc à la charge du contrôleur aérien.

- Le TCAS (***T**raffic **C**ollision **A**voidance **S**ystem*) est un système électronique embarqué dans l'avion qui prévient le (ou les) pilote(s) d'une collision imminente. La fenêtre temporelle varie entre 25 et 40 secondes, trop courte pour qu'un contrôleur aérien prenne connaissance, identifie une solution et la propose. Ce filtre propose pour le moment une solution immédiate dans le plan vertical dans le but d'éviter la collision.

Actuellement, l'emport d'un système ACAS II (***A**irborne **C**ollisions **A**voidance **S**ystem* de seconde génération) est rendu obligatoire par l'OACI (***O**rganisation de l'**A**viation **C**ivile **I**nternationale*) dans tous les avions civils à turbine à voilure fixe ayant une masse maximale au décollage de plus de 5,700 kg ou une configuration maximale approuvée de plus de 19 sièges passagers.

De nos jours, le TCAS II est le seul système satisfaisant entièrement à cette réglementation. Nous n'allons pas détailler ici le principe du TCAS, mais nous pouvons en ressortir trois défauts majeurs dans son fonctionnement :

- Il sature rapidement car il n'est pas fait pour résoudre des conflits impliquant beaucoup d'avions en même temps.
- Les solutions soumises par le TCAS ne sont pas évolutives ce qui peut poser des problèmes si, pour une quelconque raison, le pilote ne suit pas la trajectoire proposée.
- Les trajectoires sont uniquement proposées dans le plan vertical ce qui entraine une variation de poussée suivie d'une surconsommation de carburant, de nouveaux calculs à réaliser pour l'équipage, et des délais plus conséquents. Or, de nos jours, le RTA (***R**equired **T**ime of **A**rrival* ou horaire prévu d'arrivée) est un enjeu majeur pour les compagnies aériennes qui tentent d'affiner au maximum la prévision de ce paramètre pour des raisons financières.

Les défaillances de ce système nous ont incité à améliorer cet équipement et à développer de nouvelles fonctions automatiques dans la gestion des conflits aériens. Nous allons maintenant détailler les différents projets d'automatisation réalisés jusqu'à maintenant.

1.2 Les différents projets d'automatisation

La littérature nous propose différentes approches à l'initiative des autorités du contrôle aérien.

1.2.1 Le projet AERA

Les différents travaux [11] de l'organisme de recherche américain Mitre Corporation sur AERA 1 (***A**utomated **E**n-**R**oute **A**ir Traffic Control*) financés par la FAA (***F**ederal **A**viation **A**dministration*) avait l'objectif principal de prévoir les trajectoires des avions et d'établir les éventuels écarts par rapport à leur plan de vol initial. Ceci correspondait donc à un outil de soutien pour les contrôleurs aériens permettant de détecter les conflits aériens.

Le projet AERA 2, quand à lui, utilise la détection de conflits réalisés par son prédécesseur et propose une solution au contrôleur qui garde ainsi son pouvoir de décision.

Enfin AERA 3 avait pour objectif de résoudre automatiquement les conflits. L'ordinateur avait donc la responsabilité de la résolution des conflits dans le plan vertical, à vitesse constante, à l'aide de manœuvre dites « *offset*[5] ». Cependant cette approche ne pouvait gérer des *clusters* supérieurs à trois avions.

Les algorithmes mentionnés ci-dessus n'ont jamais été utilisés et le projet n'a jamais vu le jour.

1.2.2 Le projet ARC 2000

Le projet ARC 2000 [11] étudié par l'organisme européen EUROCONTROL (organisation européenne pour la sécurité de la navigation aérienne qui a pour but d'harmoniser la gestion du trafic aérien dans le ciel européen) préconisait la modélisation des trajectoires de

[5] Ce type de manœuvre est détaillé dans le chapitre 3.

l'ensemble du vol sous forme de tubes en quatre dimensions (4D), incluant ainsi les incertitudes de positions et de prédictions de trajectoires.

Le principe initial consistait à ordonner les avions du plus prioritaire au moins prioritaire en fonction de différents paramètres à définir comme la vitesse ou le type de manœuvre en cours. On note l'avion 1 le plus prioritaire et l'avion N le moins prioritaire. La méthode consiste à déterminer la trajectoire de l'avion n ($n \in [2; N]$) en ayant déjà fixé les trajectoires des avions 1, 2, 3..., n-1.

Cette méthode a pour but de simplifier le calcul des trajectoires (rendu très lourd sur un trajet complet), mais ne propose pas de solution optimisée dans son ensemble. Les dernières versions de ce projet se sont limitées à des périodes de prédiction plus courtes (de l'ordre de 20 à 30 minutes), mais le projet fut finalement abandonné au cours des années 1990.

1.2.3 Le projet SAINTEX

Le projet SAINTEX [11] élaboré au CENA (**C**entre d'**É**tude de la **N**avigation **A**érienne) aborde trois types d'approches différentes en ce qui concerne la gestion de conflits :

- Un système dit « expert » qui tente de reproduire, de manière automatique, la méthode de détection et résolution des conflits utilisée par les contrôleurs aériens. Cependant ce procédé ne pouvait gérer que des conflits pouvant survenir entre deux avions.
- Un système de modélisation de trajectoire en quatre dimensions (4D) sous forme de tubes lorsque l'avion entrait dans un zone dense, ressemblant de très prêt au principe élaboré dans le projet ARC 2000.
- Un système hybride utilisant les deux précédents en tenant compte de la densité du trafic dans la zone où se situe l'avion. Le système 4D était utilisé dans les zones denses et alors que le système expert prenait le relais dans les zones plus « calmes ».

Cependant, encore une fois, ce projet n'a pas pu offrir de solution optimale dans la globalité et n'a pas pu traiter de grands clusters (c'est à dire des conflits avec beaucoup d'avions).

1.2.4 Le projet FREER

Le projet FREER (**F**ree **R**oute **E**xperimental **E**ncounter **R**esolution) [11] a vu le jour en 1995 à EUROCONTROL et a connu plusieurs évolutions :

- **FREER-1** proposait de rendre autonomes les avions dans les zones à faible densité de trafic (au dessus du FL 390, survolant la méditerranée), prenant en compte une prédiction de trajectoire de l'ordre de 6 à 8 minutes. Les éventuels conflits étaient alors gérés par un système embarqué ne pouvant cependant gérer des *clusters* à plus de quatre avions.
- **FREER-2** ajoutait un filtre pré-tactique au sol qui évitait de surcharger les zones de vol où FREER-1 agissait, en prenant compte des prédictions de trajectoires de l'ordre de 15 à 20 minutes. Ce projet a ensuite évolué pour devenir une aide au contrôleur pour la gestion des conflits impliquant deux avions dans les zones de trafic dense. Les résultats furent concluants, mais la délégation du travail afin de décharger les opérateurs aériens reste assez faible pour autant et le problème d'une solution optimale globale ne fut toujours pas résolu ici.

1.2.5 Les projets du laboratoire LARCASE

Les systèmes d'aide au pilotage de plus en plus fréquents comme le *Head-Up Display*, ou système de pilotage *tête-haute*, qui projette les informations nécessaires au pilote sur la vitre de la cabine et les radars de plus en plus précis, nous poussent à imaginer un système performant d'anticollision embarqué.

Convaincu que l'avenir du contrôle aérien passera par une automatisation, même partiel, de son système, le laboratoire LARCASE à mené à bien un projet en ce sens. Trois sujets ont été abordés :

- le suivi de trajectoire en trois dimensions à l'aide des algorithmes génétiques ;
- l'évitement des zones de non-vol (*No-Fly Zones*) à l'aide de méthodes méta-heuristiques ;
- la gestion des conflits aériens à l'aide des algorithmes génétiques.

L'objectif de ce dernier sujet, abordé dans ce mémoire, est de proposer une amélioration du système embarqué de vol lié à la gestion de conflits aériens d'une part, ainsi que de décharger le contrôle aérien au sol lors de la gestion des conflits en croisière d'autre part.

1.3 Hypothèses de travail et environnement de validation

Le but de cette section est d'établir l'environnement d'étude, les hypothèses et leurs justifications prises en compte lors de ce travail.

Tout d'abord, nous supposerons que les avions sont capables de communiquer entre eux et donc qu'un tel système international sera mis en place. Il faut lire ce mémoire comme une étude de validation de ce type de résolution de conflits. Les problèmes de faisabilité en pratique ne sont donc pas abordés en totalité dans ce rapport.

Nous considérerons que les avions se trouvent dans leur phase de croisière et nous laisserons la gestion des conflits à la charge de l'opérateur aérien dans les zones aéroportuaires. De cette manière, les vitesses seront supposées constantes et seront entrées manuellement pour un avion donné.

L'avion sera également modélisé comme un point.

Les conflits seront résolus uniquement dans le plan horizontal conformément à ce que nous souhaitons et non pas dans un souci de simplification. Les trajectoires seront affichées en trois dimensions (2D pour les coordonnées dans le plan horizontal et 1D pour le temps).

Nous supposerons également qu'aucun avion ne sera en conflit à l'instant initial t_0, l'algorithme de gestion de conflits ayant préalablement détecté ce conflit. Il est cependant à noter que même si deux avions se trouvent en conflit réel à $t = t_0$ (et non dans une trajectoire prédite), l'algorithme proposera la meilleure solution pour gérer ce ou ces conflit(s) déjà présent(s).

Les données initiales de cap et de vitesse seront connues.

Ne disposant pas de données réelles de simulateurs de trafic aérien, les trajectoires proposées dans les exemples de résolutions seront choisies afin de prendre en compte le plus grand nombre de cas de conflits, ou tout du moins les cas les plus contraignants.

Les trajectoires prédites seront considérées comme parfaites. Ainsi, un aspect important du travail futur sera d'implémenter un modèle d'incertitude de trajectoire.

Enfin les différents types de manœuvres d'évitement proposées par l'algorithme devront être réalisables par un pilote humain et compatibles avec les performances des avions actuels. Ces manœuvres, directement pré-implémentées dans le programme, seront détaillées dans le chapitre 3.

CHAPITRE 2

L'ÉVITEMENT AÉRIEN

2.1 Différentes méthodes d'approches

Dans cette section sont répertoriées les différentes approches théoriques d'optimisation de trajectoires appliquées à la résolution de conflits aériens. Cette liste n'est pas exhaustive, mais présente les méthodes les plus pertinentes sur ce sujet.

2.1.1 Méthode des forces répulsives

Karim Zeghal [18] [19] a introduit dans sa thèse une méthode basée sur les forces exercées sur chaque avion dans le but d'ordonner leurs manœuvres.
Il a définit :

- des forces attractives qui guideraient les avions ;
- des forces répulsives qui permettraient de solutionner les conflits ;
- des forces de glissement pour contourner les obstacles.

L'intensité des différentes forces qui coordonnent les trajectoires des avions devra ensuite être choisie pour s'appliquer à notre problème et respecter les contraintes énoncées dans la section 1.3.

Cette méthode robuste solutionne les conflits mais ne propose toujours pas d'optimisation globale de la solution et les manœuvres proposées ne respectent pas toujours les contraintes énoncées précédemment.

2.1.2 Méthode neuronale

Une méthode utilisant les réseaux de neurones [14] a déjà été étudiée afin de résoudre ce genre de problème. On attribut à chaque avion un réseau de neurones identique qui, en fonction des données d'entrée (distances entre les avions, trajectoires, vitesses) va indiquer à chaque pas de temps le nouveau cap à suivre pour chacun des avions. Différents cas-tests sont utilisés pour l'apprentissage du réseau de neurones.

Cependant, les cas de résolution à plus de deux avions se compliquent, surtout pour l'apprentissage des réseaux de neurones qui demandent beaucoup trop de configurations possibles de conflits.

Une méthode d'apprentissage à l'aide des algorithmes génétiques à été testée mais a finalement nécessité trop d'ajustements pour être fiable et pour proposer une solution optimale globale.

2.1.3 Méthode de programmation semi-définie

Une méthode de programmation semi-définie initiée entre autre par Eric Féron [7] et reprise par Pierre Dodin [4] a été appliquée à la résolution de conflits aériens. Elle permet de résoudre des conflits deux à deux et de proposer les caps à suivre pour chaque avion intégré dans le *cluster*.

Cependant cette méthode ne peut pas s'appliquer dans des conditions réelles puisqu'elle impose un changement de cap à tous les avions en même temps lors d'une détection de conflits.

2.1.4 Méthode de Branch and Bound par intervalles

Cette méthode générique à été utilisé par David Gianazza [8] et appliquée à la gestion du trafic aérien. Cet algorithme par *séparation et évaluation* permet, comme son nom l'indique, de diviser un problème en différents sous-problèmes solvables en recouvrant tout l'espace initial de solutions (c'est-à-dire formant recouvrement ou idéalement une partition). La phase d'évaluation permet ensuite de déterminer la solution optimale de l'ensemble du problème.

Les résultats montrent que ce procédé s'avère efficace pour les conflits intégrant peu d'avions (4 et moins), mais la solution se dégrade pour de plus grands *clusters* si l'on exige un temps de calcul acceptable et applicable au contexte (il faudrait avoir un temps de calcul au moins inférieur au temps nécessaire pour prédire un conflit).

2.1.5 L'algorithme A*

Comme l'explique Thomas Schiex et Jean Marc Alliot [1] ainsi que Géraud Granger [10], l'algorithme A^* est un algorithme de recherche dit de *meilleur en premier*. Il permet de rechercher dans un graphe le meilleur chemin à partir d'un état initial.

Dans le cas de notre étude, la base du graphe correspondrait aux données initiales des avions présents (positions, vitesse, cap,...) et chaque arc correspondrait à une trajectoire possible de l'avion. On peut ainsi visualiser les différentes trajectoires possibles créées par l'algorithme et introduire une fonction de *coût* pour évaluer la meilleure solution d'après des critères d'évaluation choisis par l'utilisateur.

Cependant les travaux de Frédéric Médioni [15] montrent que l'algorithme A^* n'est pas adapté à la résolution de conflits à plus de deux avions, non pas à cause du temps de calcul, mais plutôt de la taille de la mémoire requise.

2.2 Les algorithmes génétiques

2.2.1 Principe général

Plusieurs travaux [5] [6] [15] ont été publiés sur la gestion de l'espace aérien au sol par l'utilisation des algorithmes génétiques. Ces derniers algorithmes permettent souvent de trouver une solution globale lorsque beaucoup de données sont à traiter.

Nous avons choisi de traiter ce problème à l'aide de ces algorithmes, convaincus qu'ils répondront à nos exigences dans le but d'obtenir une solution optimale globale en tenant compte des hypothèses mentionnées dans la section 1.3.

En 1962 naissent les algorithmes génétiques avec les travaux de John Holland [12] avant d'être popularisés en 1989 par David E. Goldberg [9]. Il s'agit d'une méthode itérative basée sur les fondements de l'évolution de Charles Darwin [2] qui consiste à optimiser une fonction (c'est-à-dire d'en déterminer un optimum). Ainsi, pour faire une analogie avec ce concept biologique, l'utilisation de termes liés à ce domaine tels que *chromosome*, *gène*, *mutation*, *croisement*,... agrémentera les explications.

Afin de mieux comprendre le principe des algorithmes génétiques, l'exemple d'une fonction $f(t)$ pour $t \in [tmin\ ;\ tmax]$ sera considéré; l'objectif étant de retrouver le maximum de la fonction $f(t)$ à l'aide de ces algorithmes. Comme la figure 2.1 nous le montre, la solution est visible sur le graphique et il s'agit de $f(topt2)$, maximum de $f(t)$ pour $t \in [tmin\ ;\ tmax]$. L'exemple à été choisi de la sorte afin de retrouver un résultat déjà établi et de simplifier la compréhension du processus.

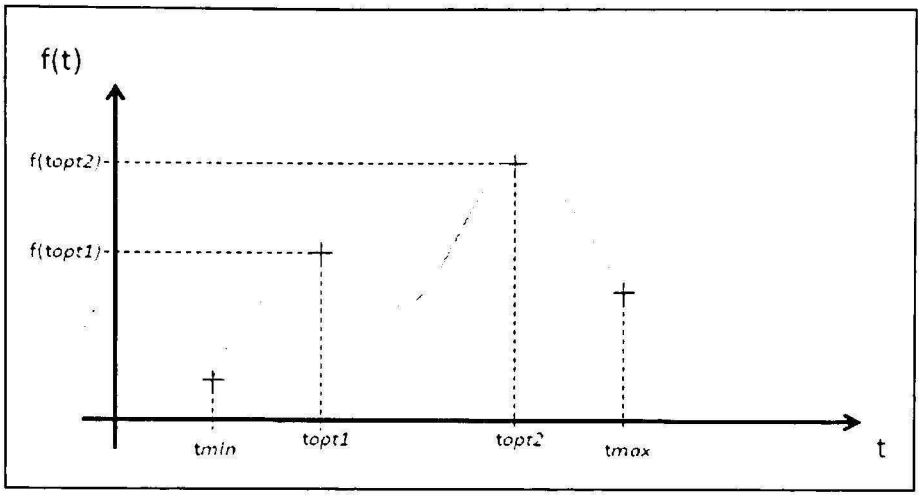

Figure 2.1 Exemple d'une fonction f(t).

À partir d'une population de dimension *n* dont les éléments sont appelés *individus* et d'une fonction d'adaptation communément appelée *fitness* définie en fonction de la problématique, l'algorithme va suivre les étapes illustrées par la figure 2.2.

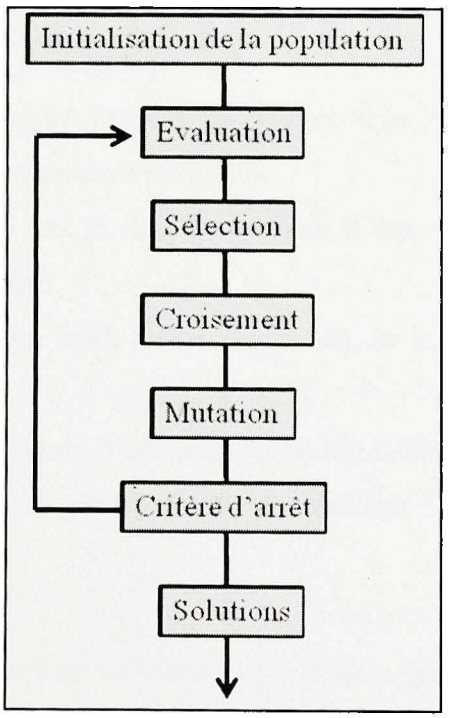

Figure 2.2 Schéma de principe d'un algorithme génétique.

On définit par *chromosome* le codage de chaque individu. Dans notre exemple, une paire de coordonnées (ou *gêne*) représente un individu et $(t,f(t))$ avec $t \in [tmin ; tmax]$ définit un chromosome.

On appelle *gêne* une séquence d'un chromosome. Soit [12, 8] les coordonnées d'un point de $f(t)$ (c'est-à-dire que $f(12)=8$), alors 12 et 8 sont les gênes de ce chromosome.

À chaque itération, une nouvelle génération d'individus est créée, la population initiale représentant la génération 0. Les étapes de l'algorithme génétique sont les suivantes :

1) Une population (de dimension n) est générée aléatoirement dans l'espace des solutions.
2) Chaque individu est évalué par la fonction d'adaptation ; les solutions les plus adaptées selon la fonction de *fitness* sont classées les « meilleures ».
3) $n/2$ couple(s) d'individus sont sélectionnés aléatoirement parmi la population sachant qu'un individu a une probabilité d'être sélectionné proportionnelle à son adaptation (évaluée par la fonction d'adaptation).
4) Chaque couple d'individus est croisé pour donner deux individus enfants ; on retrouve donc une population de n nouveaux individus.
5) Chaque individu possède une probabilité définie d'être muté, c'est-à-dire de voir ses gênes changer aléatoirement.
6) Le critère d'arrêt peut être sujet à une stagnation de la population et/ou un nombre d'itérations.
7) La solution finale est composée d'une population (de même dimension que la population initiale) dans laquelle l'on peut ressortir le meilleur individu comme solution du problème.

Ce principe est basé sur la théorie de Darwin qui pousse les individus les plus adaptés à survivre le plus longtemps pour se reproduire. Nous allons analyser plus en détail le principe de fonctionnement étape par étape en se basant sur notre exemple précédemment explicité.

2.2.2 Description détaillée

2.2.2.1 Codage des données

Le codage des données (ou des individus) est un aspect important dans la mise en place d'un algorithme génétique. Pendant longtemps, le langage binaire a été préféré au codage réel car il permettait de faciliter les opérations de croisement et de mutation, se prêtant ainsi à un découpage simple (voir la sous-section 2.2.2.5).

Cependant, ce type de codage présente un inconvénient majeur qui peut influer sur l'opération de croisement : deux chromosomes proches en termes de distance de Hamming ne correspondent pas forcément à deux individus voisins dans l'espace de solutions réelles.

La distance de Hamming, définie par Richard Hamming, associe le cardinal (dans le cas présent d'un ensemble fini, le cardinal définit le nombre d'éléments de l'ensemble) de l'ensemble des symboles d'une première chaine de caractères qui diffèrent d'une seconde chaine de caractères. Par exemple, la distance de Hamming entre MAMAN et MARIN est égale à deux (deux caractères diffèrent entre ces deux suites de symboles).

Prenons l'exemple des chromosomes de coordonnées [28 ; 8] et [16 ; 8]. Ces coordonnées codées en binaire deviennent : [11100 ; 1000] et [10000 ; 1000]. Ces deux chromosomes pas nécessairement proches dans l'espace réel, possèdent pourtant une courte distance de Hamming (une distance de 2) lorsque les coordonnées sont codées en binaire ([11100 ; 1000] et [10000 ; 1000]).

Il est important d'utiliser un codage qui répond à la structure des données du problème et de conserver, quand cela est permis, le code initial des données sans passer par un codage intermédiaire qui complexerait la méthode.

2.2.2.2 Initialisation de la population

Le choix de la population initiale de l'algorithme génétique va conditionner sa rapidité à déterminer la solution du problème, soit l'optimum recherché. Si ce dernier est totalement inconnu, il sera alors normal d'initier aléatoirement la population. Il est cependant recommandé, quand cela est possible, d'affiner l'espace des solutions dès cette étape en y appliquant les contraintes du problème afin d'accélérer la convergence de l'algorithme vers la solution finale.

Dans notre exemple, les individus de la génération 0, c'est-à-dire de la population initiale, seront contraints de se trouver sur la courbe $f(t)$ ce qui aura pour effet d'affiner la recherche de la solution et d'accélérer sa convergence.

C'est également à cette étape que la taille de la population sera fixée. Une population importante permettra d'affiner la recherche mais augmentera considérablement le nombre d'opérations. Il faut donc trouver un compromis entre la taille de la population et le nombre d'itérations de l'algorithme.

2.2.2.3 Évaluation de la population

Il faut cependant faire attention à ne pas trop contraindre la population initiale afin de ne pas systématiquement éliminer des individus qualifiés de « mauvais » par la fonction d'évaluation. En effet des individus mal adaptés au problème pourront générer des individus admissibles lors de l'étape de croisement et deviennent nécessaires au processus de brassage génétique. Il est donc difficile de gérer les contraintes efficacement et un savant dosage, souvent effectué par expérience, doit être effectué afin d'optimiser l'algorithme.

Durant cette étape d'évaluation de la population, chaque individu se verra attribuer un *poids* évaluant son admissibilité au problème à l'aide de la fonction de *fitness*.

Dans l'exemple étudié, le maximum de la fonction $f(t)$ est recherché, ainsi les chromosomes sont évalués selon la valeur de leur ordonnée (le plus grand étant le meilleur puisqu'un maximum est recherché).

2.2.2.4 Sélection de la population

L'opérateur de sélection va permettre d'identifier les meilleurs individus de la population générée et ainsi d'éliminer les plus mauvais. Il existe plusieurs méthodes de sélection ; les plus couramment utilisées sont les suivantes :

- ***Roulette Wheel Selection*** [9] ou **Sélection par la méthode de « la roulette »** (que l'on retrouvera parfois dans la littérature sous le nom de *méthode de Monte-Carlo*) s'inspire du célèbre jeu de casino. On associe à chaque individu une section de la roulette proportionnelle à son admissibilité évaluée précédemment. Les individus sont ensuite sélectionnés aléatoirement comme si l'on « tournait la roue ». Les tirages sont ainsi pondérés par la qualité des individus.

  Soit une population de 6 individus dans laquelle A est évalué comme le meilleur individu et F le moins bon parmi toute la population (A, B, C, D, E, F). La « roue » obtenue serait celle illustrée dans la figure 2.3 (les valeurs des pourcentages sont choisies aléatoirement pour illustrer la méthode).

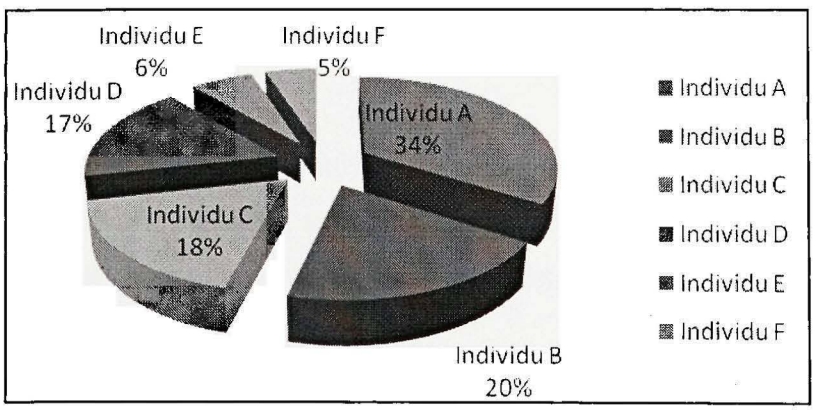

Figure 2.3 Schéma d'une *Roulette Wheel Selection.*

- ***Stochastic Remainder Without Replacement Selection*** [9] ou **méthode du reste stochastique.** Dans le cas d'une petite population (peu d'individus), les résultats obtenus par la « méthode de la roulette » ne répondent pas forcément aux attentes, en raison du faible nombre de tirages effectués. Pour palier à cette situation, la méthode du reste stochastique peut être utilisée.

  Pour chaque individu i de la population, on calcule la partie entière $E(r_i)$ où r_i est le rapport de la *fitness* fit_i de l'individu sur la moyenne des *fitness* fit_{moy}. On applique ensuite la méthode de la roulette ; expliquée ci-dessus ; sur les individus associés à leur *fitness* $r_i - E(r_i)$.

- La **sélection par rang** [3], analogue à la *Roulette Wheel Selection*, se base également sur l'implémentation d'une roulette non plus remplie en fonction de l'admissibilité des individus, mais en fonction de leur rang.

  Un rang est attribué à chaque individu en fonction de sa qualité : un rang faible pour les individus de piètres qualités et un rang élevé pour les meilleurs individus. Les rangs varient de 1 à n, avec n la dimension de la population, la section attribuée à chaque individu sur la roue étant proportionnelle à son rang.

- La **sélection par tournoi(s)** [3] : deux individus pris aléatoirement dans la population s'affrontent (on les compare en termes d'admissibilité) et le meilleur est ensuite conservé en tant que futur individu parent.

  On peut réitérer cette action autant de fois que nécessaire pour la sélection d'un individu et/ou confronter plusieurs individus en même temps (d'où le terme de tournoi). Cette sélection donne ainsi une chance aux individus les plus faibles de participer à l'amélioration de la future génération.

2.2.2.5 Opérateur de croisement

L'objectif de l'étape de croisement ou *crossover* est de brasser le code génétique afin d'enrichir la population. On obtient traditionnellement deux individus enfants à partir de deux individus parents, ce qui permet de conserver la taille de la population initiale. On passe, à cette étape, de la génération N à la génération N+1.

Les chromosomes enfants disposent ainsi, comme le montre les figures 2.4 et 2.5, du code génétique (des gênes) des deux parents.

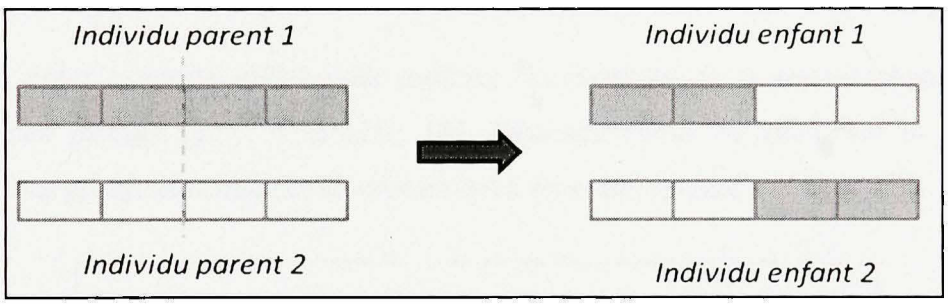

Figure 2.4 Exemple de croisement à un seul point.

Le principe du *slicing crossover* exposé à la figure 2.4 consiste à prendre aléatoirement un point de découpe dans les chromosomes de chaque parent et de transmettre une partie des gênes des deux parents à un enfant, puis les autres parties au second enfant issu de la reproduction.

Dans l'exemple montré sur la figure 2.4, nous n'avons choisi qu'un seul point de croisement, mais il est fréquent d'effectuer ce que l'on appelle du croisement à points multiples comme l'illustre la figure 2.5.

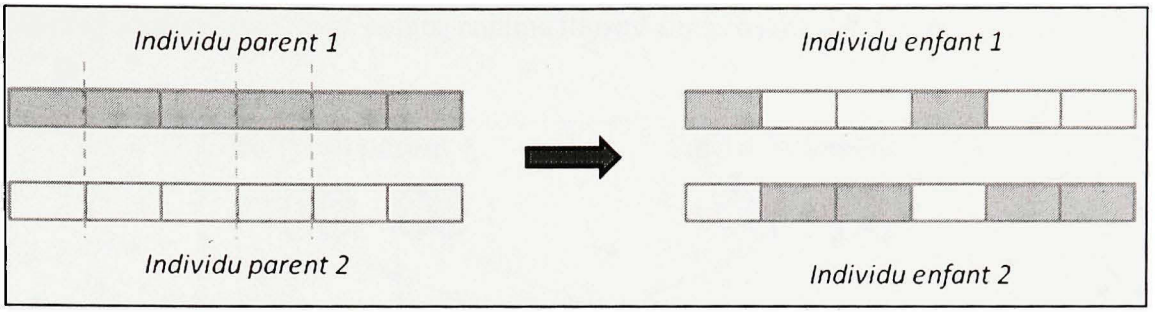

Figure 2.5 Exemple de croisement à trois points.

Chaque gène (correspondant à une petite case sur les figures 2.4 et 2.5) représente un bit en codage binaire, un chiffre ou une série de chiffres en codage réel.

Reprenons notre exemple initial pour montrer les résultats de la reproduction des deux chromosomes parents [12 ; 7] et [25 ; 18]. Dépendamment du problème et des choix effectués nous pourrions effectuer un croisement à un point, illustré à la figure 2.6 :

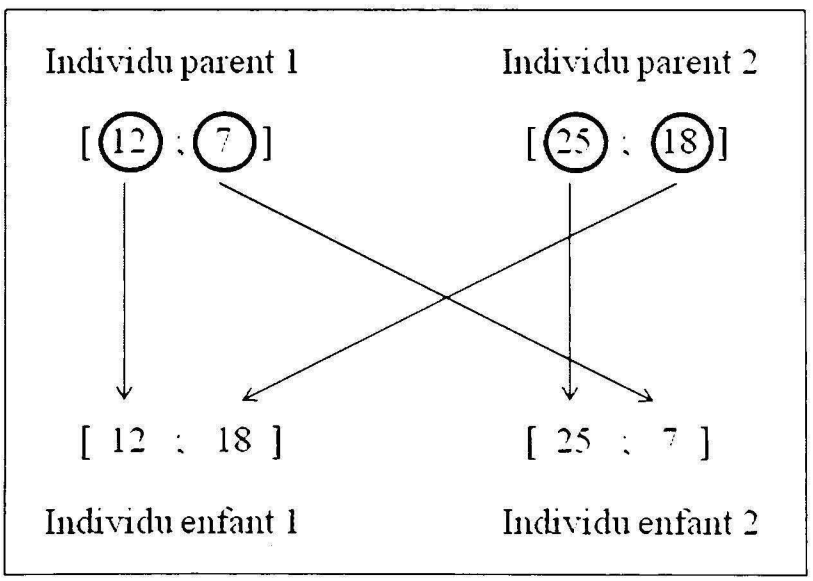

Figure 2.6 Exemple de croisement à un point.

Ou bien un croisement à trois points, comme illustré sur la figure 2.7 :

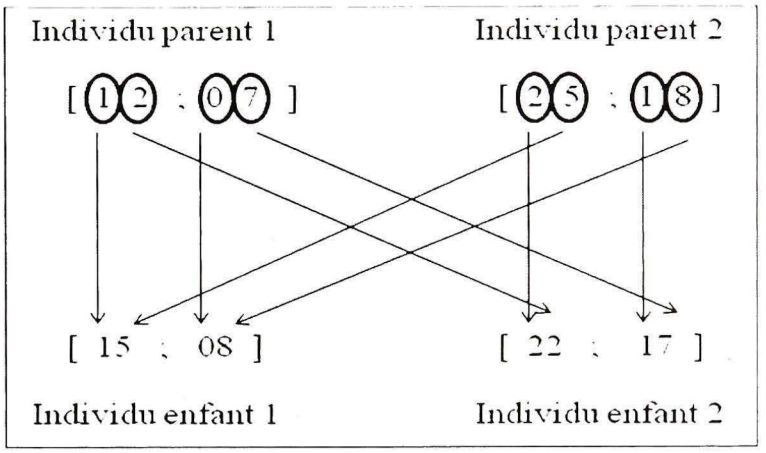

Figure 2.7 Exemple de croisement à trois points.

2.2.2.6 Opérateur de mutation

Pour éviter que l'algorithme converge vers un optimum local (t_{opt1} dans l'exemple de la figure 2.1), le principe de mutation aléatoire est introduit. Un facteur de mutation est fixé (qui ne devrait pas avoir une valeur trop élevée sous peine de basculer dans une recherche entièrement aléatoire) afin que chaque gène possède un pourcentage de chance de muter.

Prenons l'exemple d'un chromosome [12 ; 7]. L'algorithme peut faire muter le gène 12 ou bien une partie de ce gène (1 ou 2), et il en va de même pour tous les gênes de chaque chromosome. La mutation est aléatoire, même si l'opérateur peut contraindre en partie l'ensemble des gênes mutées. Dans le premier cas cité, si le gêne 12 mute, nous aurions l'exemple illustré sur la figure 2.8.

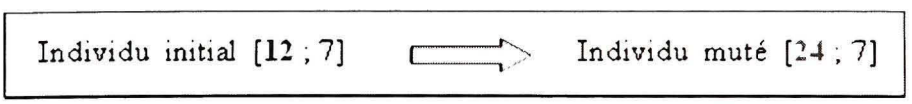

Figure 2.8 Exemple d'une mutation.

Le résultat de la mutation (ici le nombre 24) est aléatoire.

2.2.2.7 Gestion des contraintes

Les opérateurs de reproduction que sont le croisement et la mutation génèrent de nouveaux individus. Tout comme les contraintes du problème peuvent être appliquées lors de la formation de la population initiale, elles peuvent s'appliquer aux fonctions de croisement et de mutation afin de générer des individus répondant au problème. La gestion de ces contraintes est primordiale et doit être subtilement effectuée pour éviter de « sur-contraindre » l'algorithme ou de le rendre trop aléatoire.

2.2.2.8 Critère d'arrêt et solution finale

Une seconde évaluation (après celle développée à la sous-section 2.2.2.3) est introduite dans cette fonction qui va déterminer le ou les critères d'arrêt de l'algorithme. Fréquemment l'algorithme cessera lorsqu'une solution satisfaisante (à l'utilisateur de définir ce qu'il entend par satisfaisante) sera trouvée, lorsque la population sera uniforme (peuplée du même individu) ou lorsqu'un nombre prédéfini d'itérations aura eu lieu.

Une population finale est ainsi déterminée et le meilleur individu pour le problème défini constitue la solution finale trouvée par l'algorithme génétique.

2.2.3 Améliorations possibles

Des améliorations standards des fonctions détaillées ci-dessus qui peuvent être appliquées à un algorithme génétique en fonction du problème posé sont présentées dans les sections suivantes.

2.2.3.1 Élitisme

À l'issu des fonctions de croisement et de mutation, le risque de perdre la meilleure solution de la population précédente est présent. Une amélioration classique appelée ***élitisme*** consiste à préserver le ou les meilleurs individu(s) de la génération précédente pour le ou les introduire dans la nouvelle génération suite à l'application des opérateurs de croisement et de mutation comme montré sur la figure 2.8.

Le meilleur individu sera ainsi toujours conservé, et cette amélioration aura pour effet d'augmenter la convergence de l'algorithme.

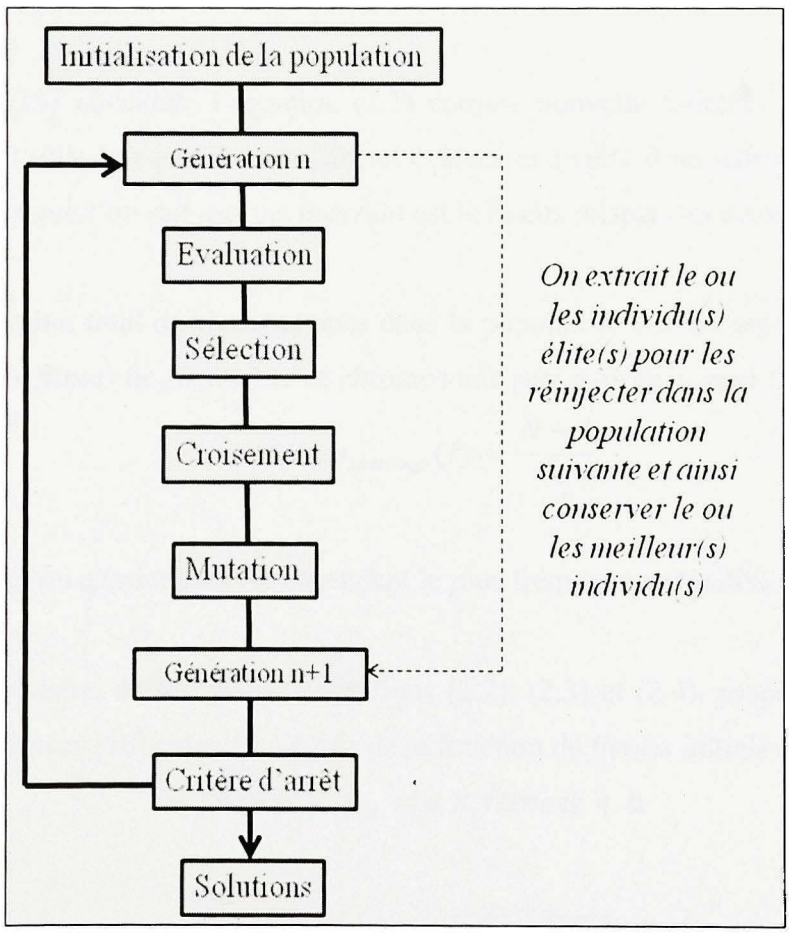

Figure 2.9 Schéma de principe de l'*élitisme*.

2.2.3.2 Scaling

Pour éviter qu'un bon individu (c'est-à-dire possédant une *fitness* élevée) devienne prépondérant dans la population et finisse par la rendre uniforme (un individu peut se reproduire avec lui-même et ainsi donner les mêmes individus enfants), on utilise un principe de ***scaling*** ou « mise à l'échelle » lors de la phase de sélection.

Tel qu'expliqué précédemment, la fonction de *fitness* est utilisée pour la sélection de la population. Le *scaling* propose de mettre à l'échelle cette fonction de *fitness* et ainsi réduire la *fitness* des chromosomes les mieux adaptés aux problèmes. Les techniques de *scaling* les plus utilisées sont les suivantes :

- Le *ranking* [15] considère l'équation (2.1) comme nouvelle fonction de *fitness*. Cette technique est utile lorsque l'on ne sait pas évaluer la qualité d'un individu par rapport à un autre mais que l'on sait que cet individu est le mieux adapté des deux.

  Soit N le nombre total de chromosomes dans la population et i le rang du chromosome, la fonction de *fitness* de *scaling* de ce chromosome sera exprimée ainsi :

$$fitness_{scaling}(i) = \frac{N - i}{N} \tag{2.1}$$

Le *scaling* linéaire ou exponentiel est cependant le plus fréquemment utilisé.

- Le *scaling* linéaire, définie par les équations (2.2), (2.3) et (2.4), propose une nouvelle fonction de *fitness* [16] calculée à partir de la fonction de *fitness* initiale :

$$fitness_{scaling} = a \times fitness + b \tag{2.2}$$

  Où :

$$a = \frac{\max(fitness_{scaling}) - \min(fitness_{scaling})}{\max(fitness) - \min(fitness)} \tag{2.3}$$

$$b = \frac{\min(fitness_{scaling}) \times \max(fitness) - \min(fitness) \times \max(fitness_{scaling})}{\max(fitness) - \min(fitness)} \tag{2.4}$$

- Le *scaling* exponentiel est défini par les équations (2.5) (2.6) [16].

Soit *fitness* la fonction de fitness, n la génération courante de la population et N le nombre total de générations, alors la *fitness de scaling* sera, en tenant compte de la méthodologie et des résultats et montrés par Nicolas Durand [6] pour le choix du facteur k :

$$fitness_{scaling} = fitness^{k(n)} \tag{2.5}$$

Où :

$$k(n) = (tan\,[(\frac{n}{N-1})\frac{\pi}{2}])^{0.1} \tag{2.6}$$

Comme le montre la figure 2.9, lorsque :

- $k(n)<1$, les écarts sont réduits et l'algorithme se comporte comme une recherche aléatoire, ce qui lui permet d' observer tout l'espace des solutions.
- $k(n)=1$, le *scaling* n'a pas lieu.
- $k(n)>1$, les écarts sont amplifiés et les meilleurs individus prennent le dessus.

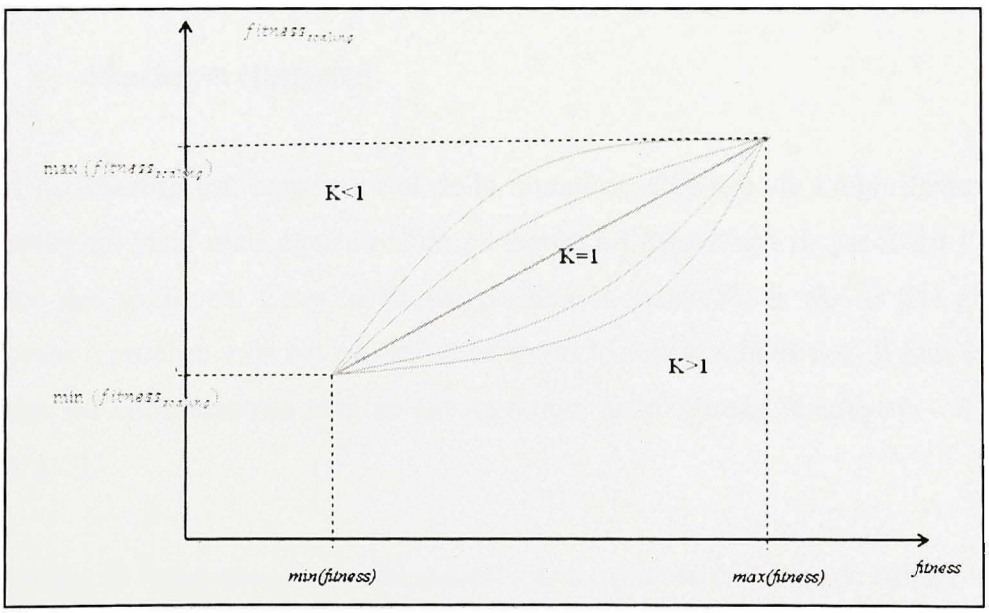

Figure 2.10 Allure de la *fitness* de *scaling* exponentielle en fonction de la « *fitness* réelle ».

Suite au choix de la fonction $k(n)$, comme le montre la figure 2.11, les individus ne sont pas favorisés dans les premières générations et laissent à l'algorithme la possibilité d'explorer tout l'espace des solutions avant que les écarts ne soient de plus en plus amplifiés et que les meilleurs individus prennent le dessus au cours des générations.

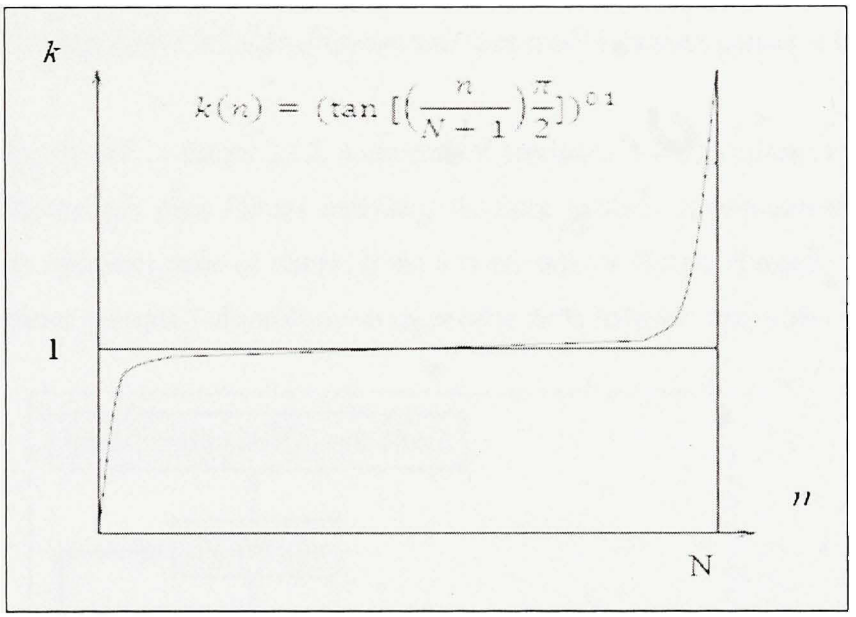

Figure 2.11 Allure du facteur *k* au cours des générations *n*.

2.2.3.3 Sharing et clustering

L'objectif du ***sharing*** est, comme celui de la mutation, d'éviter que l'algorithme converge vers un optimum local mais également de permettre à l'algorithme de parcourir l'ensemble de l'espace des solutions. Cette technique consiste à amoindrir la *fitness* des groupes de chromosomes « proches » de cet ou ces optimum(a) local(aux). Pour cela il faut évaluer les écarts entre les chromosomes afin de les regrouper sous formes de *clusters* : il s'agit du ***clustering*** [17].

Il est important de noter que cette technique est assez coûteuse en temps de calcul.

2.2.3.4 Recuit simulé

Le **recuit simulé** [13] est une technique qui, tout comme l'*élitisme*, permet de conserver les individus les mieux adaptés après la reproduction. Cette technique est appliquée juste après l'étape de croisement ; les individus enfants issus de la reproduction sont comparés à leurs parents en termes d'adaptabilité à l'aide d'un tournoi. Les meilleurs sont conservés.

Cette technique, décrite sur la figure 2.12, a cependant tendance à uniformiser la population sans laisser de chances aux plus faibles individus de faire évoluer la population. Ainsi un facteur aléatoire est introduit dans ce choix. Il est à noter que ce facteur diminue au fur et à mesure des générations lorsque l'algorithme se rapproche de la solution optimale.

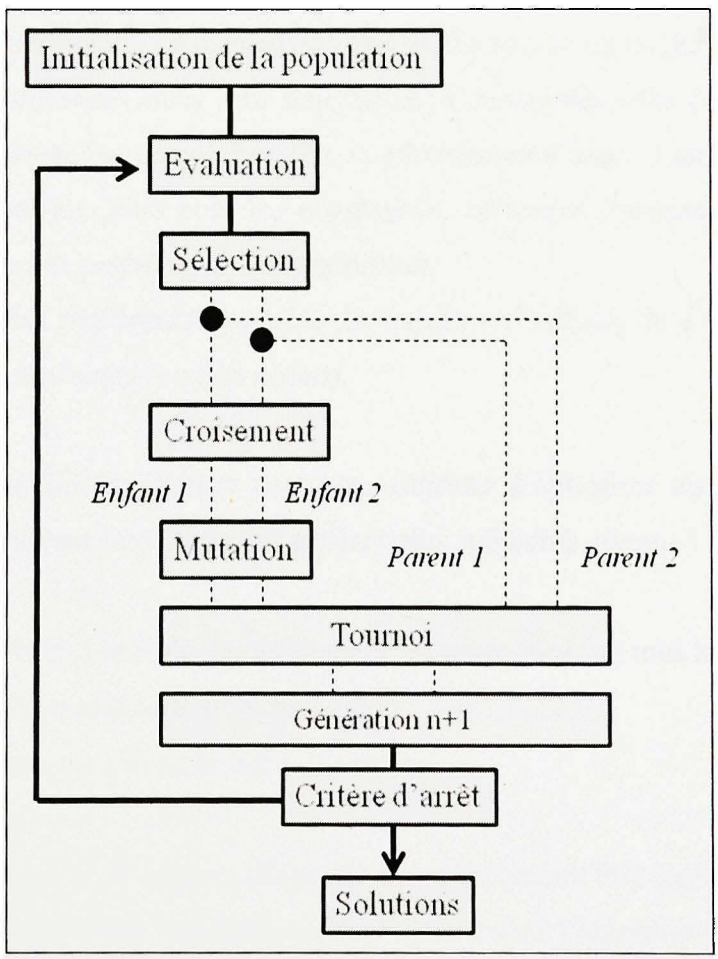

Figure 2.12 Schéma de principe du recuit simulé.

CHAPITRE 3

IMPLANTATION DE LA MÉTHODE

3.1 Objectifs d'optimisation

L'algorithme ainsi développé devra répondre aux contraintes explicitées dans le paragraphe 1.3, mais également suivre certains critères exigés ci dessous.

S'y trouve des critères objectifs :

- résoudre tous les conflits aériens présents : critère primaire de l'algorithme qui optimisera les critères suivants à partir du moment où ce dernier sera rempli.
- minimiser le nombre total de manœuvres afin de faciliter le travail du ou des pilote(s).
- minimiser les distances dues aux trajectoires d'évitements afin de réduire les coûts consommation ainsi que de vol. En effet, la prévision de l'heure d'arrivée ou RTA est un enjeu majeur de nos jours pour les compagnies aériennes souhaitant limiter les coûts excédants grâce à la précision de cette prévision.
- rejoindre le plus rapidement possible la trajectoire initiale de l'avion afin d'éviter d'engendrer de nouveaux conflits aériens.

L'algorithme de gestion de conflits aura pour objectif d'optimiser les différents facteurs énoncés auxquels peuvent se rajouter des critères plus subjectifs comme :

- déterminer s'il vaut mieux dévier faiblement les trajectoires de tous les avions plutôt que de dévier plus fortement celle d'un seul avion ;
- prendre en compte ou non de la taille de l'avion.

Cette liste non exhaustive de critères subjectifs devra cependant être légiférée par les acteurs du contrôle aérien avant d'être implémentée directement dans l'algorithme embarqué.

3.2 Modélisation du problème

3.2.1 Architecture

Le problème étant maintenant clairement posé et défini, la méthode déterminée, il ne reste plus qu'à l'implémenter en respectant les hypothèses du sujet ainsi que les objectifs fixés.

La résolution des conflits s'effectuera avec un programme d'algorithme générique appliqué au problème à l'aide du logiciel *Matlab 2007*[6]. Les manœuvres rendues possibles pour les avions seront préalablement implémentées et sont présentées dans la sous-section 3.2.2.

3.2.2 Manœuvres

3.2.2.1 Existant

Si nous nous intéressons aux manœuvres d'évitement actuellement proposées par le TCAS et le contrôle aérien, nous remarquons que celles-ci se trouvent d'une part simples à effectuer par les pilotes mais également facilement identifiables par un autre pilote situé dans le même espace aérien.

Lors du filtre tactique un contrôleur aérien dispose de quatre types d'ordre différents :

- un changement de cap ;
- une proposition de paliers intermédiaires pour les avions en phase de montée ou descente ;
- un changement de niveau de vol ;
- une variation de vitesse (particulièrement pour les avions en phase de descente).

---

[6] *Matlab* est à la fois un langage de programmation et un logiciel de développement souvent utilisé lors des phases de développement de projet. Conçu vers la fin des années 1970 il est actuellement commercialisé par la société américaine *The MathWorks*.

Comme l'on souhaite résoudre des conflits sur le plan horizontal, à vitesse constante en phase de croisière, il ne reste plus que la possibilité d'avoir une variation de cap.

Lors du filtre d'urgence le TCAS propose des ordres basiques :

- *« maintenez la vitesse verticale »* ;
- *« augmentez la montée »* ;
- *« augmentez la descente »* ;
- *« montez »* ;
- *« descendez »* ;
- *« ajustez la vitesse verticale »* ;
- *« surveillez la vitesse verticale ».*

Les manœuvres opérationnelles d'évitement proposées doivent être simples d'exécution et de compréhension, compatibles avec les performances des avions et exécutables par un pilote humain.

Nous nous sommes limitées à deux types de manœuvres standards [5] : une manœuvre dite « point-tournant » (section 3.2.2.2) et une manœuvre dite « offset » (section 3.2.2.3).

3.2.2.2 Point tournant

À la vue de la longueur des segments de route proposée par une telle manœuvre, celle-ci sera approximée continue par morceaux, tout comme l'affiche le FMS[7] au pilote (figure 3.1).

[7] Le FMS ou ***F**light **M**anagement **S**ystem* (système de gestion de vol) est un logiciel embarqué qui a pour but d'assister le ou les pilotes en fournissant des données de vol.

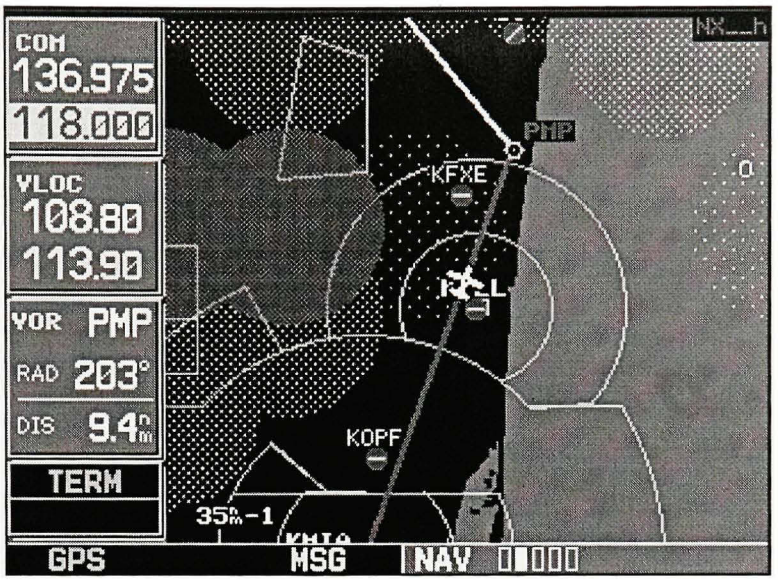

Figure 3.1 Affichage d'une trajectoire proposée par un FMS.
Tiré de *Dynasty Aviation Technologies* (2009)

La trajectoire d'évitement représentée à la figure 3.2 engendre trois changements de cap à réaliser par le pilote aux différents points P1, P2 et P3.

En ligne pointillée est indiquée la trajectoire initiale et en trait plein la trajectoire approchée déviée par une manœuvre de type « *point-tournant* ».

α représente l'angle de lacet[8] de la première manœuvre.
P1 correspond au point où débute la première manœuvre qui s'écarte de la trajectoire initiale.
P2 correspond au point où débute la seconde manœuvre qui débute le retour vers la trajectoire initiale.
P3 correspond au point où débute la troisième manœuvre afin de reprendre la route[9] initiale.
P'2 est défini de tel sorte que les distances ||P1 P2|| et ||P1 P'2|| soient identiques.
Portho2 est le projeté orthogonale de P2 sur la droite (P1 P3).
P'3 est défini de tel sorte que : ||P1 P'3|| = ||P1 P2|| + ||P2 P3||.

[8] L'angle de lacet est l'angle formé par une rotation autour de l'axe vertical (de l'avion dans le cas présent)
[9] La route, en navigation, est la direction suivie par l'avion et définie par un angle à partir du Nord géographique puis en tournant dans le sens trigonométrique inverse.

La distance d représente l'écart de position à l'issu de la manœuvre par rapport à la trajectoire initiale ou en d'autres termes l'allongement de trajectoire qu'induit la trajectoire déviée et d est définie par l'équation (3.1).

$$d = 2 \times \| Portho2\ P'2 \| \tag{3.1}$$

$$d = 2 \times (\| P1\ P'2 \| - \| P1\ Portho2 \|)$$

$$d = 2 \times (\| P1\ P2 \| - \| P1\ P2 \| \times cos(\alpha))$$

$$\boxed{d = 2 \times \| P1\ P2 \| \times [1 - cos(\alpha)]}$$

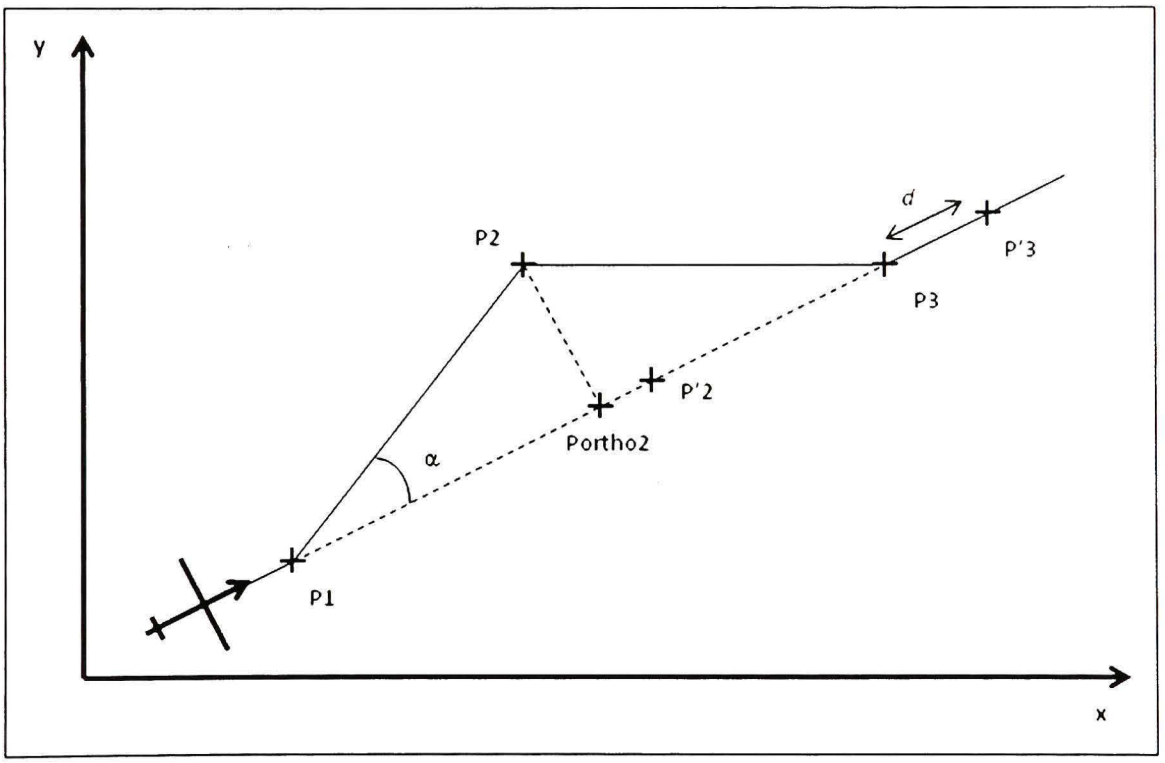

Figure 3.2 Modélisation graphique d'une manœuvre de type « *point tournant* ».

Afin de répondre aux hypothèses fixées ainsi qu'au confort des éventuels passagers, un angle α d'une valeur maximale de 30° sera fixé [11].

3.2.2.3 Offset

La trajectoire de type « *offset* », également approximée par morceaux, engendre quatre changements de cap à réaliser par le pilote aux points P1, P2, P3 et P4, et se trouve modélisée à la figure 3.3.

En pointillé est indiquée la trajectoire initiale et en trait plein la trajectoire approchée déviée par une manœuvre de type « *offset* ».

α représente l'angle de lacet de la première manœuvre.
P1 correspond au point où débute la première manœuvre qui s'écarte de la trajectoire initiale.
P2 correspond au point où débute la seconde manœuvre.
P3 correspond au point où débute la troisième manœuvre qui débute le retour vers la trajectoire initiale.
P4 correspond au point où débute la quatrième manœuvre afin de reprendre la route initiale.
P'2 est défini de tel sorte que les distances ||P1 P2|| et ||P1 P'2|| soient identiques.
P'3 est défini de tel sorte que : ||P1 P'3|| = ||P1 P2|| + ||P2 P3||.
P'4 est défini de tel sorte que : ||P1 P'4|| = ||P1 P2|| + ||P2 P3|| + ||P3 P4||.

La distance *d* représente l'écart de position à l'issu de la manœuvre par rapport à la trajectoire initiale ou en d'autre terme l'allongement de trajectoire qu'induit la trajectoire déviée. La distance *d*, définie à l'équation (8), s'applique à une manœuvre de type « *point-tournant* » pour un même angle α ainsi qu'une même distance ||P1 P2||.

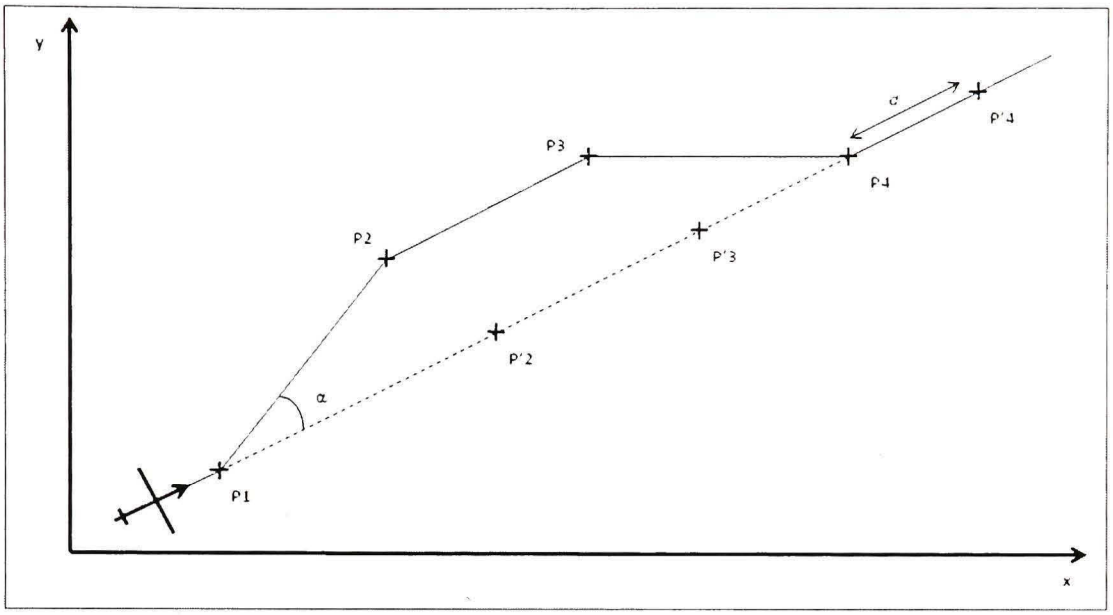

Figure 3.3 Modélisation graphique d'une manœuvre de type « *offset* ».

3.3 Implantation des algorithmes génétiques

3.3.1 Principe

L'algorithme génétique ainsi développé détermine l'apparition d'éventuels conflits à partir de la trajectoire initiale de l'avion. La fenêtre de prédiction temporelle est entrée manuellement par l'utilisateur. Plus celle-ci sera grande, plus les conflits seront prédits tôt. Ainsi les trajectoires d'évitements proposées seront mieux optimisées (suivants les différents facteurs déjà explicités), mais en contrepartie le temps de calcul augmentera. Cette fenêtre de prédiction devient un autre paramètre de l'algorithme à optimiser.

Si aucun conflit n'est décelé lors de la prédiction, la trajectoire reste inchangée et l'algorithme réitère l'opération. Si un conflit est détecté, l'algorithme propose les trajectoires d'évitements à suivre pour les avions engagés dans le *cluster*.

Il est à noter que tous les résultats affichant l'influence des différents facteurs sont explicités au chapitre 4.

3.3.2 Choix de résolution

3.3.2.1 Population initiale

Soit n le nombre d'avions embarqués dans le conflit, avec $n>1$.

P_{1_i} représente les coordonnées de l'avion i (modélisé comme un point) à l'instant t_{1_i}.

P_{2_i} représente les coordonnées de l'avion i à l'instant t_{2_i}.

α_i représente l'angle en degré de déviation de cap de l'avion i.

P_{3_i} représente les coordonnées de l'avion i à l'instant t_{3_i}.

$0 \leq t_{1_i} \leq (t_{max} - 1)$ avec t_{max} la taille de la fenêtre de prédiction temporelle.

De plus, $0 \leq t_{1_i} \leq t_{2_i} \leq t_{3_i}$ et $\alpha_i \leq 30°$.

Un chromosome est défini de la manière suivante : $\begin{pmatrix} t_{1_1} & t_{1_2} & \cdots & t_{1_N} \\ \alpha_1 & \alpha_2 & \cdots & \alpha_N \\ t_{2_1} & t_{2_2} & \cdots & t_{2_N} \\ t_{3_1} & t_{3_2} & \cdots & t_{3_N} \end{pmatrix}$.

Les coordonnées $(x ; y)$ de l'avion sont exprimées en deux dimensions et l'angle α en degré.

P_{i_n} et α_i sont définis sur la figure 3.4.

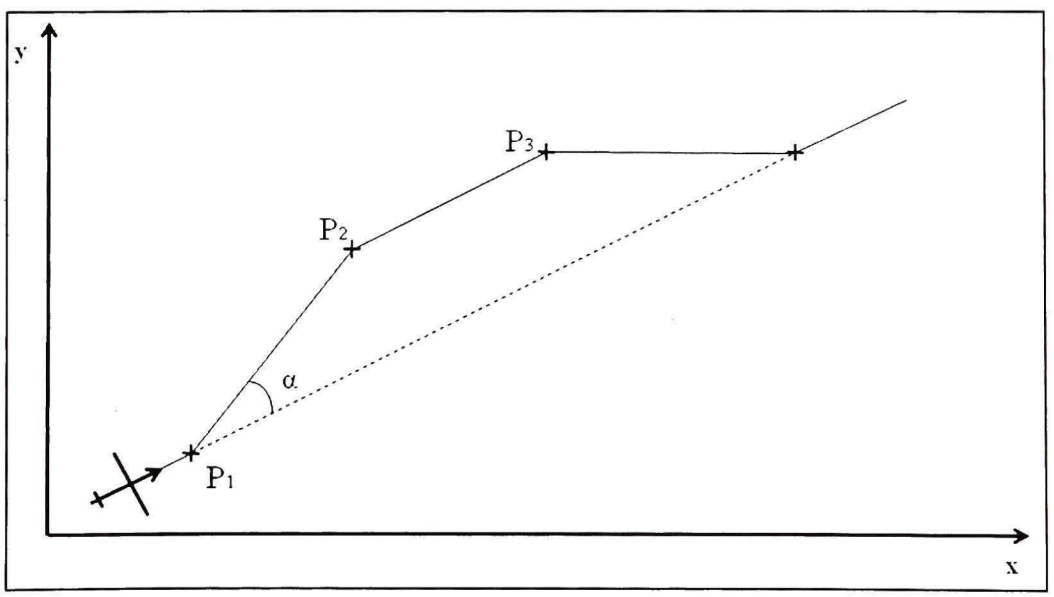

Figure 3.4 Schéma de manœuvre.

On remarque que si $t_2 = t_3$ une manœuvre de type « *point tournant* » sera effectuée alors que si $t_2 \neq t_3$, une manœuvre de type « *offset* » aura lieu.
Si $t_1 = t_2$ l'algorithme pose $t_1 = t_2 = t_3$ et la trajectoire ne sera pas déviée.

Enfin le codage des données à été conservé en chiffres réels sans nécessiter de transformation en codage binaire dans l'étape de croisement.

3.3.2.2 Fonction d'évaluation

La fonction d'évaluation va influencer l'attitude des résultats donnés par l'algorithme. Elle sera séparée en différentes sous fonctions d'évaluations traduisant les objectifs d'optimisation établis à la section 3.1.

Une première *fitness* dite *de conflit* est mise en place afin d'évaluer la distance qui sépare chaque couple ou paire d'avions. Plus la distance sera supérieure et proche de la distance minimale fixée par l'opérateur (5 Nm dans le plan horizontal), plus la valeur de la *fitness* associée sera élevée pour les chromosomes évalués. On évalue donc dans cette première sous-fonction l'écart entre la position des différents avions à chaque instant ainsi que l'écart entre la trajectoire initiale et celle proposée à minimiser.

Chaque chromosome possède sa propre *fitness totale* et ses propres sous fonction de *fitness* détaillées ci-dessous. Lorsqu'aucun conflit n'est détecté par le biais de cette sous-fonctions de *fitness*, la *fitness totale* est égale à la *fitness de conflit*. Si un conflit est décelé, la *fitness totale* est égale à la somme des *fitness de manœuvre*, de *distance*, de *retour* et de *conflits*.

La *fitness de manœuvre* évalue le nombre de manœuvres effectuées par l'ensemble des avions. Plus ce nombre sera faible, plus la valeur de la *fitness de manœuvre* associée sera élevée.

La *fitness de retour* évalue l'instant où l'avion récupère sa trajectoire initiale. Plus la somme des temps de retour de tous les avions engagés dans le *cluster* sera petite, plus la valeur de la *fitness de retour* associée sera élevée.

La *fitness de distance* évalue l'écart entre la trajectoire d'évitement proposée et la trajectoire initiale. Plus la somme de ces distances calculées pour tous les avions sera petite et plus la valeur de la *fitness de distance* associée sera élevée.

3.3.2.3 Principe de sélection

La fonction de sélection qui précède l'opération de croisement va extraire les individus parents de la génération n pour donner les individus enfants qui constitueront la génération suivante n+1.

La méthode de sélection par « roulette » à été préférée à la méthode de sélection par rang ou par tournoi afin de conserver l'aspect aléatoire de l'algorithme. La population étant suffisamment grande, la méthode de sélection du reste stochastique n'a pas été retenue.

Afin d'accélérer la convergence de la fonction de *fitness* vers la solution recherchée du problème, le principe de l'élitisme à été implémenté. Ainsi, à l'issu de l'étape de croisement, le meilleur individu de la population antérieure est réintroduit dans cette nouvelle population.

Un individu fort peut prendre le dessus sur la population et l'uniformiser. L'algorithme se stoppe donc, non pas lorsque toutes les itérations prévues ont été effectuées, mais au moment où la population devient uniforme. Afin de pallier ce problème, le principe de *scaling* exponentiel apporte de bons résultats afin de conserver une population hétéroclite.

3.3.2.4 Opérateur de croisement

Les croisements entre les individus préalablement sélectionnés sont effectués par un type de croisement à un point illustré à la figure 3.5.

Pour éviter d'alourdir l'explication, un seul avion sera pris en compte dans l'exemple qui suit. Cette configuration n'est cependant pas réalisable puisqu'un avion ne peut être en conflit avec lui-même (il faut considérer au minimum deux avions pour se trouver en présence d'un conflit aérien).

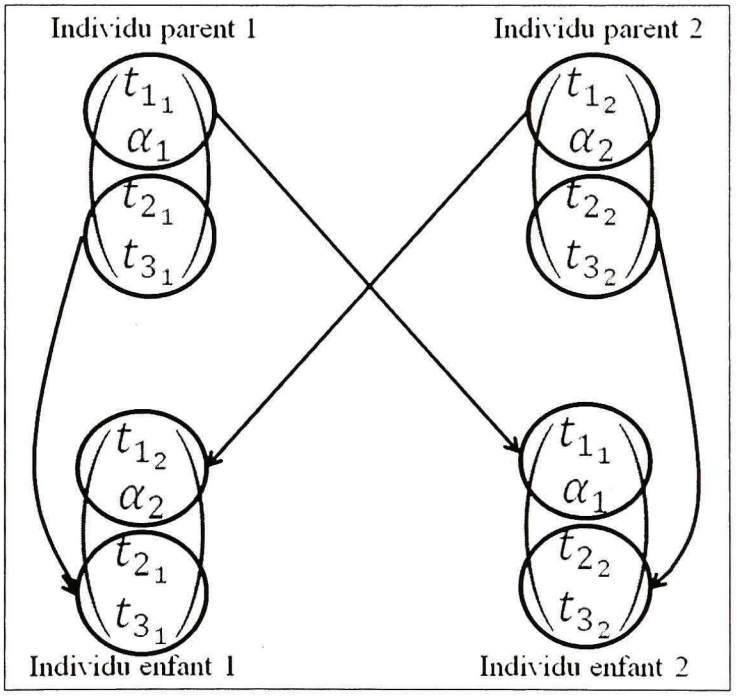

Figure 3.5 Schéma de croisement de deux chromosomes.

Notons t'_{j_i} les gênes des individus enfants.

Ainsi : $\begin{pmatrix} t'_{1_2} \\ \alpha'_1 \\ t'_{2_2} \\ t'_{3_2} \end{pmatrix} = \begin{pmatrix} t_{1_2} \\ \alpha_2 \\ t_{2_2} \\ t_{3_2} \end{pmatrix}$ et $\begin{pmatrix} t'_{1_2} \\ \alpha'_2 \\ t'_{2_2} \\ t'_{3_2} \end{pmatrix} = \begin{pmatrix} t_{1_2} \\ \alpha_1 \\ t_{2_2} \\ t_{3_2} \end{pmatrix}$.

À l'issu de la phase de croisement, l'algorithme vérifie si la condition $0 \leq t'_{1_i} \leq t'_{2_i} \leq t'_{3_i}$ est toujours vérifiée pour tous les avions *i*, et ce pour chaque chromosome.

Dans le cas contraire, l'algorithme force :

- si $t'_{2_i} < t'_{1_i}$ alors $t'_{2_i} = t'_{1_i}$.
- si $t'_{3_i} < t'_{2_i}$ alors $t'_{3_i} = t'_{2_i}$.

De cette façon la réduction de manœuvres est incitée.

3.3.2.5 Opérateur de mutation

Le taux de mutation définit le pourcentage de chance qu'un gène mute. Dans le cas étudié les gènes représentent les angles de premier virage α_i ainsi que les différents instants t_{j_i}.

3.3.2.6 Critère d'arrêt

Pour ajuster les différents paramètres de l'algorithme, le nombre d'itérations de l'algorithme constitue un premier critère d'arrêt.

Le *scaling* empêche un individu de dominer le reste de la population et devrait donc éviter d'obtenir une population uniforme, mais ce paramètre sera tout de même conservé en tant que second critère d'arrêt.

Il reste à définir à l'utilisateur ce qu'il entend par solution satisfaisante afin de gérer ce troisième critère d'arrêt. Une priorité devra ainsi être établie (faut-il préconiser l'optimisation du nombre de manœuvres par rapport à la distance totale parcourue par exemple ?).

3.3.3 Gestion des contraintes

La population initiale n'est pas générée tout à fait aléatoirement puisque certaines contraintes sont appliquées dès la phase d'initiation de la population. Les contraintes sur les gênes t_i ont déjà été explicitées à la sous-section 3.3.2.1, rajoutons que l'angle α devra être inférieur à 30°.

À l'issu de l'étape de croisement, des contraintes déjà exprimées à la sous section 3.3.2.4, sont appliquées sur les gênes t_i. Il n'est pas nécessaire de contraindre les angles α, puisque ceux-ci se verront inchangés. Le recuit simulé n'est pas utilisé afin de conserver un brassage génétique et d'éviter de trop contraindre la population qui perdrait son aspect aléatoire.

À l'issu de la mutation les mêmes contraintes appliquées lors du croisement ont lieu sur les gênes t_i. On contraint également l'angle α à être inférieur à 30°.

La gestion de la mutation donnant de bons résultats afin d'éviter de faire converger la solution vers un optimum local, le principe de sharing ne sera pas implémenté, évitant ainsi d'allonger le temps de calcul de l'algorithme.

CHAPITRE 4

EXPLOITATION DES RÉSULTATS

4.1 Influence des différents facteurs sur les résultats

Les différents choix effectués et justifiés dans les chapitres précédents seront ici expliqués en termes de résultats. Nous y observerons l'évolution de la *fitness* totale ainsi que des sous-fonctions de *fitness* en fonction de différents facteurs : les choix de développement, la fenêtre de prédiction, la nombre d'itérations de l'algorithme, la nombre de chromosomes introduits dans la population initiale et le facteur de mutation.

Tous les graphiques d'évolution des fonctions de *fitness* décrites dans le chapitre 4 présenteront en abscisse le nombre d'itérations de l'algorithme. Etant donné que la meilleure solution n'est pas connue, aucune unité ne sera volontairement affichée sur l'axe des ordonnées. Il est important ici d'observer l'allure des courbes.

Afin de faciliter la compréhension visuelle, les graphiques des évolutions de la fonction de *fitness* totale comprennent une ligne en pointillé qui nous informe lorsque l'algorithme à réussi à résoudre tous les conflits présents. Rappelons que cette optimisation est prioritaire sur les autres qui ne prendront effet qu'à partir du moment où tous les conflits auront été résolus.

Ainsi lorsque que la fonction de *fitness* se situe en dessous de ce seuil, ceci signifie que tous les conflits n'ont pas encore été résolus, et dès lors que la fonction franchit ce seuil nous comprendrons que tous les conflits ont été résolus.

4.1.1 Influence de l'élitisme sur les fonctions de fitness

Observons tout d'abord l'influence de l'*élitisme* sur les résultats de l'algorithme. Les évolutions des sous fonctions de *fitness* puis de la *fitness* totale seront comparées sans la présence de l'*élitisme* dans un premier temps puis avec par la suite.

Il est à noter que les algorithmes génétiques doivent conserver leur génération aléatoire des différentes populations. Ainsi les exemples pris avec et sans *élitisme* ne possèdent pas les mêmes populations mais conservent les mêmes conditions initiales (nombre d'avions, positions et caps), la même fenêtre de prédiction, le même facteur de mutation, le même nombre de chromosomes formant la population et le même nombre d'itérations effectuées par l'algorithme.

Nous allons considérer un conflit de trois avions convergeant vers le même point, résolu à l'aide de 50 chromosomes et 20 itérations.

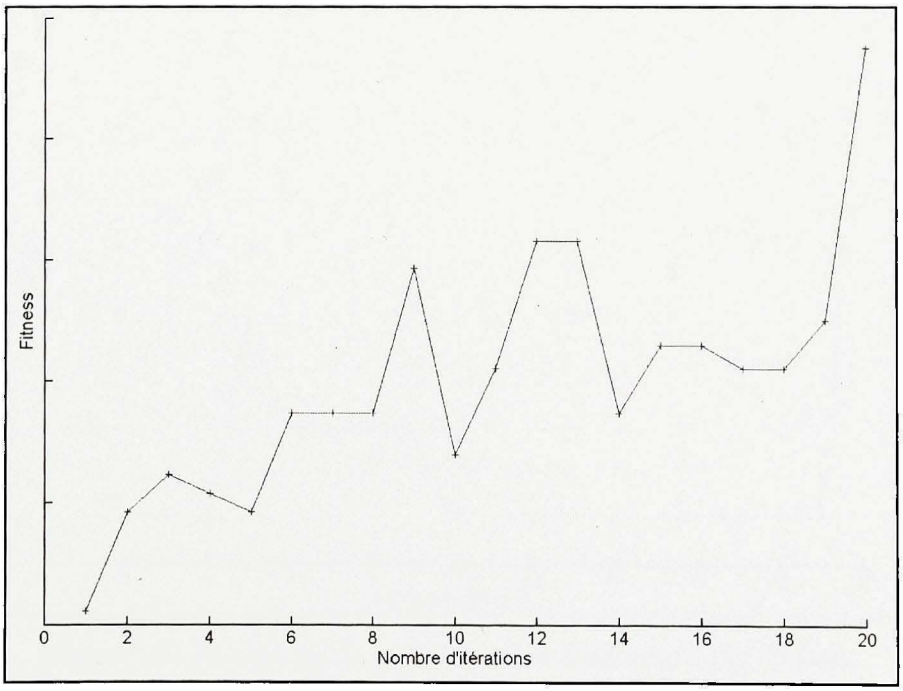

Figure 4.1 Évolution de la *fitness de retour* sans *élitisme* en fonction du nombre d'itérations.

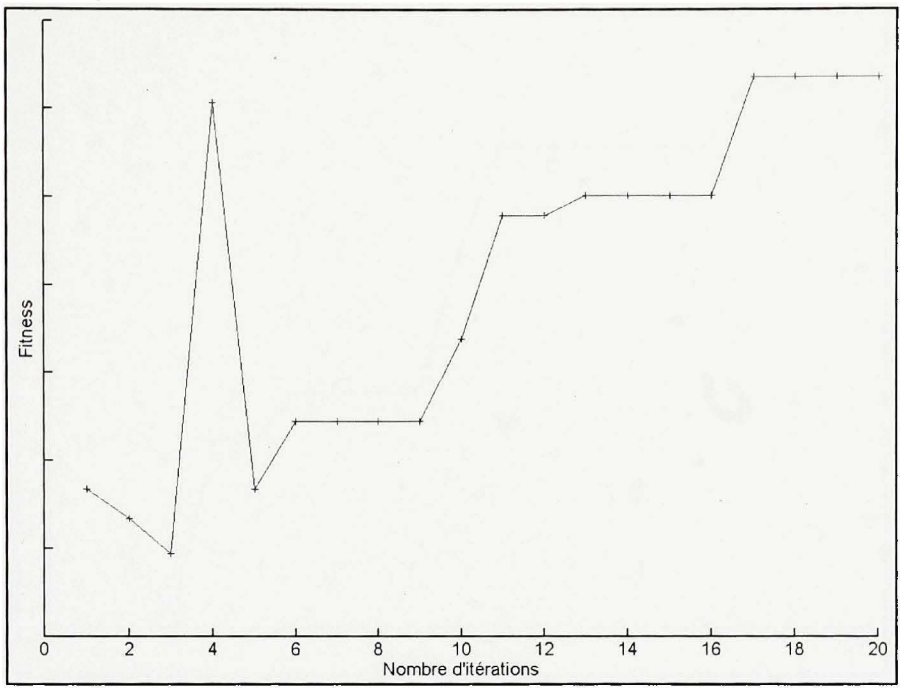

Figure 4.2 Évolution de la *fitness de retour* avec *élitisme* en fonction du nombre d'itérations.

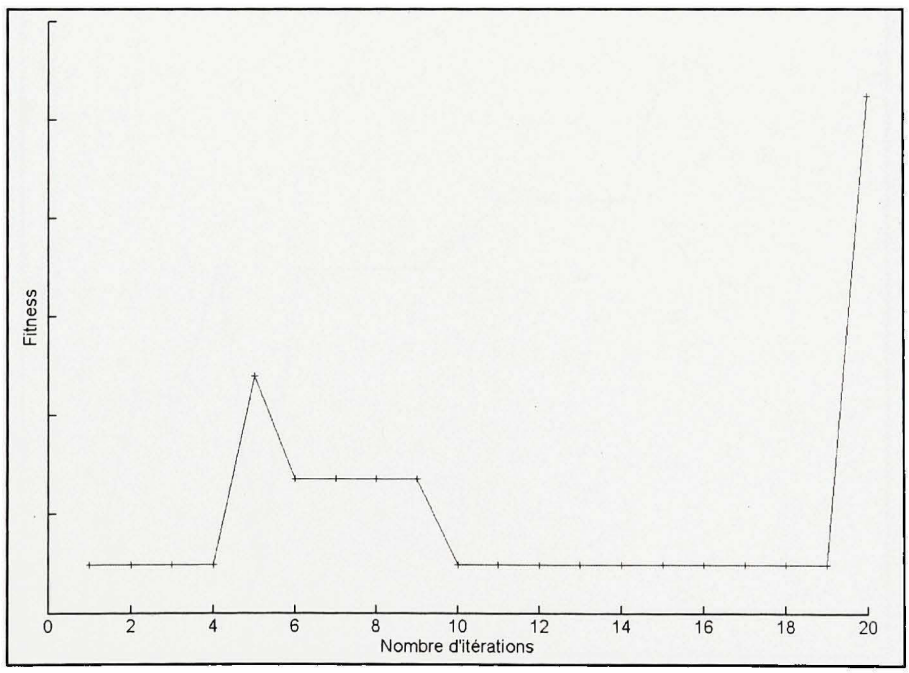

Figure 4.3 Évolution de la *fitness de manœuvre* sans *élitisme* en fonction du nombre d'itérations.

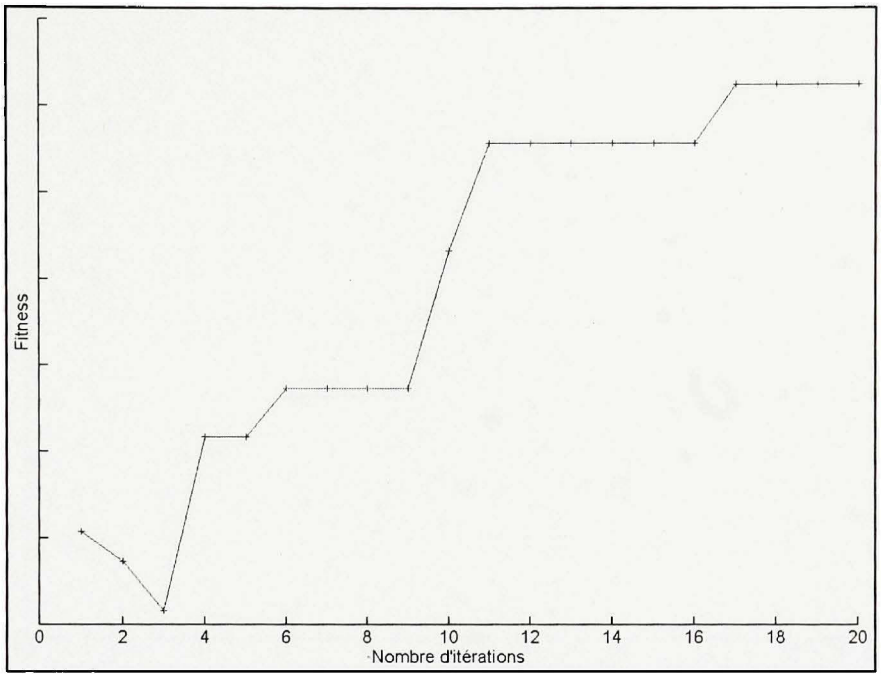

Figure 4.4 Évolution de la *fitness de manœuvre* avec *élitisme* en fonction du nombre d'itérations.

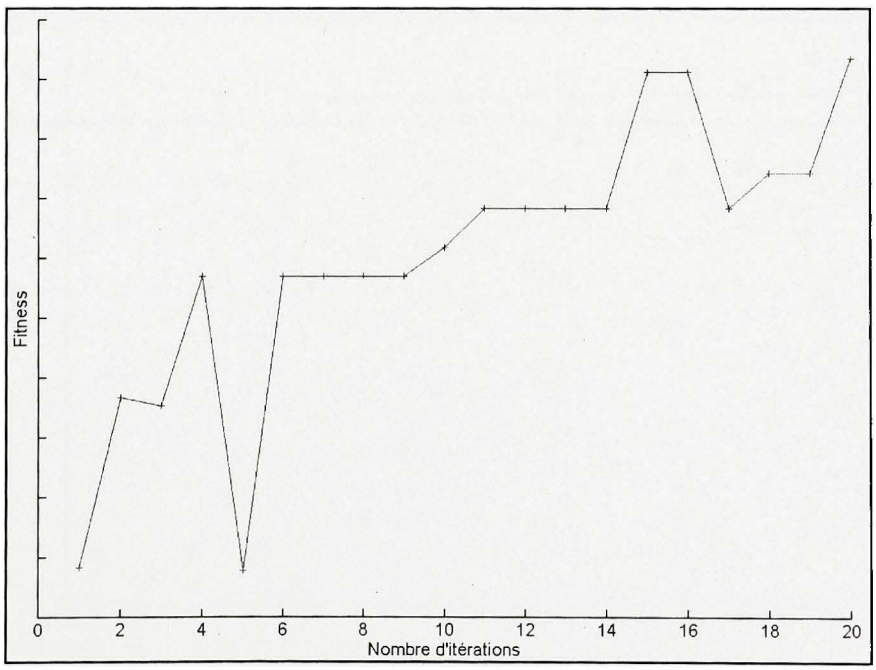

Figure 4.5 Évolution de la *fitness de distance* sans *élitisme* en fonction du nombre d'itérations.

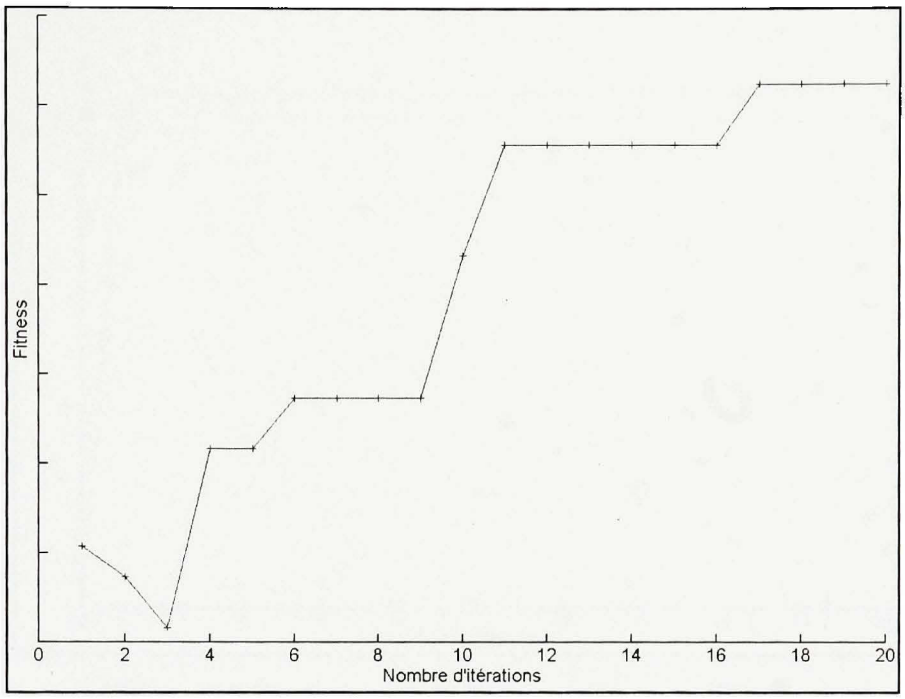

Figure 4.6 Évolution de la *fitness de distance* avec *élitisme* en fonction du nombre d'itérations.

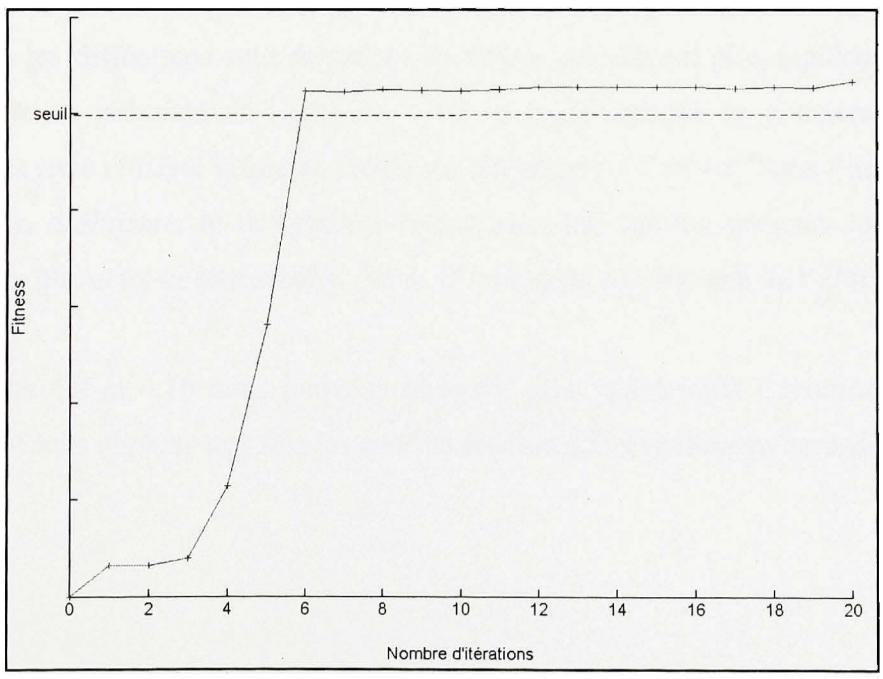

Figure 4.7 Évolution de la *fitness totale* sans *élitisme* en fonction du nombre d'itérations.

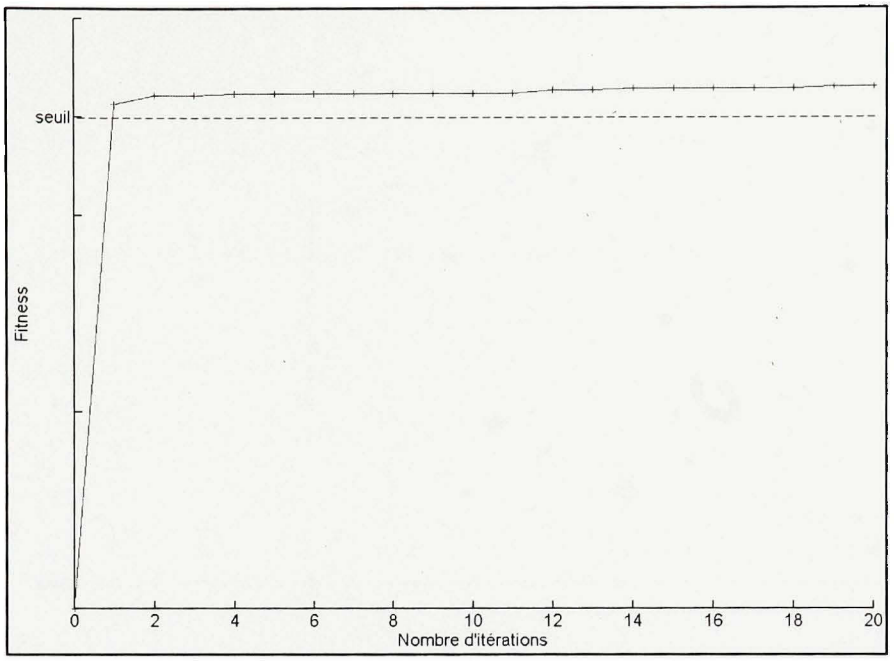

Figure 4.8 Évolution de la *fitness totale* avec *élitisme* en fonction du nombre d'itérations.

La comparaison deux à deux des figures 4.1 et 4.2, 4.3 et 4.4 ainsi que des figures 4.5 et 4.6 montrent que les différentes sous fonctions de *fitness* convergent plus rapidement vers leur solution finale en présence de l'*élitisme*. Ceci se traduit mieux en comparant les *fitness totales* avec et sans *élitisme* mises en forme sur les figures 4.7 et 4.8. Sans l'implémentation de la fonction d'*élitisme*, le programme résout tous les conflits présents lors de la 6ème itération, alors que ceux-ci sont résolus dès la 1ère itération en présence de l'*élitisme*.

Sur les figures 4.9 et 4.10 nous pouvons observer plus visiblement l'évolution des *fitness totales* avec et sans *élitisme* une fois les conflits résolus. Les conclusions sont similaires.

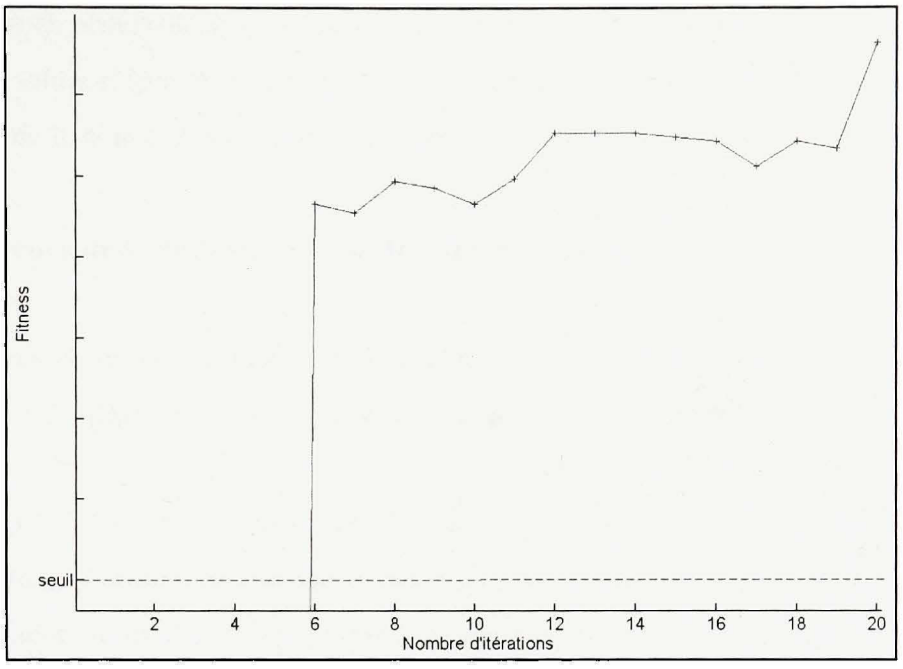

Figure 4.9 Évolution de la *fitness totale* sans *élitisme* après la résolution des conflits en fonction du nombre d'itérations.

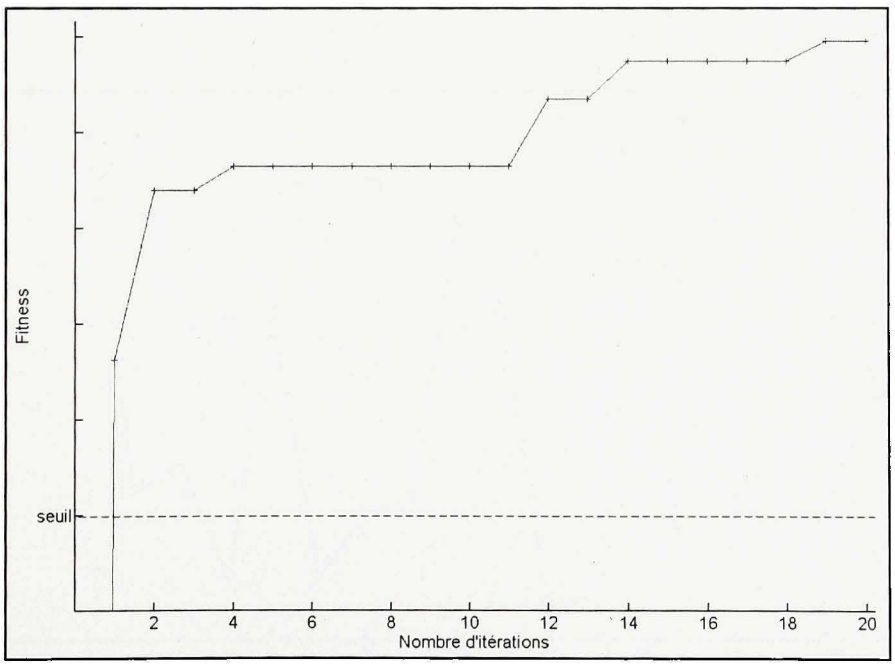

Figure 4.10 Évolution de la *fitness totale* avec *élitisme* après la résolution des conflits en fonction du nombre totale d'itérations.

Finalement nous observons que la fonction d'*élitisme* remplit son rôle en conservant toujours la meilleure solution (pas de baisse de la *fitness totale* au cours des itérations) et permet une accélération de la convergence de la *fitness totale* vers la solution finale.

4.1.2 Influence de la mutation sur les fonctions de fitness

Afin de mieux observer l'influence de la mutation sur les fonctions de *fitness*, l'*élitisme* ne sera pas pris en compte lors des exemples suivants.

Les figures 4.11, 4.12 et 4.13 montrent l'influence du facteur de mutation sur la population de chromosomes. Un facteur trop important risque de faire muter beaucoup de chromosomes dont les meilleurs individus : c'est ce que l'on observe sur la figure 4.14. L'algorithme réussi à résoudre tous les conflits dès la 4ème itération avant que le ou les meilleurs individus de la population mute(nt) et oblige(nt) l'algorithme à reformer un autre individu apportant une bonne réponse au problème.

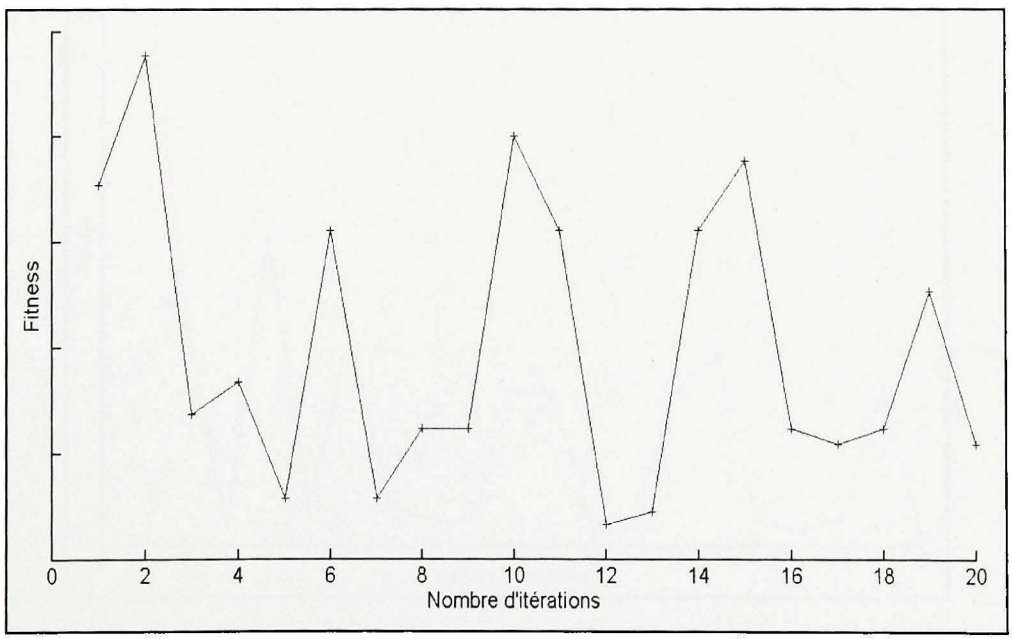

Figure 4.11 Évolution de la *fitness de retour* sans *élitisme* avec un facteur de mutation élevé (80%).

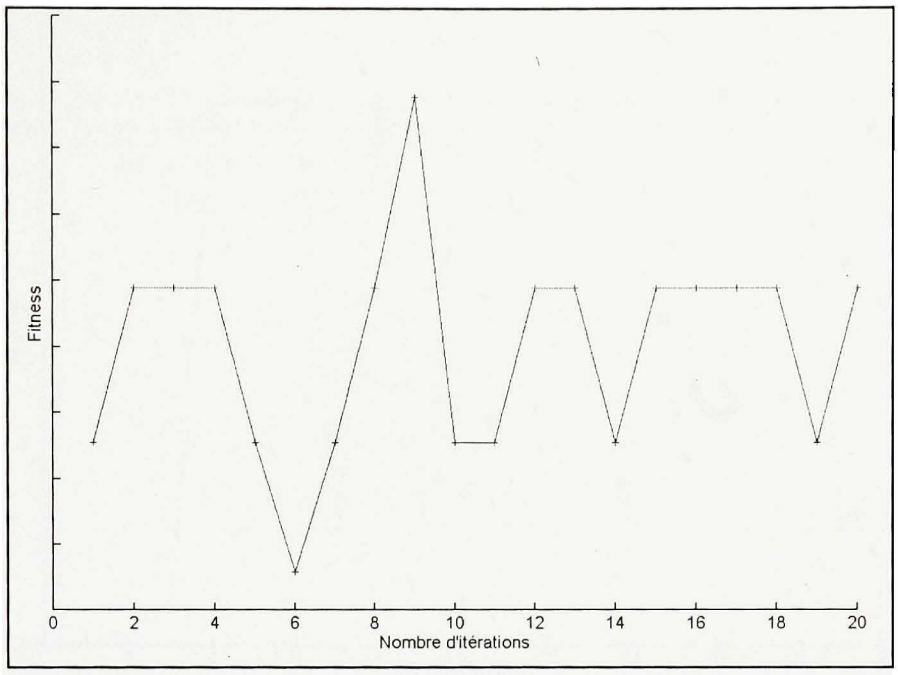

Figure 4.12 Évolution de la *fitness de manœuvre* sans *élitisme* avec un facteur de mutation élevé (80%).

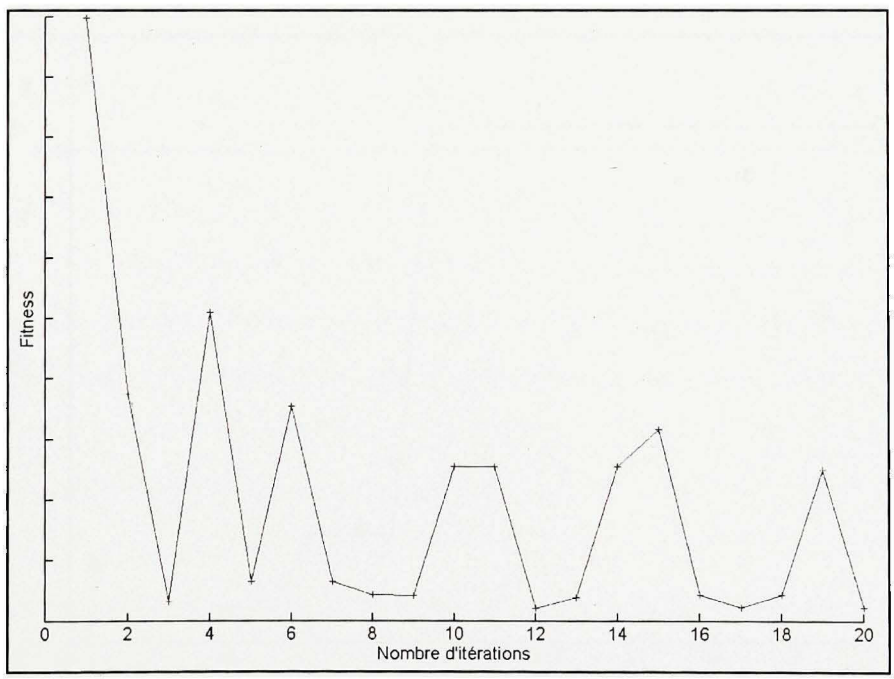

Figure 4.13 Évolution de la *fitness de distance* sans *élitisme* avec un facteur de mutation élevé (80%).

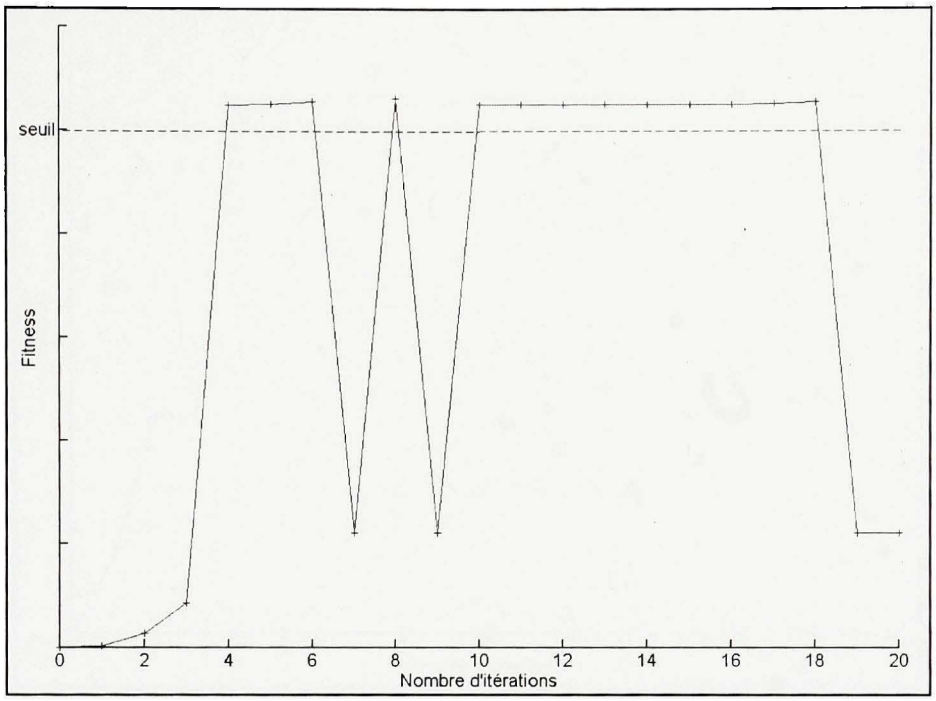

Figure 4.14 Évolution de la *fitness totale* sans *élitisme* avec un facteur de mutation élevé (80%).

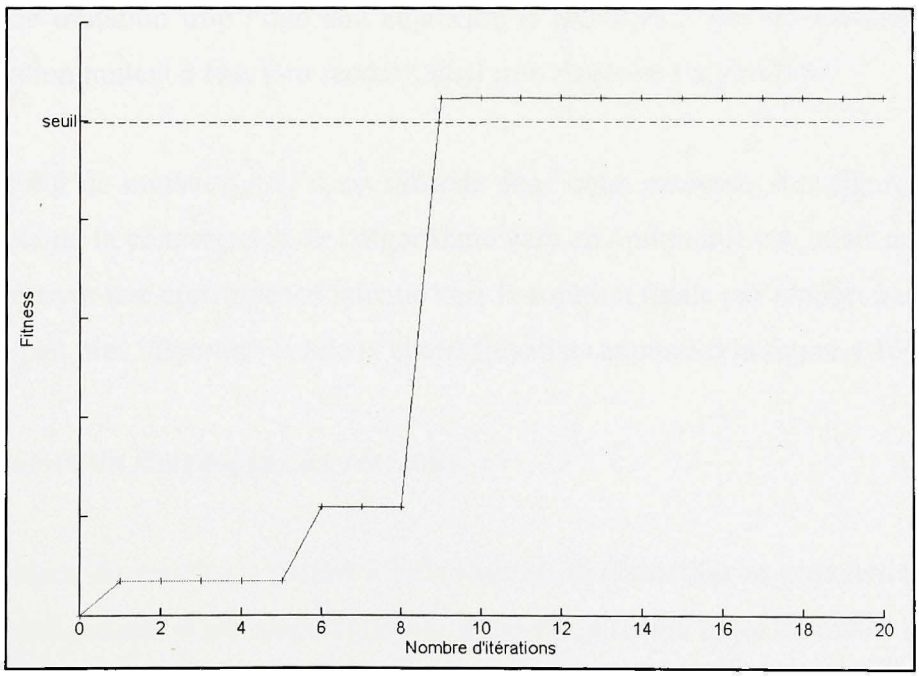

Figure 4.15 Évolution de la *fitness totale* sans *élitisme* sans mutation (0%).

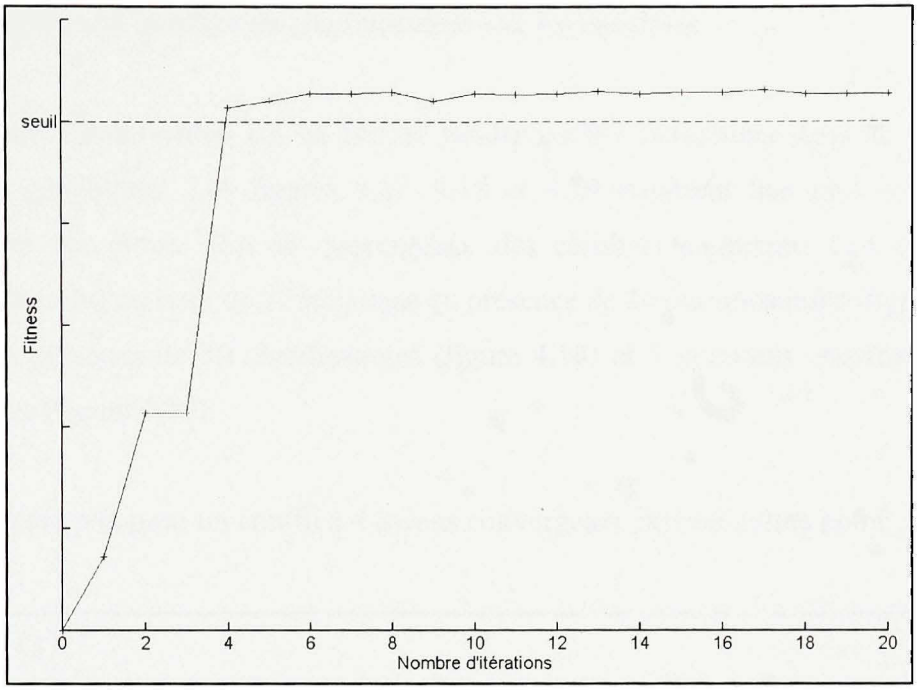

Figure 4.16 Évolution de la *fitness totale* sans *élitisme* avec un facteur de mutation de 10%.

Un facteur de mutation trop important augmente la probabilité que les meilleurs individus d'une génération mutent à leur tour rendant ainsi trop aléatoire l'algorithme.

Avec un facteur de mutation nul, il est difficile dans notre exemple, à la figure 4.15, de se rendre compte de la convergence de l'algorithme vers un optimum local, mais nous pouvons du moins observer une convergence ralentie vers la solution finale par rapport à un facteur de mutation un peu plus important et mieux choisi (résultats exposés à la figure 4.16).

4.1.3 Influence du sharing sur les résultats

La mise en place du *sharing* a permis à l'algorithme de diversifier sa population. Alors que celui-ci stagnait à cause d'une population uniforme avant même la fin de toutes les itérations demandées, l'implémentation du *sharing* a permis de palier à ce problème.

4.1.4 Influence du nombre de chromosomes sur les résultats

Le nombre de chromosomes est un facteur fondamental à déterminer dans la méthode des algorithmes génétiques. Les figures 4.17, 4.18 et 4.19 montrent que plus le nombre de chromosomes est élevé, plus la convergence des résultats augmente. Les conflits sont entièrement résolus au bout de 17 itérations en présence de 20 chromosomes (figure 4.17), 10 itérations en présence de 50 chromosomes (figure 4.18) et 5 itérations en présence de 100 chromosomes (figure 4.19).

L'exemple a été pris pour un conflit à 4 avions convergeant vers un même point.

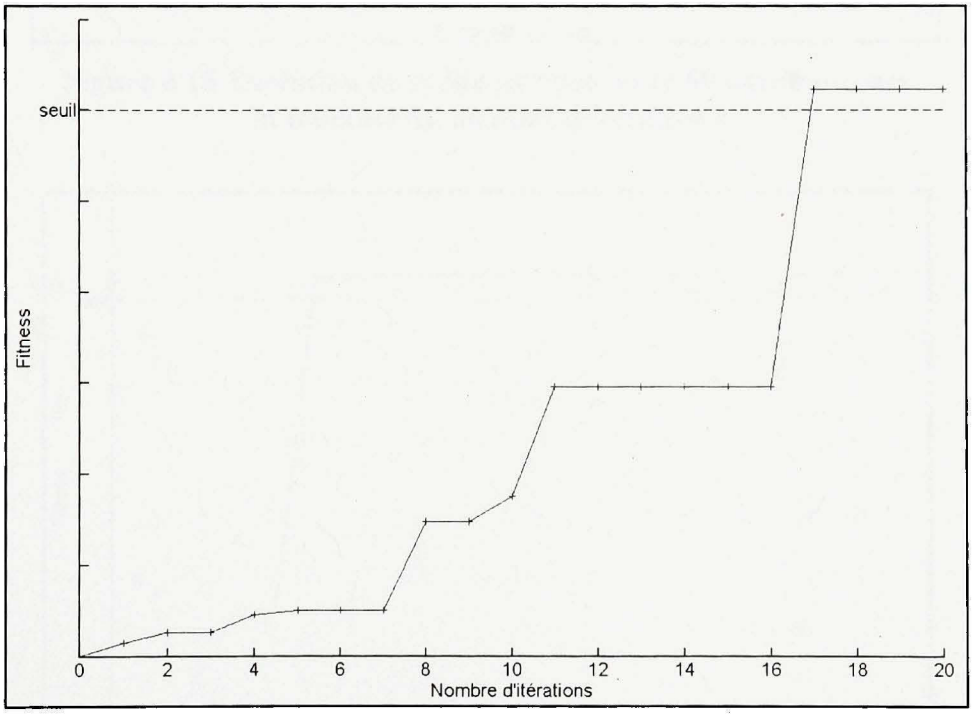

Figure 4.17 Évolution de la *fitness totale* avec 20 chromosomes en fonction du nombre d'itérations.

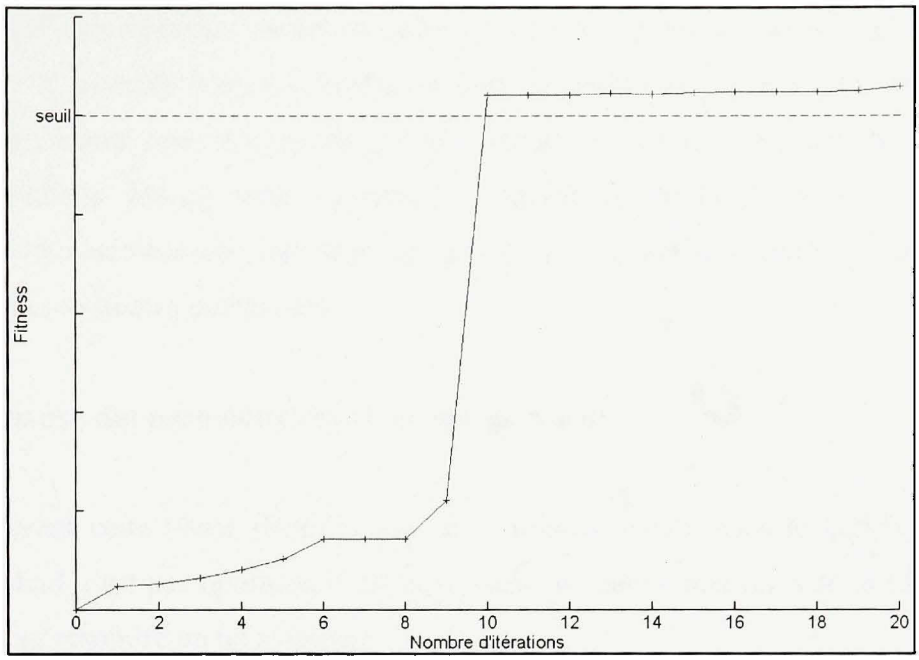

Figure 4.18 Évolution de la *fitness totale* avec 50 chromosomes en fonction du nombre d'itérations.

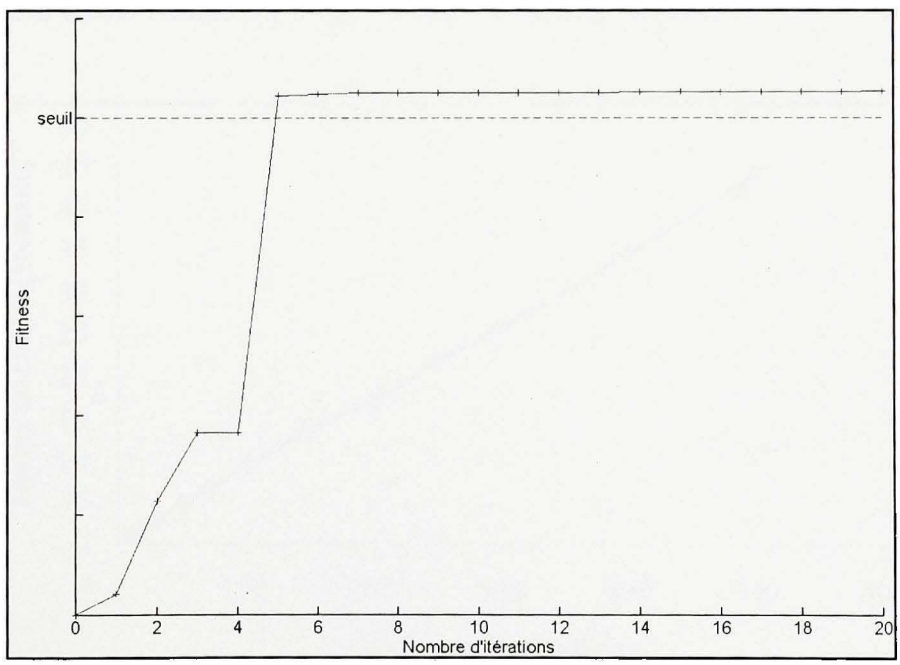

Figure 4.19 Évolution de la *fitness totale* avec 100 chromosomes en fonction du nombre d'itérations.

Le nombre de chromosomes choisi va varier suivant le nombre d'avions impliqués dans le conflit (plus le nombre d'avions impliqués dans le conflit sera élevé, plus le nombre de chromosomes requis pour trouver une solution viable sera grand), la précision des résultats que l'on souhaite obtenir mais également l'importance de la prédiction de trajectoire effectuée. Tout ceci nécessite cependant un coût en temps, facteur ô combien important dans le domaine des systèmes embarqués.

4.1.5 Influence des paramètres sur le temps de calcul

Même si durant cette phase d'étude, avec une programmation sous le logiciel Matlab, le temps de calcul n'est pas optimisé, il est intéressant de mentionner un ordre d'idée du temps qu'il faut pour résoudre un tel système.

Les simulations ont été effectuées à l'aide du logiciel Matlab R2008a sur un ordinateur doté d'un processeur AMD Athlon X2 Dual-Core, 2.3 GHz et 2048 Mo.

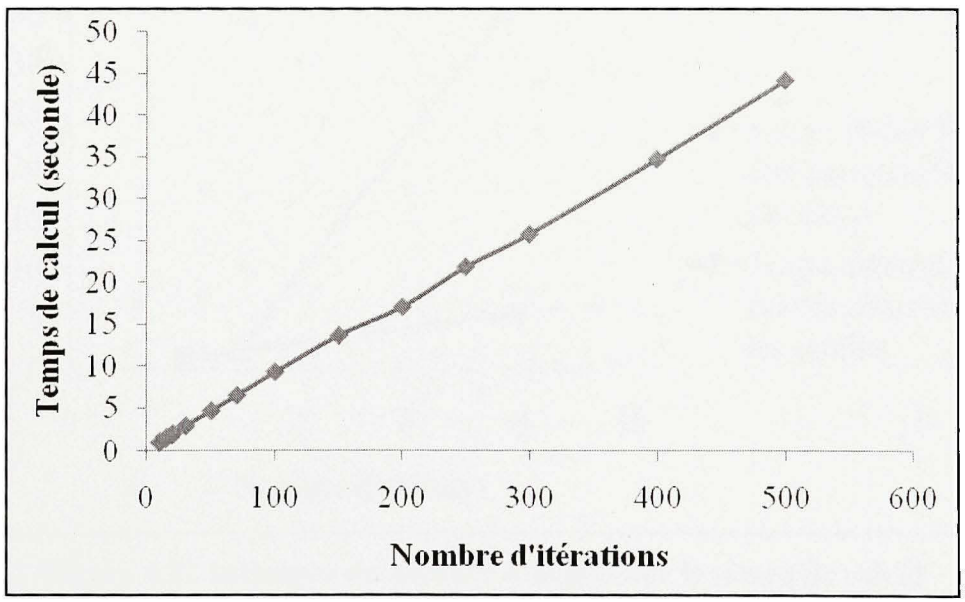

Figure 4.20 Influence du nombre d'itérations sur le temps de calcul à l'aide d'un exemple à 3 avions et 50 chromosomes.

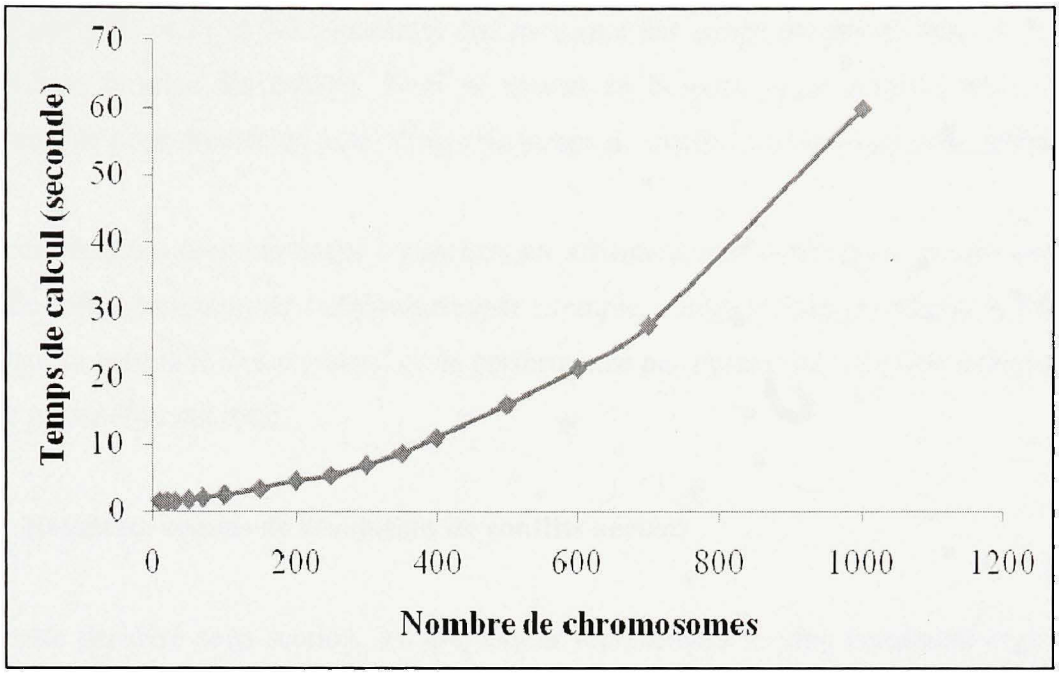

Figure 4.21 Influence du nombre de chromosomes sur le temps de calcul à l'aide d'un exemple à 3 avions et 20 itérations.

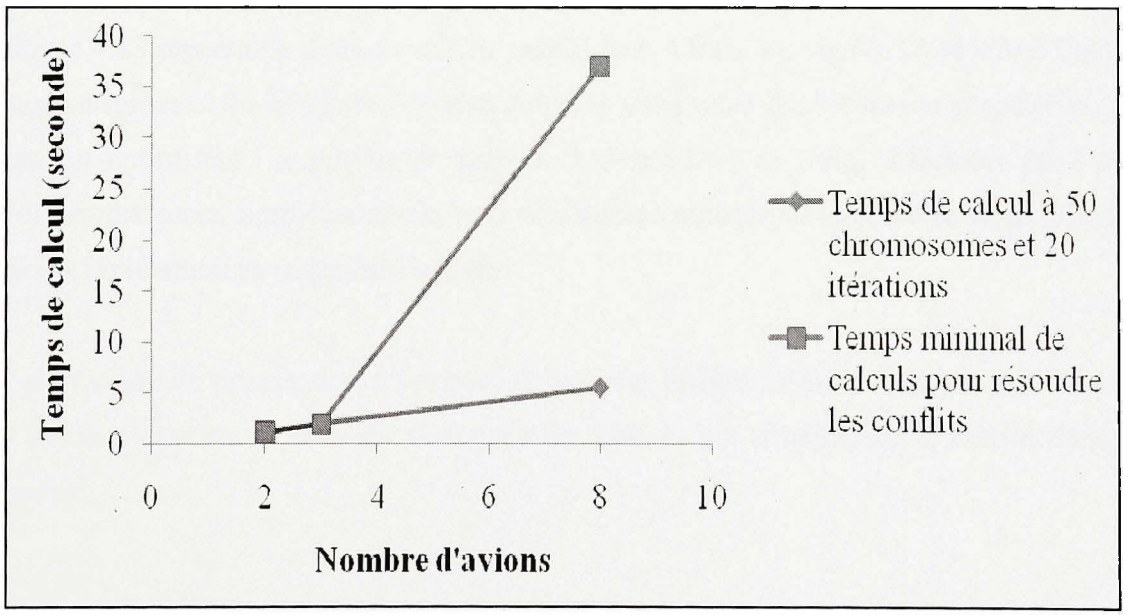

Figure 4.22 Influence du nombre d'avions sur le temps de calcul à l'aide d'un exemple à 50 chromosomes et 20 itérations.

Les figures 4.20, 4.21 et 4.22 montrent une moyenne des temps de calculs relevés pour que l'algorithme termine les calculs. Tous se situent en dessous de la minute, mais ne sont présentés que pour donner un ordre d'idée du temps de calcul nécessaire par cette méthode.

Tous ces résultats nous amènent à conclure en affirmant que l'utilisateur pourra choisir le mode de fonctionnement de l'algorithme (par exemple, vaut-il mieux privilégier le temps de calcul sur la précision des résultats, ou la performance par rapport au temps de calcul ?) pour ensuite paramétrer cet outil.

4.2 Résultats visuels de résolution de conflits aériens

Dans cette dernière sous section, les graphiques représentent le plan horizontal (x,y) où les trajectoires initiales (en pointillées) et d'évitements des avions (en traits pleins) sont représentées suivant différentes situations de vol ou d'approches initiales.

La figure 4.23 représente deux avions en conflit face à face. Le conflit est résolu à l'aide de 50 chromosomes et 20 itérations, et nous pouvons remarquer que les *fitness secondaires* sont également optimisées : le conflit est géré en 3 manœuvres au total (effectuées par l'avion dévié), les distances supplémentaires sont minimisées puisque un seul avion se trouve dévié et rejoint rapidement sa trajectoire initiale.

Aucun système de priorisation n'est pour le moment intégré ; l'algorithme choisit lui-même quel avion s'écartera le plus de sa trajectoire dans le but d'optimiser le conflit dans son ensemble.

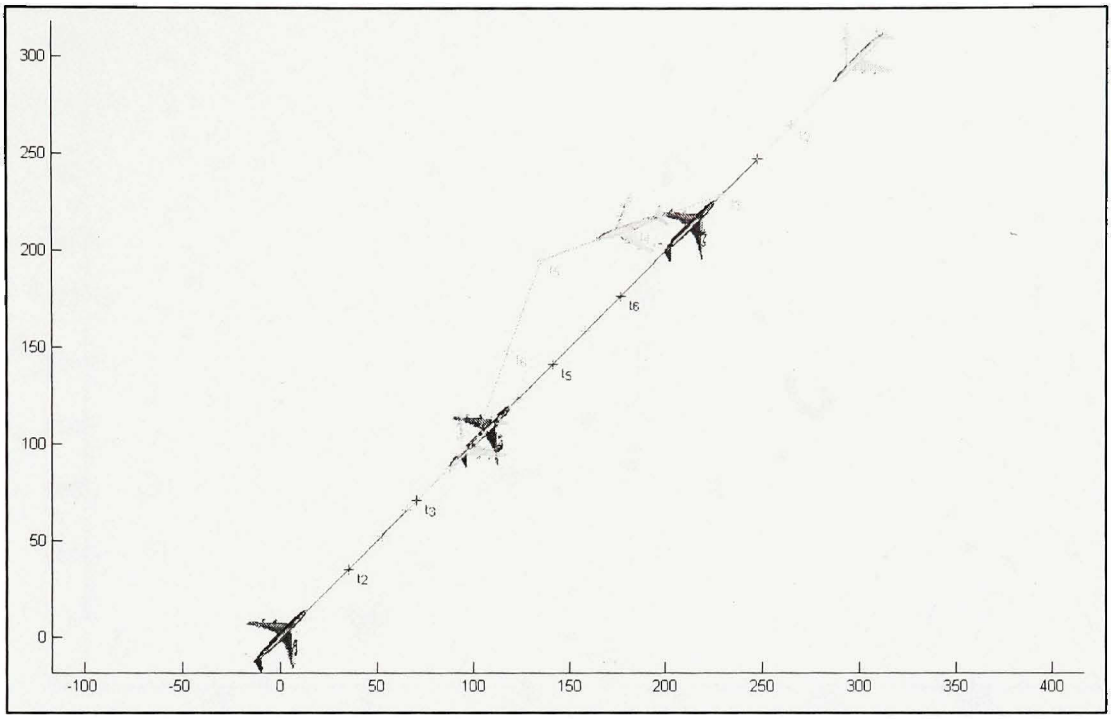

Figure 4.23 Gestion d'un conflit à deux avions en approche "face à face".

La figure 4.24 présente une autre configuration de conflit à deux avions. Ce dernier est résolu à l'aide de 50 chromosomes et 20 itérations. Dans la même optique que l'exemple précédent, un seul avion est dévié à l'aide d'une manœuvre « *point tournant* ».

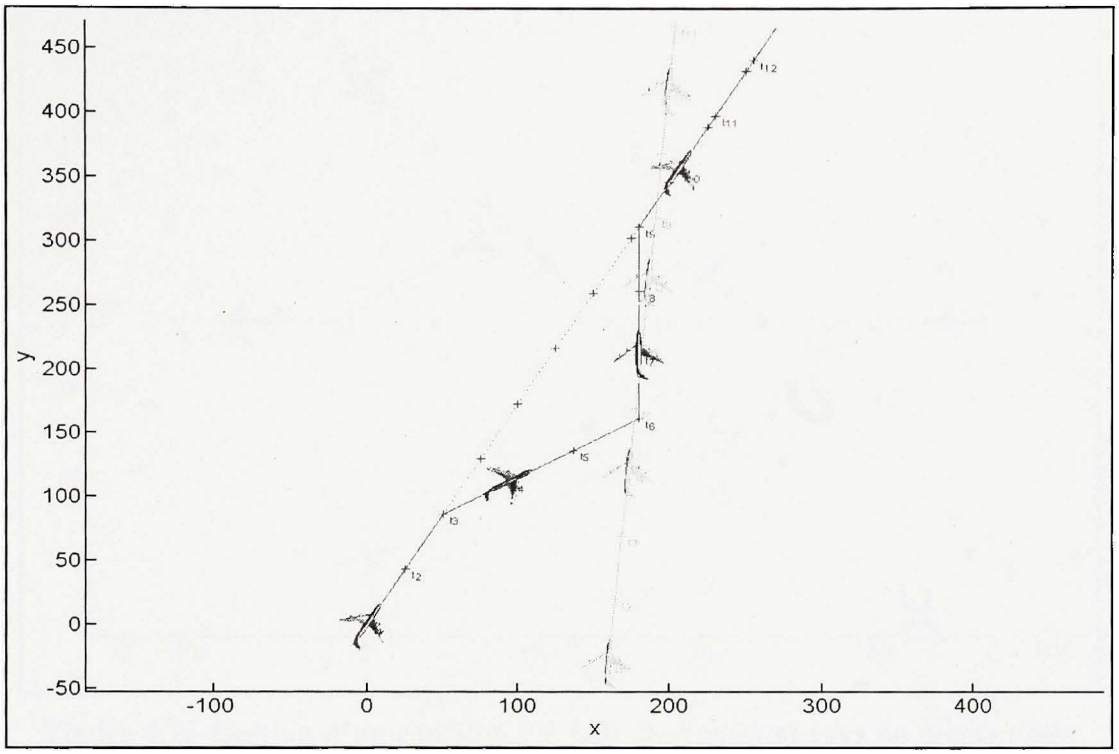

Figure 4.24 Gestion d'un conflit à deux avions en approche "côte à côte".

Observons maintenant directement sur les résultats l'influence du nombre de chromosomes et du nombre d'itérations sur la résolution d'un conflit à trois avions convergeant vers un même point.

Il est intéressant d'observer ce que nous avons exprimés précédemment avec les graphiques des *fitness* se refléter sur les trajectoires des avions.

Notons que dans les deux cas les conflits ont été gérés à l'issu de la simulation, mais l'optimisation des autres facteurs tels que le nombre total de manœuvres effectuées, la distance parcourue ou bien le retour à la trajectoire initiale est nettement visible en comparant les figures 4.25 et 4.26. L'optimisation s'affine d'une figure à l'autre : le nombre total de manœuvres reste à peu près constant, mais la distance déviée parcourue ainsi que le retour à la trajectoire initiale s'optimisent avec l'augmentation du nombre de chromosomes.

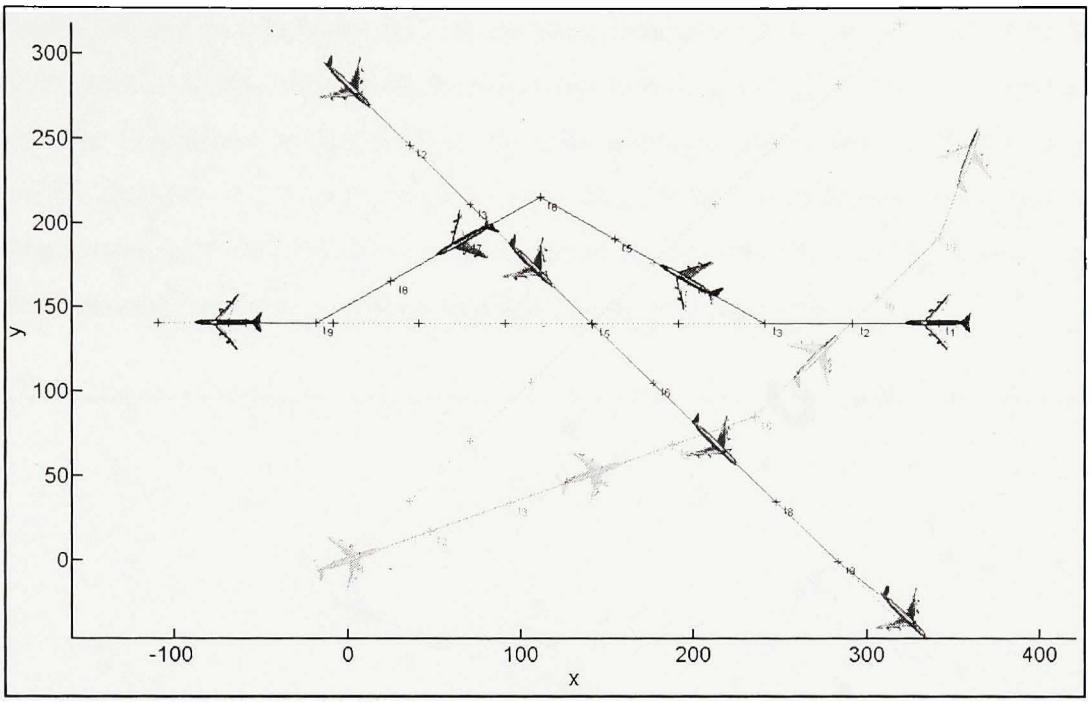

Figure 4.25 Gestion d'un conflit à 3 avions convergeant vers un même point à l'aide de 50 chromosomes et 50 itérations.

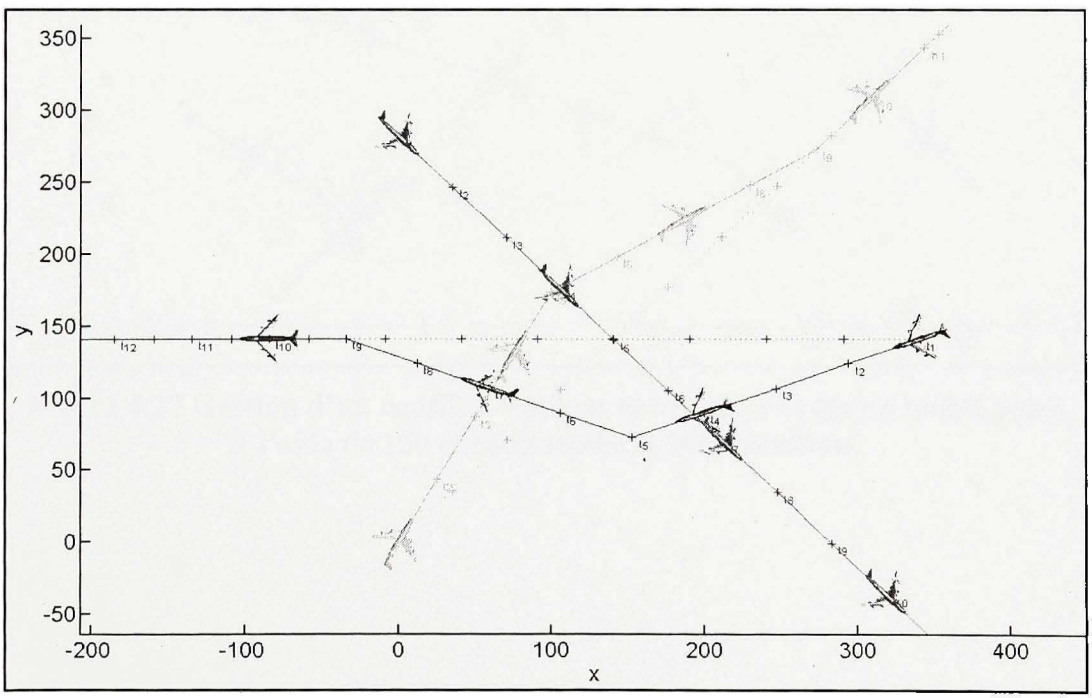

Figure 4.26 Gestion d'un conflit à 3 avions convergeant vers un même point à l'aide de 100 chromosomes et 50 itérations.

Enfin nous présentons à la figure 4.27 un cas très contraignant de six avions convergeant vers un même point. Il a fallu 150 chromosomes et 100 itérations pour le résoudre. L'exemple est présent pour démontrer la performance de cette méthode qui permet de traiter un grand nombre de données. Le résultat proposé n'est sans doute pas optimisé à son maximum, cependant rappelons que l'objectif reste de gérer les conflits aériens en croisière, ce qui limitera fortement le nombre d'avions impliqués dans un même conflit.

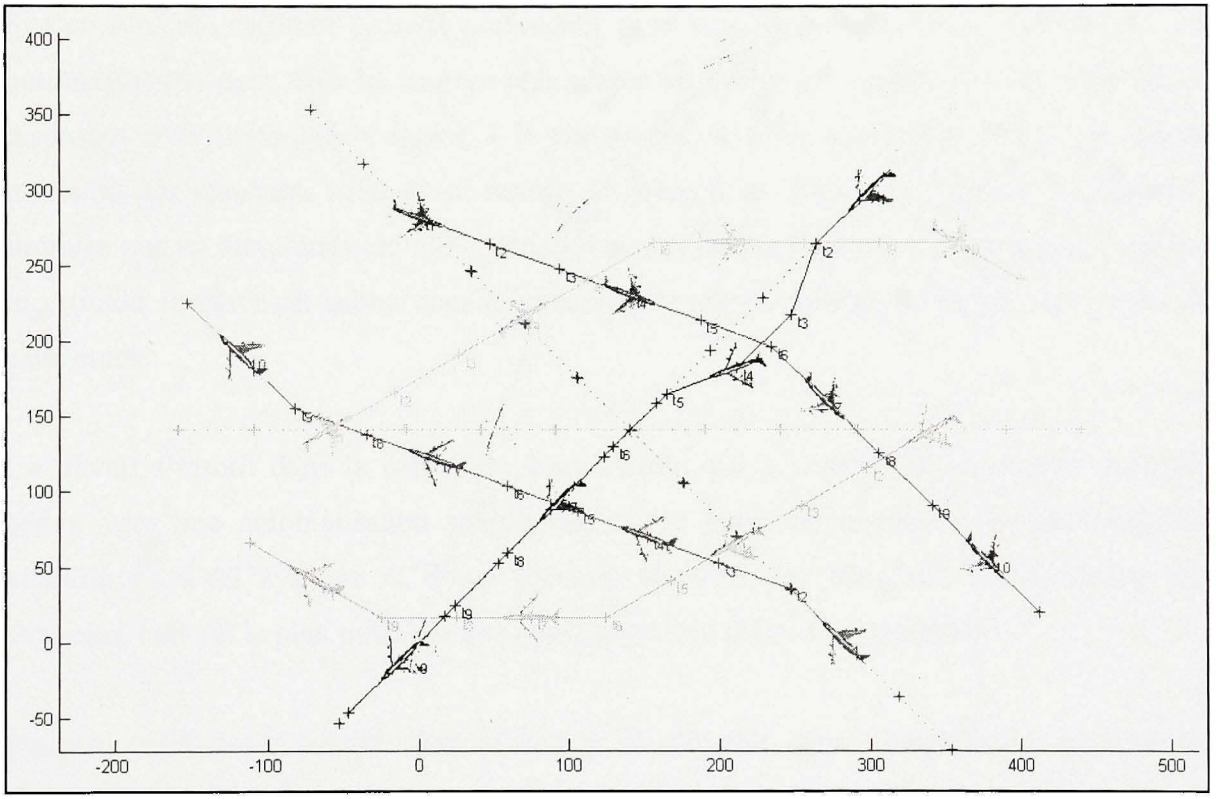

Figure 4.27 Gestion d'un conflit à 6 avions convergeant vers un même point à l'aide de 150 chromosomes et 100 itérations.

CONCLUSION

La solution proposée à l'aide d'une méthode de résolution par algorithmes génétiques présente des avantages qui ont motivés leur utilisation, mais ne constitue pas l'unique solution au problème. Ce rapport présente une étude de leur utilisation appliquée à la résolution de conflits aériens.

Si l'on souhaite explorer plus en profondeur cette voie, des améliorations subsistent : une optimisation du code dans un langage plus adapté ou encore une application plus concrète en disposant d'objectifs précis quand à la nature des résultats escomptés (temps de calcul, précision des résultats, taille de la fenêtre de prédiction). Enfin, en l'absence de données fournies par un simulateur de trafic aérien, les trajectoires prédites n'ont souffert d'aucune incertitude. Intégrer cet aspect dans un travail futur relève vraisemblablement de l'avenir de cette étude.

Ce travail s'inscrit dans la continuité d'un chemin qui devrait mener la gestion du trafic aérien vers une automatisation progressive. Cette étape ne représente qu'une étude de viabilité d'un tel système et d'une méthode donnant des résultats ; elle nécessite des avancées, tant sur le plan politique que technique avant d'être mise en place.

J'espère enfin avoir suscité chez le lecteur de l'intérêt dans l'emploie des algorithmes génétiques ainsi que l'intérêt de poursuivre ses recherches dans le domaine de la gestion de conflits aériens en route.

LISTE DES RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Alliot Jean-Marc et Schiex Thomas. 1993. *Intelligence Artificielle et Informatique Théorique*. ISBN : 2-85428-324-4. Cepadues.

[2] Darwin Charles. 1859. *The Origin of species*.

[3] Delahaye Daniel. 1995. « Optimisation de la sectorisation de l'espace aérien par algorithmes génétiques ». Thèse de doctorat en informatique, ENSAE, 220 p.

[4] Dodin Pierre. 1999. *Résolution de conflits via la programmation semi-définie*. Rapport de DEA, Paris VI.

[5] Durand Nicolas. 1996. « Optimisation de trajectoires pour la résolutions de conflits ». Thèse de doctorat en informatique, Institut national polytechnique de Toulouse, 200 p.

[6] Durand Nicolas. 2004. « Algorithme génétique et autres outils d'optimisation appliqués à la gestion de trafic aérien ». Thèse de doctorat, 218 p.

[7] Frazzoli, E., Z.H. Mao et E. Feron. 2001. « Aircraft conflict resolution via semi-definite programming ». *AIAA Journal of Guidance, Control and Dynamics*, vol. 24 i.1, p. 79-86.

[8] Gianazza, David. 2004. « Optimisation des flux de trafic aérien ». Thèse de doctorat en informatique et télécommunication, Institut national polytechnique de Toulouse, 199 p.

[9] Goldberg David. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. ISBN : 0-201-15767-5. Addison Wesley.

[10] Granger Géraud. 1998. « Résolution de conflits embarqués dans les espaces de faibles densité ». Mémoire, INPT.

[11] Granger Géraud. 2002. « Détection et résolution de conflits aériens : modélisations et analyse ». Thèse de doctorat en informatique, Polytechnique, 152 p.

[12] Holland John. 1962. « Outline for a logical theory of adaptative systems ». *Journal for the Association of Computing Machinery*, vol. 3.

[13] Mahfoud S.W. and Goldberg D.E. *Parallel recombinative simulated annealing : A genetic algorithm*. Rapport Illigal, 92002. Urbana : University of Illinois.

[14] McCulloch, W.S. et W.A. Pitts. 1943. « A logical calculus immanent in nervous activity ». *Bulletin of mathematics and biophysics*, vol. 115, n°5, (été).

[15] Medioni Frédéric. 1998. « Méthode d'optimisation pour l'évitement aérien – Systèmes centralisés, système embarqués ». Thèse de doctorat en informatique, École Polytechnique, 230 p.

[16] Michalewicz Z. 1992. « Genetic algorithms + Data Structures = Evolution Programs ». *Springer-verlag*.

[17] Yin X. et Germay N. 1993. « A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization ». In *Proceedings of the Artificial Neural Nets and Genetic Algorithms*.

[18] Zeghal Karim. 1993. *Champs de forces symétriques : La logique d'un système anticollision coordonnée.* Rapport technique de l'ONERA.

[19] Zeghal Karim. 1994. « Vers une théorie de la coordination d'actions, application à la navigation aérienne ». Thèse de doctorat, Université Paris VI.