Modélisation mathématique et résolution automatique de conflits par algorithmes génétiques et par optimisation locale continue

Clément Peyronne

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Paul Sabatier de Toulouse

Discipline ou spécialité :
Mathématiques appliquées et Informatique

Clément Peyronne

Soutenance prévue le : mercredi 12 décembre 2012 à 9h00,
à l'École Nationale de l'Aviation Civile, Toulouse

Titre :

Modélisation mathématique et résolution automatique de conflits par algorithmes génétiques et par optimisation locale continue

---

Jury :

Éric Feron : Professeur à Georgia Institute of Technology (Rapporteur)
Michel Bierlaire : Professeur à l'École Polytechnique Fédérale de Lausanne (Rapporteur)
Daniel Delahaye : Enseignant-chercheur à l'École Nationale de l'Aviation Civile (Co-directeur)
Marcel Mongeau : Enseignant-chercheur à l'École Nationale de l'Aviation Civile (Co-directeur)
Jean-Baptiste Hiriart-Urruty : Professeur à l'Université Paul Sabatier (Examinateur)
Andrew R. Conn : *Senior Researcher* au T.J. Watson Research Center d'IBM, NY (Examinateur)

École doctorale :
École Doctorale Aéronautique Astronautique (EDAA)

Unité de recherche :
MAIAA : Laboratoire de Mathématiques Appliquées Informatique et Automatique pour l'Aérien (École Nationale de l'Aviation Civile)

Table des matières

Remerciements

Tout au long de cette thèse, j'ai reçu l'aide et le soutien de nombreuses personnes. Je tiens d'abord à remercier Capgemini pour m'avoir donné l'oportunité de réaliser cette thèse avec une grande liberté et pour m'avoir permis de réaliser ce séjour chez IBM à New-York si enrichissant d'un point de vue professionnel comme personnel.

Je tiens à remercier tous les collègues des laboratoires MAIAA et PII qui ont fait du laboratoire un endroit joyeux et chaleureux et qui m'ont également bien aidé dès que j'en ai eu besoin. Un petit clin d'oeil à Nour Dougui pour ses conseils avisés.

Je remercie Laurent Lapasset, Stéphane Puechmorel et Daniel Delahaye pour leur encadrement, leur aide précieuse, et leurs blagues légères du vendredi midi.

Un grand merci à Andrew Conn, qui m'a accueilli et encadré durant mon séjour à IBM, et qui m'a par la suite aidé, encouragé, et relu maintes fois. Étant aussi têtu l'un que l'autre nous avons parfois mis un moment à tomber d'accord, mais ces discussions ont toujours été constructives et stimulantes.

Je remercie également Marcel Mongeau pour les nombreuses heures qu'il a consacré à relire nos articles et mon manuscrit, mais aussi pour son optimisme sans faille, pour toutes les fois où il a réussi à me remonter le moral, et aussi pour son accent chantant inégalable.

Je souhaite aussi remercier ma famille pour leur soutien et leur présence dans les moments difficiles, merci à ma mère de me garder les pieds sur terre, à ma soeur pour sa confiance en moi, et à mon père d'être encore là pour voir ce manuscrit.

Je pense aussi à tous mes amis qui ont dû supporter mes longs discours, relire des textes un peu obscurs, qui ont réussi à me vider la tête quand j'en avais besoin.

Enfin, merci à mon amoureuse, qui a supporté mes humeurs, qui a su me faire relativiser et me remobiliser quand il le fallait et qui a partagé mes joies et mes peines pendant ces trois années intenses.

Introduction

Cette thèse porte sur le domaine de la gestion du trafic aérien ou l'*Air Traffic Management* (ATM). L'ATM est un système qui a pour but d'assurer un déroulement fluide de l'ensemble du trafic aérien tout en garantissant le respect de normes de sécurité. Ce système est actuellement en pleine mutation. En effet, l'augmentation constante du trafic est telle que le système actuel ne pourra plus répondre, dans un futur proche, aux normes de sécurité exigées. Partant de ce constat, de grands changements sont nécessaires. Même si une stagnation du trafic est observée depuis la présente crise économique, une réflexion globale est actuellement menée des deux côtés de l'Atlantique (Europe et États-Unis) à travers deux projets de grande envergure que nous détaillerons plus loin : respectivement SESAR [1] et NextGen [2]. Ces deux projets posent les bases de l'ATM du futur tant d'un point de vue structurel que d'un point de vue technologique. Une des améliorations voulues par ces deux projets d'un point de vue structurel est une utilisation plus importante de l'automatisation dans le système. Ceci est rendu possible par le contexte technologique envisagé par SESAR et NextGen qui permet d'entrevoir de nouvelles solutions. En effet, SESAR et NextGen, en proposant l'introduction de nouvelles technologies performantes, permettent d'envisager un contexte opérationnel différent. Précision importante à apporter : bien que prônant l'automatisation, dans ces deux projets, on souhaite que les décisions finales en termes de sécurité soient prises par l'opérateur humain.

La réflexion menée dans cette thèse se place dans le cadre de ces grandes évolutions de l'ATM. Notre premier chapitre commence par une présentation du fonctionnement général du système actuel. Nous spécifions ensuite le problème précis auquel nous nous sommes intéressés : la résolution automatique de conflits. Avant de présenter le cadre opérationnel envisagé par SESAR et NextGen de manière à cibler le cadre d'application de notre travail. Nous exposons ensuite les différentes solutions existantes à notre problème en détaillant à la fois les grands projets opérationnels passés mais aussi les différentes méthodes existantes dans le domaine de la recherche. Le but de ce chapitre est de bien situer dans quel cadre se placent les méthodes que nous avons développées. Le chapitre 2 présente les deux principales contributions de cette thèse. Nous présentons d'abord les modèles de trajectoires que nous avons proposés dont le modèle de courbe parcimonieux basé sur les B-splines. Nous détaillons ensuite notre formulation mathématique du problème qui prend la forme d'un problème d'optimisation semi-infinie. Les méthodes d'optimisation que nous avons utilisées pour résoudre ce problème seront exposées dans le chapitre 3 : il s'agira d'algorithmes génétiques, de méthodes d'optimisation sans dérivées et de méthodes de points intérieurs. Enfin, nous détaillons dans le chapitre 4 les résultats numériques que nous avons obtenus et qui démontrent que les méthodes d'optimisation numérique locale continue constituent une alternative intéressante pour ce problème habituellement abordé avec des heuristiques et dans un cadre de modélisation discret.

1. *Single European Sky ATM Research*
2. *Next Generation Air Transportation System*

Chapitre 1

Contexte et état de l'art

Dans ce chapitre, nous présentons d'abord le système de gestion du traffic aérien avant de détailler les différentes méthodes existantes répondant à notre problème.

1.1 Contexte : la gestion du trafic aérien

La gestion du trafic aérien ou *Air Traffic Management* (dans la suite de la thèse, nous utiliserons l'acronyme ATM pour désigner la gestion du trafic aérien) est le terme générique désignant le système global en charge d'assurer la bonne marche du trafic aérien. L'ATM est en effet responsable de l'écoulement fluide et sûr du trafic aérien. Cette gestion engage différentes problématiques allant de la circulation des avions sur le tarmac à la planification de trajectoires, le tout en assurant la sécurité parfaite des passagers comme du personnel. L'ATM est un système complexe dans le sens où chacune de ses parties agit plus ou moins indépendamment tout en ayant une action implicite sur les autres parties du système. Par exemple, une mauvaise gestion du trafic à l'échelle européenne pourrait entraîner la formation de zones congestionnées, et ainsi, avoir de grandes répercussions sur le trafic local. Dans ce chapitre, nous détaillerons d'abord dans les premières sections les différents enjeux de l'ATM de manière générale, pour ensuite présenter dans les parties 1.1.3, 1.1.4 et 1.1.5 les différentes composantes de l'ATM. Enfin, nous présenterons le problème considéré dans cette thèse dans la partie 1.1.6.

1.1.1 Les enjeux de l'ATM

L'objectif de l'ATM est de gérer l'ensemble du trafic aérien en assurant une bonne régulation de ce dernier tout en respectant un grand nombre de normes de sécurité. Pour cela, une organisation des tâches est nécessaire, différents acteurs sont impliqués à différentes échelles géographiques et à différentes échelles de temps.

Les différents enjeux de l'ATM transparaissent en considérant le problème du point de vue du passager. Nous voyons donc apparaître deux des enjeux majeurs de l'ATM : garantir une sécurité maximale aux utilisateurs du système et respecter au mieux les horaires annoncés.

Le premier enjeu est primordial, il s'agit de la sécurité. Concrètement, cela se traduit, pour les avions, par l'évitement des zones congestionnées, des zones météorologiques dangereuses, et bien sûr, des autres avions environnants.

Le deuxième enjeu est le respect des planifications effectuées. Ceci se traduit concrètement par le respect des horaires et donc, par la volonté de rester le plus près possible des trajectoires prévues.

Le respect de ces deux enjeux peut paraître simple, mais comme nous l'avons dit précédemment, l'ATM est un système complexe, contenant beaucoup d'incertitudes. Nous verrons par la suite que le respect de ces règles requiert une grande organisation, une vision à différentes échelles, et implique de nombreux acteurs.

1.1.2 Le problème de la résolution de conflits

Pour assurer la sécurité des avions durant leurs vols, il est nécessaire de contrôler en temps réel leurs agissements. C'est le rôle du contrôle du trafic aérien (que nous appellerons par la suite ATC [1]), qui fait partie intégrante de l'ATM. La notion de conflit est essentielle dans l'ATC. Pour des raisons de sécurité, une zone de sécurité est définie autour de chaque avion. Cette zone est tri-dimensionnelle et répond non seulement à l'évitement de collision mais également aux espacements nécessaires pour éviter les turbulences créées par les avions environnants. La norme de *séparation standard horizontale* entre deux avions est de 5Nm [2]. La norme de *séparation standard verticale*, quant à elle, est fixée à 1000ft [3]. Ces distances de séparation définissent ainsi une zone de sécurité autour des avions comme illustré dans la figure 1.1.

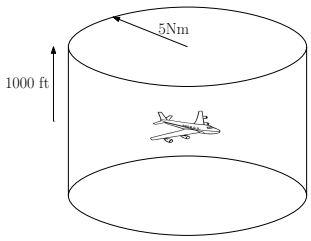

FIGURE 1.1 – Zone de sécurité d'un avion

Lorsqu'un avion pénètre dans la zone de sécurité d'un autre avion, ces deux avions sont dits en *conflit*. La notion de conflit traduit une violation des distances de séparation standard définies ci-dessus. La relation « est en conflit avec » définit une relation d'équivalence et chaque classe d'équivalence associée est appelée *cluster de conflits*.

La résolution de conflits, qui consiste à utiliser des manœuvres d'évitement pour assurer le respect des contraintes de séparation, est actuellement opérée par les contrôleurs aériens. Cependant, du fait de l'augmentation du trafic, l'automatisation partielle ou totale de cette résolution est un problème très étudié pour fournir une aide à la décision au contrôleur. La résolution automatique de conflits devra donc assurer le respect des normes de séparation sur un horizon temporel choisi.

1.1.3 Contrôle aérien et phases de vol

Au cours des dernières décennies, à mesure que le trafic augmentait, l'espace aérien a été divisé en *secteurs*. Ces zones géographiques tri-dimensionnelles ont été définies pour rendre possible le contrôle. En effet, l'espace aérien dans sa globalité présentant un trop grand nombre d'avions, il est divisé en secteurs pour répartir la charge de travail entre les contrôleurs. Il existe également deux types de zones particulières dans l'espace aérien. Les TMA [4] entourent

1. Air Traffic Control
2. *Nautical miles*, 1Nm = 1852m
3. *feet*, 1ft $\simeq 30,48$cm
4. *Terminal Maneuvring Area*, ou région de contrôle terminale, qui réunit l'espace aérien régi par le contrôle d'aérodrome et le contrôle d'approche.

les aéroports et prennent en charge les avions dans cette zone. Des zones de l'espace aérien sont réservées aux militaires. Cet espace dédié peut être *actif* ou non, s'il ne l'est pas, il peut être utilisé pour le trafic civil (sous réserve d'accord avec les autorités compétentes).

Il existe différentes *phases de vol* : le *roulage*, le *décollage*, la *montée*, la *croisière*, la *descente*, l'*atterrissage*. Le contrôle aérien est en quelque sorte compartimenté selon ces phases de vol. Le roulage, le début de la montée et la fin de la descente sont gérés par le contrôle *d'aérodrome* ; la fin de la montée et le début de la descente sont gérés par le contrôle *d'approche*. Le contrôle d'aérodrome et le contrôle d'approche sont responsables des TMA. Enfin, la phase de croisière est gérée par le contrôle dit *en-route*.

Le contrôle d'aérodrome est effectué dans les tours de contrôle. Les contrôleurs d'aérodrome gèrent les décollages et les atterrissages ainsi que le roulage. Ils surveillent le trafic visuellement mais aussi à l'aide du radar-sol.

Le contrôle d'approche a pour rôle de gérer le trafic autour des aérodromes et est généralement situé au pied de la tour de contrôle. Les contrôleurs d'approche doivent gérer les avions en montée (qui viennent de décoller) et en descente (en passe d'atterrir), mais également les avions en transit qui ne font que passer dans la zone d'approche. En montée, le rôle du contrôleur est d'emmener l'avion de son point de sortie de la zone d'aérodrome vers son niveau de vol de croisière. En descente, le contrôleur doit placer les avions de manière à les aligner sur la piste d'atterrissage en assurant une cadence définie par le contrôle d'aérodrome.

Le contrôle en-route prend en charge le trafic dans les zones de croisière entre les aérodromes. Les contrôleurs en-route sont situés dans des centres ANSP [5] (il y en a cinq en France) et travaillent par paires : le contrôleur *organique*, qui assure la communication avec les secteurs voisins, et le contrôleur *radar*, qui surveille le trafic et assure la détection et la résolution de conflits.

Il convient de préciser que le trafic aérien est stratifié. En effet, les avions ne volent pas à des altitudes choisies librement : des niveaux de vol ont été définis. Ces paliers sont espacés de 1000 ft dans les zones de trafic comprenant des avions de lignes (ce qui correspond à la distance de séparation verticale). De plus, il existe un règle dite *règle semi-circulaire* qui stipule que les niveaux de vol pairs sont réservés aux vols orientés vers l'ouest et les niveaux de vol impairs, aux vols en direction de l'est.

1.1.4 Plan de vol et FMS

Afin de structurer le trafic et de faciliter son organisation, des *routes aériennes* ont été définies. À ce jour, les avions de lignes empruntent le plus souvent ces routes aériennes. Celles-ci sont en fait des couloirs aériens (ou *airways*), définis en France par le code de l'Aviation Civile (voir figure 1.2). Ces routes présentent plusieurs intérêts. En premier lieu, elles permettent de structurer le trafic, facilitant ainsi la tâche des contrôleurs, mais également de créer un plan de vol se référant à ces routes.

5. *Air Navigation Services Providers* : fournisseur de services de navigation aérienne. En France, la Direction des Services de la Navigation Aérienne.

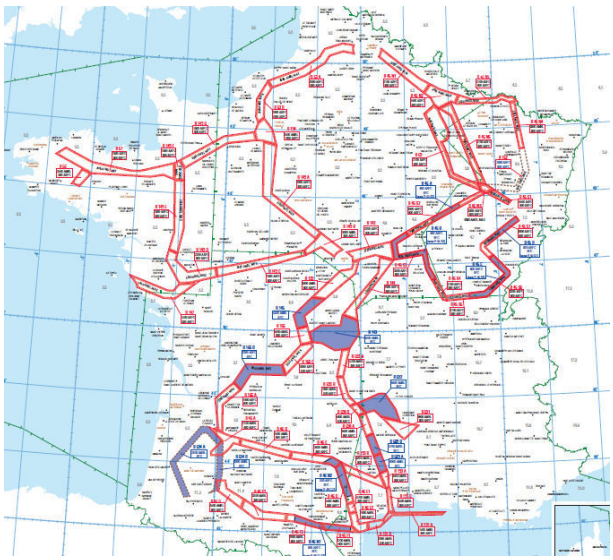

FIGURE 1.2 – Carte des routes aériennes sur la France

Physiquement, ces routes sont tracées à l'aide d'un ensemble de balises. C'est à partir de ces routes et de ces balises que sont établis les *plans de vol*. Pour qu'un avion puisse effectuer un trajet entre deux aéroports, il doit auparavant déposer un plan de vol pour informer les services de la navigation aérienne et demander leur accord. Ce plan de vol contient tous les éléments définissant le vol prévu :

- le type d'avion,
- l'heure de départ,
- le premier niveau de vol demandé pour la croisière (point d'entrée dans la phase de vol de croisière),
- l'équipement de bord,
- la route prévue, décrite par une liste de balises par lesquelles l'avion passera.

Pour suivre ce plan de vol, le pilote utilise un système de gestion de vol embarqué (FMS [6] définit, à partir du plan de vol établi, un *profil vertical* (ou profil d'altitude) et un *profil de vitesse*. Ces deux profils sont optimisés en fonction de différents critères et en considérant les informations contenues dans le plan de vol (notamment le premier niveau de vol demandé pour la phase de croisière). Le FMS dispose également de deux dispositifs d'aide au décollage et à l'atterrissage, respectivement SID [7] qui permet de modifier les procédures de départ, et STAR [8] qui permet d'insérer un point de report ou un circuit d'attente (dans le cas où l'atterrissage serait retardé, par exemple).

1.1.5 La planification du trafic aérien

Le trafic est organisé à différents horizons temporels ou *filtres* de planification du trafic. L'ensemble de ces filtres génère les plans de vol et en assure le suivi.

Le premier d'entre eux est le filtre *stratégique*. Il correspond à une planification à long terme durant laquelle le trafic est organisé de manière macroscopique. Cette planification définit les

6. *Flight Management System*). Le rôle principal du FMS est d'assister le pilote pour l'aider à suivre le plan de vol prédéfini. En premier lieu, le FMS : système embarqué de gestion de vol
7. *Standard Instrument Departure*
8. *Standard Terminal ARea*

schémas d'orientation de trafic, qui fixent les volumes et les flux de trafic de manière à optimiser la sécurité et la capacité de l'espace aérien. La planification stratégique est réalisée six mois avant la journée de trafic concernée et l'organisme qui la réalise se nomme la CFMU [9]. Il est nécessaire de préciser que cette phase de planification se fait à l'échelle européenne.

La CFMU est également en charge du filtre de planification à moyen-terme. En effet, la veille ou l'avant-veille, la journée de trafic est entièrement organisée. Lors de cette phase, la majeure partie des plans de vol sont connus. La CFMU détermine alors les capacités des différents secteurs, et ainsi le nombre maximal d'avions que les services de contrôle auront à gérer. Le nombre de contrôleurs nécessaires pour gérer le trafic des différents secteurs est calculé à partir de cette information.

Ensuite, le trafic est géré par le filtre *pré-tactique*, effectué le jour-même. Ce filtre a pour rôle d'ajuster l'organisation pré-définie de la journée de trafic en fonction des différents aléas pouvant se produire (retards, météo, grèves...). Ainsi, les horaires de décollages et les routes définies par les plans de vol peuvent être ajustés.

Enfin, pendant le vol, le filtre *tactique* prend en charge les avions pendant le vol. C'est en fait le travail des contrôleurs aériens. Les contrôleurs sont responsables du trafic sur une zone de l'espace aérien appelée *secteur aérien*. Ils ont la responsabilité d'assurer la sécurité des avions traversant leur secteur. Pour cela, ils surveillent le trafic, détectent et résolvent les conflits, et assurent le passage des avions de leur secteur à un secteur voisin.

Pour assurer la sécurité du système, il existe un dernier filtre *d'urgence* qui n'intervient qu'en cas de défaillance des filtres précédents. Au niveau du sol, c'est le *filet de sauvegarde* qui déclenche une alarme sur l'écran de contrôle lorsque des avions sont trop proches. Ce filet de sauvegarde ne propose pas de résolution de conflit. Si cet avertissement reste sans suite, un système embarqué à bord des avions appelé TCAS [10] a pour rôle d'éviter une collision imminente. Le TCAS intervient une minute avant la collision. Le TCAS de l'avion communique avec celui des avions voisins et propose un avis de résolution au pilote. Cette résolution de dernier recours se fait dans le plan vertical (il propose à un avion de monter et à l'autre de descendre). Ce système embarqué est rarement utilisé car il se déclenche lorsque tous les autres filtres ont échoués (sauf en TMA).

1.1.6 Présentation du problème considéré

Nous présentons ici le problème traité dans cette thèse, la résolution automatique de conflits. Nous définissons le cadre précis dans lequel nous avons choisi de travailler. Tout d'abord, nous nous sommes focalisés sur la résolution de conflits dans la planification tactique ; nous considérons donc un horizon temporel d'approximativement 20 minutes. Nous avons également décidé de restreindre notre étude au trafic en-route. Par conséquent, comme expliqué plus haut, nous considérons uniquement un trafic bi-dimensionnel : nous ne traitons pas les phases de montée et de descente. Les conflits que nous étudions sont donc un ensemble de trajectoires convergentes dans le plan. Pour résoudre ces conflits, nous utiliserons dans cette étude uniquement des manœuvres de résolution bi-dimensionnelles. En effet, dans le contexte opérationnel du trafic en-route, les manœuvres d'évitement verticales ne sont que très rarement utilisées : les compagnies aériennes n'en veulent pas car elles sont très coûteuses en carburant et inconfortables pour les passagers. Ce sont des manœuvres de dernier recours (pour l'évitement d'une *collision* imminente et non pour l'évitement de *conflits*) mises en œuvre par le TCAS (cf sous-partie 1.1.5), cela ne rentre donc pas dans le champs d'étude de cette thèse.

9. *Central Flow Management Unit*, située à Bruxelles
10. Traffic Collision Avoidance System

Nos méthodes de résolution se focalisent uniquement sur les 20 minutes courantes, seul l'espace aérien correspondant est considéré. Les données d'entrée de notre problème contiennent les positions d'entrée et de sortie des avions dans l'espace considéré, et la vitesse de ceux-ci. Nous considérons que les avions volent à une vitesse constante puisqu'en phase de croisière, la vitesse de l'avion ne varie que de manière minime. Nous définissons ici quelques notations pour les données d'entrée de notre problème. Pour l'avion i, les points d'entrée et de sortie dans l'espace correspondant à l'horizon temporel considéré.

$$\alpha^i_{start} = \begin{pmatrix} X^i_0 \\ Y^i_0 \end{pmatrix} \text{ et } \alpha^i_{end} = \begin{pmatrix} X^i_2 \\ Y^i_2 \end{pmatrix},$$

La vitesse constante de l'avion i est notée v_i. Ces notations sont illustrées dans la figure 1.3 où le segment de trajectoire étudié est représenté en gras.

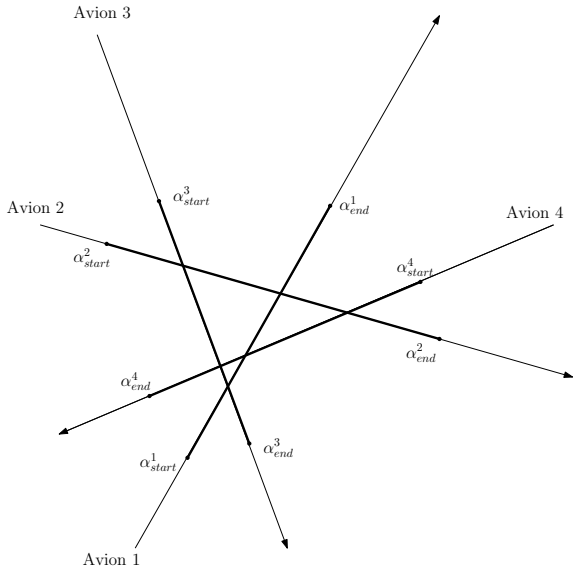

FIGURE 1.3 – Situation de conflit à 4 avions

On souligne le fait que cette thèse considère une automatisation *partielle* de la résolution de conflit ; afin d'obtenir un système de détection qui soit certifiable par les autorités compétentes, l'humain devra rester dans la boucle. On s'intéressera donc ici à l'automatisation en vue d'un outil d'aide à la décision.

Dans la partie suivante, nous détaillons plus avant les différents aspects de notre problème.

1.2 La résolution automatique de conflits

Dans cette partie, nous présentons le rôle d'une méthode de résolution automatique de conflits. Nous détaillons ensuite les projets opérationnels menés dans l'optique de cette automatisation, puis étudions les méthodes de résolution les plus utilisées dans la littérature.

1.2.1 Détection et résolution de conflits

La détection et la résolution de conflits sont intimement liées. En effet, la détection de conflits est définie comme la recherche de l'instant où une manœuvre d'évitement est nécessaire, il est ainsi naturel que détection et résolution soient interdépendantes (le moment de début d'application de la manœuvre ayant une influence sur la manœuvre elle-même). La détection de conflits (CD [11]) consiste à repérer un intervalle de temps durant lequel deux avions (ou plus) ne respectent plus les distances de séparation. La résolution de conflits (CR [12]) a pour rôle de déterminer les manœuvres nécessaires à l'évitement du conflit détecté. Souvent, la détection et la résolution sont menées conjointement par une même méthode (CDR [13]).

A l'heure actuelle, la détection et la résolution de conflits sont opérées exclusivement par le contrôleur aérien. L'automatisation de la détection et de la résolution de conflits est devenue une nécessité du fait de l'augmentation constante du trafic. Jusqu'à maintenant, afin dâĂŹabsorber le trafic supplémentaire en conservant la même charge de travail pour le contrôleur, la taille des secteurs aériens était diminuée. Cela permettait au contrôleur de continuer à gérer le même nombre d'avions. Cette méthode ne peut s'appliquer indéfiniment car il est nécessaire de maintenir une taille de secteur minimum pour permettre au contrôleur de gérer son trafic. En deçà de cette taille critique, le contrôleur passerait trop de temps à faire la liaison avec le secteur suivant par rapport à la gestion des conflits dans son secteur. L'objectif de cette automatisation est de conserver une charge de travail équivalente pour l'opérateur humain tout en augmentant le nombre total d'avions dans le système et donc absorber, in fine, la croissance du trafic.

Les différentes tentatives d'automatisation ont utilisé deux méthodes de détection différentes. La première utilise une discrétisation du temps et vérifie, à chaque pas de temps, la distance entre chaque paire d'avions. Cette méthode a l'inconvénient majeur de coûter très cher en temps de calcul. La deuxième utilise une *grille de détection*. Cette grille peut correspondre à un découpage en 4D (espace + temps), comme dans [24], elle peut aussi ne considérer qu'un découpage spatial, voire, comme dans [11], une discrétisation du plan horizontal, altitude et temps étant contrôlés alors *a posteriori* lors des rapprochements dans le plan. Nous avons mis en œuvre cette dernière approche pour tester notre approche au début de nos recherches. Pour commencer, nous avons décidé d'utiliser le nombre de conflits comme indicateur du respect des contraintes de séparation avant d'utiliser une formulation exacte que nous présentons dans la partie 2.2.1.

11. Conflict Detection
12. Conflict Resolution
13. Conflict Detection and Resolution

1.2.2 Historique des projets opérationnels d'automatisation du contrôle aérien

Depuis bientôt vingt ans, l'automatisation (partielle ou totale) de la résolution de conflits est étudiée. Dans ce but, plusieurs projets ont vu le jour mais n'ont, pour l'instant, pas apporté une réponse complète à ce problème.

Le premier projet en date est le projet américain AERA [14]. Il a été initiée par l'administration Reagan après la célèbre grève des contrôleurs américains de 1981. La première phase, AERA 1, prédit les trajectoires et permet de détecter d'éventuelles violations de séparation. AERA 1 propose donc uniquement des outils de détection de conflits. La deuxième phase, AERA 2, propose aux contrôleurs différentes résolutions traitant les conflits détectés par AERA 1. Dans AERA 3, la décision de résolution est remise à un système automatique résolvant les conflits par paire à l'aide d'un algorithme Gentle-Strict (GS) qui attend le plus longtemps possible avant d'utiliser une manœuvre latérale pour éviter le conflit. Les niveaux supérieurs se chargent d'assurer le trafic global en contrôlant les densités de zones. On peut noter que ce projet ne résolvait pas de conflits impliquant plus de trois avions. Cette tentative fut un échec (seul AERA 1 fut implanté opérationnellement) et ce projet fut abandonné malgré le financement initial de 2 milliards de dollars de la FAA [15].

Le second projet, ARC2000, lancé en 1989, prône une automatisation complète du contrôle en-route européen. Il a été lancé par l'organisme européen Eurocontrol. L'idée initiale est que, si on connait précisément la trajectoire d'un avion, il est possible de déterminer un tube 4D (tube définissant la trajectoire dans un référentiel quadri-dimensionnel espace-temps) entourant celle-ci. Au départ, ARC2000 prévoyait de résoudre les conflits à un horizon temporel de 20 à 30 minutes en optimisant globalement, c'est-à-dire en déplaçant simultanément toutes les trajectoires. Finalement, une solution différente a été envisagée. La stratégie adoptée est, lorsqu'un avion est ajouté au secteur, de construire une trajectoire qui n'entre pas en conflit avec les trajectoires précédemment considérées. Cette méthode utilise une optimisation locale classique de type gradient. Le principal problème de cette modélisation est qu'elle ne répartit pas les déviations équitablement entre les avions. En effet, le premier avion considéré obtiendra une route directe tandis que le dernier peut s'attendre à une importante déviation. Dans la suite du projet, des solutions pour remplacer le principe du « premier arrivé, premier servi » ont été envisagées, notamment une série de règles classant a priori les avions par ordre de priorité. Plus d'informations sur ce projet sont disponibles dans [17].

Un autre projet développé en France (SAINTEX) propose trois approches différentes de l'automatisation du contrôle en route. SAINTEX a été lancé par le Centre d'Études de la Navigation Aérienne dans les années 1990. Trois approches sont abordées dans ce projet :

1. L'approche détection-résolution (CDR) est un système orienté expert, c'est-à-dire qu'il calque son comportement sur celui d'un contrôleur. Le système détecte le conflit, et, selon les paramètres de celui-ci, propose une manœuvre de résolution. Cependant, cette approche gère uniquement les conflits entre deux avions.
2. L'approche 4D se rapproche fortement du projet ARC2000 dans le sens où la trajectoire du nouvel avion est calculée de manière à éviter les conflits avec des trajectoires

14. Automated En-Route Air traffic control
15. Federal Aviation Administration : agence gouvernementale chargée des règlementations et des contrôles concernant l'aviation civile aux ÃLtats-Unis

précédemment traitées, ce qui présente les mêmes inconvénients que précédemment.

3. Enfin, dans l'approche hybride, les avions stables sont gérés suivant l'approche détection-résolution et les autres par l'approche 4D.

Le projet SAINTEX ne s'intéresse qu'aux espaces aériens peu saturés, ce qui limite son application (la résolution de clusters n'est pas abordée). Plus de détails sur le projet SAINTEX sont présentés dans [1].

Le projet FREER [16] est mené par Eurocontrol à partir de 1995. Ce projet s'appuie sur la délégation du contrôle en zone de faible densité à des algorithmes embarqués. Les avions sont donc considérés comme autonome dans un espace défini : le FFA [17]. Dans cet espace, les *règles de l'air étendues* sont appliquées [14] et des algorithmes embarqués basés sur un outil appelé HIPS [18] sont utilisés pour générer des trajectoires sans conflit. Cependant, ces algorithmes ne pouvant gérer des conflits de plus de quatre avions, un contrôle pré-tactique est nécessaire au sol, réduisant ainsi la décharge de travail pour les contrôleurs. Dans la deuxième version, FREER 2, la résolution des conflits à deux avions en espace plus dense est également partiellement déléguée aux avions. Pour cela, il est nécessaire d'équiper les avions du système ASAS [19] assurant la séparation entre avions. Cette méthode n'allège donc pas la charge de travail des contrôleurs de manière significative et la gestion globale du trafic est peu changée.

Enfin, le projet ERASMUS [20][4] est un projet lancé par l'Union Européenne dans le cadre du FP6 [21]. Le but premier d'ERASMUS est de diminuer le taux d'attention nécessaire à un contrôleur pour gérer un avion. Pour cela, la solution envisagée est un pré-traitement du trafic. En effet, ERASMUS propose de résoudre le plus grand nombre de conflits en ne touchant qu'aux paramètres de vol (vitesse, niveaux de vol, etc), le principe étant que l'amplitude de ces changements soit minime. Le but de ce projet est de proposer deux modes d'utilisation : un mode entièrement automatisé et un mode d'aide à la décision pour le contrôleur. Dans le mode automatisé, le caractère minime (souvent nommé *subliminal*) des changements opérés prend tout son sens dans la mesure où la volonté d'ERASMUS est de faciliter la tâche au contrôleur. Par exemple, les changements en vitesse sont limités à $[-6\%, 3\%]$, ces changements sont dits subliminaux puisque le contrôleur ne les perçoit pas.

La résolution de conflits appliquée dans ERASMUS est basé sur le simulateur CATS [22] développé à la DSNA/DTI [23]. Ce simulateur utilise un modèle tabulé de performances avions ainsi que des données de plans de vols réels. Ce simulateur propose deux types de solveurs de conflits. Le premier solveur utilise les algorithmes génétiques ; ce solveur est appliqué à chaque cluster de conflits. Nous détaillerons cette méthode de résolution dans la partie 1.2.5. La principale propriété de ce solveur est de tenter une optimisation globale sans preuve d'optimalité globale. Le deuxième solveur s'appuie quant à lui sur une méthode d'optimisation discrète locale. C'est un solveur séquentiel distribué. Il détermine d'abord un ordre de traitement des avions. Une fois cet ordre défini, les avions impliqués dans le conflit sont traités selon leur

16. *Free-Route Experimental Encounter Resolution*
17. *Free Flight Airspace*
18. *Highly Interactive Problem Solver : jeu d'écrans permettant une visualisation du trafic environnant et l'édition d'une trajectoire d'avion tout en visualisant les zones interdites pour celle-ci*
19. *Airborne Separation Assurance System*
20. *En-Route Air traffic Soft Management Ultimate System*
21. *Sixth Framework Package* : programme de recherche et développement financé par la commission européenne
22. *Complete Air Traffic Simulator*
23. Direction des Services de la Navigation Aérienne / Direction de la Technique et de l'Innovation

ordre de priorité. L'avion en cours de traitement optimise sa trajectoire de manière à éviter les avions dont la trajectoire a déjà été fixée et de façon à minimiser sa distance parcourue. Ce problème est résolu à l'aide de l'algorithme de recherche de chemin dans un graphe A* bien connu des informaticiens et détaillé dans [18].

Ces deux solveurs s'appuient sur trois types de manœuvres pour résoudre les conflits. Le premier type de manœuvre utilise des changements de cap : l'*offset* et le *point tournant* (présentés en figure 1.4). Le deuxième utilise des changements dans le plan vertical : des changements de niveaux de vol. Le dernier type de manœuvre utilise des régulations de vitesse. Les manœuvres horizontales et verticales ne peuvent être combinées dans une manœuvre d'évitement.

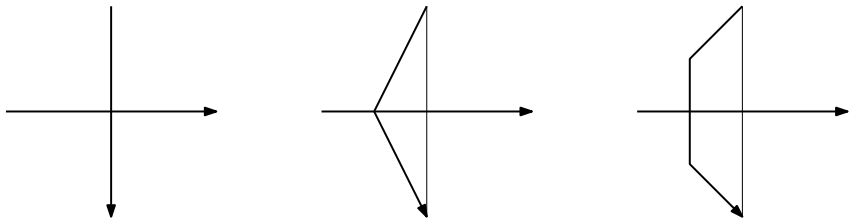

FIGURE 1.4 – Deux types de manœuvres horizontales

Après avoir présenté les différents projets opérationnels passés, intéressons-nous maintenant aux deux grands projets actuels définissant l'évolution future de l'ATM.

1.2.3 Les projets opérationnels actuels : SESAR et NextGen

Les deux grands projets actuels dans l'ATM, SESAR [24], du côté européen, et NextGen [25], du côté américain, ont été lancés au milieu des années 2000 avec un objectif commun : redéfinir le système de l'ATM pour absorber le trafic croissant (un doublement, voire un triplement du trafic est prévu à l'horizon 2020) en assurant, et même en améliorant, les standards de sécurité et de fluidité actuels du trafic, et en diminuant considérablement l'impact environnemental de l'ATM dans son ensemble. Les deux projets étant très proches, nous détaillerons particulièrement le projet SESAR pour ensuite voir les points de divergence avec NextGen.

Détaillons d'abord les ambitions de SESAR et les moyens envisagés pour atteindre ces objectifs élevés. SESAR se base sur trois grands principes : l'uniformisation des systèmes utilisés par les états membres du projet (d'où l'appellation *Single European Sky*), l'introduction de nouvelles technologies, l'introduction de l'automatisation à différents niveaux et enfin le concept de *trajectory-based operations*.

L'ambition première de SESAR était de créer un espace aérien unique pour l'Europe où les systèmes seraient uniformisés pour éliminer les incohérences entre les systèmes ATM des états membres. Dans ce cadre, la notion de FAB [26], qui consiste pour les états à accepter la définition d'espace aérien en fonction des besoins du trafic et non des frontières, a été introduite et ratifiée. Le FABEC [27] regroupant l'Allemagne, la Belgique, la France, le Luxembourg, les

24. *Single European Sky ATM Research*
25. *Next Generation Air Transportation System*
26. *Functional Airspace Block*
27. *Functional Airspace Block Europe Central*

Pays-Bas et la Suisse a ainsi été validé en 2010.

L'introduction d'automatisation intervient dans plusieurs parties de l'ATM. Le facteur limitant de l'ATM européen pour absorber le trafic supplémentaire est la charge de travail limitée d'un contrôleur puisqu'il ne peut gérer un très grand nombre d'avions simultanément. L'objectif est donc d'accroître la capacité de gestion d'un contrôleur en conservant la même charge de travail. Pour cela, plusieurs moyens directs et indirects sont envisagés. Un moyen directement mis à la disposition des contrôleurs est l'automatisation des supports de détection de conflits en développant des composants fournissant des alertes de conflits potentiels comme le STCA [28].

Passons maintenant aux aides indirectes aux contrôleurs. Tout d'abord, l'utilisation de la décongestion *a priori* va permettre de fluidifier le trafic en résolvant les conflits *prévisibles* (c'est-à-dire les conflits qui ne sont pas dus aux aléas) en amont et ainsi, de soulager de manière significative les contrôleurs de conflits "évitables". De plus, une résolution par contrôle de vitesse est en phase d'expérimentation : cet outil est appelé TCSA [29] et résout des conflits en utilisant de très faibles changements de vitesse (ceci fait suite aux travaux d'ERASMUS utilisant les changements de vitesse subliminaux), ce qui soulage également le contrôleur.

Enfin, pour des espaces de faible densité, des outils de résolution de conflits embarqués dans les avions comme ACAS [30], qui résolvent de façon coopérative les problèmes de séparation entre avions, doivent voir le jour. Dans cette configuration, la responsabilité de la sécurité de l'avion revient au pilote (le contrôleur n'étant plus impliqué). La cohabitation entre avions auto-gérés et avions sous la responsabilité des contrôleurs dans le même espace aérien sera assurée par le système ASAS.

La création d'autres outils est également planifiée. Le plus critique d'entre eux est un outil de partage de données : SWIM [31]. Cet outil permettra à tous les acteurs du système d'avoir les informations simultanément. Ceci représenterait une grande avancée, notamment grâce aux échanges d'informations instantanés entre sol et bord. Ces informations permettent à SESAR d'introduire le concept de CDM [32] permettant aux différents acteurs de prendre des décisions collaboratives. Cet outil permettra également de simplifier la gestion entre espace aérien civil et militaire puisque l'information sera immédiatement disponible et la décision collaborative. Enfin, SWIM donnera la possibilité aux contrôleurs d'envoyer des ordres plus complexes aux avions et notamment, d'utiliser des trajectoires courbes au lieu des segments de droite actuels.

Pour finir, le concept essentiel introduit par SESAR est la notion de *Business Trajectory* (BT). En effet, la BT permet d'envisager un système basé sur le respect des trajectoires définies (*trajectory-based operations*). Les trajectoires des avions seront définies de porte-à-porte, intégrant ainsi les zones aéroportuaires dans la gestion des trajectoires. Un des avantages des BT est qu'elles ne sont pas obligées d'emprunter les routes aériennes ; elles peuvent donc utiliser des routes directes. De plus, le principe de BT voudrait que l'avion suive sa BT précisément, puisqu'elle sera définie en tant que trajectoire 4D (rendant possible une plus grande précision) et ne s'en éloigne qu'en cas de situation critique. Ce système de détermination de trajectoire permettrait de plus, de par sa grande précision, d'envisager une réduction des normes actuelles de séparation (5 Nm et 1000 ft), qui prennent bien sûr en compte la précision des instruments de mesures utilisés. Pour plus de précisions sur SESAR, on pourra consulter les références [44, 45].

28. *Short Term Conflict Alert*
29. Tactical Control by Speed Adjustment
30. *Airborne Collision Avoidance System*
31. *System Wide Information Management*
32. *Collaborative Decision Making*

NextGen est l'équivalent américain de SESAR. La problématique est la même, à savoir : arriver à absorber la croissance du trafic américain à l'horizon 2020 en garantissant une sécurité au moins équivalente à celle actuelle et en réduisant l'impact environnemental. Pour cela, NextGen sa base essentiellement sur les mêmes idées que SESAR, notamment au niveau de l'automatisation de la détection et de la résolution de conflits. En effet, une délégation aux avions des tâches de séparation est envisagée de manière plus importante qu'en Europe, déchargeant considérablement les contrôleurs, la responsabilité de sécurité étant toutefois laissée à tout moment au pilote. NextGen se base également sur le concept de trajectoire 4D de grande précision et sur un système d'échange d'informations par un réseau dit Net-Centricity (équivalent de SWIM) réduisant ainsi les incertitudes, tant au niveau de la prédiction de trajectoires qu'au niveau de la prise de décision par les contrôleurs.

Le projet NextGen présente néanmoins des spécificités dues aux différences des systèmes ATM actuels entre l'Europe et les États-Unis (le trafic américain présente de grandes zones de faibles densités par exemple). En effet, le facteur limitant du système ATM américain est son système aéroportuaire, contrairement au trafic européen limité par la capacité de gestion (en nombre d'avions) des contrôleurs. Pour assurer un débit aéroportuaire suffisant pour accueillir deux à trois fois plus de trafic, deux moyens seront utilisés. De nouvelles infrastructures seront développées, puisque de nouvelles pistes seront construites dans la plupart des grands aéroports américains. Pour assurer le contrôle d'approche et d'aérodrome, des *tours virtuelles* seront créées. Le principe d'une tour virtuelle est d'éviter la localisation des contrôleurs dans la tour de contrôle, entraînant ainsi des réductions de coûts de construction, mais surtout permettant d'affecter les contrôleurs de ces tours à "n'importe quel aéroport". Ce concept permet de répondre rapidement à une demande plus importante sur tel ou tel aéroport, rendant le système flexible et réactif.

Le deuxième point sur lequel NextGen semble plus avancé est l'intégration des données météorologiques dans la prise de décision. Cette amélioration se place toujours dans l'esprit d'améliorer la précision des prédictions pour décharger le contrôleur, mais aussi pour permettre une meilleure gestion des flux dans les aéroports. Pour plus d'informations sur le projet NextGen, nous proposons la référence [2].

1.2.4 Les grandes classes de méthodes de résolution de conflits

Il existe de nombreuses méthodes de résolution automatiques de conflits. Différents critères permettent de les classer. Nous verrons dans cette partie les propriétés discriminantes qui nous permettront de placer les méthodes que nous présenterons plus loin dans ce panorama.

Centralisée vs décentralisée : qui prend en charge la résolution ?

ÃŁtre en charge de la détection et de la résolution de conflits implique de grandes responsabilités. Il est donc nécessaire d'attribuer précisément cette responsabilité pour définir clairement qui est en charge de ces tâches. Partant de ce constat, il existe deux grandes classes de méthodes : les méthodes *centralisées* et les méthodes *décentralisées*.

Dans les méthodes centralisées, la détection et la résolution sont opérées par un seul acteur qui donne les ordres de résolution aux avions. Elles centralisent donc la responsabilité. Par exemple, une méthode qui se calquerait sur l'action des contrôleurs aériens serait une méthode

centralisée. Il est important de préciser que les méthodes centralisées sont généralement des méthodes appliquées au sol.

Les méthodes décentralisées répartissent la responsabilité entre différents acteurs. Comme nous le verrons par la suite, certaines méthodes donnent une part de responsabilité aux avions. Par exemple, dans l'approche autonome, les avions voisins communiquent entre eux et déterminent ainsi une stratégie commune de résolution. Ces méthodes de résolution sont implantées à bord de l'avion.

L'approche centralisée a l'avantage d'avoir une vision globale du trafic. En effet, il est difficile, pour l'approche autonome, de pouvoir considérer le contexte global du trafic aérien sans un contrôle au sol.

Coopératives vs non-coopératives

Un conflit implique plusieurs avions. Différentes stratégies de résolution sont possibles : coopératives et non-coopératives. Dans les méthodes coopératives, les avions prennent en compte les manœuvres des autres avions impliqués pour déterminer leurs trajectoires. Par opposition, dans les méthodes non-coopératives, chaque avion décide de ses manœuvres sans considérer les autres avions ou du moins, en considérant comme fixes les trajectoires des autres avions (menant généralement à une approche séquentielle).

Prescrite, optimisée, par champ de force ou manuelle

Le problème de la résolution de conflits présente plusieurs facettes. En effet, le fait de résoudre un conflit assure seulement que la contrainte de séparation n'est plus violée, sans autre considération (l'optimalité de la résolution par rapport à certains critères de coût par exemple). C'est pourquoi il existe différentes approches de résolutions : prescrite, optimisée, par champ de force et manuelle (voir [27]).

La résolution *prescrite* consiste à définir en amont un ensemble de manœuvres de résolution (manœuvres présentées dans la figure 1.4 par exemple) que l'opérateur applique en cas d'apparition de conflits. L'avantage d'appliquer des manœuvres prédéfinies est d'accélérer le temps de réponse de l'opérateur. L'inconvénient majeur de ce type de méthode est de ne pas être adaptable à la situation, et donc de ne pas nécessairement être optimale.

La résolution *optimisée*, elle, cherche non seulement à résoudre le conflit, mais également à optimiser le coût de la résolution selon un critère donnée (distance parcourue, consommation de carburant, etc...). Les manœuvres de résolution utilisées dans cette approche varient selon les auteurs. Nous détaillerons les différentes méthodes de résolution optimisées existantes dans la partie 1.2.5 car c'est cette approche que nous considérons dans cette thèse.

La troisième approche est une approche *par champ de force*. Les avions sont considérées comme des particules qu'on laisse naviguer dans un champ de force. Par exemple, la destination de l'avion est modélisé par une charge positive, les obstacles par une charge négative, et l'avion est une particule négative, attirée par sa destination et repoussé par les obstacles. Cette approche pose généralement des problèmes de respect de certaines contraintes ATM (rayon de courbure ou modifications de vitesse inatteignables par un avion, etc).

Enfin, il existe également une approche dite *manuelle* qui consiste à laisser l'utilisateur proposer une solution et à lui fournir un retour sur la qualité de celle-ci (par essai et erreur).

Prise en compte des incertitudes

Pour détecter les conflits, il est nécessaire de prédire les trajectoires des avions dans un futur plus ou moins proche, à partir des informations de l'état présent. Or, la prédiction de la position d'un avion dans le futur est entachée d'une incertitude. Même si les nouveaux outils de géolocalisation permettent de situer un avion très précisément dans l'espace en tout instant t, la *prédiction* de la trajectoire dans le futur, quant à elle, n'est pas précise. En effet, la prédiction est effectuée au sol où toutes les informations sur les conditions de vol ne sont pas disponibles, notamment le vent. Prenons un exemple simple, la position exacte d'un avion étant connue à l'instant t, la prédiction de la trajectoire est effectuée sans considérer le vent. Ainsi, à un temps $t + \Delta t$, si l'avion est soumis à un vent de face, sa position réelle différera de façon significative de la prédiction effectuée en amont. Ainsi, dans notre contexte de planification tactique, pour un avion volant à une vitesse de 500 Nm/h subissant un vent de face de 20 Nm, présentera une erreur d'estimation de position de 6.5 Nm après $\Delta t = 20$ minutes de vol. Cela pose non seulement un problème au niveau de la prédiction de conflits mais également pour ce qui est de la bonne définition du problème de résolution de conflits anticipés.
On distingue trois types de méthodes de prédiction : *nominale*, *pire cas* et *probabiliste* selon la façon dont sont prises en compte les incertitudes (voir [27]).

L'approche nominale se base sur le vecteur vitesse courant et la position de l'avion pour prédire la trajectoire. Ce modèle n'est précis que pour un horizon temporel très réduit. Il ne prend pas du tout en compte les incertitudes.

L'approche pire cas envisage toutes les manœuvres possibles de l'avion. Cette méthode pessimiste est robuste puisque le système résout toutes les situations possibles. Cependant, elle présente l'inconvénient de surestimer grandement la complexité du trafic et de déclencher très fréquemment de fausses alarmes.

L'approche probabiliste s'appuie sur une prédiction des différentes trajectoires possibles pour chaque avion. Ainsi, un ensemble de trajectoires est considéré, chacune d'entre-elles se voyant attribuer une probabilité d'occurrence (notées p1, p2 et p3 sur la figure 1.5). La manœuvre de résolution prend alors en compte la probabilité de conflits. L'approche probabiliste apparaît comme un compromis entre les deux approches précédentes. Ces trois approches sont illustrées par la figure 1.5.

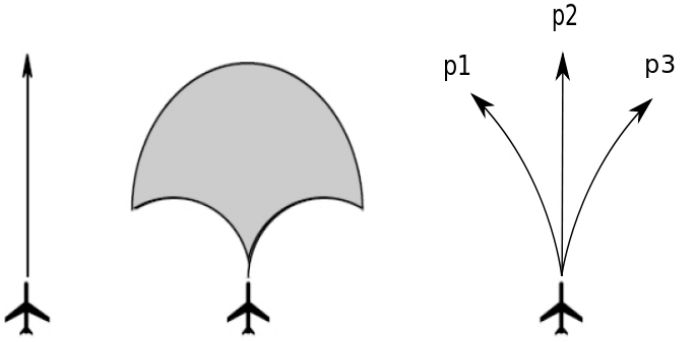

FIGURE 1.5 – Approches nominale, pire cas et probabiliste de la prédiction de trajectoire

Nous verrons dans la suite qu'il existe diverses autres manières de prendre en compte les incertitudes selon la modélisation de trajectoires utilisée. Par conséquent, pour chaque méthode, nous mentionnerons uniquement si elle prend ou pas en compte les incertitudes.

Types de manœuvres utilisés

Actuellement, lorsque les avions sont en vol, il est de la responsabilité du contrôleur d'assurer le respect des normes de séparation entre avions à tout instant. Pour cela, il utilise généralement deux types de manœuvres : le point tournant et la mise en offset (présentées dans la figure 1.4). Ces deux manœuvres se font dans le plan horizontal. Les manœuvres verticales sont utilisées uniquement dans les phases de montée ou de descente, par le biais de mise en pallier temporaire. Dans le trafic en-route, les manœuvres verticales ne sont utilisées par les contrôleurs qu'en dernier recours, de par leur coût élevé en termes de carburant.

Dans les méthodes automatiques de résolution de conflits, il existe deux approches pour modéliser les manœuvres d'évitement. La première se conforme aux manœuvres utilisées couramment, dites manœuvres *opérationnelles*. La seconde ne se préoccupe pas de cet aspect et utilise des manœuvres d'évitement non-opérationnelles. Dans ce cas, nous différencions les manœuvres ne respectant pas les contraintes ATM (de vitesse ou de courbure) et celles les respectant. Néanmoins, les méthodes utilisant des manœuvres d'évitement non-opérationnelles considèrent souvent des contraintes du cadre opérationnel futur fixé par les grands projets SESAR et NextGen, que nous détaillerons plus loin.

Preuve de convergence

On dit qu'une méthode a une preuve de convergence si on peut démontrer que la méthode trouvera une solution au problème de résolution de conflits lorsqu'une telle solution existe. Cet aspect est important pour les méthodes automatiques de résolution puisque c'est une condition nécessaire pour qu'elles soient acceptables dans un cadre opérationnel. Cette préoccupation apparaît car le problème de résolution de conflits est un problème d'optimisation globale difficile. Il peut être décomposé en deux sous-problèmes : la gestion de la combinatoire (ce qui revient ici à déterminer si l'avion i, $i = 1, 2, \ldots, N$, tourne à gauche ou à droite) et la recherche du minimum local du domaine de recherche déterminé par cette gestion de la combinatoire. Dans ce cadre, les méthodes d'optimisation locale classiques (avec preuve de convergence vers un optimum local), ont été peu à peu abandonnées par la communauté de la recherche ATM car elles ne parvenaient pas à trouver une solution optimale satisfaisante. En effet, les méthodes locales peuvent, selon le point de départ choisi, être bloquées dans un optimum local de faible qualité (elles peuvent même converger vers une solution non-réalisable) puisqu'elle ne gère pas l'aspect combinatoire du problème. À l'inverse, les méthodes d'optimisation globale sont construites pour gérer l'aspect combinatoire du problème et sont *susceptibles* de converger vers une solution réalisable : elles ne possèdent pas de preuve de convergence. Nous verrons dans cette thèse que, à l'aide d'une modélisation de trajectoire particulière et une formulation du problème efficace, ces méthodes locales peuvent néanmoins s'avérer efficaces. Les méthodes locales présentent l'avantage d'être très économes en termes de nombre d'évaluations de fonctions et présentent généralement des vitesses de convergence (vers un optimum local) asymptotiques intéressantes.

Précisons cependant qu'une preuve de convergence n'est obligatoire que dans le cas d'une automatisation totale de la résolution. En vue de la conception d'un outil d' aide à la décision

au contrôleur, la méthode automatique de résolution n'étant pas décisionnaire, une preuve de convergence n'est pas requise.

1.2.5 Méthodes automatiques de résolution de conflits pour la planification tactique

Nous résumons ici les méthodes de résolution de conflits les plus représentées dans la littérature. La résolution de conflits est un problème difficile car fortement combinatoire. Par exemple, en se plaçant dans le plan horizontal, un problème impliquant n avions présentera $\frac{n(n+1)}{2}$ conflits potentiels (un par paire d'avions) à chaque pas de temps. Lorsque le nombre d'avions augmente, il apparaît clairement que le nombre de conflits potentiels devient très important, rendant ainsi la résolution de conflits de plus en plus contrainte et par conséquent, la résolution du problème de plus en plus difficile.

Nous présentons d'abord deux méthodes de résolution qui n'utilisent pas une approche optimisée (contrairement à la méthode que nous souhaitons mettre en place). Elles constituent cependant une facette importante des méthodes automatiques de résolution de conflits. Une troisième méthode, utilisant une approche optimisée et modélisant les trajectoires comme un chemin dans un graphe, est ensuite détaillée. Nous détaillons alors deux méthodologies qui effectuent une résolution séquentielle, la deuxième offrant également une méthodologie pour une résolution globale. Nous présentons ensuite deux autres méthodes utilisant les algorithmes génétiques (une méthode d'optimisation globale). La première utilise uniquement les algorithmes génétiques tandis que la deuxième utilise les algorithmes génétiques comme une boucle de plus haut niveau pour gérer la combinatoire du problème et met en œuvre une méthode d'optimisation locale pour effectuer l'optimisation. Enfin, nous présentons un méthode de résolution de conflits utilisant uniquement une méthode d'optimisation locale, ce qui nous inspirera par la suite.

Résolution de conflits par méthode de champs de force

La première méthode de champs de force appliquée à la résolution des conflits aériens a été présentée par Karim Zeghal dans sa thèse de doctorat [48]. Cette méthode de résolution est centralisée et coopérative. Les manœuvres utilisées ne sont pas opérationnelles et ne respectent pas les contraintes ATM comme nous le verrons par la suite. Il n'existe pas de preuve de convergence pour cette méthode.

Cette méthode définit d'abord trois types de forces : les forces attractives, exercées par l'objectif (le point d'arrivée désiré) ; les forces répulsives, exercées par un obstacle (et de direction normale à celui-ci) ; et les forces de glissement qui seront tangentes à l'*équipotentielle de danger*, qui est parallèle à l'obstacle. Le principe est de considérer l'avion comme une particule chargée qui navigue selon le champ de force défini. Ainsi, l'avion est attiré par son objectif tout en étant repoussé par les obstacles sur son passage. Dans la figure 1.6, on voit les différentes forces appliquées sur un avion lors d'un passage à proximité d'un obstacle.

Dans le cas d'un croisement entre deux avions, la vitesse relative de chaque avion par rapport à l'autre est projetée sur l'équipotentielle de danger de l'avion voisin (qui est considéré comme un obstacle mouvant). Deux forces de glissement coordonnées sont ainsi obtenues comme illustré dans la figure 1.7.

Les différentes forces définies ci-dessus définissent donc un champ de force dans lequel navigue l'avion. La recherche de solutions par champs de force cherche une solution sans conflit,

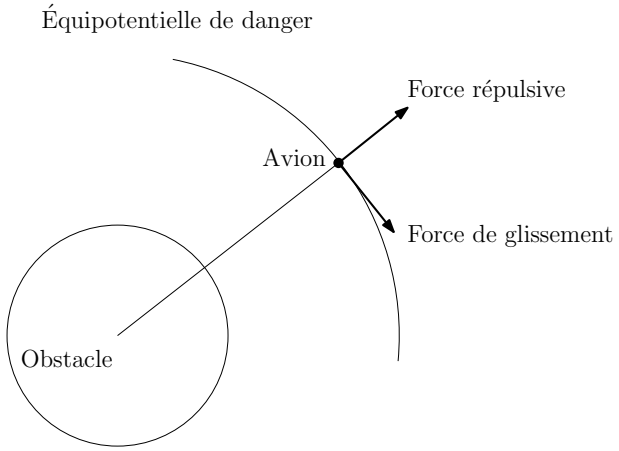

FIGURE 1.6 – Forces exercées par un obstacle sur un avion

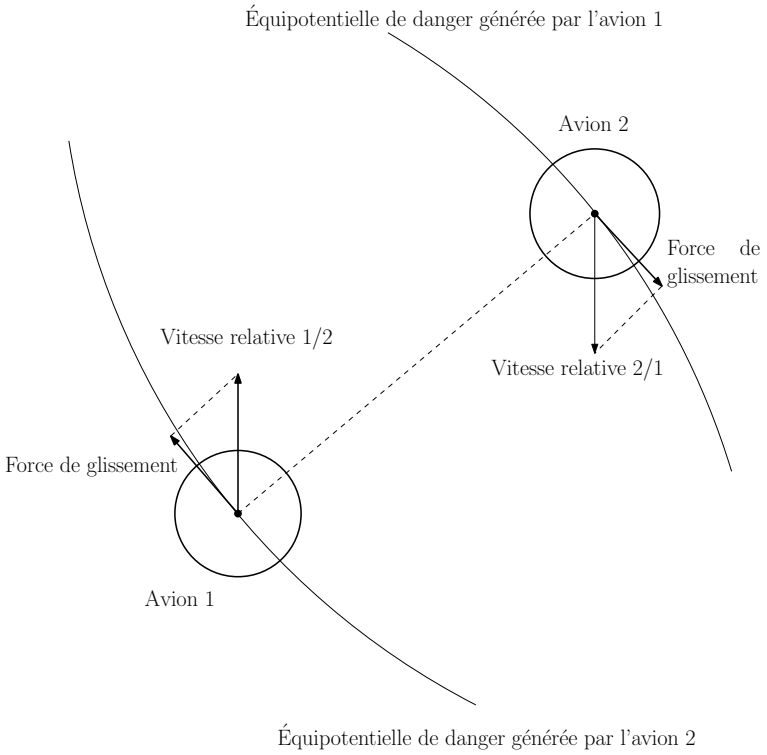

FIGURE 1.7 – Forces de glissement coordonnées entre deux avions

sans optimisation de fonction coût. Cette méthode nécessite bien sûr des aménagements pour une utilisation opérationnelle. Plusieurs utilisations ont été envisagées :

1. La première, dans un système d'avions autonomes dans lequel un système-bord gère les conflits à l'horizon temporel de 4 à 10 minutes. Les avions résolvent alors les conflits eux-mêmes.
2. La deuxième, dans une optique d'aide à la décision fournie au contrôleur. Les trajectoires initiales sont "plongées" dans le champ de force sur l'horizon temporel désiré, et un ensemble de trajectoires sans conflit est proposé au contrôleur.
3. La dernière, dans un système mixte utilisant en partie les deux solutions précédentes (certains avions autonomes, et les autres gérés par le contrôleur assisté d'une aide à la décision).

Cette approche est très intéressante car elle garantit l'obtention de solutions. Cependant, elle a plusieurs points faibles puisqu'elle ne prend pas en compte les contraintes opérationnelles de l'avion (courbure bornée notamment) et qu'elle ne recherche pas une solution optimale (que ce soit en terme de coût, de déviation, de confort passager). De plus, la garantie d'évitement est fortement liée au choix de l'intensité des différentes forces utilisées, lequel est laissé à l'utilisateur au cas par cas.

Résolution de conflits par fonctions de navigation

Les méthodes de résolution de conflits par fonctions de navigation sont en quelque sorte une évolution de la méthode précédente. Au lieu d'utiliser des champs de force, elles utilisent des champs de navigation créés à partir de fonctions présentant des propriétés particulières. La grande différence par rapport à la méthode précédente est l'existence d'une garantie d'évitement (qui peut être assimilée à une preuve de convergence). Ces méthodes sont classées dans les méthodes centralisées, coopératives et utilisent une approche champ de force. Les manœuvres utilisées ne sont pas opérationnelles et les incertitudes ne sont pas prises en compte. Nous évoquons ici plusieurs types de fonctions utilisées pour créer ces champs de navigation et nous préciserons pour chacune si les manœuvres utilisées respectent ou non les contraintes ATM.

Les **fonctions potentiel** sont les premières à avoir été utilisées, en 1985 par Khatib [26] pour l'évitement d'obstacles en robotique. La fonction potentiel donne en tout point de l'espace une valeur réelle. À l'emplacement de la destination de l'avion, la valeur de cette fonction est minimale, et elle est maximale à l'emplacement des obstacles. Pour chaque point de l'espace, la valeur de la fonction potentiel et de son gradient sont calculés et le gradient définit un champ de vecteurs dans lequel le robot naviguera. C'est le même principe que la méthode des champs de force détaillée ci-dessus, à la différence des fonctions utilisées pour créer le champ de navigation. Le mobile se déplace donc dans le champ en suivant la direction de descente définie par le gradient de la fonction potentiel jusqu'à un minimum (la destination du mobile). Le problème de cette approche est que le mobile s'arrête dès lors qu'il atteint un point de gradient nul, ce point pouvant être un minimum local ne représentant pas la destination du mobile (qui, elle, se situe à minimum global). C'est pour éviter ce problème que les fonctions de navigation ont été définies.

Les **fonctions de navigation** ont été définies en 1992 par Rimon et Koditchek [41]. Pour définir une fonction de navigation, plusieurs notions sont nécessaires. L'*espace de travail* $\mathcal{W} \subset \mathbb{R}^2$ de rayon ρ_0, correspondant à l'ensemble de l'espace où l'avion peut se déplacer, est défini par :

$$\mathcal{W} = \{q \in \mathbb{R}^2 : ||q||^2 \leq \rho_0^2\}.$$

Ensuite, les m obstacles de rayon ρ_j et de centre q_j déterminent des zones $\mathcal{O}_i$ interdites pour l'avion :

$$\mathcal{O}_i = \{q \in \mathbb{R}^2 : ||q - q_j||^2 \leq \rho_j^2\}.$$

À partir de ceci on définit l'espace des fonctions de navigation :

$$\mathcal{F} = \mathcal{W} - \bigcup_{i=1}^{m} \mathcal{O}_i.$$

Les fonctions de navigation sont une combinaison d'une fonction "distance au but", d'une fonction "obstacle" et d'une fonction de conditionnement (utilisée pour borner la fonction de

navigation). La fonction distance au but s'écrit

$$\gamma_k(q) = ||q - q_d||^{2k},$$

où k est un paramètre positif et q_d est la destination de l'avion. Remarquons que γ_k s'annule uniquement en q_d, il y a donc un unique minimum local qui est aussi minium global. Les *fonctions obstacles* prennent en compte les obstacles $\mathcal{O}_i$ mais aussi la frontière de l'espace de travail $\mathcal{W}$. Ainsi, pour les frontières, on définit une fonction β_0 :

$$\beta_0(q) = \rho_0^2 - ||q||^2,$$

et pour les obstacles, les fonctions

$$\beta_j = ||q - q_j||^2 - \rho_j^2, \quad 1 \leq j \leq m$$

. Enfin, deux autres fonctions sont requises : la fonction $\rho_k(x) = x^{\frac{1}{k}}$ et une *fonction de conditionnement*, notée σ_λ, est définie pour faire de la fonction de navigation une fonction à valeurs dans $[0; 1]$:

$$\sigma_\lambda(x) = \frac{x}{\lambda + x}, \quad x \geq 0,$$

où λ et k sont des paramètres choisis par l'utilisateur. Enfin, la fonction de navigation est définie comme suit :

$$\phi_k(q) = (\rho_k \circ \sigma_1 \circ \frac{\gamma_k}{\beta})(q) = \frac{||q - q_d||^2}{[||q - q_d||^{2k} + \beta(q)]^{\frac{1}{k}}}.$$

Cette fonction présente des propriétés intéressantes. D'abord, elle n'a qu'un seul minimum en q_d qui représente la configuration objectif (la destination), ce qui évite la présence nuisible de minimums locaux, comme dans le cas des fonctions potentiel. De plus, comme toutes les fonctions de navigation, cette fonction garantit l'évitement des obstacles. En revanche, ϕ_k a l'inconvénient de générer des trajectoires ne respectant pas forcément les contraintes opérationnelles de l'avion (vitesse et courbure bornées).

Cette fonction a donc été reprise et améliorée par Dimarogonas *et al.* dans [9]. En effet, de manière à améliorer la robustesse pour les résolutions à plusieurs avions ainsi que pour garantir une bonne orientation de la trajectoire à proximité de la destination, ces auteurs définissent les **fonctions de navigation dipolaires**. Nous ne détaillerons pas ici tous les termes de cette fonction. Précisons seulement que ce type de fonctions utilise un potentiel répulsif artificiel pour aligner les trajectoires à la destination suivant l'orientation d'arrivée souhaitée. Ceci garantit que la trajectoire soit réalisable (en assurant le respect des contraintes ATM) pour les avions, en évitant par exemple les rotations instantanées sur un point.

Malgré leurs performances remarquables dans le contrôle de mouvement de mobiles, puisqu'elles garantissent l'évitement et l'atteinte de la destination, les fonctions de navigation restent néanmoins pour l'instant inadaptées au trafic aérien car elles ne prennent pas en compte toutes les contraintes opérationnelles (vitesses bornées, trajectoires lisses). Une méthode a été mise au point par Lygeros dans [43] utilisant un Modèle de Contrôle Prédictif (MPC [33]) pour assurer le respect de ces contraintes ATM. Les fonctions de navigation introduites dans [43] sont utilisées en tant qu'outil de modélisation de trajectoires. Le MPC minimise une fonction coût qui

33. Model Predictive Control

traduit les objectifs de l'avion (à savoir, arriver à destination le plus vite possible tout en minimisant le nombre de changement de cap). Ainsi, à chaque pas de temps, t, le MPC fabrique une destination intermédiaire optimale pour chaque avion à l'horizon temporel T ($T > t$) et les fonctions de navigation génèrent ensuite les trajectoires pour rallier la destination intermédiaire à partir de sa position courante et ce, sans conflit. Cette association entre MPC et fonctions de navigation ne présente cependant pas de résultats probants puisqu'elle fournit des trajectoires loin d'être optimales engendrant de grandes déviations. De plus, cette approche génère des trajectoires présentant des variations de vitesse irrégulières qui ne sont pas acceptables dans un contexte opérationnel.

Résolution de conflits par colonies de fourmis

Cette méthode, introduite par Olive dans sa thèse de doctorat [34], est une méthode centralisée, coopérative et utilise une approche de résolution optimisée. Elle s'intéresse aux conflits dans le plan horizontal sans considérer les incertitudes et les manœuvres utilisées sont opérationnelles. Cette méthode ne fournit pas de preuve de convergence. Cette méthode a par la suite été appliquée dans [16].

L'unique manœuvre d'évitement considérée par Olive est le point tournant (cf figure 1.4). En revanche, plusieurs angles de manœuvre sont autorisés : 10, 20 et 30°, et la durée de manœuvre est libre (la manœuvre de résolution peut être appliquée sur toute la trajectoire). Dans cette approche, le temps est discrétisé et la trajectoire d'un avion est modélisée par un chemin dans un graphe. Chaque nÅŞud de ce graphe correspond à une position d'un avion à un temps donné. Pour déterminer les chemins optimaux et sans conflit des N avions, Olive utilise un algorithme basique d'optimisation par colonies de fourmis (ou *Ant Colony Opitmization*, ACO) qui, au lieu d'utiliser un paquet de fourmis pour déterminer un chemin, utilise N paquets pour déterminer le chemin des N avions.

L'algorithme se déroule par génération. Le principe étant que chaque génération de fourmis dépose des phéromones sur son parcours qui guident la génération suivante. La façon dont ces phéromones sont déposées est donc un aspect crucial de cet algorithme. Nous en expliquons le principe ici.

À chaque génération, pour chaque avion i ($i = 1, 2, \ldots, N$), p fourmis (où p est fixé par l'utilisateur, par exemple, $p = 10$ dans [16]) sont lancées dans le graphe représentant une grille discrète dans le plan horizontal à partir du nœud représentant la position initiale de l'avion i. Elles se déplacent de nœud en nœud vers leurs objectifs respectifs. À chaque chemin, ou *path*, emprunté correspond un score s_{path}. Ce score est déterminé en fonction de la manœuvre utilisée. Pour déterminer ce score, trois types de mouvements sont autorisés pour une fourmi : le mouvement U par lequel elle suit sa trajectoire initiale (ligne droite), le mouvement V par lequel elle s'écarte de sa trajectoire, et le mouvement W par lequel elle revient vers sa trajectoire initiale (voir figure 1.8). Durant le chemin de la fourmi, le score du chemin est incrémenté : pour chaque arête, le score est inchangé si la fourmi fait un mouvement U, il est incrémenté de 2 si la fourmi fait un mouvement V, et de 1 si elle fait un mouvement W. De plus, si une fourmi en rencontre une autre sur un même nÅŞud (ce qui correspond à un conflit), ces deux fourmis sont dites *perdues*. Chaque arête, e, ayant mené directement à un conflit, comptabilise le nombre de fourmi perdues dans n^e_{out} (le nombre courant de fourmis perdues à l'arête e). Sur chaque chemin (ensemble d'arêtes consécutives) qu'elles empruntent, les fourmis laissent une certaine quantité de phéromones en fonction de la *qualité* du chemin emprunté. Cette qualité d'un chemin dépend du score s_{path}, qui traduit le rallongement de la trajectoire, et du nombre

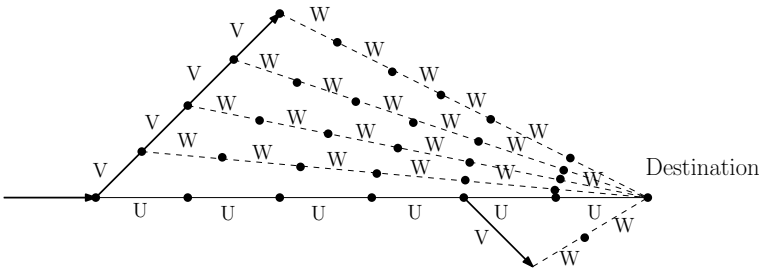

FIGURE 1.8 – Exemple de mouvements dans le graphe pour une trajectoire

de fourmis perdues, n^e_{out} où $e \in path$). Plus précisément, la quantité de phéromone τ laissée sur une arête e est déterminée comme suit :

$$\tau = \begin{cases} 0 & \text{si le chemin contient un conflit} \\ \frac{N-n^e_{out}}{N} \times \frac{\tau_0}{s_{path}} & \text{sinon,} \end{cases}$$

où τ_0 est la quantité initiale de phéromones présente sur le chemin en question.

L'algorithme s'arrête lorsque le score obtenu par l'ensemble des N fourmis ne diminue plus ou lorsqu'un temps maximal de résolution est atteint. Il est important de noter que les phéromones déposées s'évaporent au fur et à mesure des générations selon une loi d'évaporation déterminée à l'avance par l'utilisateur pour éviter une convergence prématurée de l'optimisation.

La méthode d'optimisation appliquée ici utilise des variables d'optimisation mixtes (discrètes et continues), entraînant l'apparition d'une combinatoire. Le critère à optimiser n'est ni continu, ni dérivable. Par conséquent, une telle modélisation ne permet pas le recours aux méthodes d'optimisation numérique locale.

Résolution de conflits par analogie lumineuse

Cette méthode, introduite par Dougui dans sa thèse de doctorat [11] et présentée dans [12], est une méthode centralisée, non-coopérative et optimisée. Les manœuvres utilisées ne sont pas opérationnelles dans le système actuel mais répondent aux spécifications du projet SESAR. Les incertitudes sont prises en compte mais il n'existe pas de preuve de convergence.

Cette méthode se base sur une analogie avec le déplacement de la lumière dans un milieu à indice de réfraction variable. Chaque trajectoire est considérée comme un rayon lumineux dévié par l'indice de réfraction du milieu qui traduit la proximité des obstacles à étudier. Chaque trajectoire suit donc une géodésique (le chemin le plus court, selon une métrique définie, ici le temps, entre les points de départ et d'arrivée de l'avion) calculée approximativement par discrétisation grâce à un algorithme de Branch and Bound (BnB). Le point de départ de l'avion est ainsi considéré comme une source lumineuse à partir de laquelle un front d'onde est émis et propagé par BnB en tenant compte de l'indice de réfraction à chaque pas de temps, définissant ainsi la trajectoire de l'avion. La géodésique étant, par définition, le trajet de temps minimum dans le milieu concerné, la trajectoire retenue pour un avion particulier correspondra au premier rayon arrivé à la destination de cet avion.

La résolution de conflits se fait séquentiellement à travers le calcul de trajectoires opérés par le BnB. Plus précisément, le premier avion, considérant un espace aérien vide, ne dévie pas de sa trajectoire ; le deuxième considérera la trajectoire du premier comme une contrainte fixée dont tiendra compte l'indice de réfraction de ce deuxième avion et ainsi de suite. Le dernier

avion se retrouvera donc logiquement avec une trajectoire plus largement déviée que celle du premier puisqu'elle doit éviter tous les autres avions.

Le point crucial est la définition de l'indice de réfraction du milieu pour que la géodésique évite les trajectoires déjà définies. Chaque trajectoire est définie comme un tube 4D (3D + temps) représentant la zone de protection de l'avion en 4D. Il faut donc que les différents tubes ne s'intersectent pas. Pour cela, l'indice de réfraction, qui est une fonction à valeurs sur $\mathbb{R}^3$, prend une valeur constante élevée, I_{max} (fixée par l'utilisateur), à l'intérieur de chaque tube et prend une valeur minimale unitaire partout ailleurs. Si on considère $X(t) \in \mathbb{R}^3$ un point de l'espace à l'instant t et $Y_i(t), i = 1, \cdots, n-1$ les positions des $n-1$ avions déjà traités, l'indice de réfraction pour le n-ième avion est défini par :

$$I(X(t)) = \begin{cases} I_{max} & \text{Si } \exists i \in \{1, \cdots, n-1\} \text{ tel que } (d_v(X(t, Y_i(t)) \leq 1000\text{ft}) \\ & \text{et } (d_h(X(t, Y_i(t)) \leq 5\text{Nm} \\ 1 & \text{sinon,.} \end{cases}$$

où d_v et d_h sont respectivement les distances dans le plan horizontal et dans le plan vertical.

Cette méthode de résolution a été testée sur une journée entière de trafic sur la France (données historiques). Elle a été associée, à cette fin, à une méthode de détection de conflits. La détection est menée en deux phases : une détection macroscopique se basant sur le chevauchement de parallélépipèdes rectangles encadrant les trajectoires (si les parallélépipèdes se chevauchent, alors l'union des deux parallélépipèdes est considérée comme un cluster de conflits) et une détection microscopique se basant sur une discrétisation de l'espace. Les résultats obtenus sont très concluants. En effet, dans le cas où les incertitudes ne sont pas considérées, 100% des conflits sont résolus. En considérant les incertitudes, le nombre de conflits augmente énormément car les tubes 4D deviennent alors de entonnoirs s'élargissant, s'intersectant ainsi davantage. La méthode est alors adaptée de façon à se donner une marge supplémentaire, avec la possibilité d'imposer à certains avions des points RTA [34] (cf. partie 1.1.4) qui diminuent l'incertitude à certains points de l'espace 4D. Le nombre de points RTA utilisés doit être minimisé par la méthode. Cette méthodologie résout alors 93% des conflits, ce qui se rapproche de l'objectif opérationnel Airbus de 95% de résolution.

Cette méthode par analogie lumineuse est donc prometteuse. Elle présente cependant l'inconvénient d'être appliquée séquentiellement, ce qui peut dévier lourdement un avion plutôt que de répartir les déviations entre les avions (problème d'équité). L'ordre de priorité de traitement des avions ne semble cependant pas avoir de grande influence sur la viabilité de la résolution puisqu'un ordre de traitement aléatoire a toujours permis l'obtention de bons résultats.

Résolution de conflits par Branch & Bound par intervalles

L'approche présentée ici propose une résolution de conflits dans le plan horizontal pour des avions volant à vitesse constante et sans prise en compte des incertitudes. Cette méthode, introduite par Médioni dans [29], est une méthode centralisée et utilise une approche de résolution optimisée. Les manœuvres utilisées ne sont pas opérationnelles. Il n'existe pas de preuve de convergence pour cette méthode et elle ne prend pas en compte les incertitudes.

34. *Required Time of Arrival* : impose à l'avion d'arriver à un point donné de l'espace dans une fenêtre temporelle réduite.

Cette approche se limite aux manœuvres d'évitement par point tournant d'angle constant α (paramètre fixé par l'utilisateur). Le problème qui se pose ici est donc de déterminer les valeurs optimales du temps de début de manœuvre, t_0, de fin de manœuvre, t_1, et du sens de celle-ci (α ou $-\alpha$) pour chaque avion du conflit considéré.

L'algorithme de résolution utilisé est le Branch and Bound par intervalles, notée BnBI par Médioni. Dans une première phase, se concentrant uniquement sur la recherche d'un grand ensemble de solutions satisfaisantes contraintes de séparation, les variables réelles, t_0 et t_1 sont remplacées par des *intervalles* de temps de début et de fin de manœuvre : $T_0 = [t_{0min}, t_{0max}]$ et $T_1 = [t_{1min}, t_{1max}]$. L'angle α prend les valeurs discrètes α ou $-\alpha$ selon le sens de déviation de l'avion considéré. Cette première phase engendrera donc pour chaque avion un intervalle de début de manœuvre T_0, un intervalle de fin de manœuvre T_1 et un sens de déviation. On appelle *pavé*, le produit cartésien $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_N, b_N] \subset \mathbb{R}^N$ de N intervalles ($a_i \leq b_i, i = 1, 2, \ldots, N$). Du fait du traitement par analyse par intervalles, on se retrouvera dans une des trois situations suivantes :

- un pavé admissible ne contenant que des solutions qui respectent les contraintes de séparation,
- un pavé non-admissible ne contenant aucune solution satisfaisant toutes les contraintes de séparation,
- un pavé indéterminé pour lequel on ne peut pas garantir se trouver dans un des deux cas précédents. Typiquement, ce pavé contiendra à la fois des solutions satisfaisant les contraintes de séparation et des solutions violant des contraintes de séparation.

Deux approches différentes sont présentées dans [29]. Une approche globale, où tous les avions sont traités en même temps, et une approche séquentielle qui traite les N avions un par un dans un ordre pré-établi.

L'**approche globale** formule d'abord un problème d'optimisation pour le problème à N avions sur un horizon temporel $[0, t_f]$ avec, pour chaque avion i, deux variables réelles, t_0^i et t_1^i (dates de début et de fin de manœuvre) et une variable discrète, α^i, indiquant le sens de la manœuvre ($\alpha^i \in \{\alpha, -\alpha\}$. Ainsi, l'ensemble de recherche du problème de résolution de conflit sera :

$$E_N = ([0, t_f] \times [0, t_f] \times \{\alpha, -\alpha\})^N$$

où t_f est le temps d'arrivée de l'avion à destination (fin de l'horizon considéré).
La fonction à minimiser f_{opt}^N est définie comme suit pour tout $s \in E_N$. Pour déterminer la solution optimale, l'algorithme nécessite plusieurs outils. Une fonction d'inclusion $f_{opt}^n(t_r)$, ($t_r \in E_{opt}^n$) qui détermine la borne inférieure de l'intervalle considéré. Cette fonction sert en fait à quantifier la qualité de la solution considérée (ici, elle traduit le temps durant lequel les avions sont déviés et le respect des contraintes). Formellement, on définit f_{opt}^n de la sorte :

$$f_{opt}^n(s) = \begin{cases} \sum_{i=1}^n t_1^i - t_0^i, & \text{si } s \text{ respecte les contraintes de séparation} \\ G_r, & \text{sinon,} \end{cases}$$

où G_r est un réel suffisamment grand (supérieur à la pire performance d'une solution respectant les contraintes) et fixé par l'utilisateur. La fonction-objectif représente le temps total durant lequel les avions sont déviés pour une solution satisfaisant les contraintes de séparation. La méthode BnBI cherche donc à minimiser la fonction f_{opt}^N sur E_N pour obtenir un point $\overline{s}$ qui définit les trajectoires optimales des N avions :

$$\overline{s} = \{(\bar{t}_0^1, \bar{t}_1^1, \overline{\alpha}^1), ..., ((\bar{t}_0^N, \bar{t}_1^N, \alpha^N) | \bar{t}_0^i \leq \bar{t}_1^i, i = 1, .., N\}$$

Pour obtenir une telle solution optimale, l'algorithme subdivise récursivement l'ensemble de recherche initial E_N en manipulant une file de pavés qui sont en fait des sous-ensembles de l'espace de recherche. ÃLtant donné qu'il existe 2^N combinaisons possibles, il y a donc initialement 2^N pavés considérés par l'algorithme qui les place dans une file d'intervalles notée F. Chaque intervalle de la file sera traité comme suit :

1. Le pavé P considéré est partitionné en deux pavés P_1 et P_2..
2. Pour les deux pavés obtenus, l'intervalle image est calculé (pour P_1 par exemple, l'intervalle image est défini par $[\min_{P_1} f^N_{opt}, \max_{P_1} f^N_{opt}]$).
3. Pour chacun des deux intervalles images, si la borne inférieure ($\min_{P_1} f^N_{opt}$ dans notre exemple) est supérieure à la valeur $\hat{f}$ de la meilleure solution rencontrée précédemment, le pavé est supprimé car il ne peut pas contenir l'optimum. Sinon, il existe encore deux possibilités. Soit le pavé est indéterminé et il est intégré à la file F, soit le pavé est admissible. Dans ce dernier cas, le pavé est alors réduit à son point central $\hat{s}$, comparé à la meilleure solution courante $\hat{f}$, s'il est moins bon que celle-ci, le pavé est éliminé ; sinon cette solution $\hat{s}$ devient la meilleure solution courante.

Cette approche globale est rapide pour $N = 2$ avions, mais dès que le nombre d'avions augmente, le nombre de sub-division, l'espace mémoire et le temps de calcul requis croissent de façon rédhibitoire.

L'**approche séquentielle** du BnBI fixe d'abord un ordre de priorité sur les N avions et traite ensuite dans cet ordre les avions un par un. Ainsi, le premier avion aura sa trajectoire directe et sera considéré comme une contrainte pour les suivants et ainsi de suite. La qualité des résultats dépend donc fortement de l'ordre de priorité fixé *a priori*. De plus, le dernier avion considéré a de fortes chances de voir sa trajectoire grandement déviée pour éviter tous les avions dont les trajectoires ont été fixées préalablement. Cette approche a été testé pour des conflits à $N = 5$ avions. Il existe, pour 5 avions, $5! = 120$ ordres de priorité possibles. Sur ces 120 ordres de priorité, 90 ont abouti à une solution acceptable (sans conflit).

Même si les méthodes de BnBI semblent s'adapter pour une gestion de trafic dans le plan horizontal (trafic en-route par exemple), elles présentent cependant des inconvénients majeurs. En effet, l'approche globale ne résout les problèmes dans un temps acceptable que pour un nombre limité d'avions et l'approche séquentielle présente une qualité de résolution aléatoire car dépendant fortement de l'ordre de priorité fixé *a priori*.

La méthode d'optimisation appliquée ici utilise des variables d'optimisation mixtes (discrètes et continues), entraînant l'apparition d'une combinatoire. Le critère à optimiser n'est ni continu, ni dérivable. Par conséquent, cette modélisation du problème permettrait l'utilisation de méthodes d'optimisation numérique locale.

Résolution de conflits par Algorithme Génétique

Les Algorithmes Génétiques (que nous noterons désormais AG) appliqués à la résolution automatique de conflits se classent dans les méthodes de résolution centralisées et coopératives et utilisent une approche de résolution optimisée. Dans la méthode que nous présentons ici, les manœuvres autorisées (point tournant et offset) sont opérationnelles et les incertitudes sont prises en compte.

Les AG font partie des méthodes d'optimisation dites *évolutionnaires*, qui constituent elles-mêmes un sous-ensemble des méthodes stochastiques pour l'optimisation globale. Il s'agit de

méthodes heuristiques sans preuve de convergence. Les méthodes évolutionnaires s'inspirent de l'évolution naturelle afin de trouver l'optimum d'un problème d'optimisation. Les AG font évoluer une population de solutions de génération en génération, en suivant des principes de l'évolution comme la *sélection naturelle*, le *croisement* ou encore la *mutation*.

Nous présentons ici la méthode de résolution proposée par Durand dans [15]. Dans cette thèse, Durand propose une résolution de conflits appliquée à la planification tactique (horizon temporel de moins de 12 minutes) en considérant des trajectoires 4D (3D spatiale + temps) qui prennent en compte les incertitudes de prédiction. Les conflits sont résolus à l'aide de manœuvres opérationnelles présentées en figure 1.4, à savoir l'offset et le point tournant. Les AG minimisent ici un critère prenant en compte à la fois le nombre de manœuvres utilisées pour résoudre les conflits et le retard induit par ces manœuvres. Ce critère est pénalisé en cas de non-respect des contraintes de séparation. Dans ces travaux, les AG ne prennent donc pas en charge les contraintes indépendamment de la fonction objectif. Les contraintes de séparation sont prises en compte dans la fonction-objectif à minimiser via l'ajout d'un terme de pénalisation mesurant la violation de ces contraintes.

En pratique, pour résoudre les conflits en temps réel, Durand utilise un simulateur qui effectue, à intervalles de temps réguliers ($\delta = 2$ ou 3 minutes), une détection des paires de conflits sur un horizon temporel plus long ($\Delta = 10$ ou 12 minutes). C'est le principe de la fenêtre temporelle glissante. Les paires d'avions en conflit sont regroupées en clusters selon la règle : si l'avion A est en conflit avec l'avion B et si B est en conflit avec C, alors A, B et C sont en conflit. Un cluster est ainsi une classe d'équivalence de la relation « est en conflit avec » . Ensuite, ces clusters de conflits constituent autant de sous-problèmes à résoudre par l'AG indépendamment les uns des autres, i.e. pour chaque cluster, l'AG ne prend en compte que les avions du cluster considéré. Les différentes solutions obtenues sont alors renvoyées au simulateur qui vérifie si les manœuvres de résolution de chaque cluster n'entraîne pas un nouveau conflit dans le trafic global. Si un conflit apparaît, cette procédure (détection-clusterisation-résolution) est réitérée à partir de la nouvelle situation.

On distingue trois périodes dans l'horizon temporel. Tout d'abord la période *verrouillée*, qui dure δ secondes, pendant laquelle le simulateur évalue la situation. Aucune trajectoire n'est modifiée pendant cette période. Vient ensuite la période *définitive* durant laquelle sont décidées les manœuvres à opérer lors du prochain pas de temps (δ), celles-ci ne pourront être modifiées par l'exécution suivante. Enfin, la période *de manœuvres prévues* durant laquelle les manœuvres prévues au pas de temps précédent sont appliquées (dans la mesure où l'AG a bien trouvé une solution au conflit).

Dans cette méthode, les incertitudes sont modélisées par une modification de la représentation de la position de l'avion. Dans le plan horizontal, à $t = 0$ (instant présent), la position de l'avion est représenté par un point ; sa position exacte est connue. Lorsqu'il se déplace en ligne droite, à $t = \delta$ (prédiction), elle est représentée par un segment, et lorsque la prédiction comprend des changements de cap, la position de l'avion est alors représentée par un polyèdre encadrant les positions possibles de l'avion. Dans le plan vertical, à $t = 0$, la position de l'avion est également modélisée par un point et pour $t = \delta$, par un rectangle. Cette modélisation est illustrée dans la figure 1.9.

Dans sa modélisation, Durand utilise un ensemble de manœuvres prenant en compte les réalités opérationnelles afin de générer des trajectoires utilisables par le contrôleur.

Dans le plan horizontal, Durand n'utilisera que l'offset et le point tournant. Ces manœu-

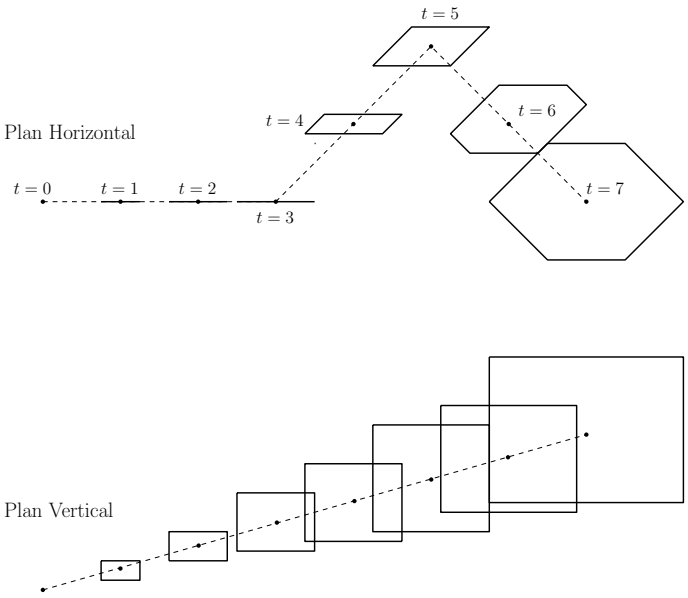

FIGURE 1.9 – Modélisation de l'incertitude sur la position de l'avion dans les plans horizontal et vertical

vres sont ici définies par plusieurs paramètres et présentées dans la figure 1.10 : la valeur du changement de cap, notée α, qui pourra prendre les valeurs 10, 20 et 30° ; la date de début de la manœuvre d'éloignement, t_0 ; celle de fin de la manœuvre d'éloignement, t_1 ; et enfin, celle de retour à la trajectoire initiale t_2. Les quatre paramètres α, t_0 et t_1 et t_2 de chaque avion (définissant ainsi une situation de trafic pour le cluster concerné), constituent les variables d'optimisation. Leurs valeurs sont contenues dans un chromosome. Pour une situation à N avions, le chromosome est donc de taille $4N$ Ce codage représente une solution au problème, il est ainsi utilisé par l'AG.

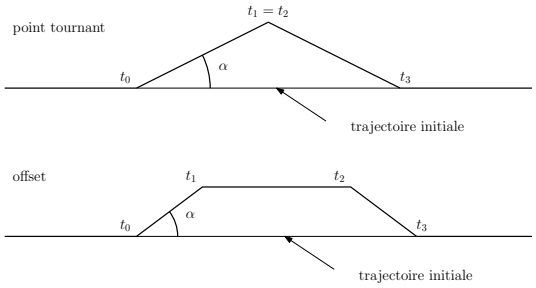

FIGURE 1.10 – Manœuvres utilisées dans le plan horizontal et variables d'optimisation

Dans le plan vertical, la phase de montée peut être interrompue à t_0 et reprise à t_1. Lors de la phase de croisière, l'avion peut être descendu au niveau de vol inférieur à t_0 et remonté à son niveau initial à t_1. En phase de descente, aucune manœuvre n'est possible à l'exception d'une réduction de vitesse de 10, 20 ou 30%. En revanche, il est possible d'anticiper la descente en effectuant un pallier. En effet, 50 Nm avant le début de la descente, l'avion peut descendre à t_0, marquer un pallier à t_1 jusqu'à rattraper le profil de descente. Ces manœuvres sont représentées dans la Figure 1.11. Il est important de noter que le modèle interdit de mener de front une manœuvre horizontale et une manœuvre verticale, et qu'une manœuvre commencée ne peut être remise en cause. Les paramètres de ces manœuvres sont utilisés et stockés dans un chromosome par l'AG.

L'AG explore l'espace des solutions en faisant « évoluer » une population de chromosomes

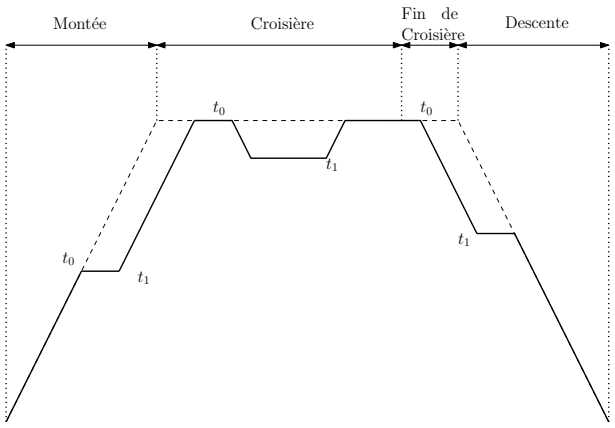

FIGURE 1.11 – Manœuvres utilisées dans le plan vertical

selon les principes de l'évolution. Cette étude utilise une population dont la taille est proportionnelle au nombre d'avions impliqués dans le cluster ($10N$).

Pour pouvoir opérer une sélection dans cette population, une fonction de mesure de la qualité de la solution dite de *fitness* est définie, elle traduit la capacité d'un individu à résoudre le problème, c'est-à-dire, dans notre contexte, à résoudre les conflits tout en effectuant le moins de déviations possibles. Pour cela, une matrice, F, triangulaire inférieure de taille $N \times N$ est utilisée. Chaque élément diagonal, F_{ii}, de cette matrice représente l'allongement de la trajectoire de l'avion i ainsi que le nombre de manœuvres utilisées. Chaque élément non-diagonal, F_{ij}, mesure la violation de la contrainte de séparation entre les avions i et j ($F_{ij} = 0$ si les avions i et j ne sont pas en conflit). À partir de cette matrice, la fitness f est définie :

$$f(F) = \begin{cases} \frac{1}{2} + \frac{1}{1+\sum_i F_{ii}} & \text{si } F_{ij} = 0 \quad \forall i \neq j, \\ \frac{1}{2+\sum_{i \neq j} F_{ij}} & \text{sinon.} \end{cases}$$

La fitness est définie de telle sorte que $f(F) > 0.5$ et on ne tient compte que des allongements et du nombre de manœuvres pour une situation sans conflit, et $f(F) < 0.5$ et on ne considère que les violations de contraintes de séparation pour des situations où il reste des conflits.

Afin de cibler les avions toujours en conflit pour les opérateurs génétiques (notamment pour le croisement), une *fitness locale* est de plus définie pour chaque avion i : $f_i(F) = \sum_{i \neq j} F_{ij}$ pour mesurer les violations de séparation impliquant l'avion i. L'opérateur de croisement, qui a pour objectif d'exploiter les bonnes solutions déjà disponibles afin d'en trouver de meilleures, est défini comme suit. Deux solutions parents A et B sont choisies. Les fitness locales $f_i(A_i)$ et $f_i(B_i)$ sont comparées pour chaque avion i. Si $f_i(A_i) \leq f_i(B_i) - N\tau$ (on rappelle que τ est la distance de séparation standard), alors les deux solutions enfants héritent de la trajectoire de l'avion i du parent A. De même, si $B_i \leq A_i - n\tau$, les deux enfants héritent de la trajectoire de l'avion i de B. Sinon, un croisement *barycentrique*[35] est appliquée aux deux parents A et B. La probabilité qu'une telle démarche de croisement soit appliquée à deux individus de la génération précédente est fixé à $p_c = 0.5$.

L'opérateur de *mutation*, qui a pour objectif de favoriser l'exploration de tout l'espace des solutions, consiste à choisir un avion dont la fitness locale est supérieure à un seuil donné, et

35. Le croisement barycentrique de deux solutions parents P_1 et P_2 consiste à choisir le même *gène* i (correspondant aux paramètres de la trajectoire de l'avion i) de chacun des parents et à affecter aux enfants E_1 et E_2 le gène i correspondant à une pondération des paramètres de trajectoires contenu dans le gène i de chacun des parents.

à modifier les valeurs des paramètres de trajectoire de cet avion, soit de 12 secondes pour les variables de temps, soit de 10°pour la variable d'angle. La probabilité qu'une telle mutation soit appliquée à un avion donné, à une génération donnée, est fixée à $p_m = 0.15$.

Le critère d'arrêt de cette méthode d'optimisation est un nombre d'itérations (de générations) maximal fixé.

L'algorithme de résolution a été mis en œuvre et testé par Durand. Il résout des clusters comprenant jusqu'à $N = 20$ avions convergeant vers un même point. Cependant, les AG n'ayant de pas preuve de convergence, cette méthode ne peut être envisagée pour une automatisation complète.

La méthode d'optimisation appliquée ici utilise des variables d'optimisation discrètes, entraînant l'apparition d'une combinatoire. De plus, le critère à optimiser n'est ni continu, ni dérivable. Par conséquent, cette modélisation du problème ne permet pas l'utilisation de méthodes d'optimisation numérique locale.

Résolution de conflits par AG et programmation linéaire

Nous présentons ici une approche proposée par Médioni *et al.* dans [30]. Cette approche s'intéresse au même problème que celui présenté dans la partie 1.2.5 ci-dessus ; c'est également une méthode de résolution centralisée et coopérative qui utilise une approche de résolution optimisée. Les manœuvres utilisées sont opérationnelles mais les incertitudes ne sont pas considérées. La méthode présentée ici ne possède pas de preuve de convergence.

Le cadre du problème est réduit par rapport à la méthode précédente puisqu'ici, la seule manœuvre de résolution proposée est l'offset. De plus, seul le trafic dans le plan est considéré (2D). Le principe de base de cette méthode est simple : pour que N avions soient séparés, il faut et il suffit que chaque paire d'avions soit séparée. De fait, dans ce cadre, pour deux avions i et j partant des points de départs D_i et D_j et arrivant aux points d'arrivées A_i et A_j, dont les trajectoires initiales forment un angle Φ_{ij} (voir figure 1.12), la condition de séparation est formulée :

$$v_i^2 v_j^2 sin(\Phi_{ij})^2 (t_{ij}^i - t_{ij}^j)^2 \geq d^2(v_i^2 + v_j^2 - 2v_i v_j cos(\Phi_{ij})) \tag{1.1}$$

où v_i et v_j sont les vitesses respectives des avions i et j, d est la norme de séparation choisie, et t_{ij}^i (resp. t_{ij}^j) l'instant auquel la trajectoire non déviée de l'avion i (resp. j) coupe celle de l'avion j (resp. i).

En cas de non-respect de cette condition de séparation pour la paire d'avions i, j, la méthode de Médioni *et al* utilise une manœuvre d'offset pour résoudre le conflit. L'offset de l'avion i est défini par trois paramètres : l'angle de mise en offset ,β, fixé *a priori* par l'utilisateur ; la valeur de l'offset, d_i, (distance entre la trajectoire initiale de l'avion et la route parallèle induite par l'offset) ; et le sens de l'offset (droite ou gauche). On a donc ici finalement une formulation mixte avec variables continues (les valeurs d'offset, d_i) et variables discrètes (le sens de l'offset). Le sens de l'offset, dans lequel réside la combinatoire du problème, est déterminé a priori par l'algorithme génétique qui fixe celui-ci pour engendrer un sous-problème de programmation linéaire. Ainsi, les variables du sous-problème d'optimisation de programmation linéaire sont uniquement les valeurs des offsets d_i (sur la figure 1.12 la valeur de l'offset appliqué à l'avion j est noté d_j). Cette modification de trajectoire entraîne une modification de la condition de séparation (1.1) qui peut alors s'écrire comme une inégalité linéaire en d_i et d_j. Par exemple, si les offsets des deux avions sont orientés vers la droite et que l'avion i passe après l'avion j ($t_{ij}^i > t_{ij}^j$), (1.1) devient :

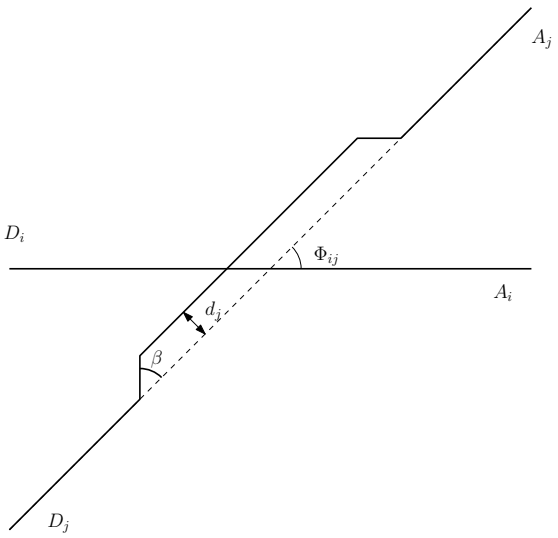

FIGURE 1.12 – Conflit à deux avions et manœuvres de résolution

$$v_i v_j sin(\Phi_{ij})\left(t_{ij}^i + \frac{d_i tan(\frac{\beta}{2})}{v_i} + \frac{d_i cot(\Phi_{ij})}{v_i} + \frac{d_j}{cot(\Phi_{ij})v_i} - t_{ij}^j - \frac{d_j tan(\frac{\beta}{2})}{v_j} - \frac{d_j cot(\Phi_{ij})}{v_j} - \frac{d_i}{cot(\Phi_{ij})v_j}\right) \geq d\sqrt{v_i^2 + v_j^2 - 2v_i v_j cos(\phi_{ij})}$$

Le respect de cette séparation est traité comme une contrainte linéaire dans le problème d'optimisation. La séparation des avions étant traitée par paire d'avions, il y aura donc $\frac{N(N-1)}{2}$ contraintes linéaires. Une contrainte de borne est rajoutée pour chaque d_i (d_i étant la valeur de l'offset, $0 \leq d_i \leq d_{max}$) pour envisager uniquement des déviations « raisonnables ». Le problème traité présentera donc $\frac{N(N+1)}{2}$ contraintes.

En ce qui concerne la fonction-objectif, dans un conflit à N avions, la méthode d'optimisation résout les conflits par offset en minimisant le retard occasionné par ces modifications de trajectoires. La fonction-objectif est en fait la somme des retards des avions :

$$S(d_1, d_2, ..., d_N) = 2\sum_{i=1}^{N} \frac{d_i tan(\frac{\beta}{2})}{v_i},$$

où on rappelle que β est une constante. C'est ainsi que, pour un conflit donné, en supposant connues les variables discrètes de sens d'offset, Médioni, Durand et Alliot obtiennent un problème d'optimisation linéaire à n inconnues (la valeur des offsets d_i) soumis à $\frac{N(N+1)}{2}$ contraintes. Ce sous-problème en d_i est résolu par l'algorithme du simplexe. Cependant comme nous l'avons écrit plus haut, le problème global, lui, étant fortement combinatoire, la possibilité de tomber dans un optimum local est grande et dépend de la configuration initiale choisie (sens de l'offset et ordre de passage des avions). Il existe en effet, pour une situation impliquant N avions, $2^{\frac{N(N+1)}{2}}$ possibilités de configuration initiale ! Par conséquent, leur méthode a recours à un AG pour traiter la combinatoire du problème en amont et déterminer une bonne configuration initiale afin qu'une méthode de programmation linéaire de type simplexe résolve le sous-problème résultant.

Il faut noter que jusqu'à $N = 5$ avions, le nombre total de configurations initiales à considérer reste abordable (32768). Dans ce cas, le problème est résolu par la méthode du simplexe

en partant de la meilleure configuration initiale. Pour une situation à 5 avions, on a constaté que l'AG trouvait dans tous les tests la configuration initiale optimale, en revanche, à partir de $N = 6$ avions, l'AG ne la trouve pas systématiquement. De plus, lorsque la taille du problème augmente significativement (20 avions), le coût d'évaluation d'une configuration devient trop important ; les problèmes de grande taille ne sont donc pas abordés.

Cet inconvénient n'est pas la seule cause de l'abandon de cette méthode. En effet, cette modélisation ne permet pas de prendre en compte les incertitudes sur la position des avions. De plus, la modélisation par droites n'est pas réaliste pour des phases de montée ou de descente. Enfin, cette approche fait l'hypothèse forte qu'aucun conflit n'avait lieu pendant les mises en offset ou les retours sur trajectoire, ce qui rend cette méthode inapplicable dans des zones congestionnées.

La méthode d'optimisation appliquée ici utilise des variables d'optimisation mixtes (discrètes et continues), entraînant l'apparition d'une combinatoire. Le critère à optimiser est continu, et *a priori* dérivable. Par conséquent, cette modélisation du problème permettrait l'utilisation de méthodes d'optimisation numérique locale.

Résolution de conflits par programmation linéaire mixte

Cette méthode, introduite par Pallatino *et al.* dans [35], est une méthode centralisée, coopérative et utilise une approche de résolution optimisée. Les manœuvres utilisées sont opérationnelles mais les incertitudes ne sont pas considérées. Cette méthode s'intéresse à la résolution de conflits dans le plan horizontal, et considère des avions volant à vitesse constante. Dans leur article, Pallatino *et al.* présentent deux approches distinctes, l'une utilisant des manœuvres de changement de vitesse, et l'autre utilisant des manœuvres de changement de cap. Dans les deux cas, le problème d'optimisation est traité comme un problème d'optimisation linéaire mixte. Il n'y pas de preuve de convergence pour aucune de ces deux variantes.

La première approche utilise une variation q_i sur la vitesse v_i de l'avion i pour résoudre les conflits. Cependant, cette variation est limitée pour répondre aux contraintes opérationnelles sur les avions de ligne (puisqu'on s'intéresse aux vols de croisière) :

$$v_{i_{min}} \leq v_i + q_i \leq v_{i_{max}},$$

où $v_{i_{min}}$ et $v_{i_{max}}$ sont les vitesses minimales et maximales autorisées. Elles sont définies ici telles que :

$$\frac{v_{i_{max}} - v_{i_{min}}}{v_{i_{min}}} \leq 0.1.$$

Les variables d'optimisation de cette formulation sont les variations de vitesse q_i, l'objectif étant de minimiser ces variations tout en garantissant l'absence de conflits. Les contraintes doivent donc être exprimées en fonction des variables d'optimisation.

Dans cette modélisation, un avion i est représenté par un triplet (x_i, y_i, θ_i) contenant la position de l'avion dans le plan horizontal et sa direction θ_i. On définit d'abord les angles, notés l_{12} et r_{12}, entre les droites sécantes tangentes aux disques de rayon $\frac{d}{2}$ (où d est la distance de séparation horizontale) représentant les zones protégées des avions $i = 1$ et $j = 2$, et l'axe « horizontal » du plan, est ou ouest par exemple (voir figure 1.13). L'expression de la contrainte de séparation pour une situation à deux avions peut ainsi s'écrire comme suit :

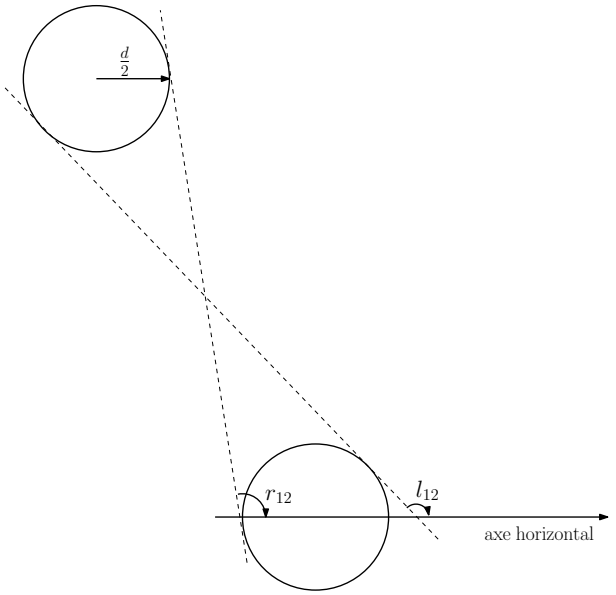

FIGURE 1.13 – Situation à deux avions et angles associés

$$\begin{cases} \frac{(v_1+q_1)\sin(\theta_1)-(v_2+q_2)\sin(\theta_2)}{(v_1+q_1)\cos(\theta_1)-(v_2+q_2)\cos(\theta_2)} \geq \tan(l_{12}) \\ \\ \text{ou} \\ \\ \frac{(v_1+q_1)\sin(\theta_1)-(v_2+q_2)\sin(\theta_2)}{(v_1+q_1)\cos(\theta_1)-(v_2+q_2)\cos(\theta_2)} \leq \tan(r_{12}) \end{cases}$$

Afin de traduire cette condition disjonctive de séparation, (qui contient un opérateur logique « ou »), en contraintes linéaires, des variables booléennes sont introduites (nous ne détaillons pas leur utilisation ici). Cette reformulation conduit à un programme linéaire mixte en nombres entiers (PLNE ou *mixed-integer linear programming*, MILP en anglais). Dans cette étude, le logiciel Cplex [23] d'IBM est utilisé) pour minimiser la somme des variations de vitesse $\sum_{i=1}^{n} -q_i$ sous des contraintes linéaires de séparation. Cette condition de séparation pour $N = 2$ avions est étendue à N avions en considérant chaque paire (i, j) d'avions indépendamment.

L'inconvénient principal de cette approche est que l'on ne peut pas nécessairement résoudre toutes les situations. En effet, il est évident que cette formulation ne pourra pas trouver de solution réalisable pour deux avions arrivant face à face puisqu'elle n'autorise pas de changements de direction. La deuxième approche proposée par ces auteurs évite cet écueil.

La deuxième approche s'intéresse au même problème, mais au lieu d'utiliser des changements de vitesse, elle utilise des changements de cap comme manœuvres d'évitement. Ici, les N avions volent à vitesse constante v_i mais peuvent, en revanche, opérer un changement de direction instantané d'angle p_i par rapport à leur direction initiale θ_i. Ainsi, le problème consiste à trouver le changement de direction optimal de chaque avion tout en garantissant l'absence de conflits.

Les variables d'optimisation considérées ici sont les changements de direction p_i. Les contraintes doivent donc être exprimées en fonction des variables d'optimisation.

En utilisant les angles l_{12} et r_{12} définis dans la première approche, la condition de séparation peut alors être exprimée de la manière suivante pour une situation à deux avions $i = 1$ et $j = 2$:

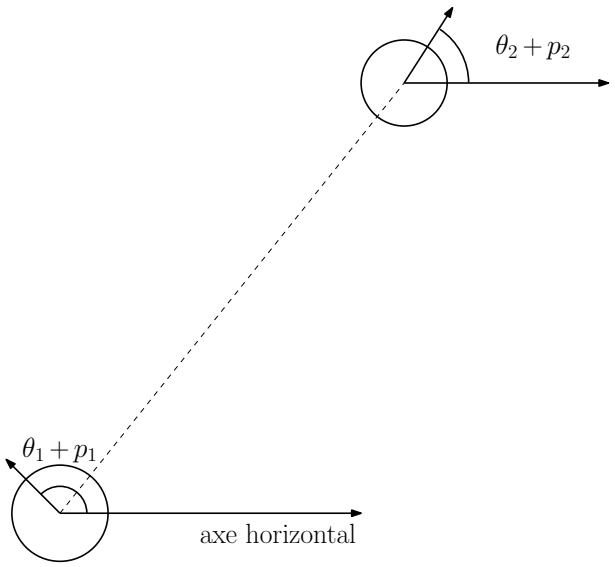

FIGURE 1.14 – Situation à deux avions et changements de direction associés

$$\begin{cases} \frac{\theta_1+p_1+\theta_2+p_2+\pi}{2} \leq r_{12} \\ \text{ou} \\ \frac{\theta_1+p_1+\theta_2+p_2+\pi}{2} \geq l_{12} \end{cases}$$

Cette condition disjonctive de séparation est de nouveau linéarisée via l'ajout de variables booléennes puis étendue à N avions en considérant chaque paire d'avions (i, j) indépendamment. La même méthode (Cplex) de PLNE est utilisée pour minimiser la fonction coût : $\max_{i=1,2,\ldots,N} p_i$ (qui est facilement linéarisée) sous les contraintes linéaires de séparation obtenues.

Les tests numériques présentés dans [35] montrent que la méthode de résolution par changements de vitesse est plus rapide en termes de temps de calcul mais elle ne parvient pas à résoudre tous les conflits, comme expliqué plus haut. L'approche par changements de cap résout un plus grand nombre de conflits tout en produisant des trajectoires généralement moins coûteuses en termes de temps de vol. Les deux approches n'obtiennent cependant pas toujours de solution réalisable. L'utilisation d'une combinaison entre changement de cap et changement de vitesse en utilisant cette modélisation n'est pas possible de par la complexité de la linéarisation des contraintes disjonctives. Il est important de noter que dans le cas de l'approche par changement de cap, le retour sur la trajectoire initiale n'est pas assumée.

La méthode d'optimisation appliquée ici utilise des variables d'optimisation continues. De plus, le critère à optimiser est continu et dérivable. Par conséquent, cette modélisation du problème permettrait l'utilisation de méthodes d'optimisation numérique locale.

1.2.6 Bilan

Nous avons vu dans ce chapitre que la résolution de conflits est un problème qui a été et est toujours beaucoup étudié. De nombreuses approches ont échoué, et toutes présentent des avantages et des inconvénients. Ceci est principalement dû à l'aspect fortement combinatoire du problème et aux contraintes exigeantes de l'ATM qui font qu'il est difficile d'obtenir une méthode répondant à tous les critères. La table 1.1 présente un bilan des différentes méthodes

détaillées plus haut en spécifiant leurs caractéristiques selon les critères vus dans la partie 1.2.4.

Les méthodes que nous développons dans cette thèse sont centralisées, coopératives (pour éviter les déviations importantes d'un avion par rapport aux autres), et utilisent une approche de résolution optimisée. Nous utilisons des manœuvres non-opérationnelles dans le contexte actuel mais adaptées au cadre opérationnel du projet européen SESAR. Nos méthodes travaillent dans le plan horizontal car nous considérons uniquement le trafic en-route et donc des avions stables en altitude. En revanche, nous ne prenons pas en compte les incertitudes. Nous présenterons deux approches, l'une est une heuristique stochastique, sans preuve de convergence ; l'autre utilise des méthodes d'optimisation locale, ayant une preuve de convergence.

Méthode	Centralisée	Coopératives	Incertitudes	Manœuvres	Preuve de convergence
AG	Centralisée	Oui	Oui	Opérationnelles	Non
AG et Prog. Linéaire	Centralisée	Oui	Non	Opérationnelles	Non
Champ de force	-	Oui	Non	Non-opérationnelles	Non
Fonctions de navigation	-	Oui	Non	Non-opérationnelles	Oui
BnBI global	Centralisée	Oui	Non	Opérationnelles	Non
BnBI séquentiel	Centralisée	Non	Non	Opérationnelles	Non
Colonies de fourmis	Centralisée	Oui	Non	Opérationnelles	Non
Prog. Lin. Mixte	Centralisée	Oui	Non	Opérationnelles	Non
Analogie lumineuse	Centralisée	Non	Oui	Cadre SESAR	Non

TABLE 1.1 – Synthèse des méthodes de résolution présentées et de leurs caractéristiques

Chapitre 2

Modélisation de trajectoire et formulation mathématique du problème

Dans ce chapitre, nous présentons les deux modélisations de trajectoire que nous avons utilisées puis nous écrivons la formulation mathématique du problème de résolution automatique de conflits sous la forme d'un problème d'optimisation.

2.1 Modélisation de trajectoire

Dans cette partie, nous présentons les différents modèles mathématiques de trajectoires que nous avons introduits dans cette thèse avant d'expliquer de manière concrète comment nous les avons utilisé. Une des contributions de cette thèse est l'utilisation de trajectoires courbes en se plaçant dans le cadre opérationnel du projet européen SESAR. Cependant dans cette partie, nous présentons également le modèle de trajectoire linéaire par morceaux que nous avons testé de manière à évaluer la viabilité de nos méthodologies en utilisant les exigences opérationnelles actuelles. Rappelons que notre travail porte sur la planification tactique de trajectoire sur le trafic en-route. Par conséquent, comme expliqué dans la sous partie 1.1.6, nos manœuvres se font uniquement dans le plan horizontal.

2.1.1 Trajectoire linéaire par morceaux

Nous utilisons une trajectoire linéaire par morceaux qui correspond aux manœuvres de point tournant présentées dans la partie 1.2.4. Ce type de manœuvre correspond exactement à l'ordre donné par un contrôleur à un pilote pour résoudre un conflit. Ce choix de modélisation est donc adapté à une application opérationnelle immédiate.
Notre modélisation considère la trajectoire uniquement sur un horizon temporel donné. Nous nous intéressons ici à la planification tactique et donc à un horizon temporel d'approximativement 20 minutes. Nous considérons les points d'entrée et de sortie dans cet horizon temporel respectivement comme le point de départ et d'arrivée de l'avion. On suppose ici que la manœuvre de point tournant est imposée sur l'intégralité de notre horizon temporel et la trajectoire initiale est définie comme la ligne droite reliant point de départ et d'arrivée. Le point tournant est placé sur la médiatrice de la trajectoire initiale comme montré sur la figure 2.1.

Le point tournant sera en fait placé sur la médiatrice, pour tous les avions, par la méthode d'optimisation en charge de la résolution de conflits. La trajectoire de l'avion i sera notée γ_i.

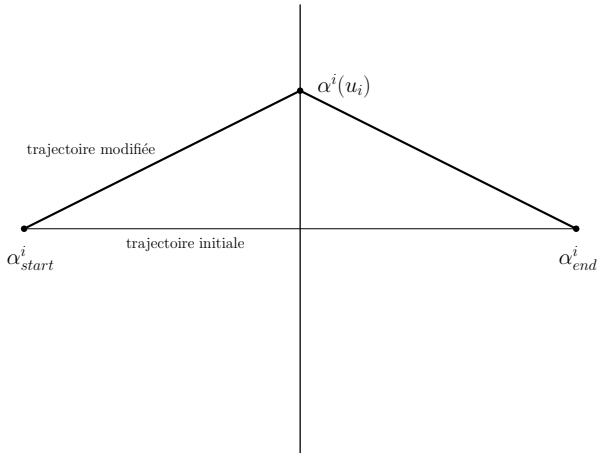

FIGURE 2.1 – Modélisation de trajectoire linéaire par morceaux avec point tournant

Pour chaque avion i ($i = 1, 2, \ldots, N$) impliqué dans le conflit que nous cherchons à résoudre, nous disposons du point d'entrée et de sortie dans la zone de conflit :

$$\alpha^i_{start} = \begin{pmatrix} X^i_0 \\ Y^i_0 \end{pmatrix} \text{ et } \alpha^i_{end} = \begin{pmatrix} X^i_2 \\ Y^i_2 \end{pmatrix},$$

ainsi que de la vitesse de l'avion v_i que nous considérons constante.

À partir de ces données, et pour garantir que la trajectoire modifiée ne rallonge pas de manière excessive la trajectoire initiale, nous définissons une *bande de déviation maximale* qui définit l'ensemble des positions que peut prendre le point tournant sur la médiatrice de la trajectoire initiale (voir figure 2.2). Ainsi, nous définissons pour chaque avion i, $D^i_{max} := \lambda D^i_{init}$, où $D^i_{init} := ||\alpha^i_{end} - \alpha^i_{start}||$ et où λ est une proportion fixée à l'avance par l'utilisateur (nous avons choisi empiriquement $\lambda = 0.3$ dans tous nos tests, ce qui engendrait un rallongement maximal de 10% de la trajectoire). Le point tournant, dont les coordonnées sont notées $\alpha^i(u_i)$, sera donc placé à l'aide d'un pourcentage de cette bande de déviation maximale : $u_i := \pm 100 \frac{d^i}{D^i_{max}}\%$ où d^i est la distance entre le point tournant et la trajectoire initiale (de l'avion i). C'est ce paramètre u_i qui sera déterminé par la méthode d'optimisation. En d'autres termes, le vecteur $u = (u_1, \cdots, u_N)$ a comme composantes nos variables d'optimisation.

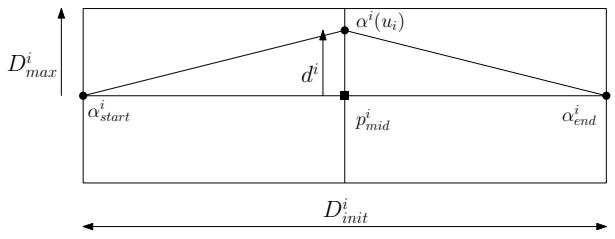

FIGURE 2.2 – Modélisation de trajectoire linéaire par morceaux

Pour calculer la trajectoire i à partir de u_i, les coordonnées du point tournant de l'avion i, contenues dans le vecteur $\alpha^i(u_i)$, sont nécessaires. Pour déterminer ces coordonnées à partir de notre paramètre u_i, on procède comme suit. On définit d'abord le vecteur directeur de la trajectoire de l'avion i : $\delta^i := \alpha^i_{end} - \alpha^i_{start}$. À partir de cela, nous calculons la position de la

base de la médiatrice notée p^i_{mid} (voir figure 2.2) :

$$p^i_{mid} = \alpha^i_{start} + \frac{1}{2}\delta^i.$$

Nous devons également définir le vecteur q^i orthogonal à la trajectoire initiale et de norme D_{max}. Pour cela, nous résolvons le système (2.1).

$$\begin{cases} (\delta^i)^T q^i = 0 \\ \\ ||q^i|| = D^i_{max}. \end{cases} \tag{2.1}$$

Enfin, les coordonnées du point tournant sont déterminées de la manière suivante :

$$\alpha^i(u_i) = p^i_{mid} + u_i q^i,$$

La manœuvre de point tournant ainsi définie décrit une trajectoire déviée sur la gauche de la trajectoire initiale lorsque $u_i > 0$, et sur la droite si $u_i < 0$. Avec cette modélisation, il nous est possible d'utiliser une courbe paramétrée par le temps (nous verrons que ce ne sera plus possible avec la modélisation B-splines). La trajectoire, γ_i, de l'avion i est donc complètement déterminée par la valeur u_i et par le temps, t, comme suit.

$$\gamma^i(u_i, t) = (\gamma^i_x(u_i; t), \gamma^i_y(u_i; t)). \tag{2.2}$$

En utilisant les définitions précédentes et le fait que l'avion se déplace à vitesse constante, nous obtenons la position de l'avion en tout temps t :

$$\gamma^i(u_i, t) = \begin{cases} \alpha^i_{start} + tv_i \frac{(\alpha^i(u_i) - \alpha^i_{start})}{\|\alpha^i(u_i) - \alpha^i_{start}\|}, & \text{si } t \in [0, t^i_{mid}] \\ \\ \alpha^i(u_i) + (tv_i - t^i_{mid} v_i) \frac{(\alpha^i_{end} - \alpha^i(u_i))}{\|\alpha^i_{end} - \alpha^i(u_i)\|}, & \text{si } t \in [t^i_{mid}, t^i_{end}], \end{cases}$$

où $t^i_{mid} = \frac{||\alpha^i(u_i) - \alpha^i_{start}||}{v_i}$ et $t^i_{end} = 2t^i_{mid}$ sont respectivement, le temps d'arrivée en $\alpha^i(u_i)$, et le temps d'arrivée en α^i_{end}.

Nous avons donc défini notre modèle de trajectoire pour un avion en fonction du paramètre u_i défini plus haut. L'ensemble des variables u_i, pour $i = 1, \ldots, N$ seront utilisées par la méthode d'optimisation pour résoudre les conflits de manière optimale.

La modélisation de trajectoire linéaire par morceaux, bien qu'elle soit adaptée au cadre opérationnel actuel, présente plusieurs points négatifs pour les méthodes automatiques de résolution de conflits. Premièrement, la trajectoire linéaire par morceaux n'est pas volable sous cette forme puisque l'avion effectue un virage courbe au niveau du point tournant. Ce décalage entre manœuvre prédite et manœuvre effectuée entraîne une désynchronisation entre la nouvelle trajectoire engendrée par la méthode automatique de résolution et la trajectoire réelle, ce qui se traduit par une incertitude supplémentaire sur la prédiction de conflits en aval (rappelons que notre méthodologie doit être appelée successivement suivant le principe de fenêtre temporelle évoquée en introduction). Nous choisissons donc de recourir à un modèle de trajectoire courbe. Quel serait alors le modèle de trajectoire courbe ayant les meilleures propriétés pour notre contexte applicatif ?

2.1.2 Trajectoire B-spline

Dans cette sous-partie, nous proposons de modéliser mathématiquement des trajectoires courbes, à l'aide de B-splines cubiques. Cet outil mathématique de la théorie de l'approximation nous permet de définir une courbe (c'est-à-dire une infinité de points) à l'aide d'un nombre très limité de paramètres (voire un seul dans la modélisation choisie ici).

Les B-splines cubiques présentent plusieurs avantages significatifs. Tout d'abord, elles produisent des trajectoires volables (car de continuité $\mathcal{C}^2$) et *a priori* adaptée au futur FMS souhaité par le projet européen SESAR. De plus, par construction, elles minimisent les variations d'accélération. Cette propriété est très intéressante puisque l'accélération, et par conséquent la consommation de carburant, sont minimisées, ce qui satisfait les exigences environnementales du projet SESAR.

Les B-splines sont des fonctions de $\mathbb{R}$ dans $\mathbb{R}$ cubiques par morceaux qui permettent d'approcher un ensemble de points, ici de $\mathbb{R}^2$, que l'on appelle *points de contrôle*. On considère d'abord une suite de *nœuds* $s_0, s_1, \ldots, s_k \in \mathbb{R}$. Chaque paire de nœuds consécutifs forme un intervalle. La restriction de la B-spline sur chaque intervalle est un polynôme de degré 3. Ces polynômes sont définis de façon à assurer des raccords deux fois continûment différentiables en chaque nœud. Plus précisément, ce polynôme cubique est déterminé par quatre nœuds : deux nœuds s_k, s_{k+1} définissant l'intervalle courant, un nœud s_{k-1} définissant l'intervalle précédent et un nœud s_{k+2} de l'intervalle suivant.

Dans ce paragraphe, nous nous intéressons à des courbes du plan. On utilisera donc des B-splines pour chacune des deux composantes de la trajectoire (on peut faire de même pour des courbes en dimension 3, etc.). On considère des points de contrôle $(X_j, Y_j) \in \mathbb{R}^2$, $j = 0, \ldots, N_{cp} - 1$, où N_{cp} est le nombre de points de contrôle que nous souhaitons utiliser. Les nœuds que nous utilisons ici seront des entiers. À partir de ces points de contrôle, une courbe d'approximation B-spline, entièrement déterminée par ces points de contrôle, peut être définie :

$$\gamma(s) = (\gamma_x(s), \gamma_y(s)), s \in [0, N_{cp} - 1],$$

où s est le *paramètre naturel* de la B-spline bi-dimensionnelle.
Un exemple de B-spline bi-dimensionnelle est présenté figure 2.3.

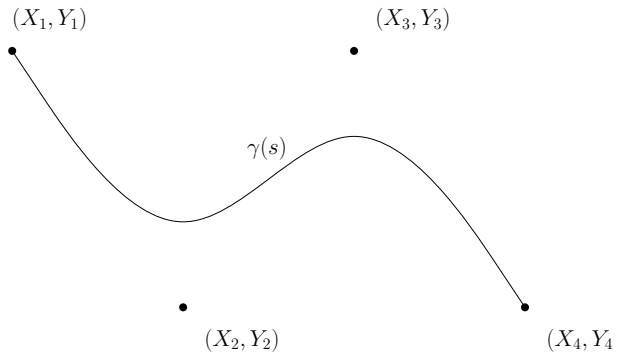

FIGURE 2.3 – Exemple d'approximation de 4 points de contrôles du plan par la B-spline γ paramétrée par s

On note bien sur cette figure que la courbe d'approximation B-spline obtenue ne passe pas nécessairement par les points de contrôles (sauf pour le premier et le dernier point de contrôle). C'est là précisément la différence entre interpolation et approximation. L'approximation élimine les effets oscillatoires notoires qui sont souvent engendrés par l'interpolation [40]. Cet outil

mathématique est donc mieux adapté.

Une courbe d'approximation B-spline s'obtient en utilisant une base de fonctions B-spline $(B_k)_{k=-1,2,\ldots,N_{cp}}$; chaque élément, B_k, de cette base de fonction est un polynôme de degré 3 illustré en figure 2.4. La fonction de base B_0 est la *spline cubique naturelle d'interpolation* [20] des points suivants : $(-2,0)$, $(-1,\frac{1}{6})$, $(0,\frac{2}{3})$, $(1,\frac{1}{6})$, $(2,0)$. Elle est centrée en 0 et vaut zéro hors de l'intervalle $[-2,2]$. Les autres éléments de la base sont simplement obtenus par translations unitaires de B_0.

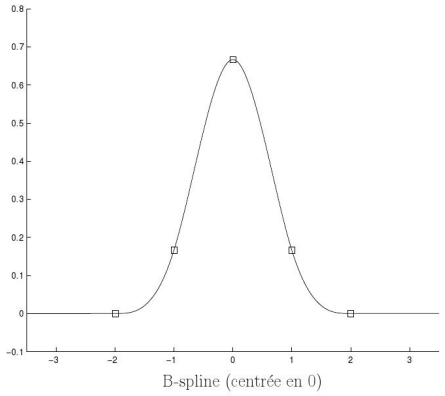

FIGURE 2.4 – Fonction de base B_0 centrée en le nœud 0

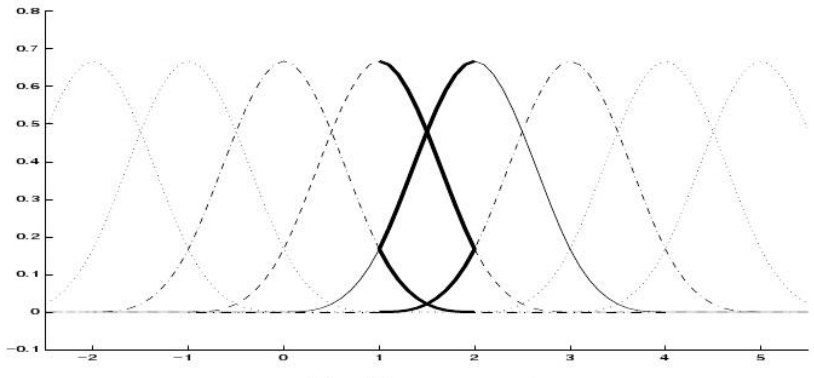

FIGURE 2.5 – Les 4 fonctions de base impliquées dans le calcul de l'approximation dans l'intervalle $[1,2]$

Les fonctions de la base B-spline utilisées pour calculer la courbe $\gamma(s)$ sur un intervalle particulier, par exemple pour $s \in [1,2]$ sont celles à valeur non-nulle dans cet intervalle (en gras sur la figure 2.5). Considérant ceci, la courbe B-spline qui approche les points de contrôles (X_k, Y_k) est définie par la combinaison linéaire suivante des fonctions de base B_k définies plus haut :

$$\gamma_x(s) = \sum_{k=-1}^{N_{cp}+1} X_k B_k(s),$$

et

$$\gamma_y(s) = \sum_{k=-1}^{N_{cp}+1} Y_k B_k(s).$$

En pratique, la construction préalable des B_k, qui est relativement technique [20], n'est pas nécessaire. En effet, en utilisant les valeurs décrites dessous, qui découlent de la construction des fonctions de base, le calcul de l'approximation B-spline est plus efficace. Voyons d'abord cela pour la composante γ_x de notre trajectoire à modéliser. Définissons γ_x^k, $\frac{d\gamma_x^k}{ds}$, $\frac{d^2\gamma_x^k}{ds^2}$ et $\frac{d^3\gamma_x^k}{ds^3}$, la valeur respectivement, de la courbe B-spline et de ses dérivées première, seconde et troisième au point $s = s_k$, où s_k est la valeur du paramètre correspondant au k-ième point de contrôle (s_k est appelé *nœud*). Rappelons que nous avons choisi d'utiliser des nœuds équidistants définis tels que $s_k = k$. On peut montrer qu'en utilisant les relations :

$$\begin{aligned}
\gamma_x^k = \gamma_x(s_k) &= \tfrac{X_{k+1}+4X_k+X_{k-1}}{6}, \\
\tfrac{d\gamma_x^k}{ds} = \tfrac{d\gamma_x}{ds}(s_k) &= \tfrac{X_{k+1}-X_{k-1}}{2h}, \\
\tfrac{d^2\gamma_x^k}{ds^2} = \tfrac{d^2\gamma_x}{ds^2}(s_k) &= \tfrac{X_{k+1}-2X_k+X_{k-1}}{h^2}, \\
\tfrac{d^3\gamma_x^k}{ds^3} = \tfrac{d^3\gamma_x}{ds^3}(s_k) &= \tfrac{X_{k+2}-3X_{k+1}+3X_k-X_{k-1}}{h^3},
\end{aligned} \tag{2.3}$$

où h est la longueur du pas entre les nœuds (dans notre cas, $h = 1$) et en écrivant le développement de Taylor d'ordre 3 entre s_k et s_{k+1} (ce développement de Taylor d'ordre 3 est exact pour un le polynôme est de degré 3 – B-splines cubiques), on obtient la forme explicite suivante pour la B-spline cherchée :

$$\gamma_x(s) = \gamma_x^k + (s - s_k)\frac{d\gamma_x^k}{ds} + \frac{(s - s_k)^2}{2}\frac{d^2\gamma_x^k}{ds^2} + \frac{(s - s_k)^3}{6}\frac{d^3\gamma_x^k}{ds^3} \tag{2.4}$$

On obtient $\gamma_y(s)$ par un raisonnement identique. Précisons ici que, dans la suite de cette étude, nous utiliserons la convention de notation suivante : les indices i et j feront référence aux avions i et j, alors que l'indice k fera référence aux points de contrôles.

Nous détaillons maintenant le calcul de la première composante, $\gamma_x^i(s)$, de la trajectoire $\gamma^i(s)$ de l'avion i pour le cas particulier où trois points de contrôle sont utilisés. Nous reprenons ici le même principe et les mêmes notations que pour le modèle de trajectoires linéaires par morceaux : le point d'entrée, α^i_{start}, le point tournant, $\alpha^i(u_i)$, et le point de sortie, α^i_{end}, devenant ici des points de contrôle. Les notations sont détaillées dans la figure 2.6 pour l'avion i.

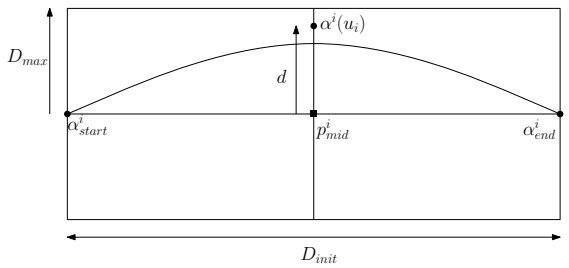

FIGURE 2.6 – Modélisation de trajectoire par approximation B-spline

Comme nous l'écrivions plus haut, chaque polynôme d'ordre 3 définissant la courbe entre deux nœuds s_k et s_{k+1} est déterminé par quatre points de contrôles X_{k-1}, X_k, X_{k+1}, X_{k+2}. Voyons comment nous définissons γ_x^i entre les points α^i_{start} et α^i_{end}, c'est-à-dire sur les deux intervalles $[\alpha^i_{start}, \alpha^i(u_i)]$ et $[\alpha^i(u_i), \alpha^i_{end}]$. Associons X_0 au point d'entrée $\alpha^i_{start_x}$, X_1 au point

$\alpha^i(u_i)$ et X_2 au point de sortie α^i_{end}. On peut montrer [40] que si on souhaite que notre B-spline *interpole* les points d'entrée, α^i_{start}, et de sortie, α^i_{end}, il suffit d'introduire des points de contrôle supplémentaires X_{-1} et X_3 dits points de contrôle *fantômes* de la manière suivante : $X_{-1} := 2X_0 - X_1$ et $X_3 := 2X_2 - X_1$. Or, dans notre cas, on a $X_1 = \alpha^i(u_i) = p^i_{mid} + u_i q^i$. Nous obtenons donc, sur le premier intervalle, pour $s \in [0,1]$ $(k = 0)$:

$$\begin{cases} X_{k-1} &= X_{-1} = 2X_0 - (p^i_{mid_x} + u_i q^i_x), \\ X_k &= X_0, \\ X_{k+1} &= p^i_{mid_x} + u_i q^i_x, \\ X_{k+2} &= X_2, \end{cases} \tag{2.5}$$

et, sur le second intervalle, pour $s \in [1,2]$ $(k = 1)$:

$$\begin{cases} X_{k-1} &= X_0, \\ X_k &= p^i_{mid_x} + u_i q^i_x, \\ X_{k+1} &= X_2, \\ X_{k+2} &= X_3 = 2X_2 - p_{mid_x} - u_i q_x. \end{cases} \tag{2.6}$$

Ainsi, en appliquant (2.5) et (2.6) aux définitions (2.3) et (2.4), on obtient l'expression de $\gamma^i_x(s)$ entre les points d'entrée, α^i_{start}, et de sortie α^i_{end}, c'est-à-dire pour $s \in [0,2]$:

$$\gamma^i_x(u_i;s) = \begin{cases} X_0 + (p^i_{mid_x} + u_i q^i_x - X_0)s \\ +(X_2 - 2p^i_{mid_x} - 2u_i q^i_x + X_0)\frac{s^3}{6,} & \text{si } s \in [0,1] \\ \\ \frac{1}{6}(X_2 + 4p^i_{mid_x} + 4u_i q^i_x + X_0) \\ +\frac{(s-1)}{2}(X_2 - X_0) \\ +\frac{(s-1)^2}{2}(X_2 - 2p^i_{mid_x} - 2u_i q^i_x + X_0) \\ +\frac{(s-1)^3}{6}(2p^i_{mid_x} + 2u_i q^i_x - X_2 - X_0), & \text{si } s \in [1,2]. \end{cases}$$

De même, on obtient $\gamma^i_y(u_i;s)$.

$$\gamma^i_y(u_i;s) = \begin{cases} Y_0 + (p^i_{mid_y} + u_i q^i_y - Y_0)s \\ +(Y_2 - 2p^i_{mid_y} - 2u_i q^i_y + Y_0)\frac{s^3}{6,} & \text{si } s \in [0,1] \\ \\ \frac{1}{6}(Y_2 + 4p^i_{mid_y} + 4u_i q^i_y + Y_0) \\ +\frac{(s-1)}{2}(Y_2 - Y_0) \\ +\frac{(s-1)^2}{2}(Y_2 - 2p^i_{mid_y} - 2u_i q^i_y + Y_0) \\ +\frac{(s-1)^3}{6}(2p^i_{mid_y} + 2u_i q^i_y - Y_2 - Y_0), & \text{si } s \in [1,2]. \end{cases}$$

Nous avons utilisé une modélisation de trajectoire basée sur les B-splines en raison de leur efficacité d'un point de vue temps de calcul : la trajectoire est obtenue directement à partir de quelques opérations élémentaires mettant en jeu les coordonnées de nos trois points de contrôle. Le caractère « parcimonieux » des B-splines est un avantage essentiel dans une volonté d'optimisation de courbe. En effet, elles nous permettront de déterminer la trajectoire d'un avion à partir d'un seul scalaire, comme le montre la partie suivante.

2.2 Modélisation du problème sous la forme d'un problème d'optimisation

Comme nous l'avons déjà évoqué plus haut, cette thèse vise à développer une approche de résolution de conflits optimisée (cf partie 1.2.4). Nous présentons donc ici une formulation mathématique du problème de résolution de conflits sous la forme d'un problème d'optimisation. L'approche optimisée a pour but d'obtenir une situation sans conflit tout en optimisant le tracé des trajectoires selon des critères choisis (consommation, temps de trajet, équité entre avion, etc...). Nous nous concentrons spécifiquement plus spécifiquement sur la minimisation de la moyenne des distances parcourues par les avions. Nous présenterons d'abord la formulation semi-infinie de manière générale pour ensuite détailler les expressions de la fonction-objectif et des contraintes et de leurs dérivées pour les deux modélisations de trajectoire utilisées (linéaire par morceaux et B-spline).

2.2.1 Formulation semi-infinie

Dans cette section, nous introduisons la formulation de type SIP [1][21] qui traite des problèmes minimisant une fonction-objectif soumise à une infinité non-dénombrable de contraintes. Plus précisément, les contraintes sont des fonctions de la variable d'optimisation et d'un paramètre continu. C'est ce paramètre continu qui engendre une infinité de contraintes. La formulation classique d'un problème de SIP est :

$$\begin{cases} \min\limits_{x} f(x) \\ \text{s.c.} \quad g(x;t) \leq 0 \quad \forall t \in [t_1, t_2] \end{cases}$$

où x est ici la variable d'optimisation de dimension finie, f est la fonction-objectif, g la fonction contrainte et t le paramètre continu appelé paramètre *semi-infini*. La contrainte doit donc être satisfaite pour une infinité de valeurs de t ($t \in [t_1, t_2]$ et t est continu).

Cette approche nous permet de modéliser directement les contraintes de séparation à l'aide d'une seule contrainte pour chaque paire d'avions. En effet, les distances de séparation devant être satisfaites *en tout temps*, le temps est donc notre paramètre semi-infini. Nos variables d'optimisation sont, comme annoncé dans le chapitre 2.1, les u_i déterminant la position du point de contrôle (ou le point tournant) de chaque avion i. Nous définissons un vecteur u contenant la valeur u_i des N avions. La dimension de notre problème d'optimisation est donc réduite au nombre d'avions impliqués dans le conflit considéré.

Nous définissons la fonction-objectif $f(u)$ comme la moyenne des $f^i(u_i)$ sur les N avions :

$$f(u) = \frac{1}{N} \sum_{i=1}^{N} f^i(u_i),$$

où chacune des fonctions $f_i(u_i)$ est représentative de la différence relative de distance parcourue par l'avion i entre sa trajectoire modifiée et sa trajectoire initiale. Nous détaillerons la fonction-objectif plus précisément dans la partie 2.2.2.

Intéressons-nous maintenant aux contraintes de notre problème d'optimisation : le respect des distances de séparation. Pour cela, reprenons les notations du chapitre 2.1 : $\gamma^i(u_i; t)$

1. *Semi-Infinite Programming*

représente la trajectoire 2D de l'avion i. La distance de séparation à l'instant t pour les avions i et j est :

$$\|\gamma^i(u_i;t) - \gamma^j(u_j;t)\|_2, \tag{2.7}$$

où $||\cdot||_2$ est la norme l_2. Nous définissons la fonction $c^{ij}(u;t)$ comme étant la distance au carré :

$$c^{ij}(u;t) := \|\gamma^i(u_i;t) - \gamma^j(u_j;t)\|_2^2,$$

afin de simplifier les calculs qui suivent.

Rappelons que nous travaillons uniquement dans le plan et que nous utilisons donc la distance de séparation horizontale $\tau := 5\text{Nm}$. La valeur $t^{ij}_{min} = \min(t^i_{end}, t^j_{end})$, avec t^i_{end} le temps de sortie de l'avion i (le temps où celui-ci atteint son objectif α^i_{end}). Ainsi, la contrainte de séparation doit être respectée pour tout $t \in [0, t^{ij}_{min}]$. Par conséquent, nous pouvons formuler notre problème d'optimisation comme suit :

$$\begin{cases} \min\limits_u f(u) = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} f^i(u_i) & \\ \text{s.c.} \quad c^{ij}(u;t) \geq \tau^2 \qquad \forall t \in [0, t^{ij}_{min}]; & \begin{array}{l} i = 1, 2, \ldots, N-1; \\ j = i+1, 2, \ldots, N. \end{array} \end{cases}$$

La valeur $t^{ij}_{min} = \min(t^i_{end}, t^j_{end})$, avec t^i_{end} le temps de sortie de l'avion i (le temps où celui-ci atteint son objectif α^i_{end}), ainsi la distance de séparation standard doit être respectée pour tout $t \in [0, t^{ij}_{min}]$.

En utilisant les B-splines, les trajectoires $\gamma^i(u_i;s)$ et $\gamma^j(u_j;s)$ ne sont pas paramétrées par le temps directement. Par conséquent, afin d'utiliser une méthode d'optimisation, nous devrons échantillonner numériquement les trajectoires en temps (par interpolation linéaire) pour obtenir $\gamma^i(u_i;t)$ et $\gamma^j(u_j;t)$ pour toute évaluation des fonctions contraintes c^{ij}.

2.2.2 La fonction-objectif et ses dérivées

Ici, nous détaillons la fonction-objectif et ses dérivées dans le cas d'une modélisation linéaire par morceaux des trajectoires puis dans le cas d'une modélisation B-spline. En optimisation numérique, la connaissance de la sensibilité de la fonction-objectif par rapport aux variables d'optimisation est en effet une information cruciale qui doit être exploitée lorsque disponible.

Cas de la modélisation linéaire par morceaux des trajectoires

L'expression de la fonction-objectif est triviale dans ce cas puisque la distance parcourue par un avion i en termes de u_i s'obtient directement (cf figure 2.2) . Soit $\hat{T}^i(u_i)$ la distance parcourue par l'avion i en utilisant le point tournant défini par u_i. Par le théorème de Pythagore et en utilisant le fait que $D_{max} = \lambda D_{init}$), on obtient :

$$\hat{T}^i(u_i) = 2D_{init}\sqrt{\lambda^2 u_i^2 + \frac{1}{4}}$$

À partir de cela, nous définissons précisément la fonction $f^i(u_i)$ comme suit :

$$f^i(u_i) = \frac{(\hat{T}^i(u_i))^2 - (\hat{T}^i(0))^2}{(\hat{T}^i(1))^2 - (\hat{T}^i(0))^2}$$

Nous utilisons le carré de la distance parcourue car, sans inconvénient majeur du point de vue modélisation de notre problème opérationnel, nous parvenons ainsi à simplifier le calcul des dérivées. Nous normalisons de manière à considérer l'impact de la déviation par rapport à la trajectoire initiale (notons que $\hat{T}^i(0)$ correspond à la distance parcourue par la trajectoire initiale et que $\hat{T}^i(1)$ correspond à la pire déviation autorisée). Un simple calcul conduit finalement à $f^i(u_i) = u_i^2$. La fonction-objectif s'exprime donc :

$$f(u) = \frac{1}{N}\sum_{i=1}^{N} f^i(u_i) = \frac{1}{N}\sum_{i=1}^{N} u_i^2. \tag{2.8}$$

La dérivée de notre fonction-objectif (2.8) par rapport à nos variables d'optimisation u_i est donc immédiate. Pour $i = 1, 2, \ldots, N$, on a :

$$\boxed{\frac{\partial f(u)}{\partial u_i} = \frac{2u_i}{N}},$$

ou encore :

$$\boxed{\nabla f(u) = \frac{2u}{N}}.$$

Cas de la modélisation B-spline des trajectoires

Dans le cas des B-splines, l'expression de la fonction-objectif et de son gradient sont plus complexes. Nous utilisons ici la notation définie dans la sous-partie 2.1.2, notamment $\gamma^i(u_i; s) = \Big(\gamma_x^i(u_i; s), \gamma_y^i(u_i; s)\Big)$. La distance totale parcourue $\hat{T}^i(u_i)$ en fonction de la position du point de contrôle définie par u_i pour l'avion i est :

$$\hat{T}^i(u_i) := \int_0^2 \sqrt{\Big(\frac{d\gamma_x^i(u_i; s)}{ds}\Big)^2 + \Big(\frac{d\gamma_y^i(u_i; s)}{ds}\Big)^2}\, ds. \tag{2.9}$$

C'est en fait l'expression de l'abscisse curviligne d'une courbe paramétrée par s (rappelons que dans notre contexte à trois points de contrôle, s varie de 0 à 2, c'est pourquoi l'intégration se fait sur $[0, 2]$). L'abscisse curviligne correspond à l'expression d'une énergie, et nous obtenons de la géométrie riemannienne [10] que minimiser $\hat{T}^i(u_i)$ est équivalent à minimiser la fonction $T^i(u_i)$ définie comme suit :

$$T^i(u_i) := \int_0^2 \Big(\frac{d\gamma_x^i(u_i; s)}{ds}\Big)^2 + \Big(\frac{d\gamma_y^i(u_i; s)}{ds}\Big)^2 ds. \tag{2.10}$$

Dans un souci de normalisation de notre fonction-objectif, toujours en regard de notre volonté de considérer l'impact de la déviation par rapport à la trajectoire initiale, nous définissons ici l'expression de $f^i(u_i)$:

$$f^i(u_i) = \frac{T^i(u_i) - T^i(0)}{T^i(1) - T^i(0)}.$$

La fonction-objectif s'écrit donc :

$$f(u) = \frac{1}{N}\sum_{i=1}^{N} \frac{T^i(u_i) - T^i(0)}{T^i(1) - T^i(0)}. \tag{2.11}$$

Proposition 1. *Le gradient de la fonction-objectif $f(u)$ présentée dans (2.11) s'écrit sous la forme explicite suivante :*

$$\boxed{\nabla f(u) = \frac{32u\left(\left(q_x^i\right)^2 + \left(q_y^i\right)^2\right)}{15N\left(T^i(1) - T^i(0)\right)}}. \tag{2.12}$$

Démonstration. Le terme $\frac{1}{T^i(1)-T^i(0)}$ étant une constante de normalisation (indépendante des variables d'optimisation u_i), nous pouvons, sans perte de généralité, l'ignorer dans le détail de nos calculs. Dans les calculs qui suivent, chacun des deux termes de l'intégrale (2.10) dans l'expression de $T^i(u_i)$ sera considéré séparément. Il nous suffira de détailler le terme impliquant γ_x^i, le terme impliquant γ_y^i se déduisant directement.

Nous rappelons ici l'expression de $\gamma_x^i(u;s)$, la première composante de $\gamma^i(u;s)$. Par souci de simplification de la notation, par la suite, l'indice i est abandonné dans la notation des points de contrôle X_k^i puisque nous considérons uniquement la trajectoire de l'avion i (et on fait de même pour les ordonnées Y_i^k en jeu dans l'expression de $\gamma_y^i(u_i;s)$) . Or, nous avions :

$$\gamma_x^i(u_i;s) = \gamma_x^i(u_i;s_k) + (s-s_k)\frac{d\gamma_x^i(u_i;s_k)}{ds} + \frac{(s-s_k)^2}{2}\frac{d^2\gamma_x^i(u_i;s_k)}{ds^2} + \frac{(s-s_k)^3}{6}\frac{d^3\gamma_x^i(u_i;s_k)}{ds^3}. \tag{2.13}$$

Nous rappelons également l'expression de $\gamma_x^i(u_i;s_k)$, $\frac{d\gamma_x^i(u_i;s_k)}{ds}$, $\frac{d^2\gamma_x^i(u_i;s_k)}{ds^2}$ et $\frac{d^3\gamma_x^i(u_i;s_k)}{ds^3}$ en termes des points de contrôle qui nous vient de la théorie des B-splines [13] :

$$\begin{cases} \gamma_x^i(u_i;s_k) & = (X_{k+1} + 4X_k + X_{k-1})/6 \\ \frac{d\gamma_x^i(u_i;s_k)}{ds} & = (X_{k+1} - X_{k-1})/2 \\ \frac{d^2\gamma_x^i(u_i;s_k)}{ds^2} & = X_{k+1} - 2X_k + X_{k-1} \\ \frac{d^3\gamma_x^i(u_i;s_k)}{ds^3} & = X_{k+2} - 3X_{k+1} + 3X_k - X_{k-1}. \end{cases}$$

Dans notre modélisation, une trajectoire est déterminée par trois points de contrôle. Par conséquent, la courbe d'approximation B-spline est exprimée sur deux intervalles $[s_0, s_1]$ (cas où $k = 0$) et $[s_1, s_2]$ ($k = 2$). Ces deux parties sont calculées séparément.

Sur le premier intervalle, la trajectoire est exprimée comme suit :

$$\gamma_x^i(u_i;s) = \gamma_x^i(u_i;s_0) + (s-s_0)\frac{d\gamma_x^i(u_i;s_0)}{ds} + \frac{(s-s_0)^2}{2}\frac{d^2\gamma_x^i(u_i;s_0)}{ds^2} + \frac{(s-s_0)^3}{6}\frac{d^3\gamma_x^i(u_i;s_0)}{ds^3}.$$

Le problème est maintenant de déterminer les dépendances en u_i selon l'intervalle considéré (ici, $[s_0, s_1]$. Pour chaque intervalle $[s_k, s_{k+1}]$, le calcul de la B-spline utilise quatre nœuds correspondant aux points de contrôle suivants : X_{k-1}, X_k, X_{k+1}, X_{k+2}. Par conséquent, selon l'intervalle, le point de contrôle déterminé par la méthode d'optimisation ($X_1 = \alpha(u_i)$) correspond, soit à X_{k+1} sur $[s_0, s_1]$, soit à X_k sur $[s_1, s_2]$. Ce point de contrôle n'est pas le seul impliquant u_i puisque, comme nous l'expliquons dans la partie 2.1.2, la méthodologie B-spline utilise des points de contrôle fantômes $X_{-1} = 2\alpha_{start_x}^i - (p_{mid_x}^i + u_i q_x^i)$ et $X_3 = 2\alpha_{end_x}^i - p_{mid_x} - u_i q_x$.

Pour $s \in [s_0, s_1]$, X_{k+1} est le point défini par la méthode d'optimisation. Les points de contrôle utilisés sont :

$$\begin{cases} X_{k-1} = X_{-1} = 2\alpha_{start_x}^i - (p_{mid_x}^i + u_i q_x^i) \\ X_k = X_0 = \alpha_{start_x}^i \\ X_{k+1} = X_1 = p_{mid_x}^i + u_i q_x^i \\ X_{k+2} = X_2 = \alpha_{end_x}^i \end{cases}$$

Ainsi, la trajectoire $\gamma_x^i(u;s)$ s'exprime comme suit pour tout $s \in [0,1]$:

$$\gamma_x^i(u_i;s) = \alpha^i_{start_x} + (p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x})s + (\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x})\frac{s^3}{6}.$$

Il en découle pour tout $s \in [0,1]$:

$$\frac{d\gamma_x^i(u_i;s)}{ds} = (p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x}) + (\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x})\frac{s^2}{2}.$$

Par conséquent, on a pour tout $s \in [0,1]$:

$$\begin{array}{ll}\Big(\frac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 & = (p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x})^2 + \frac{s^4}{4}(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x})^2 \\ & +s^2(p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x})(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}).\end{array}$$

Pour obtenir $\frac{\partial T^i}{\partial u_i}$, il est nécessaire de calculer l'intégrale $\int_0^2 \Big(\frac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 + \Big(\frac{d\gamma_y^i(u_i;s)}{ds}\Big)^2 ds$. Comme mentionné plus haut, seuls les termes impliquant γ_x^i sont détaillés. Ici, nous nous concentrons sur l'intervalle $s \in [0,1]$. On a :

$$\begin{array}{ll}\int_0^1 \Big(\frac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 ds = \int_0^1 & \Big[(p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x})^2 \\ & +\frac{s^4}{4}(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x})^2 \\ & +s^2(p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x}) \\ & (\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x})\Big] ds.\end{array}$$

Nous obtenons alors :

$$\begin{array}{ll}\int_0^1 \Big(\frac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 ds = & (p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x})^2 + \frac{1}{5}(\frac{\alpha^i_{end_x}}{2} - p^i_{mid_x} - u_i q^i_x + \frac{\alpha^i_{start_x}}{2})^2 \\ & +\frac{1}{3}(p^i_{mid_x} + u_i q^i_x - \alpha^i_{start_x})(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}).\end{array}$$

Ainsi, en dérivant par rapport à u_i, nous obtenons :

$$\frac{\partial}{\partial u_i}\int_0^1 \Big(\frac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 ds = q^i_x\Big[\frac{16}{15}(p^i_{mid_x} + u_i q^i_x) - \frac{6}{5}\alpha^i_{start_x} + \frac{2}{15}\alpha^i_{end_x}\Big].$$

Passons maintenant à l'intervalle $s \in [s_1, s_2]$ (le cas où $k = 1$). En utilisant le développement de Taylor (2.13) sur cet intervalle, l'expression suivante est obtenue :

$$\gamma_x^i(u_i;s) \;\; = \gamma_x^i(u_i;s_1) + (s-s_1)\tfrac{d\gamma_x^i(u_i;s_1)}{ds} + \tfrac{(s-s_1)^2}{2}\tfrac{d^2\gamma_x^i(u_i;s_1)}{ds} + \tfrac{(s-s_1)^3}{6}\tfrac{d^3\gamma_x^i(u_i;s_1)}{ds}.$$

En suivant le même raisonnement que précédemment, les quatre points de contrôle utilisés pour le calcul de la B-spline sur cet intervalle sont (le point défini par la méthode d'optimisation étant le point X_k) :

$$\begin{cases} X_{k-1} = X_0 = \alpha^i_{start_x} \\ X_k = X_1 = p^i_{mid_x} + u_i q^i_x \\ X_{k+1} = X_2 = \alpha^i_{end_x} \\ X_{k+2} = X_3 = 2\alpha^i_{end_x} - p^i_{mid_x} - u_i q^i_x. \end{cases}$$

L'expression de la trajectoire $\gamma_x^i(u_i;s)$ pour tout $s \in [1,2]$ devient donc :

$$\begin{array}{ll}\gamma_x^i(u_i;s) = & \frac{1}{6}(\alpha^i_{end_x} + 4p^i_{mid_x} + 4u_i q^i_x + \alpha^i_{start_x}) \\ & +\frac{(s-1)}{2}(\alpha^i_{end_x} - \alpha^i_{start_x}) \\ & +\frac{(s-1)^2}{2}(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}) \\ & +\frac{(s-1)^3}{6}(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x}).\end{array}$$

En dérivant par rapport à s, nous obtenons pour tout $s \in [1,2]$:

$$\begin{aligned}\tfrac{d\gamma_x^i(u_i;s)}{ds} = \; & \tfrac{1}{2}(\alpha^i_{end_x} - \alpha^i_{start_x}) + (s-1)(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}) \\ & + \tfrac{(s-1)^2}{2}(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x}).\end{aligned}$$

Par conséquent, on a pour tout $s \in [1,2]$:

$$\begin{aligned}\Big(\tfrac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 = \; & \tfrac{1}{4}(\alpha^i_{end_x} - \alpha^i_{start_x})^2 + (s-1)^2(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x})^2 \\ & + \tfrac{(s-1)^4}{4}(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x})^2 \\ & + (s-1)(\alpha^i_{end_x} - \alpha^i_{start_x})(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}) \\ & + \tfrac{(s-1)^2}{2}(\alpha^i_{end_x} - \alpha^i_{start_x})(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x}) \\ & + (s-1)^3(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x})(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}).\end{aligned}$$

Nous intégrons alors sur $[1,2]$ pour obtenir :

$$\begin{aligned}\int_1^2 \Big(\frac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 ds = \; & \tfrac{1}{4}(\alpha^i_{end_x} - \alpha^i_{start_x})^2 + \tfrac{1}{3}(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x})^2 \\ & + \tfrac{1}{20}(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x})^2 \\ & + \tfrac{1}{2}(\alpha^i_{end_x} - \alpha^i_{start_x})(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}) \\ & + \tfrac{1}{6}(\alpha^i_{end_x} - \alpha^i_{start_x})(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x}) \\ & + \tfrac{1}{4}(2p^i_{mid_x} + 2u_i q^i_x - \alpha^i_{end_x} - \alpha^i_{start_x})(\alpha^i_{end_x} - 2p^i_{mid_x} - 2u_i q^i_x + \alpha^i_{start_x}).\end{aligned}$$

Puis en dérivant par rapport u_i, il vient :

$$\frac{\partial}{\partial u_i}\int_1^2 \Big(\frac{d\gamma_x^i(u_i;s)}{ds}\Big)^2 ds = q^i_x\Big[\frac{16}{15}(p^i_{mid_x} + u_i q^i_x) + \frac{2}{15}\alpha^i_{start_x} - \frac{6}{5}\alpha^i_{end_x}\Big].$$

Comme mentionné précédemment, nous en déduisons directement le terme impliquant γ_y^i. On obtient ainsi la formule exacte de la dérivée de notre fonction-objectif par rapport à u_i, pour $i = 1, 2, \ldots, N$:

$$\begin{aligned}\tfrac{\partial}{\partial u_i} f^i(u_i) \; & = q^i_x\Big[\tfrac{16}{15}(p^i_{mid_x} + u_i q^i_x) - \tfrac{6}{5}\alpha^i_{start_x} + \tfrac{2}{15}\alpha^i_{end_x}\Big] \\ & + q^i_x\Big[\tfrac{16}{15}(p^i_{mid_x} + u_i q^i_x) + \tfrac{2}{15}\alpha^i_{start_x} - \tfrac{6}{5}\alpha^i_{end_x}\Big] \\ & + q^i_y\Big[\tfrac{16}{15}(p^i_{mid_y} + u_i q^i_y) - \tfrac{6}{5}Y_0 + \tfrac{2}{15}Y_2\Big] \\ & + q^i_y\Big[\tfrac{16}{15}(p^i_{mid_y} + u_i q^i_y) + \tfrac{2}{15}Y_0 - \tfrac{6}{5}Y_2\Big] \\ \\ & = \tfrac{32}{15}q^i_x(p^i_{mid_x} + u_i q^i_x) + \tfrac{32}{15}q^i_y(p^i_{mid_y} + u_i q^i_y) \\ & - \tfrac{16}{15}q^i_x(\alpha^i_{end_x} + \alpha^i_{start_x}) - \tfrac{16}{15}q^i_y(Y_2 + Y_0).\end{aligned}$$

Étant donné que, par construction, $p^i_{mid_x} = \frac{\alpha^i_{end_x} + \alpha^i_{start_x}}{2}$ et $p^i_{mid_y} = \frac{Y_2 + Y_0}{2}$, la i-ième composante du vecteur gradient $\nabla_i f(u)$ s'écrit :

$$\frac{\partial f(u)}{\partial u_i} = \frac{\frac{32}{15}u_i\Big((q^i_x)^2 + \Big(q^i_y\Big)^2\Big)}{N\Big(T^i(1) - T^i(0)\Big)}.$$

Ainsi, on obtient l'expression du gradient (2.12). □

2.2.3 Les contraintes et leurs dérivées

Les contraintes semi-infinies ne peuvent être évaluées directement numériquement (puisqu'il y en a une infinité). Par conséquent, pour l'expression des contraintes, nous nous sommes inspirés de [7] et [46] qui, comme ici, intègrent la contrainte sur le paramètre semi-infini pour la transformer en une contrainte unique. On se concentre sur la fonction contrainte $C^{ij}(u)$:

$$C^{ij}(u) = \int_0^{t^{ij}_{min}} \Big(\max\{\tau^2 - c^{ij}(u;t); 0\} \Big)^2 dt = 0. \tag{2.14}$$

La contrainte n'est vérifiée que sur $[0, t^{ij}_{min}]$ car au-delà de t^{ij}_{min}, l'avion « arrivé » en premier n'étant plus considéré, la contrainte n'a plus lieu d'être vérifiée.

Ce faisant, nous obtenons une contrainte égalité unique pour chaque paire d'avions (i, j). Si cette contrainte n'est pas violée, la distance de séparation sera respectée en tout temps de l'intervalle d'intérêt $[0, t^{ij}_{min}]$. En effet, si les avions i et j ne respectent pas la distance de séparation en un certain temps t, alors, par continuité de c^{ij}, on a $\tau^2 - c^{ij}(u;t) > 0$. L'intégrale devient alors non-nulle, et la contrainte (2.14) est violée.

C'est parce que l'expression mathématique max n'est pas dérivable que nous avons plutôt utilisé cette modélisation mettant en jeu le carré de la violation de la contrainte de séparation. Nous pouvons maintenant obtenir la dérivée de $C^{ij}(u)$ de la manière suivante :

$$\begin{aligned} \frac{\partial C^{ij}(u)}{\partial u_i} &= \tfrac{\partial}{\partial u_i} \left[\int_0^{t^{ij}_{min}} \Big(\max\{\tau^2 - c^{ij}(u;t); 0\} \Big)^2 \quad dt \right] \\ &= \int_0^{t^{ij}_{min}} 2 \max\{\tau^2 - c^{ij}(u;t); 0\} \frac{\partial}{\partial u_i} \Big(\tau^2 - c^{ij}(u;t) \Big) \quad dt. \end{aligned}$$

Or, nous avons $\frac{\partial}{\partial u^i}\tau^2 = 0$ (τ étant une constante). Ainsi, nous pouvons écrire :

$$\boxed{\frac{\partial C^{ij}(u)}{\partial u_i} = - \int_0^{t^{ij}_{min}} 2 \max\{\tau^2 - c^{ij}(u;t); 0\} \frac{\partial}{\partial u_i} c^{ij}(u;t) \quad dt}. \tag{2.15}$$

Nous détaillons ci-dessous uniquement l'expression de $\frac{\partial}{\partial u_i} c^{ij}(u;t)$ pour chaque modélisation de trajectoire.

Cas de la modélisation linéaire par morceaux de la trajectoire

Pour la modélisation linéaire par morceaux, nous disposons de trajectoires, $\gamma^i(u_i;t)$, explicitement paramétrées par le temps. Nous avons donc l'expression suivante pour $\dfrac{\partial}{\partial u_i} c^{ij}(u;t)$:

$$\frac{\partial}{\partial u_l} c^{ij}(u;t) = \begin{cases} 2\Big(\gamma^i(u_i;t) - \gamma^j(u_j;t)\Big) \frac{d}{du_i}\gamma^i(u_i;t) & \text{si } l = i \\ -2\Big(\gamma^i(u_i;t) - \gamma^j(u_j;t)\Big) \frac{d}{du_j}\gamma^j(u_j;t) & \text{si } l = j \\ 0 & \text{sinon} \end{cases} \tag{2.16}$$

Il nous reste donc plus qu'à déterminer l'expression de $\dfrac{\partial}{\partial u_l}\gamma^i(u_i;t)$, pour tout $l = 1, 2, \ldots, N$.

Avant toute chose, pour simplifier les expressions dans les calculs qui vont suivre, nous définissons : $d_i(u_i) = \alpha^i(u_i) - \alpha^i_{start}$ et $D_i(u_i) = \alpha^i_{end} - \alpha^i(u_i)$.

Proposition 2. *La dérivée* $\frac{\partial}{\partial u_i}\gamma^i(u_i;t)$ *a pour expression analytique pour tout* $t \in [0, t^i_{end}]$ *et pour* $i = 1, 2, \ldots, N$ *:*

$$\frac{\partial}{\partial u_i}\gamma^i(u;t) = \begin{cases} tv_i\left(\frac{q_i}{\|d_i(u_i)\|} - \frac{(q_i\cdot d_i(u_i))d_i(u_i)}{\|d_i(u_i)\|^3}\right) & Si\ t \in [0, t^i_{mid}] \\ q_i + (tv_i - \|d_i(u_i)\|) & \\ \left(-\frac{q_i}{\|D_i(u_i)\|} + \frac{(q_i\cdot D_i(u_i))D_i(u_i)}{\|D_i(u_i)\|^3}\right) & \\ -\frac{(q_i\cdot d_i(u_i))D_i(u_i)}{\|d_i(u_i)\|\|D_i(u_i)\|} & Si\ t \in [t^i_{mid}, t^i_{end}] \end{cases}. \tag{2.17}$$

Pour $l \neq i$*, cette dérivée est évidemment nulle puisque la trajectoire* $\gamma^i(u_i;t)$ *dépend uniquement de* u_i*. À partir de cette information et des équations (2.15), (2.16) et (2.17), nous obtenons une forme explicite des dérivées des fonctions contraintes pour la modélisation linéaire par morceaux pour toute paire d'avion* (i, j)*.*

Démonstration. La dérivée de $\gamma^i(u_i;t)$ s'écrit en utilisant la dérivée de la norme de d_i (son argument, $d_i(u_i)$, étant ici toujours non-nul). Nous la présentons ici :

$$\frac{\partial}{\partial u_i}\gamma^i(u_i;t) = tv_i\frac{q_i\frac{\partial\alpha^i(u_i)}{\partial u_i}\|d_i(u_i)\| - \frac{\partial}{\partial u_i}\left(\|d_i(u_i)\|\right)d_i(u_i)}{\|d_i(u_i)\|}.$$

Or, nous avons :

$$\frac{\partial\alpha^i(u_i)}{\partial u_i} = q_i, \tag{2.18}$$

et :

$$\frac{\partial}{\partial u_i}\left(\|d_i(u_i)\|\right) = \frac{q_i\cdot d_i(u_i)}{\|d_i(u_i)\|}, \tag{2.19}$$

où la notation « $\cdot$ » représente le produit scalaire usuel.

Nous obtenons donc pour $t \in [0, t^i_{mid}]$ l'expression suivante pour la dérivée de $\gamma^i(u_i;t)$:

$$\frac{\partial}{\partial u_i}\gamma^i(u_i;t) = \frac{tv_i}{\|d_i(u_i)\|}\left(q_i - \frac{(q_i\cdot d_i(u_i))d_i(u_i)}{\|d_i(u_i)\|^2}\right) \tag{2.20}$$

La trajectoire $\gamma^i(u_i;t)$ ne dépendant que de u_i, les dérivées $\frac{\partial}{\partial u_l}\gamma^i(u_i;t)$ sont nulles pour $l \neq i$.

Passons maintenant à l'expression de $\frac{\partial}{\partial u_l}\gamma^i(u_i;t)$ sur $t \in [t^i_{mid}, t^i_{end}]$. Sur cet intervalle, la trajectoire $\gamma^i(u_i;t)$ s'exprime :

$$\gamma^i(u_i;t) = \alpha^i(u_i) + (tv_i - t^i_{mid}v_i)\frac{D_i(u_i)}{\|D_i(u_i)\|}$$

En dérivant par rapport à u_i, l'expression suivante est obtenue pour $t \in [t_{mid}, t_{end}]$:

$$\frac{\partial}{\partial u_i}\gamma^i(u_i;t) = \frac{\partial\alpha^i(u_i)}{\partial u_i} + tv_i\frac{\partial}{\partial u_i}\left(\frac{D_i(u_i)}{\|D_i(u_i)\|}\right) - \frac{\partial}{\partial u_i}\left(t^i_{mid}v_i\frac{D_i(u_i)}{\|D_i(u_i)\|}\right) \tag{2.21}$$

Le premier terme est obtenu directement par (2.18) et le second terme découle, comme précédemment, la dérivée de la norme (ici encore, on a $D_i(u_i) \neq 0$) :

$$\begin{aligned}\frac{\partial}{\partial u_i}\left(\frac{D_i(u_i)}{\|D_i(u_i)\|}\right) &= \frac{\frac{\partial}{\partial u_i}\left(D_i(u_i)\right)\|D_i(u_i)\| - \frac{\partial}{\partial u_i}\left(\|D_i(u_i)\|\right)}{\|D_i(u_i)\|^2} \\ &= -\frac{q_i}{\|D_i(u_i)\|} + \frac{(q_i\cdot D_i(u_i))\,D_i(u_i)}{\|D_i(u_i)\|^3}\end{aligned}$$

Pour le troisième terme de (2.21), nous utilisons également la dérivée de la norme, mais aussi le fait que $t^i_{mid} v_i = \|d_i(u_i)\|$ (cf partie 2.1.1) :

$$\begin{aligned}\frac{\partial}{\partial u_i}\left(t^i_{mid} v_i \frac{D_i(u_i)}{\|D_i(u_i)\|}\right) &= \frac{\partial}{\partial u_i}\left(\|d_i(u_i)\|\right)\frac{D_i(u_i)}{\|D_i(u_i)\|} \\ &\quad + \|d_i(u_i)\|\frac{\partial}{\partial u_i}\left(\frac{D_i(u_i)}{\|D_i(u_i)\|}\right).\end{aligned} \tag{2.22}$$

Nous avons déjà calculé le terme $\frac{\partial}{\partial u_i}\left(\|d_i(u_i)\|\right)$ dans (2.19). Calculons la dérivée qui apparaît dans le deuxième terme de (2.22) :

$$\frac{\partial}{\partial u_i}\left(\frac{D_i(u_i)}{\|D_i(u_i)\|}\right) = -\frac{q_i}{\|D_i(u_i)\|} + \frac{(q_i \cdot D_i(u_i))D_i(u_i)}{\|D_i(u_i)\|^3}$$

Nous obtenons donc pour le troisième terme du membre de droite de (2.21) :

$$\begin{aligned}\frac{\partial}{\partial u_i}\left(t^i_{mid} v_i \frac{D_i(u_i)}{\|D_i(u_i)\|}\right) &= \frac{(q_i \cdot d_i(u_i))D_i(u_i)}{\|d_i(u_i)\|\|D_i(u_i)\|} \\ &\quad + \|d_i(u_i)\|\left(-\frac{q_i}{\|D_i(u_i)\|} + \frac{(q_i \cdot D_i(u_i))D_i(u_i)}{\|D_i(u_i)\|^3}\right)\end{aligned}$$

Ainsi, suivant (2.21), la dérivée de $\gamma^i(u_i;t)$, sur $t \in [t^i_{mid}, t^i_{end}]$, s'écrit de la manière suivante :

$$\begin{aligned}\frac{\partial}{\partial u_i}\gamma^i(u;t) &= q_i + (tv_i - \|d_i(u_i)\|) \\ &\quad \left(-\frac{q_i}{\|D_i(u_i)\|} + \frac{(q_i \cdot D_i(u_i))D_i(u_i)}{\|D_i(u_i)\|^3}\right) - \frac{(q_i \cdot d_i(u_i))D_i(u_i)}{\|d_i(u_i)\|\|D_i(u_i)\|}\end{aligned} \tag{2.23}$$

En combinant les résultats (2.20) et (2.23) obtenus sur chacun des deux sous-intervalles, on obtient le résultat. □

L'expression du terme $\frac{\partial}{\partial u_l}c^{ij}(u;t)$ pour $l = 1, 2, \ldots, N$ peut alors s'écrire comme suit pour tout $i = 1, 2, \ldots, N$:

$$\boxed{\frac{\partial}{\partial u_i}c^{ij}(u;t) = \begin{cases} 2\left(\gamma^i(u_i;t) - \gamma^j(u_j;t)\right)\left(tv_i\left(\frac{q_i}{\|d_i(u_i)\|} - \frac{(q_i \cdot d_i(u_i))d_i(u_i)}{\|d_i(u_i)\|^3}\right)\right) & \text{Si } t \in [0, t^i_{mid}] \\ -2\left(\gamma^i(u_i;t) - \gamma^j(u_j;t)\right)\left[q_i + (tv_i - \|d_i(u_i)\|)\right. & \\ \left.\left(-\frac{q_i}{\|D_i(u_i)\|} + \frac{(q_i \cdot D_i(u_i))D_i(u_i)}{\|D_i(u_i)\|^3}\right) - \frac{(q_i \cdot d_i(u_i))D_i(u_i)}{\|d_i(u_i)\|\|D_i(u_i)\|}\right] & \text{Si } t \in [t^i_{mid}, t^i_{end}] \end{cases}.} \tag{2.24}$$

Par conséquent, en intégrant (2.24) sur $[0, t^{ij}_{min}]$ on obtient l'expression de $\frac{\partial}{\partial u_l}C^{ij}(u)$ pour tout $l = 1, 2, \ldots, N$.

Cas de la modélisation B-spline de la trajectoire

Rappelons l'expression du terme à dériver pour obtenir la dérivée de nos contraintes :

$$\frac{\partial}{\partial u_l}c^{ij}(u;t) = \begin{cases} 2\Big(\gamma^i(u_i;t) - \gamma^j(u_j;t)\Big)\frac{d}{du_i}\gamma^i(u_i;t) & \text{si } l = i \\ -2\Big(\gamma^i(u_i;t) - \gamma^j(u_j;t)\Big)\frac{d}{du_j}\gamma^j(u_j;t) & \text{si } l = j \\ 0 & \text{sinon} \end{cases}$$

Or, la trajectoire de l'avion i, γ^i est une B-spline dépendant de la i-ième composante de u et paramétrée par son paramètre naturel s_i. Ainsi, $\gamma_i : U \times [0,2] \to \mathbb{R}^2$, où $U = [-1,1]$ est le domaine admissible de nos variables d'optimisation u_i. Il convient donc de trouver la relation liant le paramètre naturel s_i et le temps t. Cette relation est en fait l'abscisse curviligne de la courbe :

$$t = \theta_i(u_i; s_i) = t_i + \frac{1}{v_i} \int_0^{s_i} \left\| \frac{\partial \gamma^i}{\partial \xi}(u_i; \xi) \right\| d\xi,$$

où t_i est le temps de départ de la trajectoire i et v_i la vitesse constante de l'avion i. Pour tout $u_i \in U$, la relation $s_i \mapsto \theta_i(u_i; s_i)$ est un difféomorphisme (i.e. une bijection différentiable dont la bijection est différentiable) du fait des propriétés usuelles des B-splines et son inverse est notée $\theta_i^{-1}(u_i; t)$.

Proposition 3. *Pour tout $(u_i, t) \in U \times \theta(U; [0,2])$, on a :*

$$\boxed{\frac{d}{du_i}\gamma^i\left(u_i; \theta_i^{-1}(u_i; t)\right) = -\frac{\frac{\partial \gamma^i}{\partial s_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)}{\left\|\frac{\partial \gamma^i}{\partial s_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)\right\|} + \frac{\partial \gamma^i}{\partial u_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)}. \tag{2.25}$$

Démonstration. Tout d'abord, pour tout couple (u_i, t) avec $t \in \{\theta_i(\nu; \sigma), \nu \in [-1; 1] \text{ et } \sigma \in [0,2]\}$ et $u_i \in U$:

$$\theta_i\left(u_i; \theta_i^{-1}(u_i, t)\right) = t. \tag{2.26}$$

En utilisant la dérivée totale de cette expression pour un instant t fixé et le théorème de dérivation des fonctions composées, on obtient :

$$\frac{d\theta_i}{du_i} = \frac{\partial \theta_i}{\partial s_i}\left(u_i; \theta_i^{-1}(u_i; t)\right) \frac{\partial \theta_i^{-1}}{\partial u_i}(u_i; t) + \frac{\partial \theta_i}{\partial u_i}\left(u_i; \theta_i^{-1}(u_i; t)\right) = 0.$$

Ainsi, on déduit pour un t donné :

$$\frac{\partial \theta_i^{-1}}{\partial u_i}(u_i; t) = -\frac{\frac{\partial \theta_i}{\partial u_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)}{\frac{\partial \theta_i}{\partial s_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)} = -\frac{\frac{\partial \theta_i}{\partial u_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)}{\left\|\frac{\partial \gamma^i}{\partial s_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)\right\|}. \tag{2.27}$$

Or, en appliquant à nouveau le théorème de dérivation des fonctions composées à $\gamma^i\left(u_i; \theta_i^{-1}(u_i; t)\right)$, on obtient :

$$\frac{d}{du_i}\gamma^i\left(u_i; \theta_i^{-1}(u_i; t)\right) = \frac{\partial \gamma^i}{\partial s_i}\left(u_i; \theta_i^{-1}(u_i; t)\right) \frac{\partial \theta_i^{-1}}{\partial u_i}(u_i; t) + \frac{\partial \gamma^i}{\partial u_i}\left(u_i; \theta_i^{-1}(u_i; t)\right) \tag{2.28}$$

En utilisant (2.28) et (2.27), on obtient (2.25). □

L'expression (2.26) étant valable pour tout instant t, l'expression du terme $\frac{\partial}{\partial u_l} c^{ij}(u; t)$ pour $l = 1, 2, \ldots, N$ peut alors s'écrire comme suit :

$$\boxed{\frac{\partial}{\partial u_l} c^{ij}(u; t) = \begin{cases} 2\left(\gamma^i\left(u_i; \theta_i^{-1}(u_i; t)\right) - \gamma^j\left(u_j; \theta_j^{-1}(u_j; t)\right)\right) \cdot \\ \left(\frac{\partial \gamma^i}{\partial s_i}\left(u_i; \theta_i^{-1}(u_i; t)\right) \frac{\partial \theta_i^{-1}}{\partial u_i}(u_i; t) + \frac{\partial \gamma^i}{\partial u_i}\left(u_i; \theta_i^{-1}(u_i; t)\right)\right) & \text{si } l = i \\ \\ -2\left(\gamma^i\left(u_i; \theta_i^{-1}(u_i; t)\right) - \gamma^j\left(u_j; \theta_j^{-1}(u_j; t)\right)\right) \cdot \\ \left(\frac{\partial \gamma^j}{\partial s_j}\left(u_j; \theta_j^{-1}(u_j; t)\right) \frac{\partial \theta_j^{-1}}{\partial u_j}(u_j; t) + \frac{\partial \gamma^j}{\partial u_j}\left(u_j; \theta_i^{-1}(u_j; t)\right)\right) & \text{si } l = j \\ \\ 0 & \text{sinon.} \end{cases}} \tag{2.29}$$

Pour obtenir les meilleurs résultats possibles, les méthodes classiques d'optimisation numérique requièrent que nous programmions les dérivées de la fonction-objectif de même que celles des fonctions contraintes. Nous devons donc fournir une sous-routine informatique qui, étant donné un vecteur courant u, retourne les gradients des fonctions C^{ij} évalués en u. Précisons tout d'abord que le fait qu'un ré-échantillonnage est nécessaire pour le calcul de l'intégrale (2.15). En effet, aucune relation explicite n'existant entre le paramètre naturel de la spline et le temps (relation traduite par l'expression $\theta_i^{-1}(u_i;t)$, définie comme étant l'inverse de la fonction $\theta_i^{-1}(u_i;s_i) = t_i + \frac{1}{v_i}\int_0^{s_i} \left\| \frac{\partial \gamma^i}{\partial \xi}(u_i;\xi) \right\| d\xi$). C'est pourquoi d'ailleurs, nous n'avons pu avoir recours à la différentiation automatique (qui, étant donnée une sous-routine informatique évaluant une fonction $f(x)$ à partir d'une entrée x, retourne une autre sous-routine évaluant non seulement $f(x)$ à partir d'une entrée x mais évalue aussi le gradient, $\nabla f(x)$, de f). Nous avons donc programmé les gradients des fonctions contraintes C^{ij} en construisant numériquement cette relation entre le paramètre naturel de la spline et le temps.

Afin de vérifier la qualité du gradient utilisé comme référence pour notre calcul d'erreur, nous avons utilisé des schémas de différences finies d'ordre supérieur. Cette étude comparative, avec différents pas de discrétisation dans les schémas différences finies, a mis en évidence un problème d'ordre numérique dans notre mise en œuvre informatique des gradients des C^{ij} issues de (2.29). Après avoir méticuleusement vérifié la fiabilité de notre expression analytique, nous avons cherché des possibles sources d'erreurs numériques. Les composantes de notre gradient, bien que présentant une allure très similaire aux composantes correspondantes calculées à l'aide de schémas de différences finies classiques, accusent des erreurs d'amplitudes significatives. Or, pour effectuer le calcul de chacune des composantes du gradient des fonctions contraintes, deux intégrations sont nécessaires. Ces deux intégrations, combinées au ré-échantillonnage en temps des trajectoires, semblent être la cause de nos problèmes numériques. Nos travaux de recherche actuels portent sur l'élimination de ces intégrations par l'utilisation des intégrales elliptiques, qui nous permettrait d'obtenir une forme explicite de $\theta_i^{-1}(u_i;t)$, rendant ainsi possible un calcul formel du gradient des contraintes (notamment des intégrales, supprimant de fait l'intégration numérique et le ré-échantillonnage qui ont été identifiés comme la cause des erreurs numériques).

Nous nous contenterons donc, dans cette thèse, d'exploiter le gradient explicite de la fonction-objectif, mais pas celui des fonctions contraintes.

Chapitre 3

Méthodes d'optimisation utilisées

Nous présentons dans ce chapitre les différentes méthodes d'optimisation que nous avons appliquées à notre problème. Nous commençons par une présentations des algorithmes génétiques pour ensuite décrire le fonctionnement des deux méthodes d'optimisation locale que nous avons testées.

3.1 Utilisation des algorithmes génétiques

Dans cette partie, nous présentons les algorithmes génétiques en détaillant la manière dont nous avons adapté leurs composantes au problème de résolution de conflits et en expliquant nos choix pour leur mise en œuvre. Les algorithmes génétiques bénéficient d'une expertise et d'un savoir-faire de la part de l'équipe d'encadrement de cette thèse ainsi que du laboratoire d'accueil dans cette méthode, et notamment dans l'application des algorithmes génétiques aux particularité des problèmes de l'ATM.

L'objectif de la thèse étant de d'évaluer la capacité des méthodes locale à résoudre ces problèmes d'optimisation globale NP-difficile, nous utilisons les AG afin d'obtenir des résultats de référence avec une méthode de recherche stochastique adaptée aux problèmes et susceptibles de trouver des solutions réalisables à valeurs proche de la valeur globalement optimale (quitte à les laisser tourner pendant un temps qui ne serait pas acceptable dans un cadre opérationnel).

3.1.1 Généralités sur les algorithmes génétiques

Un algorithme génétique (AG) est une méthode d'optimisation mathématique basée sur les principes de l'évolution naturelle. Cette méthode est apparue suite aux travaux de biologistes américains qui ont simulés le fonctionnement de structures biologiques sur ordinateur dans les années cinquante. John Holland [22] a ensuite développé les principes fondamentaux de leur application à l'optimisation. Cependant, l'utilisation numérique efficace des AG a dû attendre l'amélioration des ordinateurs, qui n'avaient pas la puissance de calcul nécessaire à l'époque. Ainsi, l'ouvrage de Goldberg [19] marquera un tournant en présentant l'utilisation pratique des AG appliqués à des problèmes concrets.

Les AG reproduisent le processus de sélection naturelle en milieu hostile pour trouver la solution optimale à un problème donné. Le vocabulaire utilisé dans ces méthodes est le même que dans la génétique naturelle. Ainsi, des concepts identiques, comme le croisement, la mutation ou la sélection, sont utilisés. Le fonctionnement général d'un algorithme génétique consiste

à faire évoluer une population de solutions (d'individus) du problème considéré, de génération en génération, selon les principes de l'évolution naturelle.

Le principe algorithmique d'un AG est présenté dans la figure 3.1. Ce schéma représente l'évolution d'une population de la génération k à la génération $k+1$ L'AG sélectionne d'abord

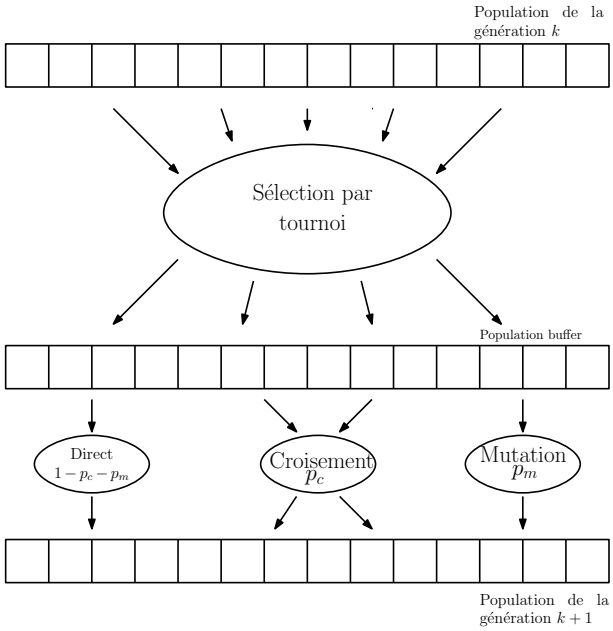

FIGURE 3.1 – Processus d'évolution de la génération k à la génération $k+1$ durant l'AG

plusieurs individus de la génération k (nous verrons comment par la suite) pour constituer une population dite *buffer*. L'AG applique ensuite différents opérateurs génétiques à chaque individu pour diversifier la population dans l'espace d'état : la mutation et le croisement qui sont appliqués respectivement selon des probabilités p_m et p_c fixées à l'avance par l'utilisateur. Les individus ainsi obtenus sont placés dans la génération $k+1$. Ce processus est répété jusqu'à ce que la condition d'arrêt choisie par l'utilisateur soit atteinte.
Il existe donc 6 éléments nécessaires au fonctionnement d'un AG :

- Tout d'abord, il faut définir l'encodage d'une solution de l'espace d'état. En effet, il est nécessaire de définir la structure de données qui contient toutes les informations nécessaires à la définition d'une solution. Cette structure de données est appelée *chromosome*.
- L'utilisateur doit fixer *a priori* les paramètres de dimensionnement de l'algorithme, notamment la taille de la population et le nombre de générations (qui est très souvent utilisé comme condition d'arrêt de l'algorithme).
- On engendre ensuite une population initiale de chromosomes. Cette génération peut se faire aléatoirement si aucune information n'est disponible quant à la position de la solution optimale. En revanche, on connaît un sous-ensemble du domaine réalisable dans lequel se trouve la solution, générer la population initiale dans celui-ci améliore la convergence de l'algorithme. De même, toute connaissance *a priori* du problème peut être exploitée ici. L'utilisation d'heuristiques simples peut également être envisagée.
- Pour pouvoir apprécier la qualité d'un individu, un critère est défini de manière à traduire quantitativement la capacité de l'individu à répondre au problème posé. Ce critère est appelé la *fitness*. La fitness peut par exemple correspondre à la fonction-objectif à maximiser ou à une combinaison de la fonction-objectif et de la violation des contraintes.
- La manière dont les individus sont sélectionnés dans la génération précédente est égale-

ment un point important. Selon la méthode utilisée, la pression sélective est plus ou moins forte. Une pression sélective forte favorise l'exploitation des bonnes solutions tandis qu'une pression sélective plus faible favorise l'exploration de l'espace d'état.
– Les opérateurs génétiques doivent être définis de manière à avoir du sens par rapport au problème considéré. Les deux opérateurs les plus couramment utilisés sont le *croisement*, qui a pour but de favoriser l'exploitation des bons individus déjà présents en « brassant » les *gènes* (éléments du chromosome) de la population, et la *mutation* qui permet d'explorer l'espace d'état en créant de nouveaux gènes qui n'ont pas encore été rencontrés durant le processus d'optimisation.

Nous présentons un survol de la littérature pour chacun de ces éléments pour ensuite exposer notre choix.

3.1.2 Codage du chromosome

Initialement, dans les premiers AG, le chromosome représentant un individu (un point de l'espace d'état) était constitué d'une chaîne de bits. Celui-ci devait contenir toutes les informations nécessaires pour décrire le point de l'espace d'état concerné. Ce codage a évolué pour aboutir à une très grande diversité permettant, par exemple, d'utiliser directement les variables réelles d'un problème d'optimisation dont le domaine de recherche est $\mathbb{R}^N$ [47]. La seule condition nécessaire du codage est qu'un chromosome détermine complètement une solution et qu'il puisse être manipulé par les opérateurs de croisement et de mutation.

Nous avons choisi d'utiliser directement les variables d'optimisation u_i (cf partie 2.2) en tant que gènes de notre chromosome : le i-ème gène d'un chromosome contient la valeur u_i. Ainsi, la taille d'un chromosome est directement lié au nombre d'avions impliqués dans le conflit. Par exemple, pour un conflit à $N = 6$ avions, le chromosome est représenté dans la figure 3.2. Chaque gène de notre chromosome correspond au placement du point de contrôle

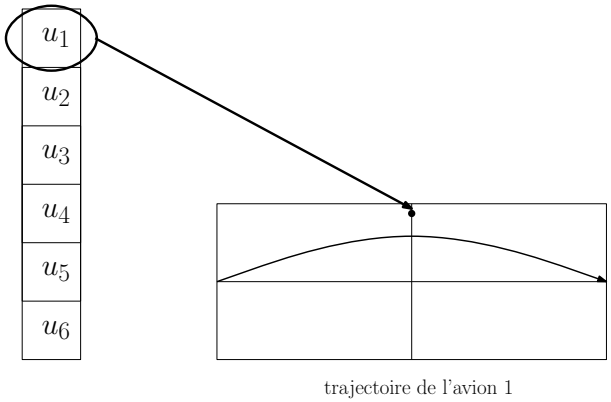

FIGURE 3.2 – Encodage d'un chromosome pour une situation à $N = 6$ avions

(ou point tournant, en modélisation linéaire par morceaux) d'un avion et définit entièrement la trajectoire de celui-ci.

3.1.3 Génération de la population initiale

Pour générer la population initiale, nous avons, comme évoqué plus haut, deux possibilités. Dans le cas où aucune information sur la position de la solution n'est disponible, le but est de recouvrir au mieux l'espace d'état. Une génération aléatoire est donc engendrée par des

tirages uniformes sur chaque gène du chromosome. Dans le cas où l'utilisateur connaîtrait un sous-domaine de l'espace d'état où la solution se trouve, la population initiale est générée dans ce sous-espace.

Dans le cas d'une génération engendrée de façon aléatoire, il existe deux éventualités. Soit il est possible de savoir à l'avance si un point respecte les contraintes du problème d'optimisation, la population est alors générée aléatoirement dans le domaine admissible. Soit il n'est pas possible de le savoir, le respect des contraintes sera alors assuré via l'ajout d'une pénalisation dans la fonction-objectif et la population est uniformément distribuée sur tout l'espace d'état.

Dans notre cas, le problème étant un problème d'optimisation NP-difficile, il n'est pas facile de savoir dans quel sous-domaine se trouve une solution. Il est également difficile de savoir à l'avance si un individu donné respecte les contraintes. Nous avons donc utilisé une pénalisation de la fonction-objectif (comme nous le verrons dans la partie suivante) et nous avons généré aléatoirement la population initiale en utilisant un tirage uniforme pour chaque composante.

3.1.4 Calcul de la fitness

Nous détaillons ici les deux méthodes d'évaluation de la fitness que nous avons utilisées avec les algorithmes génétiques. La première utilise une grille de détection comme cela se fait couramment dans la littérature traitant de la résolution automatique de conflits (cf partie 1.2.1) alors que la deuxième se base sur la formulation semi-infinie détaillée dans le chapitre 2.2.

Évaluation des contraintes à l'aide d'une grille de détection

Pour commencer, et afin de déterminer le nombre de conflits de manière rapide, nous avons utilisé une discrétisation de l'espace aérien à l'aide d'une grille de détection présentée dans la figure 3.3 pour tester rapidement notre méthodologie AG utilisant une modélisation de trajectoire par B-spline. Cette grille est composée de carrés de 5Nm de côté (la distance de séparation standard horizontale).

L'évaluation de la fitness d'une solution donnée se fait comme suit. Après avoir calculé l'ensemble des N trajectoires, la détection de conflits s'opère en deux phases. Dans un premier temps, l'algorithme parcourt chacune des trajectoires dans la grille et stocke dans chaque case traversée de la grille (en gris sur la figure 3.3) le temps d'entrée et de sortie de la case ainsi que l'indice de l'avion considéré. Après avoir parcouru ainsi toutes les trajectoires, l'algorithme passe en revue toutes les cases contenant au moins un avion. Pour chacune d'entre elles, on vérifie s'il y a un autre avion dans cette case et dans les huit cases voisines. Si c'est le cas, les temps d'entrée et de sortie des deux avions sont comparés et, si un chevauchement est détecté, alors le nombre de conflits est incrémenté de 1. Ainsi, après avoir parcouru toutes les cases, tous les conflits présents sont détectés. Cette méthode présente l'avantage d'être rapide du point de vue du temps de calcul. Cependant, elle ne fournit qu'un comptage discret du *nombre* de conflits (un conflit plus long ne sera pas plus pénalisant). De plus, ce nombre est surestimé. En effet, les cases étant de 5Nm de côté, deux avions distants de plus de 5Nm peuvent être considérés en conflit (voir figure 3.4). Cette surestimation mène donc en quelque sorte a un problème aux contraintes biaisées. C'est pour cela que nous avons décidé, après avoir effectué des tests numériques préliminaires encourageants avec cette méthode de détection par grille, d'utiliser une formulation exacte que nous présentons dans la sous-section 2.2.1.

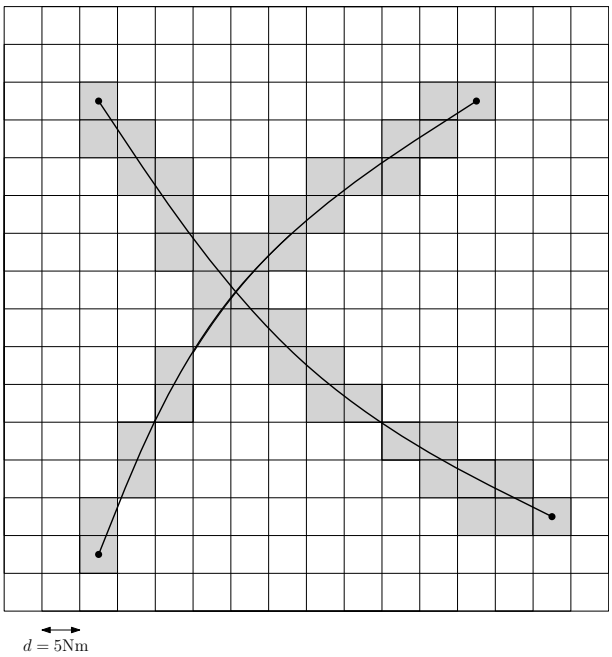

FIGURE 3.3 – Discrétisation de l'espace pour une situation à 2 avions

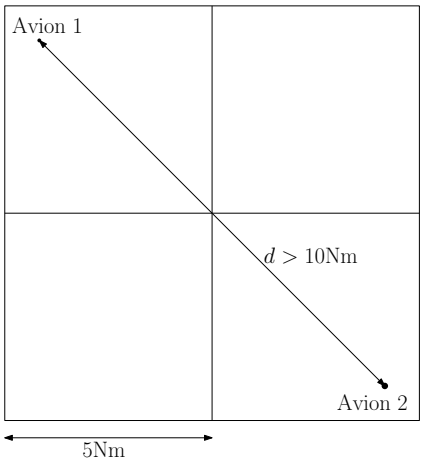

FIGURE 3.4 – Deux avions considérés à tort en conflit

Évaluation des contraintes à l'aide de la formulation SIP

Dans un souci d'homogénéité entre les méthodes, nous avons ensuite employé la fonction-objectif et les fonctions contraintes utilisées par les méthodes d'optimisation locale dans le calcul de la fitness. Cependant, les algorithmes génétiques ne prenant pas en charge les contraintes directement, nous avons pris compte des contraintes dans la définition de la fitness en ajoutant à la valeur de la fonction-objectif une pénalisation proportionnelle à la violation des contraintes. En reprenant les notations de la partie 2.2, la fitness est donc définie comme suit :

$$F(u) = \frac{1}{N}\sum_{i=1}^{N} f^i(u_i) + \omega \sum_{i=1}^{N}\sum_{j=1}^{N} C^{ij}(u),$$

où ω est le poids des contraintes dans la fitness (paramètre de pénalisation). La valeur de ce paramètre est fixée à l'avance par l'utilisateur. On pourra également envisager de faire varier ce paramètre de pénalité dans notre optimisation. Une telle approche requiert l'utilisation

d'une boucle extérieure augmentant ω chaque fois que l'algorithme converge vers un optimum ne satisfaisant pas les contraintes, et démarrant avec une valeur initiale suffisamment petite pour permettre d'abord une exploration puis l'augmentant ensuite itérativement pour forcer la réalisabilité.

Cette formulation de la fitness permet de ne pas sur-évaluer la détection de conflits mais aussi, d'avoir une fitness continue, contrairement au cas où une grille de détection était utilisée (3.1.4) . En effet, l'utilisation du nombre de conflits (valeurs entières) entraînait des discontinuités pour la fitness.

3.1.5 Opérateur de sélection

L'opérateur de sélection simule le processus de sélection naturelle, en prélevant de bons individus (mais pas uniquement) dans la génération précédente auxquels différents opérateurs génétiques sont appliqués. La fitness des individus est la seule information nécessaire pour la sélection.

Il existe différents principes de sélection dans la littérature :

- la *roulette-wheel selection*, pour laquelle chaque individu à une chance proportionnelle à sa fitness d'être sélectionné [19],
- la *sélection par rang*, pour laquelle les meilleurs individus de la population sont sélectionnés [19],
- la *sélection par tournoi stochastique*, détaillée ci-dessous [33],
- la *stochastic-remainder-without-replacement selection* qui duplique les individus plus ou moins de fois selon leur fitness avant que la sélection ne soit opérée sur cette nouvelle population [19].

Le tournoi stochastique consiste à tirer aléatoirement λ individus dans la génération précédente pour ensuite sélectionner les μ ($\lambda > \mu$) meilleurs. Ce procédé est ensuite répété pour constituer la population intermédiaire. Le rapport (choisi par l'utilisateur) entre λ et μ permet de régler la *pression sélective*. Le choix du tournoi stochastique est motivé par sa robustesse généralement reconnue ; il assure également une bonne diversité de la population au long des générations et évite en général les phénomènes de convergence prématurée.

3.1.6 Opérateur de croisement

L'opérateur de croisement (ou *crossover*) a pour rôle d'enrichir la population de solutions par brassage des gènes des bons individus. Pour cela, des chromosomes enfants sont créés à partir de chromosomes parents en manipulant les gènes de ceux-ci. L'efficacité du croisement dépend donc fortement de la structure du chromosome. Dans le cas d'un codage binaire, le croisement utilisé est par exemple le *slicing crossover* qui consiste à tirer aléatoirement une position inter-gènes pour les parents, et à échanger les sous-chaînes de bits terminales (il existe d'autres croisements pour les codages binaires). Deux enfants sont ainsi créés. Le slicing crossover peut également être étendu en utilisant plusieurs positions inter-gènes et en inversant certaines parties des chromosomes parents. Pour k positions inter-gènes utilisées, le croisement est dit *croisement à k points* (un exemple est présenté dans la Figure 3.5).

Ce type de croisement convient très bien aux problèmes d'optimisation discrète. En revanche, pour les problèmes continus, le croisement dit *barycentrique* est plus adapté. Ce type de croisement consiste à choisir un gène i dans chaque parent, et à utiliser deux transformations barycentriques pour obtenir les deux enfants comme suit. Soit $P_1(i)$ et $P_2(i)$ les gènes i de chaque parent et $E_1(i)$ et $E_2(i)$ ceux des enfants, le croisement barycentrique s'opère de la

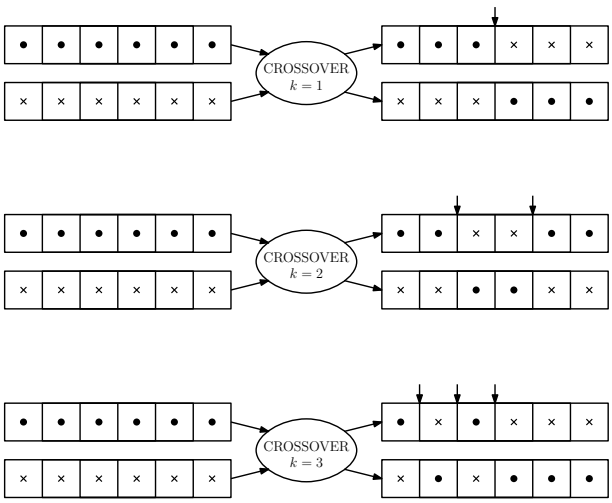

FIGURE 3.5 – Slicing crossover à k points, pour $k = 1, 2, 3$

sorte :

$$\begin{cases} E_1(i) = \alpha P_1(i) + (1-\alpha)P_2(i) \\ E_2(i) = (1-\alpha)P_1(i) + \alpha P_2(i), \end{cases}$$

où α est un coefficient de pondération tiré aléatoirement à chaque croisement et déterminé de façon à assurer que les gènes enfants restent dans le domaine autorisé.

Dans notre implémentation, nous avons utilisé le croisement barycentrique puisque nous étudions un problème continu en u. La figure 3.6 illustre le résultat d'un croisement par une visualisation des trajectoires avant et après croisement.

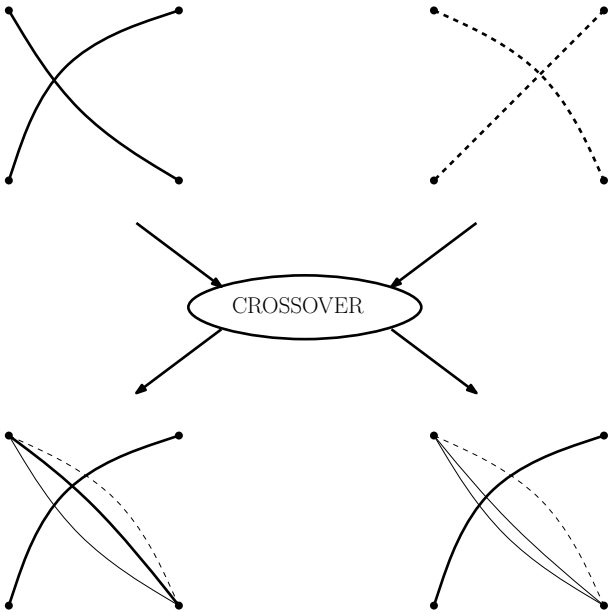

FIGURE 3.6 – Trajectoires correspondant aux parents (en haut) et aux enfants issus d'un croisement barycentrique (en bas)

3.1.7 Opérateur de mutation

L'opérateur de mutation, en générant de nouveaux gènes, a pour rôle de permettre d'explorer la totalité (en théorie) de l'espace d'état, ce qui correspond à la propriété d'ergodicité

de parcours d'espace, essentielle aux AG pour leurs propriétés de convergence. En effet les preuves théoriques de convergence [6] des AG peuvent fonctionner sans croisement, mais pas sans mutation.

L'opérateur de mutation fonctionne comme suit. Pour les problèmes discrets, un gène du chromosome est tiré aléatoirement et sa valeur est remplacée par une des autres valeurs possibles (tirée aléatoirement elle aussi). Dans le cas des problèmes continus, le gène est également tiré aléatoirement, et remplacé par une valeur aléatoire du *domaine d'extension des gènes* (espace d'état).

Pour notre problème, nous avons choisi de sélectionner un gène aléatoirement et de remplacer sa valeur par un tirage uniforme sur $[-100\%, +100\%]$. Notre opérateur de mutation est illustré par la figure 3.7 qui propose les trajectoires avant et après mutation, pour une situation à deux avions.

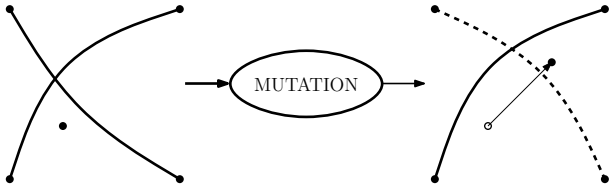

FIGURE 3.7 – Trajectoires correspondant à un chromosome avant et après mutation (déplacement du point de contrôle d'une des trajectoires)

3.2 Utilisation de méthodes d'optimisation locale continue

Dans cette partie, nous présentons les méthodes d'optimisation locale que nous utilisons. Nous commençons par la méthode d'optimisation sans dérivées (BOBYQA [1]) mise en œuvre et présentons ensuite rapidement les méthodes de points intérieurs.

3.2.1 Optimisation sans dérivée

Des méthodes d'optimisation sans dérivées avec preuve de convergence sont apparues dans les années 1990 [28, 37]. Elles répondent à une forte demande pour des méthodes d'optimisation résolvant des problèmes de type boîte noire pour lesquels la fonction-objectif ou les fonctions-contraintes sont coûteuses à évaluer (par exemple lorsque celles-ci sont issues de lourdes simulations numériques ou expérimentales) et sans dérivées explicites. Jusque-là, ces problèmes étaient résolus par des heuristiques, telles que les AG et la méthode du polytope mouvant (ou de Nelder & Mead) [31] ne présentant pas de preuves de convergence, ou encore par des variantes « différences finies » des méthodes classiques d'optimisation numérique. Ces dernières présentent certains inconvénients d'une part dans les cas de fonctions bruitées , et d'autre part dus aux $n + 1$ évaluations (coûteuses !) qu'elles requièrent pour approcher le gradient, sans compter les pertes de précision numériques dues aux erreurs d'annulation et aux divisions par petits nombres que le calcul de gradient par différences finies met en jeu.

L'algorithme BOBYQA est une méthode d'optimisation sans dérivées (*derivative-free optimization*) utilisant les méthodes de région de confiance (ou *trust region*). BOBYQA est l'une des méthodes les plus performantes pour la résolution de problèmes non-linéaires pour lesquels

1. *Bounded Optimization BY Quadratic Approximation*

les dérivées ne sont pas disponibles et pour lesquels la fonction-objectif est trop coûteuse à évaluer pour envisager l'utilisation d'une approximation des gradients par différences finies. Cette méthode repose sur une approximation quadratique de la fonction à minimiser considérée sur une région pré-déterminée (la région de confiance) plutôt que d'utiliser la fonction elle-même. Pour une présentation générale des méthodes d'optimisation sans dérivées, nous orientons le lecteur vers le livre [8].

Principe général

L'algorithme BOBYQA résout des problèmes d'optimisation sous des contraintes de bornes pour lesquels on ne dispose pas des dérivées. Le problème type est formulé comme suit :

$$\begin{cases} \min & F(x) \\ \text{s.c.} & a_i \leq x_i \leq b_i \quad \forall i = 1, \ldots, n. \end{cases} \tag{3.1}$$

L'idée de BOBYQA est de définir, à chaque itération k, un modèle quadratique Q_k considéré « valable » sur la région de confiance de centre x_k (l'itéré courant) et de rayon Δ_k qui sera mis à jour d'une itération à l'autre. Ce modèle est défini à partir des m conditions d'interpolation et traduisant le fait, qu'en m points le modèle doit être égal à la fonction qu'il approxime :

$$Q_k(y_j) = F(y_j), \quad \forall j = 1, \ldots, m,$$

où les y_j sont les points d'interpolation utilisés pour définir le modèle. L'indice j est utilisé pour numéroter les points d'interpolation et k désignera l'itération courante. Le choix de m est fait par l'utilisateur mais doit vérifier $m \in [n+2, \frac{1}{2}(n+1)(n+2)]$ pour assurer un nombre minimum nécessaire de points d'interpolation d'une part. Typiquement, les points d'interpolation correspondent aux évaluations de la fonction-objectif effectuées lors des précédentes itérations.

Une itération de BOBYQA fonctionne comme suit. À partir du modèle quadratique représentant la fonction à minimiser, une phase d'optimisation est lancée pour trouver le nouvel itéré optimal pour la région de confiance considérée. Si le modèle considéré est valable, et si ce nouvel itéré améliore la valeur de la vraie fonction-objectif, il est conservé, et le rayon de la région de confiance est agrandi (étant donné que le modèle considéré semble fiable), sinon, le procédé est relancé sur une région de confiance plus petite.

Initialisation

Pour initialiser BOBYQA, plusieurs informations doivent être fournies par l'utilisateur : tout d'abord, un point de départ pour la recherche, x_0 ; deux vecteurs, a et b, dont les composantes sont les bornes a_i et b_i utilisées dans (3.1). Sont également nécessaires : le rayon initial, Δ_1, de la région de confiance et le nombre de conditions d'interpolations, m, pour la construction du modèle quadratique.

Détermination du modèle quadratique

Lors de la première itération, l'ensemble de points d'interpolation est entièrement construit à partir de x_0, de Δ_1 et des vecteurs de base e_i de l'espace considéré comme suit et illustré en figure 3.8 pour un problème à deux dimensions.

$$\begin{cases} y_{i+1} = x_0 + \Delta_1 e_i & \text{and} \quad y_{n+i+1} = x_0 - \Delta_1 e_i & \text{si } a_i \leq (x_0)_i \leq b_i \\ y_{i+1} = x_0 + \Delta_1 e_i & \text{and} \quad y_{n+i+1} = x_0 + 2\Delta_1 e_i & \text{si } (x_0)_i = a_i \\ y_{i+1} = x_0 - \Delta_1 e_i & \text{and} \quad y_{n+i+1} = x_0 - 2\Delta_1 e_i & \text{si } (x_0)_i = b_i \end{cases}$$

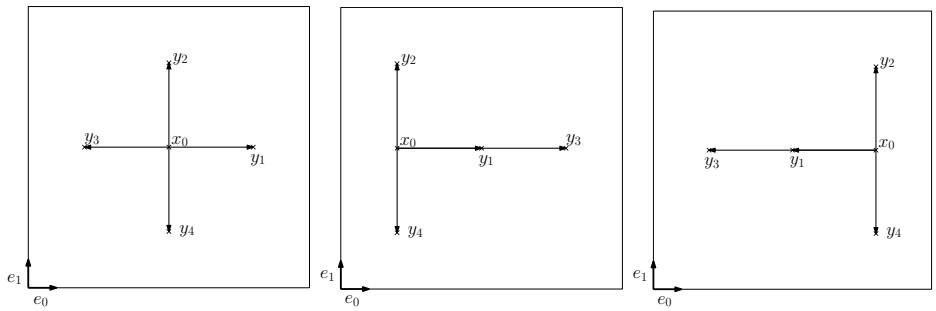

FIGURE 3.8 – Placement des y_j à partir de x_0, où $a_1 \leq (x_0)_1 \leq b_1$ dans les trois cas, et $a_0 \leq (x_0)_0 \leq b_0$ à gauche, $(x_0)_0 = a_0$ au milieu, $(x_0)_1 = a_1$ à droite

Dans le cas où m, le nombre de conditions d'interpolation demandé, est supérieur à $2n+1$, il faut ajouter des points y_j à l'ensemble détaillé ci-dessus. Ils sont déterminés à l'aide de combinaisons linéaires des y_j précédemment définis (pour plus de détails voir [39]).

Le modèle quadratique Q_1 est de plus choisi de façon à minimiser la norme de Frobenius de la Hessienne du changement de modèle, $\|\nabla^2 Q_1\|_F$, ($\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$), ce qui permet d'accélérer la convergence de l'algorithme. Cet aspect est pris en compte dans la construction de Q_1 à travers la résolution d'un système d'équations linéaire. Le gradient du modèle est quant à lui calculé uniquement à partir des points d'interpolation.

Lors des itérations suivantes, le modèle est mis à jour comme nous le verrons plus loin.

Recherche du nouvel itéré

À l'itération k, le nouvel itéré x_{k+1} est de la forme $x_{k+1} = x_k + d_k$, où d_k est une direction de recherche. L'objectif est donc ici de trouver une direction d_k qui optimise la progression de x_k à x_{k+1}. Il existe cependant deux types d'itérations. En effet, selon la qualité observée du modèle, sera appliquée une itération de type région de confiance, ou une itération dite *alternative* [38]. Nous verrons lors de la phase de mise à jour la condition exacte pour effectuer ce choix.

Si le modèle calculé lors de l'itération précédente est évalué comme fiable, une itération de type région de confiance est appliquée. La direction de recherche est alors calculée en résolvant le sous-problème d'optimisation suivant :

$$\begin{cases} \min\limits_{d} & Q_k(x_k + d) \\ \text{s.c.} & a \leq x_k + d \leq b \quad \text{et } \|d\| \leq \Delta_k \end{cases}$$

Le but de cette opération est donc de trouver le vecteur d_k qui minimise la valeur du modèle en x_{k+1} ($= x_k + d_k$) sur la région de confiance, ce qui revient à maximiser la progression effective sur le modèle durant l'itération. Ce sous-problème est traité par un algorithme de gradient conjugué tronqué (que nous ne détaillons pas ici) appliqué aux composantes de d correspondant aux composantes de x ne saturant pas les contraintes de bornes $a_i \leq x_i \leq b_i$. Les autres composantes de d sont fixées à 0. De plus, si durant l'exécution du gradient conjugué une contrainte est saturée, la composante correspondante de d est fixée à 0 et le gradient conjugué est relancé à partir de la solution courante. Lorsque la condition d'arrêt est satisfaite, l'algorithme passe à la phase de mise à jour et remplace (ou pas) un point de l'ensemble des points d'interpolation par le point obtenu par la recherche.

En revanche, si le modèle de l'itération précédente n'est pas considéré comme fiable (le progrès attendu en regard du modèle quadratique ne se confirme pas lors de l'évaluation de la

fonction-objectif), une itération alternative est utilisée. Dans ce cas, avant toute optimisation, l'ensemble des points d'interpolation est modifié. Le point le plus distant de x_k est supprimé avant le calcul du nouveau modèle Q_k et sera le point remplacé par x_{k+1} à la fin de l'itération (il est donc choisi avant l'optimisation, contrairement à l'itération de type région de confiance). Ensuite, le sous-problème d'optimisation suivant est considéré :

$$\begin{cases} \max\limits_{d} & \Lambda_t(x_k) + d^T \nabla \Lambda_t(x_k) \\ \text{s.c.} & a \leq x_k + d \leq b \quad \text{et } \|d\| \leq \Delta_k, \end{cases} \tag{3.2}$$

où $\Lambda_t(x_k)$ est une fonction quadratique qui satisfait les conditions d'interpolation de Lagrange et minimise $\|\nabla^2 \Lambda_t(x_k)\|_F$, la norme de Frobenius de sa dérivée seconde. La fonction-objectif de ce problème est en fait une approximation linéaire de $\Lambda_t(x_k+d)$ utilisée pour optimiser le temps de calcul. La valeur de $\Lambda_t(x_k)$ traduit la qualité du positionnement des points d'interpolation (et donc la qualité du modèle quadratique). Par conséquent, durant les itérations alternatives, la méthode ne cherchera pas à trouver la valeur de d_k qui minimise la fonction-objectif mais celle qui améliore le plus la qualité du modèle.

La maximisation de (3.2) est également effectuée à l'aide d'un algorithme du gradient conjugué tronqué fonctionnant exactement comme le précédent. À la suite de cette maximisation, l'algorithme passe à la phase de mise à jour.

Mise à jour

Le nouvel itéré $x_{k+1} = x_k + d_k$ ayant été trouvé à l'étape précédente, l'algorithme procède à la mise à jour des valeurs de x_k, de l'ensemble des points d'interpolation et du rayon, Δ_k, de la région de confiance. Tout d'abord, la valeur de la véritable fonction-objectif F (et non celle du modèle quadratique Q_k) est calculée en $x_k + d_k$. En fonction du progrès effectif en F, le nouvel itéré est déterminé comme suit :

$$x_{k+1} := \begin{cases} x_k & \text{si } F(x_k + d_k) \geq F(x_k) \\ x_k + d_k & \text{si } F(x_k + d_k) < F(x_k). \end{cases}$$

Si $x_k + d_k$ est accepté, il entre alors dans l'ensemble des points d'interpolation pour remplacer le point le plus éloigné de x_k.

La mise à jour du rayon, Δ_k, de la région de confiance dépend du type d'itération utilisée. Dans le cas d'une itération k de type région de confiance, un ratio r_k, traduisant la qualité du modèle, est d'abord défini :

$$r_k = \frac{F(x_k) - F(x_k + d_k)}{Q(x_k) - Q(x_k + d_k)} = \frac{Q(x_k) - F(x_k + d_k)}{Q(x_k) - Q(x_k + d_k)},$$

car x_k est un point d'interpolation. À partir de ce ratio, le rayon de la région de confiance est mis à jour comme suit :

$$\Delta_{k+1} = \begin{cases} \min[\frac{1}{2}\Delta_k, \|d_k\|] & \text{si } r_k \leq 0.1 \\ \max[\frac{1}{2}\Delta_k, \|d_k\|] & \text{si } 0.1 < r_k \leq 0.7 \\ \max[\frac{1}{2}\Delta_k, 2\|d_k\|] & \text{si } r_k > 0.7. \end{cases}$$

Cette mise à jour revient à agrandir la région de confiance lorsque le modèle est de bonne qualité (dernière ligne), à garder la même lorsque le modèle est acceptable (deuxième ligne) et

enfin à réduire la région de confiance dans le cas où le modèle n'est pas bon (première ligne). L'objectif en réduisant la région de confiance est d'améliorer la qualité du modèle. Dans le cas où r_k est inférieur à 0.1, une itération alternative est automatiquement appliquée lors de l'itération $k+1$.

Dans le cas d'une itération k alternative, la région de confiance est conservée pour l'itération suivante, $\Delta_{k+1} = \Delta_k$. En effet, le but de ces itérations étant uniquement d'améliorer le modèle, changer la région de confiance (et donc modifier la qualité du modèle obtenu) n'aurait pas de sens.

À partir du nouvel ensemble de points d'interpolation, de la nouvelle région de confiance, de l'ancien modèle Q_k, ainsi que de la condition de minimisation de la norme de Frobenius, le nouveau modèle Q_{k+1} est déterminé. Nous ne détaillons pas plus avant cette procédure technique ; plus de détails sont présentés dans la section 4 de [39].

Pour résumer, si $x_k + d_k$ fait progresser la minimisation de F, il est accepté ; sinon, la recherche d'un nouvel itéré est relancé. Dans les deux cas, selon la qualité du modèle, la région de confiance est modifiée.

Pour finir, la condition d'arrêt de cet algorithme d'optimisation porte sur le gradient du modèle. De la même manière que pour les méthodes d'optimisation classique, lorsque ∇Q_k se rapproche suffisamment de zéro, l'optimisation est arrêtée.

3.2.2 Méthodes de points intérieurs

Les méthodes de points intérieurs constituent une classe importante de méthodes d'optimisation. D'abord développées pour obtenir des résultats de complexité polynomiale en programmation linéaire [25], ces méthodes ont ensuite pris davantage d'essor alors qu'elles ont été étendues à des classes beaucoup plus générales de problèmes d'optimisation, telles l'optimisation non-linéaire et l'optimisation semi-définie (*semi-definite programming*). Le principe général étant, comme le nom l'indique, de rester à l'intérieur de l'espace admissible durant la recherche, par opposition aux méthodes de contraintes actives où l'on cherche à se déplacer sur la frontière du domaine réalisable. Différentes méthodes de points intérieurs existent, nous détaillons ici la plus intuitive : les méthodes *barrières*. Ces méthodes définissent une fonction-objectif auxiliaire à partir de la fonction-objectif du problème considéré. Elle est égale à la fonction-objectif plus une pénalité qui tend vers l'infini à mesure qu'on se rapproche des frontières du domaine réalisable. La minimisation de cette fonction auxiliaire empêche que l'on atteigne les frontières de l'ensemble admissible. Nous avons décidé d'utiliser la méthode de points intérieurs fournie par l'*optimization toolbox* de Matlab dans la routine *fmincon*. Nous détaillons moins le fonctionnement des méthodes de points intérieurs puisqu'elles sont mieux connues (plus anciennes) que les méthodes d'optimisation sans dérivée.

La méthode de points intérieurs mise en œuvre dans fmincon est décrite dans [5]. Nous la décrivons dans les grandes lignes. Le problème général considéré est :

$$\begin{cases} \min\limits_{x} & f(x) \\ \text{s.c.} & h(x) = 0 \\ & g(x) \leq 0. \end{cases}$$

Pour résoudre ce problème, une méthode barrière logarithmique est mise en œuvre (cf chapitre 19 de [3]). Par conséquent, une fonction-objectif modifiée f_μ est utilisée et le nouveau problème

d'optimisation auxiliaire est :

$$\begin{cases} \min\limits_{x,s} & f_\mu(x,s) = f(x) - \mu \sum\limits_{i=1}^{m} \ln(s_i) \\ \text{s.c.} & h(x) = 0 \\ & g(x) + s = 0, \end{cases} \tag{3.3}$$

où μ_k est une suite de paramètres (dits de pénalité) fixant la « hauteur » des barrières aux frontières à l'itération k et dont la valeur décroit vers zéro au long des itérations, s (vecteur de composantes s_i) est le vecteur contenant les variables d'écart (*slack*), utilisées pour la relaxation des m contraintes d'inégalités.

Recherche du nouvel itéré

La recherche du nouvel itéré se fait en utilisant deux itérations distinctes : une itération dite de *direct step* ou une itération gradient conjugué. À l'exception du cas où la fonction f_μ n'est pas convexe au voisinage de l'itéré courant, une itération direct step est d'abord tentée. En fonction de la réussite de cette cette itération, une itération gradient conjugué est appliquée ou pas.

Une itération direct step a pour but de résoudre les équations de Karush-Kuhn-Tucker(détaillées par exemple dans le chapitre 12.3 de [32]), portant sur le lagrangien et sur les multiplicateurs de Lagrange du problème (3.3). Cette résolution a pour but de déterminer les pas de recherche Δx et Δs, elle revient en fait à résoudre un système d'équations. Si cette étape échoue, du fait du conditionnement de la hessienne de f_{μ_k}, une itération de gradient conjugué est lancée.

L'itération de gradient conjugué cherche à résoudre le problème (3.3) en appliquant une méthode de gradient conjugué tronqué utilisant l'approximation quadratique de f_{μ_k} sur une région de confiance définie de rayon r_k soumise aux contraintes linéarisées. Nous ne détaillons pas ici la gestion de la région de confiance car elle est similaire à celle décrite pour la méthode BOBYQA.

Mise à jour

Pour déterminer si le nouvel itéré est accepté, une fonction mérite est alors définie :

$$F(x,s) = f_\mu(x,s) + v\|(h(x), g(x)+s)\|,$$

où v est un paramètre qui peut varier au fil des itérations (il est possible d'augmenter sa valeur pour forcer un mouvement vers des solutions admissibles). Si le nouvel itéré ne fait pas décroître la fonction mérite (c'est-à-dire si $F(x_{k+1}, s_{k+1}) \geq F(x_k, s_k)$), il n'est pas accepté. La procédure de recherche d'un nouvel itéré est alors relancée.

La condition d'arrêt utilisée se base sur les conditions d'optimalité de premier ordre de Karush-Kuhn-Tucker [32].

Chapitre 4

Résultats numériques et perspectives opérationnelles

Dans ce chapitre, nous présentons les résultats numériques obtenus sur plusieurs problèmes tests avant de discuter des perspectives opérationnelles de nos méthodes.

4.1 Résultats numériques

Nous présentons ici le raisonnement qui a guidé nos travaux de recherche. Notre volonté première était d'utiliser un nouveau modèle de trajectoire courbe permettant de déterminer une trajectoire à partir d'un nombre très limité de paramètres. Nous avons donc souhaité tester les B-splines. Cette modélisation permet d'approcher toutes manœuvres de résolution de type offset ou point tournant tout en offrant une trajectoire plus courte, plus économique en termes de consommation et plus confortable pour les passagers. De plus, le cadre opérationnel défini par SESAR (et notamment le développement de FMS-4D, cf partie 1.2.3), permet d'envisager le suivi de trajectoires courbes par un avion.

Dans un premier temps, nous avons testé notre modèle de trajectoire en utilisant des méthodes d'optimisation communément utilisées dans le domaine de la résolution de conflits. Par conséquent, nous avons mis en œuvre un algorithme génétique (AG) pour générer les B-splines décrivant les trajectoires [36] et utilisant une grille de détection pour calculer les fonctions définissant les contraintes de séparation. Ces premiers tests validant l'utilisation des B-splines, nous avons cherché à obtenir une formulation mathématique du problème d'optimisation plus intéressante. Les dérivées du problème obtenu semblant explicitement calculables, nous nous sommes orientés vers des méthodes classiques d'optimisation locale très efficaces. Cependant, le calcul de ces dérivées s'est avéré lourd et a surtout soulevé de nombreux problèmes numériques (tout spécialement le calcul du gradient des contraintes). Par conséquent, afin de valider la viabilité des méthodes d'optimisation locale, nous avons utilisé fmincon avec une approximation du gradient des contraintes par différences finies. Devant l'effort numérique occasionné par les schémas de différences finies (qui requièrent plusieurs évaluations - coûteuses - de la fonction pour le calcul d'un gradient), nous avons également testé des méthodes d'optimisation sans dérivée.

Nous utilisons le nombre d'évaluations de fonctions ainsi que la valeur finale de la fonction-objectif en tant que principaux critères de comparaison des différentes méthodes d'optimisation citées précédemment. Nous n'insistons pas sur le temps de calcul requis car ces méthodes sont codées avec des langages de programmation très divers et il en va donc de même pour les sous-routines évaluant les fonctions objectif et contraintes. Néanmoins, à titre d'exemple et

afin de donner une idée de l'ordre de grandeur, nous comparons l'évaluation de la fonction-objectif et des fonctions contraintes pour une même situation et pour le même point de l'espace d'état sur un processeur 2.53GHz Intel Core 2 Duo utilisant un système d'exploitation Ubuntu 12.04 LTS. Les méthodes locales utilisant Matlab pour cette opération présentent une temps d'exécution de 26 millisecondes contre 7 millisecondes pour l'AG en Java. Notons de plus que cette différence peut varier selon le point de l'espace d'état considéré puisque, plus les avions sont en conflit longtemps, plus le calcul des contraintes prendra de temps (une plus longue durée de conflit occasionnant en effet davantage de pas de discrétisation lors de l'intégration numérique). Nous ne nous baserons donc pas sur le temps de calcul pour comparer l'efficacité de nos méthodes, les différences de performance entre les langages utilisés étant trop importantes pour pouvoir effectuer une comparaison juste. Afin de donner une meilleure idée du temps de calcul, nous donnons ici maintenant le temps de calcul pour une résolution *entière*, tout en rappelant qu'il ne peut être utilisé comme critère de comparaison :

- Pour $N = 2$ avions, fmincon en Matlab trouve une solution en 2.35 s quand l'AG en Java prend 12 s,
- Pour $N = 6$ avions, fmincon en Matlab trouve une solution en 330 s quand l'AG en Java prend 160 s,
- Pour $N = 16$ avions, fmincon en Matlab trouve une solution en 1853 s quand l'AG en Java prend 207 s.

4.1.1 Paramètres numériques

Nous détaillons ici les caractéristiques générales de notre implémentation numérique ainsi que les valeurs des paramètres devant être réglés par l'utilisateur pour chaque méthode.

L'algorithme génétique que nous avons développé est implémenté en Java. Comme mentionné auparavant, la méthode d'optimisation locale différentiable est la routine fmincon de l'Optimization toolbox de Matlab. La méthode d'optimisation sans dérivées BOBYQA est implémentée en Fortran 77 et utilisée à partir de Matlab par une interface mex. C'est précisément à cause de ces différences de langage, qui entraînent des différences de performance significatives, que nous utilisons le nombre d'évaluations de fonction ainsi que la valeur de la fonction-objectif de la solution obtenue comme point de comparaison et non le temps de calcul. Cette pratique est courante pour les problèmes d'optimisation dits *boîtes noires* où l'essentiel du temps de calcul est de toute façon passé pour l'évaluation de la fonction-objectif et/ou des fonctions contraintes.

Nous présentons ci-dessous les valeurs des paramètres numériques que nous avons utilisées pour les différents algorithmes d'optimisation :

1) Taille de la population de l'AG : 100 ; nombre de générations : 100 (par conséquent, l'AG évaluera 10000 fois la fonction-objectif et les fonctions contraintes)
2) Probabilité de mutation : $p_m = 0.3$
3) Probabilité de croisement : $p_c = 0.6$
4) Paramètre de pénalisation des contraintes : $\omega = 0.01$
5) Critère d'arrêt pour les méthodes locales : $||u_{k+1} - u_k|| < 10^{-6}$

Nous avons choisi empiriquement une valeur de paramètre de pénalité ω, suffisamment faible pour permettre l'exploration de l'espace d'état. Il s'est avéré que cette unique valeur initiale, $\omega = 0.01$, a également permis d'assurer chaque fois l'obtention d'une solution réalisable, l'utilisation d'une boucle externe gérant une augmentation de la valeur du paramètre de pénalité

n'a donc pas été utile (ω a été maintenu à cette valeur constante pour l'ensemble de nos expérimentations).

Nous avons choisi le nombre de générations et la taille de la population pour l'AG de manière empirique. Pour ne pas changer les paramètres à chaque exécution, selon la complexité du problème, nous nous sommes basé sur l'un des problèmes-tests les plus complexe que nous présentons ici : le rond-point à $N = 8$ avions que nous présentons dans la partie suivante. Nous avons alors choisi les paramètres nécessaires à l'AG pour, premièrement trouver un sous-domaine réalisable, et deuxièmement affiner l'optimum réalisable trouvé. La taille de la population a été fixée à 100 en prenant pour exemple les travaux de Durand [15] et en se basant sur l'observation qu'avec notre modélisation, l'espace d'état semblait bien recouvert par une telle taille de population. Pour le choix du nombre de générations, nous avons observé l'évolution de la fitness au fil des générations. La figure 4.1, pour le cas particulier du problème du rond-point avec $N = 8$ avions, est représentative de ce que nous avons observé. Cette figure

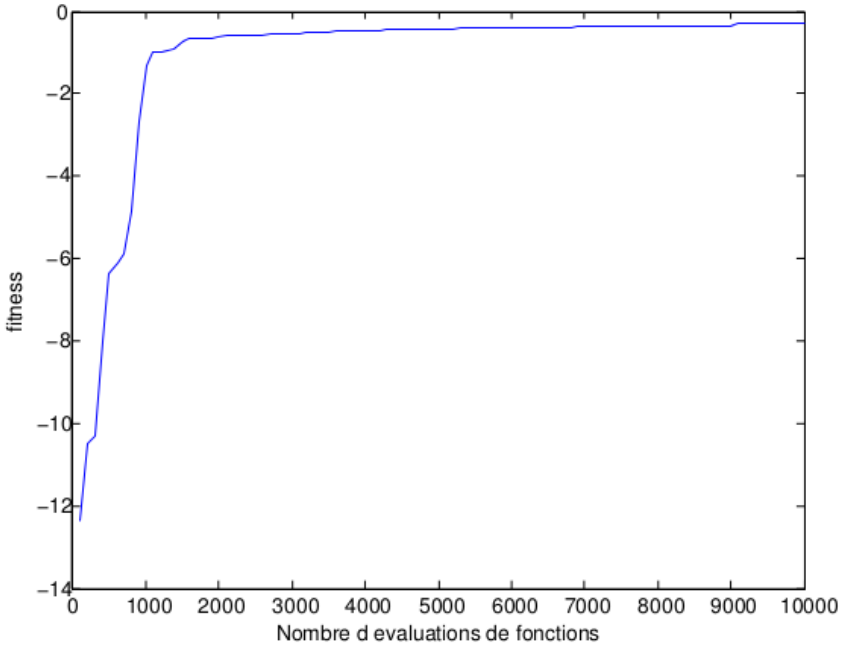

FIGURE 4.1 – Évolution de la fitness par rapport au nombre d'évaluations de fonctions pour le problème du rond-point à $N = 8$ avions

est typique de l'évolution de la fitness pour un algorithme génétique qui progresse très vite au début puis utilise énormément d'évaluations pour affiner la recherche. Nous avons voulu donner le temps à l'AG de converger vers une solution de qualité. L'AG pourrait continuer à progresser au-delà (on voit par exemple une amélioration après 9000 évaluations de fonction) mais l'investissement en évaluations de fonctions ne semble pas utile au vu du gain correspondant en fitness. Nous considérons donc qu'après 100 générations (10000 évaluations) la valeur de la solution proposée est de qualité suffisante. On constatera, sans surprise, qu'une méthode d'optimisation locale pourra toujours améliorer (de façon marginale) une telle solution.

4.1.2 Situations de trafic artificielles classiques

Dans cette partie, nous présentons les résultats numériques obtenus sur des situations de trafic artificielles du contrôle du trafic aérien qui sont souvent utilisées comme problèmes tests

dans la littérature. Chacune de ces situations artificielles traduit cependant une réalité opérationnelle et permet ainsi d'évaluer la viabilité de nos méthodes. Seuls les résultats utilisant une modélisation de trajectoire B-spline sont détaillés ici, la modélisation linéaire par morceaux présentant des résultats similaires. De plus, nous présentons uniquement le détail de la comparaison entre l'AG et la méthode de points intérieurs (fmincon) puisque la méthode sans dérivées (BOBYQA) obtient des résultats similaires à la méthode de points intérieurs. Une analyse succincte des légères différences observables entre ces deux méthodes d'optimisation locale sera néanmoins présentée en fin de partie.

Dans l'analyse suivante, les résultats obtenus par l'AG sont utilisés comme référence. En effet, de par sa propriété d'ergodicité, l'AG peut atteindre, dans un voisinage suffisamment, petit n'importe quel point de l'espace d'état, avec 10000 évaluations de fonction, on s'attend donc à une valeur de fonction-objectif suffisamment proche de la valeur optimale. De plus, l'AG est une méthode globale qui gère la combinatoire du problème, contrairement des méthodes d'optimisation locale. L'AG semble donc être un bon élément de comparaison pour tester les performances de nos méthodes locales. Lorsque nous parlons de solution attendue ou souhaitée, nous évoquons les solutions reconnues comme optimales par les contrôleurs aériens et les spécialistes de la résolution automatique de conflits [15]

Nous allons ici chercher à démontrer que les méthodes locales peuvent se montrer aussi efficaces sur des problèmes de résolution de conflits que des méthodes couramment utilisées pour ce problème. Nous présentons d'abord les résultats obtenus sur des situations de trafic artificielles, mais représentatives de situations opérationnelles. Nous utilisons un seul point de départ pour notre recherche locale choisi de manière aléatoire (tirage suivant une loi uniforme pour chaque composante de u). Nous verrons que différentes solutions sont obtenues selon le point de départ de la recherche. Pour les situations plus compliquées, un pré-traitement sera appliqué.

Le croisement face à face

Nous commençons par une situation critique : deux avions arrivant l'un en face de l'autre sur la même route. Cette situation est présentée dans la figure 4.2 (les traits représentant les trajectoires sont légèrement décalées pour plus de lisibilité mais les avions volent bien exactement sur la même route). La résolution par AG renvoie la solution présentée dans la

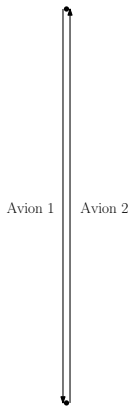

FIGURE 4.2 – Croisement de deux avions face à face

figure 4.3. Les méthodes d'optimisation locale fournissent des résultats différents selon le point de départ de la recherche. Sur cet exemple simple, ces méthodes trouvent, selon le point de

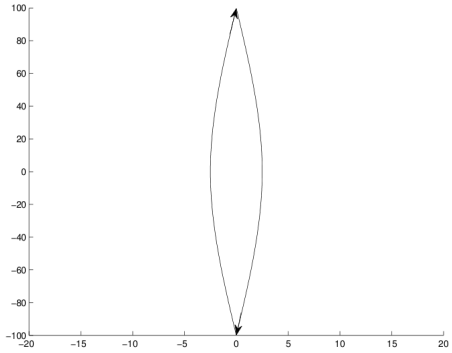

FIGURE 4.3 – Trajectoires obtenues par AG

départ, une des deux solutions attendues au problème présentées dans la figure 4.4. On voit

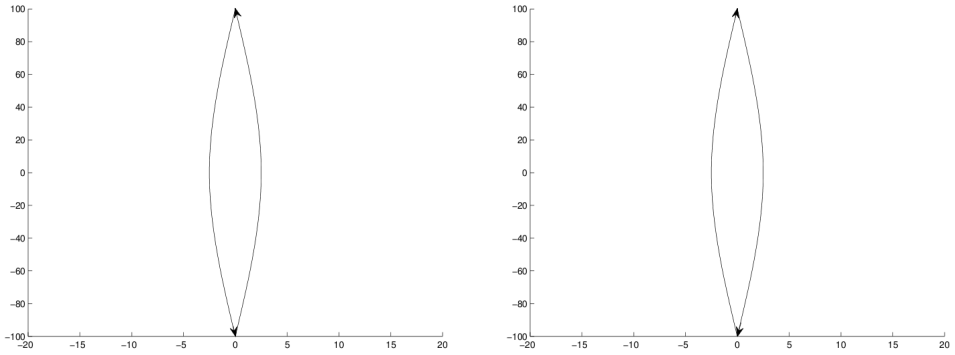

FIGURE 4.4 – Trajectoires obtenues par méthode locale des points intérieurs avec deux points de départ différents

que les deux solutions obtenues sont symétriques : à droite, les deux avions tournent à droite pour s'éviter, et à gauche, ils tournent tous les deux à gauche. On remarque également que la distance minimale entre les deux avions (au milieu de leurs trajectoires respectives, en $y = 0$) est de 5 Nm. Les trajectoires obtenues respectent donc les contraintes de séparation.

Du point de vue de la qualité de la solution, l'AG et la méthode de points intérieurs fournissent des résultats quasi-identiques. De plus, la méthode des points intérieurs trouve une des deux solutions attendues quel que soit le point de départ. La méthode de points intérieurs trouve une solution très légèrement meilleure (la valeur de sa fonction-objectif est inférieure de 0.01% à celle de l'AG) malgré un nombre beaucoup plus faible d'évaluation de 79.

Le croisement à 90°

Comme expliqué dans la partie 1.1.4, le trafic aérien est organisé selon des routes aériennes qui se croisent en des balises. Le trafic dans le voisinage de ces balises a donc tendance à converger vers ce point de l'espace. Il est donc courant que deux, voire plusieurs avions, se dirigent vers le même point pour ensuite conserver leurs caps. Nous avons donc construit un second problème-test représentatif d'une situation dans laquelle deux avions arrivent au même point au même moment en conservant leurs caps respectifs. Cette situation est illustrée en figure 4.5. On suppose que les deux avions parcourent la même distance, à la même vitesse constante. Ils arrivent donc au centre de la situation considérée au même instant, occasionnant ainsi un conflit.

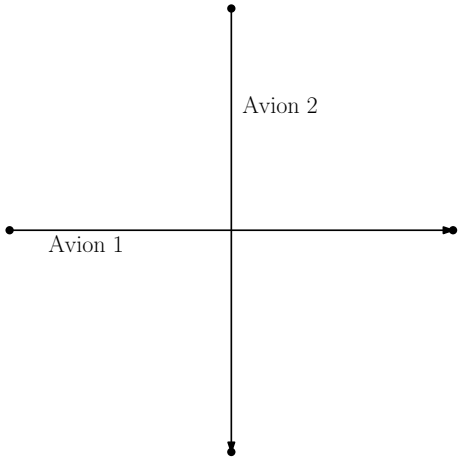

FIGURE 4.5 – Croisement de deux avions à 90°

Nous comparons maintenant sur ce second problème-test les résultats obtenus par les différentes méthodes d'optimisation que nous avons retenues. Présentons d'abord la solution obtenue par l'AG. Elle est présentée dans la figure 4.6. Les deux avions tournent légèrement

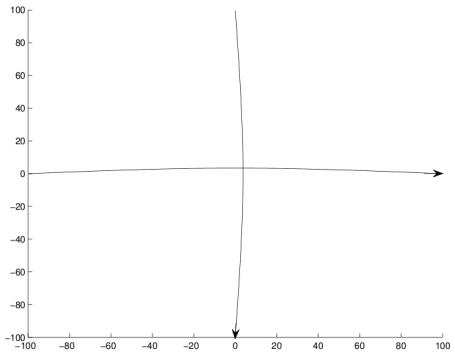

FIGURE 4.6 – Trajectoires obtenues par AG

vers la gauche de manière à s'éviter, l'avion 1 (horizontal) passant derrière l'avion 2 (vertical). On peut constater qu'avec notre approche coopérative, les déviations sont réparties sur les deux avions. Voyons maintenant différentes solutions obtenues avec la méthode de points intérieurs et représentées dans la figure 4.7. On voit, ici aussi, que les deux solutions sont symétriques. Dans les deux cas, les contraintes de séparation sont respectées. En ce qui concerne la qualité des solutions, les solutions obtenues par les méthodes de points intérieurs sont très légèrement meilleures puisque la valeur de la fonction-objectif est inférieure de 0.01% à celle obtenue avec l'AG malgré un nombre d'évaluation beaucoup plus faible : 127.

La fusion

La situation présentée ici est également typique du trafic aérien opérationnel. Deux avions suivant deux caps différents peuvent se rejoindre en une balise pour ensuite suivre la même route. On appelle cette situation une *fusion* de routes. Elle peut devenir problématique lorsque les deux avions arrivent en ce point de fusion en même temps. C'est cette situation que nous avons cherché à traduire dans ce troisième problème-test, avec deux avions ayant pour origines deux points d'un même cercle, et dont les routes fusionnent au centre de ce cercle. Ici, l'avion

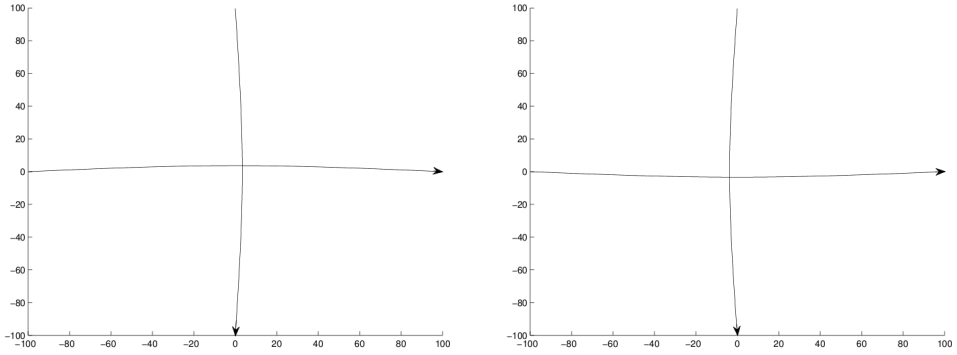

FIGURE 4.7 – Trajectoires obtenues par méthode locale des points intérieurs avec deux points de départ différents

1 rejoint la route suivie par l'avion 2. Les deux avions ayant la même vitesse constante, ils arrivent en même temps au centre du cercle. Cette situation est illustrée dans la figure 4.8.

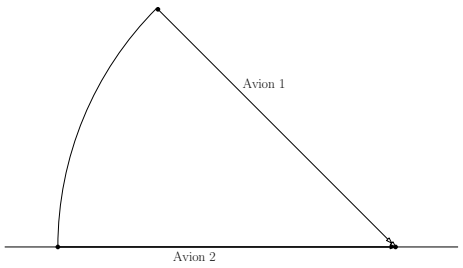

FIGURE 4.8 – Fusion des routes de deux avions

Présentons maintenant les résultats obtenus par nos méthodes. L'AG conduit à la solution présentée dans la figure 4.9. Le résultat obtenu n'est pas conforme à ce que l'on souhaiterait.

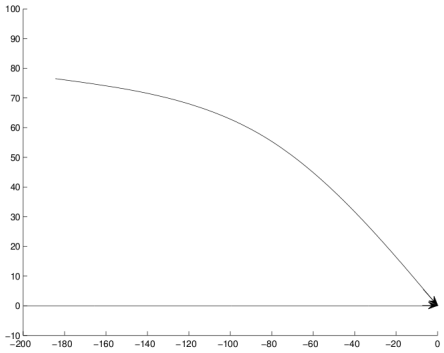

FIGURE 4.9 – Trajectoires obtenues par AG

En effet, dans la solution attendue, l'avion 1 viendrait rejoindre la route de l'avion 2 de manière asymptotique. Cependant, la fonction-objectif ne discriminant pas deux solutions sans conflit présentant le même rallongement, l'AG choisit une des deux solutions de manière aléatoire, c'est ce qui semble se passer ici.

Intéressons-nous aux solutions obtenues par la méthode des points intérieurs. Nous montrons trois de solutions équivalentes dans la figure 4.10. La solution attendue en vue de poursuivre sur la route fusionnée est présentée en haut à droite de la figure 4.10. On voit en revanche

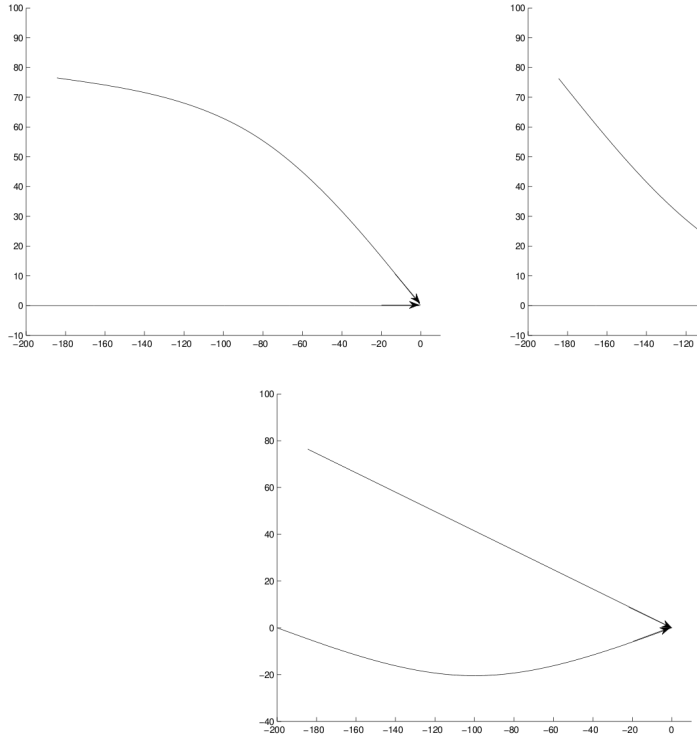

FIGURE 4.10 – Trajectoires obtenues par méthode locale de points intérieurs avec trois points de départ différents

que ce n'est pas nécessairement la solution trouvée selon le point de départ choisi. Les trois solutions trouvées ici correspondent à différents optima locaux du problème. La solution en haut à droite de la figure 4.10 est préférable pour les contrôleurs car elle satisfait une contrainte opérationnelle (cachée), à savoir que l'avion 1, qui rejoint la route de l'avion 2, doit la rejoindre de manière tangente. Cette situation montre en effet les limites de notre modèle pour cette situation très particulière. Un modèle B-spline de trajectoire pourrait être adapté à ce cas particulier de façon à forcer la tangence à une direction donnée par construction. Néanmoins, en proposant les trois différentes solutions à l'opérateur (le contrôleur aérien), notre méthodologie reste viable dans la mesure où le contrôleur utilise son expérience pour discriminer la solution la plus adaptée.

On peut également souligner que, pour une même structure de solution, c'est-à-dire, pour des allures de trajectoires similaires, les points intérieurs trouvent une meilleure solution que l'AG (encore une fois, très légèrement meilleure). Cette observation est cohérente puisque les algorithmes d'optimisation locale, à convergence asymptotique, arrivent à affiner, à améliorer une solution, là où un AG cherche de manière plus chaotique. Enfin, en moyenne, les méthodes de points intérieurs utilisent beaucoup moins d'évaluations de fonction : 153 en moyenne contre 10000 pour notre AG. Il est cependant nécessaire de préciser que lorsque l'AG trouve une solution légèrement meilleure que la solution obtenue par la méthode locale, il atteint le même niveau de qualité que cette dernière en moyenne en 400 évaluations de fonction.

Le dépassement

On décrit maintenant un quatrième problème-test classique qui correspond à une situation également très courante : le dépassement. Elle se produit lorsque deux avions de vitesses différentes suivent la même route aérienne. Lorsque l'avion en retrait vole plus vite que le premier

et doit dépasser celui-ci, on parle de *rattrapage*. Nous avons donc créé le test illustré dans la figure 4.11 (les traits représentant les trajectoires sont légèrement décalés pour plus de lisibilité mais les avions volent bien exactement sur la même route) où l'avion 2 est rattrapé par l'avion 1 qui vole à une vitesse deux fois supérieure. De fait, les deux avions se retrouvent en conflit au milieu de l'horizon temporel considéré.

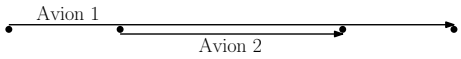

FIGURE 4.11 – Situation de dépassement entre deux avions

Présentons maintenant les résultats obtenus par nos méthodes d'optimisation. Notre AG obtient la solution présentée dans la figure 4.12. Cette solution est conforme à nos attentes. Il

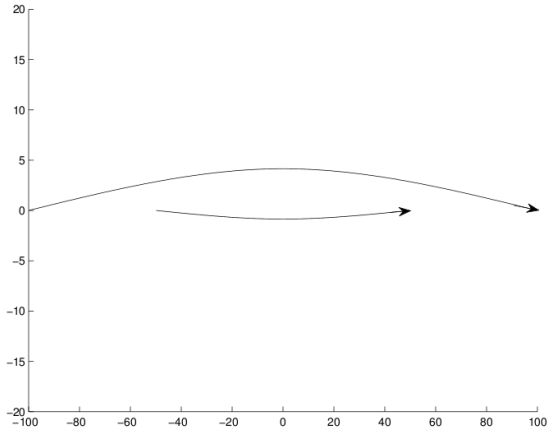

FIGURE 4.12 – Trajectoires obtenues par résolution par AG

existe deux solutions équivalentes puisque l'avion en rattrapage peut passer à droite où à gauche de l'avion 2. Une de ces deux solutions est toujours trouvée par la méthode de points intérieurs. Selon le point de départ, on trouvera l'une ou l'autre. elles sont présentées dans la figure 4.13. On voit que les deux solutions sont symétriques. Elles ont la même valeur de fonction-objectif.

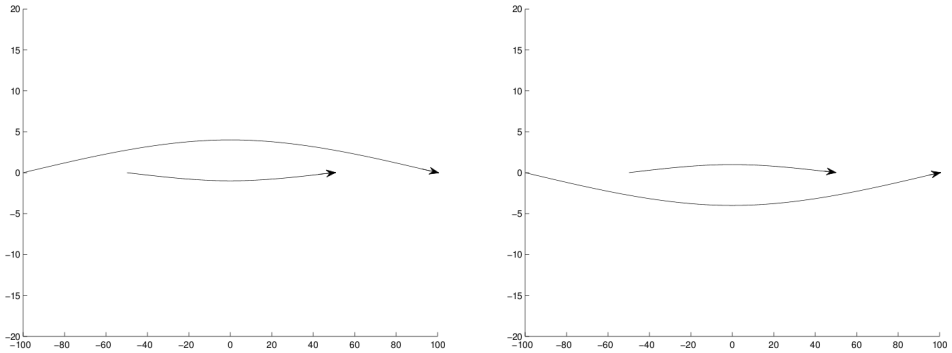

FIGURE 4.13 – Trajectoires obtenues par résolution par méthode locale des points intérieurs avec deux points de départs de la recherche différents

Elles sont meilleures que la solution obtenue par notre AG de 0.3%. Il apparaît également qu'ici encore, les déviations sont réparties entre les deux avions. Sur cette situation de trafic, la méthode de points intérieurs est économe en nombre d'évaluation de fonction : 186.

Le rond-point

Le problème dit du rond-point est une situation relativement complexe très largement utilisée pour tester les méthodes automatique de résolution de conflits. La solution souhaitée est présentée largement dans la littérature [15] et correspond à l'action qu'utiliserait un contrôleur aérien dans une telle situation. La solution consiste donc à faire tourner tous les avions dans le même sens, puis à les rediriger vers leur route originale. Cette situation consiste à faire voler plusieurs avions dont les points d'origines sont équirépartis sur un cercle vers le point diamétralement opposé. Ainsi, les avions volant tous à la même vitesse constante, on crée un conflit impliquant tous les avions au centre du cercle. Nous considérons un cinquième problème-test représentant cette situation avec un cercle de rayon de 100 Nm et N avions volant à une vitesse de 600 Nm/h. L'horizon temporel est donc d'approximativement 20 minutes. Ce problème peut être traité pour différents nombre d'avions N. Nous présentons dans la figure 4.14 un exemple de rond-point à $N = 6$ avions. Nous traitons ici le problème du rond-point

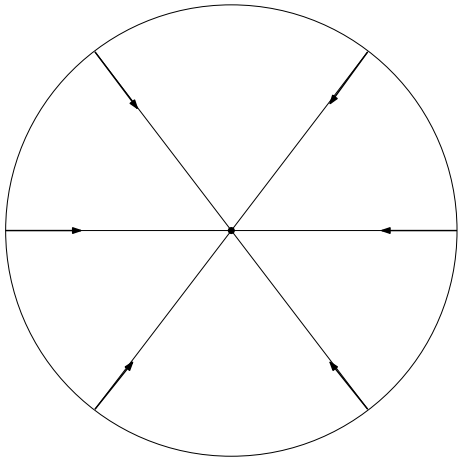

FIGURE 4.14 – Problème du rond-point pour $N = 6$ avions

pour différentes tailles de problèmes : $N = 4, 6, 8$.

Commençons par le problème du rond-point à $N = 4$ avions. La solution obtenue par l'AG est présentée dans la figure 4.15. La solution correspond à la résolution attendue d'une

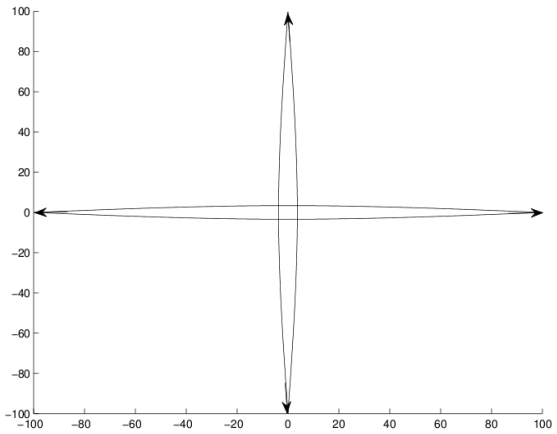

FIGURE 4.15 – Trajectoires obtenues par résolution par AG

méthode automatique de résolution de conflits : tous les avions tournent dans le même sens.

Pour ce problème-test, les solutions fournies par la méthode de points intérieurs présentent des différences significatives tant d'un point de vue structure de la solution que d'un point de vue qualité de la solution. Les différentes solutions sont présentées en figure 4.15 et 4.16. On

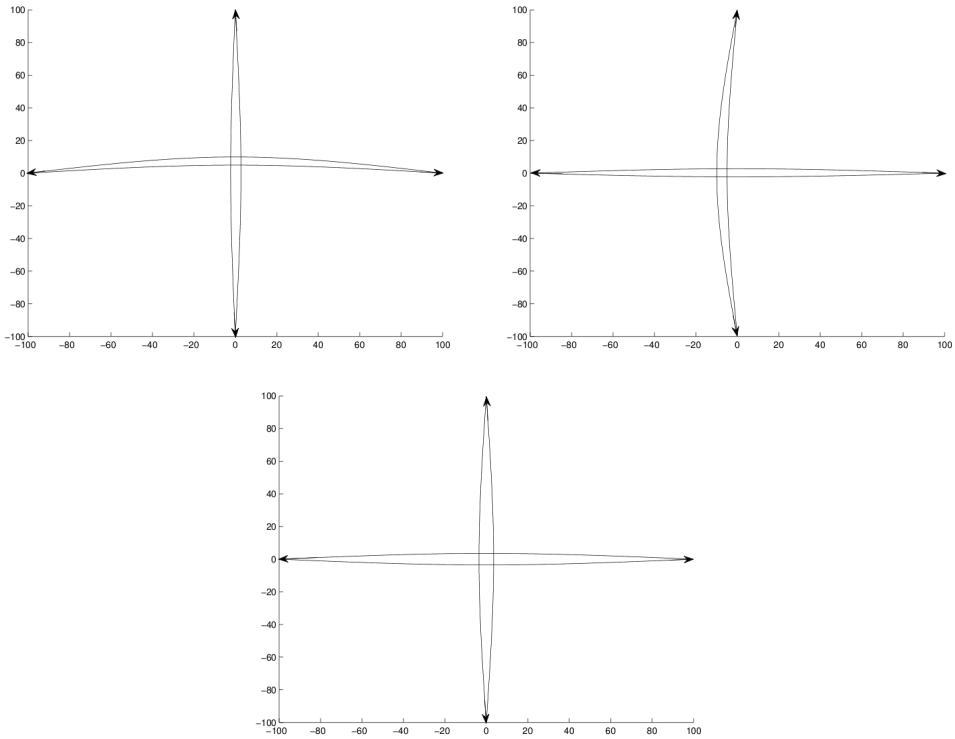

FIGURE 4.16 – Trajectoires obtenues par méthode locale des points intérieurs avec deux points de départs différents

voit bien dans les deux solutions du haut que tous les avions ne tournent pas dans le même sens, ce n'est pas la solution attendue. De plus, ces deux solutions occasionnent des déviations importantes par rapport à la solution obtenue par l'AG : 50% d'augmentation de la valeur de la fonction-objectif. Ces deux solutions sont bien réalisables. La troisième solution (bas de la figure 4.16), quant à elle, répond à nos attentes puisque tous les avions tournent dans le même sens. La valeur de la fonction-objectif de cette solution est, en outre, 5% meilleure que celle obtenue par l'AG. Enfin, d'un point de vue du nombre d'évaluations de fonction, la méthode de points intérieurs est toujours bien plus économe : cette troisième solution est trouvée en 286 évaluations.

Dans le cas du rond-point à $N = 6$ avions, on obtient des résultats analogues et les observations sont similaires. La solution obtenue par l'AG est présentée en figure 4.17. On observe à nouveau que, comme attendu, tous les avions tournent dans le même sens pour s'éviter mutuellement. Les différentes solutions obtenues par la méthode de points intérieurs sont présentées en figure 4.18.

Les déviations sont beaucoup plus importantes lorsque tous les avions ne tournent pas dans le même sens. La valeur de la fonction-objectif est jusqu'à trois fois supérieure à celle obtenue à l'aide de l'AG. En revanche, ici aussi, lorsque les avions tournent tous dans le même sens : la valeur de la fonction-objectif est 0.7% meilleure que celle de la solution de l'AG. Le nombre d'évaluation est toujours bien plus faible que celui des AG : 513 évaluations pour la meilleure solution obtenue par méthode de points intérieurs (contre 10000 pour l'AG bien que l'AG

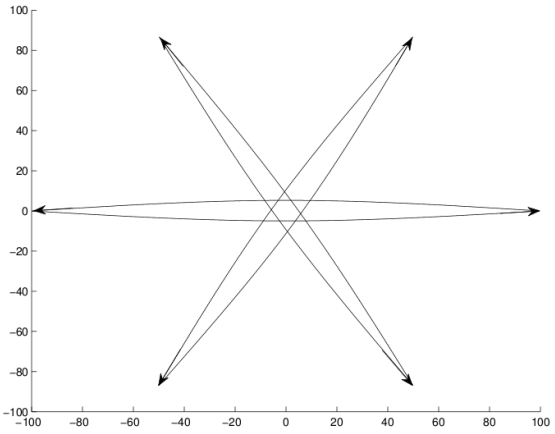

FIGURE 4.17 – Trajectoires obtenues par AG

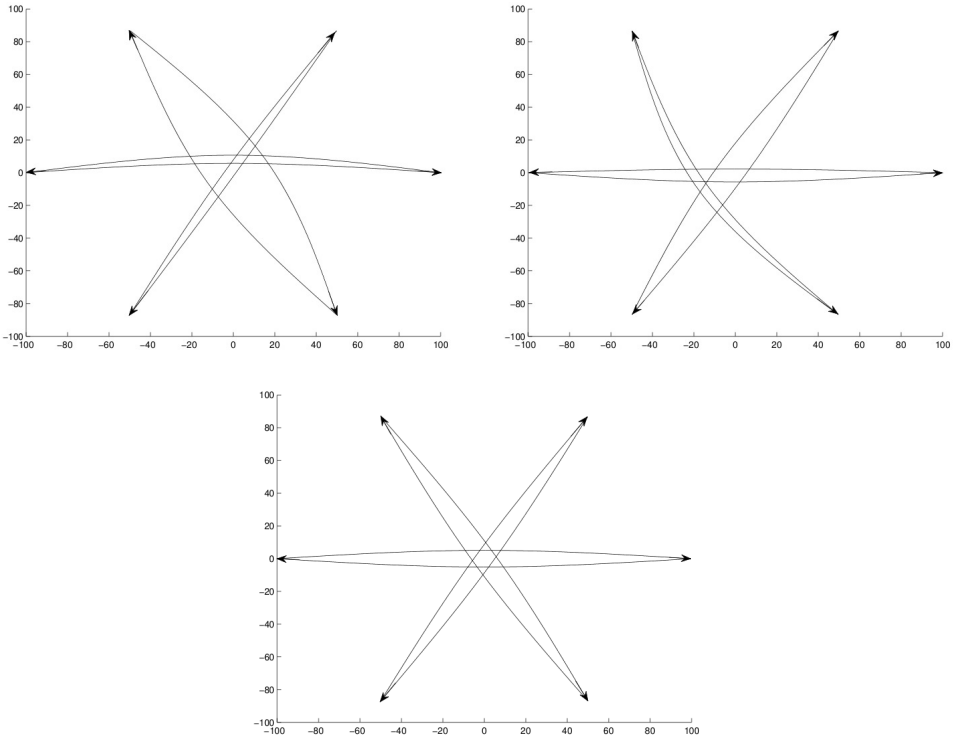

FIGURE 4.18 – Trajectoires obtenues par résolution par méthode locale des points intérieurs avec deux points de départs de la recherche différents

trouve sa meilleure solution après seulement 6700 évaluations).

Notons que jusqu'à $N = 6$ avions, la méthode de points intérieurs trouve chaque fois une solution réalisable aux problèmes testés. La qualité de cette solution est variable selon le point de départ mais elle satisfait toujours les contraintes. À partir de $N = 8$ avions, cela n'est plus le cas. Si le point de départ ne traduit pas le fait que les avions tournent dans le même sens (tous les u_i de même signe), aucune solution réalisable n'est trouvée par la méthode de points intérieurs. L'AG trouve quant à lui la solution attendue, présentée en figure 4.19. La méthode des points intérieurs peut ne pas trouver de solution pour un tel problème si

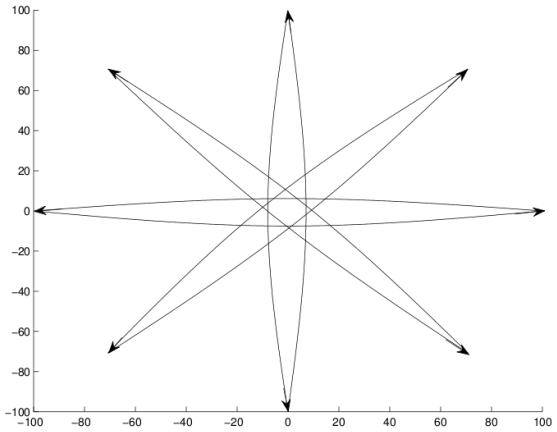

FIGURE 4.19 – Trajectoires obtenues par AG

le positionnement du point de départ de la recherche n'est pas bon. Nous avons donc mis en œuvre une stratégie très simple pour trouver un bon point de départ pour la recherche, à savoir : évaluer la fonction-objectif en 100 points de l'espace d'état tirés aléatoirement (exactement de la même manière que la population initiale de l'AG est engendrée), et faire démarrer la méthode de points intérieurs à partir du meilleur point réalisable parmi ces 100 points. Cette stratégie, très basique, s'est avérée suffisante puisqu'elle permet de trouver facilement une solution au problème à toutes les instances étudiées. La solution trouvée par cette adaptation de la méthode de points intérieurs pour le rond-point à 8 avions est présentée en figure 4.20. La solution ainsi

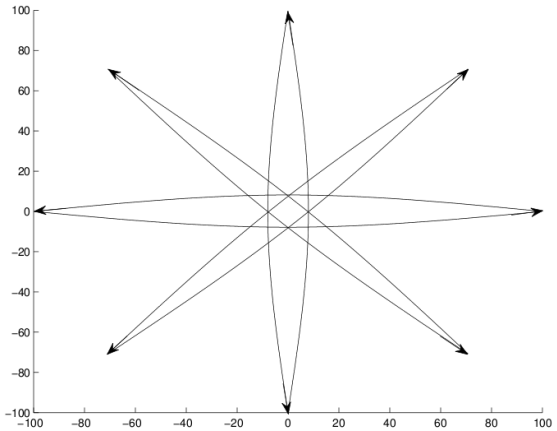

FIGURE 4.20 – Solution optimale obtenue par méthode de points intérieurs démarrant d'un bon point de départ

obtenue par la méthode de points intérieurs répond alors à nos attentes et elle obtient un résultat légèrement meilleur (la valeur de la fonction-objectif est 3% meilleure que celle de la solution obtenue avec l'AG). Ce qui se traduit, au centre de la figure 4.20 par une symétrie parfaite, par opposition à la figure 4.19 qui représente une bonne solution mais qui n'est pas localement optimale (la méthode de points intérieurs, lorsqu'elle est démarrée avec le meilleur des 100 points générés aléatoirement trouve l'optimum local concerné). De plus, en prenant en compte le nombre d'évaluations de fonction nécessaire à l'obtention du point de départ de la recherche, la méthode de points intérieurs ne requièrent que 914 évaluations de fonction au total pour trouver la solution souhaitée du problème.

Observations complémentaires

Nous avons choisi de ne pas présenter les résultats fournis par la deuxième méthode d'optimisation locale, BOBYQA. En effet, les résultats obtenus par BOBYQA étant de qualité similaire à ceux par la méthode de points intérieurs, nous ne détaillerons que ceux-ci. Nous avons cependant noté quelques différences entre les résultats obtenus avec les deux méthodes locales que nous avons mis en œuvre. Premièrement, lorsque le même optimum local est trouvé par les deux méthodes, BOBYQA converge en général plus finement vers l'optimum local considéré (très légèrement) malgré l'avantage significatif que possède la méthode de points intérieurs avec la dérivée de la fonction-objectif. Deuxièmement, il arrive à BOBYQA de trouver « seulement » un optimum différent quand la méthode de points intérieurs trouve la même solution que l'AG, ce qui peut paraître étrange. En effet, au premier abord, on pourrait s'attendre à ce qu'une méthode sans dérivées telle BOBYQA, qui étale ses points initiaux d'évaluation, ait davantage de chance d'explorer un grand sous-ensemble du domaine réalisable, par opposition à une méthode avec recherche de pas (*line search*) classique, telle fmincon, qui n'évalue la fonction-objectif qu'en chaque itéré. Cependant, les méthodes sans dérivées s'appuient sur une approximation quadratique et notre fonction-objectif étant une somme de carré facilement approché par un modèle quadratique, la région de confiance se réduit donc généralement très rapidement autour du point de départ. Il n'est donc pas surprenant que la méthode d'optimisation sans dérivées trouve ici un optimum local qui est dans le voisinage direct du point de départ (dans le même bassin d'attraction).

Nous avons également choisi de ne pas présenter les figures des résultats obtenus avec la modélisation de trajectoire linéaire par morceaux, les résultats étant en tout point semblables à ceux obtenus avec la modélisation B-spline.

Bilan

Dans cette partie, nous avons testé nos méthodes d'optimisation sur des situations de trafic artificielles. Les méthodes locales trouvent des solutions de qualité comparables à celles trouvées par l'AG. De plus, en utilisant différents points de départ pour la recherche locale, il est possible d'obtenir différents optimas locaux, qui, dans l'éventualité d'un passage à l'opérationnel, pourraient être discriminés par l'opérateur (le contrôleur aérien). Il apparaît également à travers ces tests, que, lorsque le nombre d'avions impliqués dans le conflit augmente, une boucle extérieure peut devenir nécessaire à la résolution. Cette boucle reste très simpliste puisqu'il s'agit uniquement de lancer la recherche locale d'un bon point de départ (le meilleur point parmi 100 points tirés aléatoirement). Nous utiliserons cette technique pour les prochains tests.

Afin de montrer le gain effectif qu'apporterait l'obtention du gradient des contraintes pour la modélisation B-splines, nous présentons ici un tableau contenant le nombre d'évaluations de fonctions et le nombre d'itérations effectuées pour la modélisation linéaire par morceaux avec les différences finies et avec le gradient exact pour chaque situation artificielle présentée plus haut. On voit clairement l'apport considérable des dérivées, particulièrement lorsque le nombre d'avions, et donc la dimension du problème, augmente. Ces résultats appuient donc la continuation de nos travaux dans le but de trouver une technique numériquement efficace pour le calcul exact du gradient des contraintes des B-splines.

	Différences finies		Gradient exact	
	itérations	évaluations	itérations	évaluations
Face à face	7	112	7	80
Croisement	10	122	20	87
Fusion	18	140	19	28
Dépassement	28	187	22	23
Rond-point $N = 4$	7	146	7	82
Rond-point $N = 6$	50	820	35	179
Rond-point $N = 8$	19	340	16	21

TABLE 4.1 – Comparaison de l'effort numérique avec différences finies et gradient exact

4.2 Perspectives Opérationnelles

Dans cette partie, nous discutons de la possibilité d'un transfert de nos méthodes locales vers le domaine opérationnel du contrôle aérien, que ce soit en vue d'une (peu probable) automatisation complète des tâches actuelles du contrôleur aérien, ou en vue du développement d'outils d'aide à la décision venant s'ajouter aux instruments du contrôleur. Tout d'abord, nous présentons un problème-test plus représentatif d'une situation opérationnelle. Nous présentons dans un second temps une étude statistique des résultats obtenus. Enfin, nous discutons des développements restant à faire pour envisager une application opérationnelle de notre méthodologie (modélisation B-spline de trajectoire et formulation du problème d'optimisation numérique).

4.2.1 Problème du rond-point bruité

Pour créer une base de situations tests plus représentatives de situations réelles, nous nous inspirons du problème du rond-point présenté dans la partie 4.1.2. Nous introduisons un bruit aléatoire dans les positions de départ et d'arrivée des avions sur le cercle ainsi que dans la vitesse de chaque avion pour obtenir une situation où N avions convergent vers une *zone* réduite de l'espace. Nous fixons dans cette étude le nombre d'avions à $N = 6$. Selon les contrôleurs, il est exceptionnel que plus de 4 avions soient impliqués dans le même conflit dans des situations de trafic en-route (le contexte de cette thèse). Pour générer nos situations de rond-point bruité, nous avons procédé de la manière suivante. Pour chacun des 6 avions :

- le point de départ de la situation de rond-point classique est déplacé sur le cercle d'un angle tiré aléatoirement suivant une loi uniforme sur $[-20^{\circ}; 20^{\circ}]$
- Le point diamétralement opposé au point ainsi obtenu est alors à sont tour déplacé de la même manière pour obtenir le point d'arrivée de l'avion.

Par conséquent, la symétrie inhérente à la situation du rond-point classique est détruite. La vitesse est également bruitée : la vitesse de chaque avion i est ainsi générée aléatoirement suivant une loi uniforme de manière à ce que $v_i \in [550; 650]$ knots (Nm/h). Un exemple de situation obtenue par ce procédé est présenté dans la figure 4.21. Nous générons ainsi 100

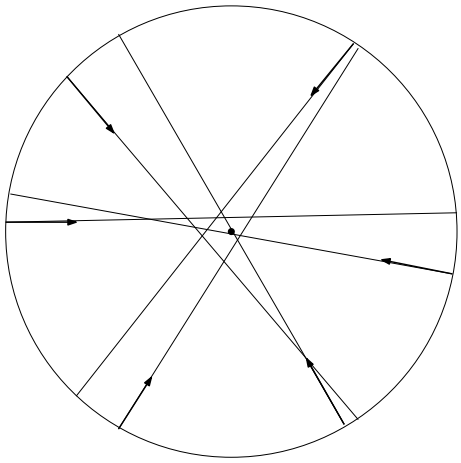

FIGURE 4.21 – Exemple de situation aléatoire pour $N = 6$ avion

situations différentes qui constitueront autant d'instances sur lesquelles nous allons appliquer les méthodes d'optimisation afin de conduire notre étude statistique sur les résultats obtenus.

4.2.2 Résultats

Dans cette section nous comparons les résultats des méthodes d'optimisation locale vues précédemment sur les 100 instances créées en section précédente, les résultats obtenus par l'AG sont utilisés comme référence étant donné que, avec 10000 évaluations de fonctions, on peut s'attendre à ce que l'AG trouve ce que nous considérons comme solution de référence. La valeur de fonction-objectif correspondante sera notée $f^\star$.

Plusieurs critères sont pris en compte pour comparer nos méthodes. Premièrement, le pourcentage de succès des différentes méthodes sont rapportés (par succès, il est entendu que la méthode a convergé vers un optimum local réalisable). Des statistiques sur la valeur de la fonction-objectif de la solution sont ensuite présentées : la moyenne, le pire et le meilleur élément sur les résultats (correspondant à chacune des 100 instances). Pour chaque instance, la différence entre $f^\star$ et la valeur de fonction-objectif de la solution fournie par la méthode locale (lorsque la méthode locale a convergé vers une solution réalisable). Lorsqu'on donnera la meilleure (respectivement la pire) différence de valeurs de fonction-objectif, on fera référence à l'instance pour laquelle la différence entre $f^\star$ et la valeur de fonction-objectif est la plus grande (respectivement la plus petite). Enfin, le nombre d'évaluations de fonctions requises pour atteindre l'optimum sera aussi donnée. On rappelle que pour l'AG, il s'agit systématiquement de 10000 évaluations de fonctions et que, pour les deux méthodes d'optimisation locale fmincon et BOBYQA, ce nombre inclus les 100 évaluations préalables qui ont conduites à la détermination d'un bon point de départ. Tous ces résultats sont rassemblés dans la table 4.2, où les meilleurs résultats obtenus sont affichés en caractères gras et où « f » fait référence à la valeur de la fonction-objectif (représentant la déviation totale, qu'on souhaite minimiser).

		AG	fmincon	BOBYQA
Taux de succès		**100%**	95%	**100%**
Moyenne de f		**0,044**	0,049	0,091
Médiane de f		**0,044**	0,070	0,11
$f - f^\star$	moyenne	-	**0,029**	0,067
	médiane	-	**0,006**	0,047
	pire	-	0,34	**0,29**
	meilleur	-	**-0,054**	**-0,069**
Nombre d'évaluations de fonction	moyenne	10000	1135	**400**
	médiane	10000	820	**393**
	pire	10000	6825	**1033**
	Meilleur	10000	**178**	194

TABLE 4.2 – Comparaison des résultats numériques sur 100 instances du rond-point bruité

On peut constater que le pourcentage de succès des méthodes d'optimisation locale est élevé, ce qui montre que ces méthodes sont une solution de rechange crédibles pour peu qu'on utilise une modélisation de trajectoire et une formulation du problème d'optimisation adaptées. Les différences de valeurs de la fonction-objectif aux optima nous révèle que les méthodes locales fournissent même dans certains cas des solutions de meilleure qualité. En effet, l'algorithme fmincon trouve le meilleur optimum pour 48 instances (sur les 95 instances pour lesquelles une solution réalisable est trouvée), ce qui est très prometteur. L'algorithme d'optimisation sans dérivées BOBYQA trouve une meilleure solution pour 23 instances, ce qui reste significatif. Nous discutons de l'effort numérique (nombre d'évaluations de fonctions nécessaires) pour parvenir à ce résultat plus bas. En revanche, pour les autres instances, là où les méthodes locales

produisent des solutions de moins bonne qualité, ces solutions sont en général de qualité très inférieure à celles fournies par l'AG (du point de vue de la valeur de la fonction-objectif). En fait, les méthodes locales ont convergé vers des minimas locaux réalisables de qualité moindre.

Comme mentionné précédemment, l'AG effectuent 10000 évaluations de fonctions pour chaque instance, même lorsqu'une solution sans conflit est trouvée bien avant d'avoir consommé ces 10000 évaluations. On peut légitimement se demander après combien d'évaluations l'AG trouve une solution sans conflit. En moyenne, la première solution sans conflit est trouvée très tôt : après 110 évaluations, ce qui correspond presque à une recherche purement aléatoire. Cependant, ces solutions sont de très mauvaise qualité mettant en jeu des grandes déviations qui ne sont pas acceptables. Dans tous les cas, l'obtention de la valeur de $f^\star$ requiert beaucoup plus d'évaluations (environ 6 fois le nombre moyen d'évaluations nécessaires à fmincon et 15 fois celui de BOBYQA). Les méthodes locales sont donc bien plus économes que l'AG en termes de nombre d'évaluations de fonctions pour l'obtention d'une qualité comparable. Pour finir, rappelons que le nombre d'évaluations de fonctions de fmincon est fortement biaisé par l'utilisation des différences finies pour le calcul du gradient des contraintes : en N dimensions, chaque approximation du gradient par différences finies requiert $N+1$ évaluations de fonctions, et le gradient est calculé au moins une fois par itération. En effet, le nombre moyen d'*itérations* de fmincon est de seulement 54. Par conséquent, lorsque les gradients des contraintes seront finalement implémentés efficacement (travaux de recherche actuels), on peut s'attendre à ce que l'effort numérique lié aux évaluations de fonctions et de gradients diminue, comme montré dans la sous-partie 4.1.2 pour la modélisation linéaire par morceaux. Cela rendra les méthodes d'optimisation numérique locale classiques (points intérieurs ou programmation quadratique séquentielle) encore plus efficaces pour le problème de résolution de conflits.

4.3 Bilan

Nous avons montré ici que les méthodes d'optimisation locale, associées à une modélisation de trajectoire avantageuse et à une formulation mathématique astucieuse du problème, pouvaient se révéler efficaces pour le problème de la résolution. Le point faible de ces méthodes est de ne pas garantir l'obtention d'un minimum de qualité comparable à celui obtenu par une méthode global lorsque le nombre d'avion augmente. On peut quand même souligner qu'en associant ces méthodes à une stratégie d'optimisation globale à deux phases basique (en l'occurrence ici nous avons utilisé la plus triviale, le multi-start), nous avons souvent pu trouver un optimum de qualité comparable à celui obtenu par la méthode globale (à défaut, un optimum local réalisable est toujours trouvé) pour les situations de conflit impliquant jusqu'à 8 avions. La voie des méthodes d'optimisation globale à deux phases (une phase globale et l'approche locale que nous avons proposé) dont certaines présentent des conditions pour garantir l'obtention de l'optimum global [42]. Ce type de méthode serait d'autant plus intéressant lorsque le calcul numérique du gradient des fonctions contraintes sera validé. En effet, le nombre d'évaluations de fonctions nécessaires à une méthode locale pour converger devenant ainsi très faible, l'effort numérique supplémentaire engendré par une stratégie globale devrait rester acceptable.

Conclusions et perspectives

Le système en charge de la gestion du trafic aérien est en pleine mutation. Pour continuer à assumer ses prérogatives, à savoir garantir un écoulement sûr et fluide de l'ensemble du trafic aérien alors que le trafic continue de croître, des innovations sont nécessaires. Les deux grands projets SESAR et NextGen posent les bases de travail pour ces innovations en définissant le cadre opérationnel du futur. Parmi ces innovations, le problème de résolution de conflits est particulièrement étudié puisque l'apport d'un outil d'aide à la décision pour le contrôleur serait une avancée en vue d'une automatisation partielle du contrôle aérien.

Dans cette thèse, nous avons cherché à utiliser les avancées technologiques prévues par SESAR pour proposer un modèle de trajectoire avantageux ainsi qu'une nouvelle formulation mathématique du problème de résolution de conflits.

La première contribution de cette thèse est l'introduction d'un modèle de trajectoire courbe, par B-splines, envisageable dans le cadre opérationnel défini par SESAR. Ce modèle de trajectoire, en plus d'être plus économe que la modélisation linéaire par morceaux habituelle des manœuvres classiques d'offset et de point tournant, présente l'avantage de définir une trajectoire par un nombre très limité de paramètres : un seul dans notre cas.

La deuxième contribution de cette thèse réside dans la nouvelle formulation mathématique du problème de résolution de conflits : un problème d'optimisation semi-infinie mettant en jeu des fonctions différentiables, ouvrant la voie à l'utilisation de méthodes d'optimisation locale avec preuve de convergence et qui sont reconnues pour leur efficacité. Nous avons successivement utilisé les récentes méthodes d'optimisation sans dérivées, les méthodes de points intérieurs avec approximation des gradients des fonctions contraintes par différences finies et avec le gradient explicite que nous avons pu obtenir pour la fonction-objectif.

Les tests numériques effectués tendent à montrer la viabilité de notre méthodologie (association entre modèle de trajectoire parcimonieux et méthodes d'optimisation numérique locale). Notre approche coopérative (chaque avion prend en compte l'agissement des avions voisins) permet de plus d'obtenir une répartition équitable des déviations entre les avions impliqués dans les conflits. Les méthodes d'optimisation numérique locale fournissent des résultats comparables à ceux obtenus par les algorithmes génétiques, très utilisés pour ce problème, pour peu que ces méthodes locales soient combinées avec une naïve stratégie de multi-start. Ces résultats numériques sont prometteurs et ouvrent la voie à la réhabilitation de l'utilisation des méthodes d'optimisation locale pour le problème de résolution de conflits. Les méthodes locales ont plusieurs avantages : elles présentent une preuve de convergence et elles sont également beaucoup plus économes numériquement, nécessitant beaucoup moins d'évaluations de fonction pour converger et ce, avec des vitesses de convergence asymptotique inconnues des populaires heuristiques qui sont habituellement utilisées pour la résolution de conflits.

Ce travail ouvre donc de nombreuses perspectives. Nous avons été surpris à maintes reprise durant cette thèse des possibilités qui s'ouvraient à nous, nous permettant d'aller plus loin dans des directions que nous n'avions pas envisagées initialement. Nous avons par conséquent exploré

de nombreuses pistes à différents niveaux, et notamment plusieurs méthodes d'optimisation. De ce fait, nous avons manqué de temps pour faire évoluer notre méthodologie vers un contexte plus opérationnel. Cependant, les résultats numériques sont encourageants et augurent d'un bon comportement de notre méthodologie B-splines/méthodes d'optimisation sur du trafic réel.

Parmi les axes de travail à mener à partir des travaux présentés dans cette thèse, notons tout d'abord le travail entamé sur le calcul d'une forme explicite du gradient des fonctions contraintes. La piste que nous avons commencé à explorer, si elle s'avérait fructueuse, pourrait améliorer significativement notre méthodologie, du point de vue de l'efficacité numérique comme de celui de la qualité des résultats obtenus. Une autre piste de recherche prometteuse serait de remplacer la boucle externe basée sur une naïve stratégie multi-start par une approche à deux phases plus élaborée (par exemple la méthode *multi-level single-linkage* présentée dans [42] permettrait de lancer l'optimisation locale de façon relativement économe depuis différents bassins d'attraction). Des travaux de recherche dans cette direction pourraient permettre de garantir l'obtention d'optima globaux et/ou de fournir aux contrôleurs un éventail de solutions localement optimales. Les contrôleurs pourraient alors faire leur choix en fonction du trafic global ou d'autres contraintes cachées (non prises en compte par la formulation mathématique).

Bibliographie

[1] L. Angerand and H. Lejeannic. Bilan du projet SAINTEX. Technical Report R92009, Centre d'études de la navigation aérienne, Toulouse.

[2] D. Arbuckle. NextGen ATM concept of operations : ASAS reliant. Technical report, Centre d'Études de la Navigation Aérienne, septembre 2007.

[3] M. Bierlaire. *Introduction à l'optimisation différentiable.* Presse polytechniques et universitaires romanes, 2006.

[4] D. Bonini, C. Dupré, and G. Granger. How ERASMUS can support an increase in capacity in 2020. In *Proceedings of the 7th International Conference on computing communications and control technologies (CCCT)*, Orlando, Florida, 2009.

[5] R. H. Byrd, J. C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89 :149–185, 2000.

[6] R. Cerf. *Une Théorie Asymptotique des Algorithmes Génétiques.* PhD thesis, Université Montpellier II (France), 1994.

[7] A. R. Conn and N. Gould. An exact penalty function for semi-infinite programming. *Mathematical Programming*, 37(1) :19–40, 2000.

[8] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization.* MPS-SIAM Series on Optimization, 2009.

[9] D. V. Dimarogonas and K. J. Kyriakopoulos. A feedback stabilization and collision avoidance scheme for multiple independent nonholonomic non-point agents. In *Proceedings of the IEEE International Symposium on Robotics and Automation*, 2005.

[10] M. P. Do Carmo. *Riemannian Geometry.* Birkhauser, Boston, MA, 1992.

[11] N. Dougui. *Planification de trajectoires avion : approche par analogie lumineuse.* PhD thesis, Université Toulouse III, Paul Sabatier, Décembre 2011.

[12] N. Dougui, D. Delahaye, S. Puechmorel, and M. Mongeau. A light-propagation model for aircraft trajectory planning. *To appear in Journal of Global Optimization*, 2012.

[13] M. Duncan. *Applied Geometry for Computer Graphics and CAD.* Springer, 2nd edition, 2005.

[14] V. Duong, E. Hoffman, J. Nicolaon, L. Floc'Hic, and A. Bossu. Extended flight rules to apply to the resolution of encounters in autonomous airborne separation. Technical report, Eurocontrol, Brétigny, 1996.

[15] N. Durand. *Optimisation de trajectoires pour la résolution de conflits.* PhD thesis, Institut National Polytechnique de Toulouse, Mai 1996.

[16] N. Durand and J.-M. Alliot. Un algorithme de colonie de fourmis pour résoudre des conflits aériens. In *ROADEF, Toulouse*, 2010.

[17] X. Fron, B. Maudry, and J.-C. Tumelin. ARC 2000 : Automatic radar control. Technical report, Eurocontrol, 1993.

[18] D. Gianazza. Algorithme évolutionnaire et A* pour la séparation en 3D des flux de trafic aérien. *Journal Européen de systèmes automatisés*, 2005.

[19] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Reading MA Addison Wesley, 1989.

[20] T. Guillod. Interpolations, courbes de Bézier et B-splines. In *Bulletin de la Société des enseignants Neuchâtelois de Sciences*, 2008.

[21] R. Hettich and K. O. Kortanek. Semi-infinite programming : theory, methods, and applications. *SIAM Rev.*, 35(3) :380–429, sept 1993.

[22] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan press, 1975.

[23] ILOG. *Cplex 9.1 User's Manual.* IBM, 2005.

[24] M. Jardin. Grid-based strategic air traffic conflict detection. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA ; USA, aout 2005.

[25] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4) :373–395, décembre 1984.

[26] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 500–505, St. Louis, MO, USA, Mars 1985.

[27] J. K. Kuchar and L. C. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transcations on Intelligent Transportation Systems*, 1 :179–189, 2000.

[28] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. Technical report, ICASE, NASA Langley Research, 1996.

[29] F. Médioni. *Méthode d'optimisation pour l'évitement aérien : systèmes centralisés, systèmes embarqués.* PhD thesis, Thèse doctorat informatique de l'Ecole Polytechnique, 1998.

[30] F. Médioni, N. Durand, and J.-M. Alliot. Algorithmes génétiques et programmation linéaire appliqués à la résolution de conflits atc. In *Evolution Artificielle 94.*

[31] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7 :308–313, 1965.

[32] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, 2nd edition, 2006.

[33] C. Oei, D. Golberg, and S. Chang. Tournament selection, niching, and the preservation of diversity. Technical Report 0, Illinois Genetic Algorithms Laboratory, 1991.

[34] X. Olive. *Résolution de conflits par algorithmes stochastiques parallèles.* PhD thesis, École Nationale Supérieure de l'Aéronautique et de l'Espace, 2006.

[35] L. Pallottino, E. Feron, and A. Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems*, 3(1) :3 –11, mars 2002.

[36] C. Peyronne, D. Delahaye, M. Mongeau, and L. Lapasset. Optimizing B-splines using genetic algorithms applied to air-traffic conflict resolution. In *International Conference on Evolutionnary Computation 2010 proceedings*, Valencia, Espagne, Octobre 2010.

[37] M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7 :287–336, 1998.

[38] M. J. D. Powell. Developments of NEWUOA for minimization without derivatives. Technical Report NA2009/06, Department of Applied Mathematics and Theoretical Physics, Cambridge, England, 2007.

[39] M. J. D. Powell. The bound optimization by quadratic approximation (BOBYQA) algorithm for bound constrained optimization without derivatives. Technical Report NA05, Department of Applied Mathematics and Theoretical Physics, Cambridge, England, 2009.

[40] C. Rabut. *Courbes, outils pour la CAO, approximation à une variable*, 2008. INSA-Toulouse lecture notes.

[41] E. Rimon and D. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5) :501–518, 1992.

[42] A. Rinnooy Kan, G. T. Timmer, and L.Stougie. Stochastic global optimization methods Part II : Multi level methods. *Mathematical Programming*, 39 :57–78, 1987.

[43] G. P. Roussos, G. Chaloulos, K. J. Kyriakopoulos, and J. Lygeros. Control of multiple non-holonomic air vehicles under wind uncertainty using model predictive control and decentralized navigation functions. In *Proceedings of the IEEE International Conference on Decision and Control*, pages 1225–1230, 2008.

[44] SESAR consortium. The ATM target concept d3. Technical Report DLM-0706-001-02-00, EUROCONTROL, Toulouse, septembre 2007.

[45] SESAR consortium. SESAR master plan. Technical Report DLM-0710-001-02-00, Eurocontrol, Bruxelles, avril 2008.

[46] C. Visweswariah, R. Haring, and A. R. Conn. Noise considerations in circuit optimization. *IEEE transactions on computer aided design of integrated circuits and systems*, 19(6) :679–690, 2000.

[47] A. Wright. Genetic algorithms for real parameter optimization. In *Proceeding of the Foundation Of Genetic Algorithms*, pages 205–218. FOGA, 1991.

[48] K. Zeghal. *Vers une théorie de la coordination d'actions, application à la navigation aérienne.* PhD thesis, Université Paris VI, 1994.

Résumé

La gestion du trafic aérien est un système complexe. Actuellement en pleine mutation, une des problématiques essentielles à l'évolution du système est la recherche de méthodes automatiques de résolution de conflits. Nous présentons d'abord un nouveau modèle de trajectoire courbe basé sur les B-splines et permettant de définir une trajectoire à l'aide d'un nombre très limité de paramètres. À partir de cette modélisation, nous arrêtons une nouvelle formulation du problème de résolution de conflits pour obtenir un problème d'optimisation continue. Celle-ci repose sur une formulation dite *semi-infinie* de la contrainte de séparation entre deux avions. La manière dont nous avons défini la fonction-objectif et les fonctions contraintes nous permettent également d'en calculer les gradients.

Nous utilisons trois différentes méthodes d'optimisation pour résoudre notre problème. Une méthode globale stochastique est d'abord testée : les algorithmes génétiques, couramment utilisés pour le problème de résolution de conflits. Deux méthodes d'optimisation locale sont aussi mises en œuvre, une méthode de points intérieurs et une méthode d'optimisation sans dérivées. Enfin, nous présentons des résultats numériques prometteurs montrant la viabilité de l'optimisation locale pour le problème de résolution de conflits. Notre méthodologie, alliant une modèle de trajectoire courbe parcimonieux et une méthode d'optimisation locale appliquée à notre formulation mathématique du problème, est une option crédible pour le problème de résolution de conflits aériens.

Abstract

Air traffic management is a complex system. Currently in transition, one of the main evolutions to this system is the search for an automatic method for solving air-traffic conflict problems. In this thesis, we present a new way to tackle the problem. First, we introduce a new smooth trajectory model based on B-splines thus allowing us to define a trajectory using only a few parameters. Using this trajectory model, we define a new formulation for the conflict resolution problem yielding a continuous optimization problem. It relies on a semi-infinite programming formulation of the separation constraints between two aircraft. Our formulation enables the calculation of the functions involved in the optimization problem.

We use three different optimization methods to solve our problem. A stochastic global optimization method has first been tested : a genetic algorithm, of the type commonly used in air-traffic conflict resolution. Two local optimization methods are also implemented : an interior point method and a derivative-free optimization method. Finally, we present promising numerical results showing the viability of local optimization method for the air-traffic conflict resolution problem. More precisely, our methodology, relying on a parsimonious smooth trajectory model and local optimization methods applied to our mathematical formulation of the problem, is a credible alternative for the air-traffic conflict resolution problem.