# Méthodes d’optimisation pour l’évitement aérien : systèmes centralisés, systèmes embarqués

*Drôle de pilote, drôle de tête... Sûr, on va tomber !*

Barbara, *Vol de nuit*

*On n'y prend pas garde, tout d'abord ; puis, c'est horrible ; cela s'attache à nous par la seule vertu du poids. On ne s'en débarasse pas. Il faut porter jusqu'à la fin toutes les idées qu'on soulève...*

André Gide, *Paludes*

# Remerciements

Au moment d'achever la rédaction de ma thèse, je tiens à remercier Jean-Marc Alliot. Il a été présent à toutes les étapes d'un chemin long et parfois difficile. C'est en suivant ses cours à l'ENAC que j'ai pris goût à l'informatique. C'est lui qui m'a montré l'intérêt des problèmes posés par l'évitement aérien, qui m'a convaincu de m'inscrire en DEA et qui a encadré mon stage de DEA. Il m'a présenté à Marc Schoenauer, qui allait devenir mon directeur de thèse, a collaboré à l'élaboration de mon sujet de thèse et a encadré mon travail au Laboratoire d'Optimisation Globale qu'il dirige à Toulouse, tout au long de ces années. Cela suffirait très largement à mériter toute ma reconnaissance, mais ce n'est pas tout : Jean-Marc Alliot m'a aussi et surtout soutenu dans des moments de doute, de désespoir même. Il sait ce que je lui dois, je ne l'oublierai pas.

Merci Jean-Marc.

Marc Schoenauer possède de nombreuses qualités, qui font de lui un excellent directeur de thèse. Il s'est de plus, avec moi, montré patient et compréhensif. Il m'a soutenu, aidé, supporté. Je l'en remercie chaleureusement.

Je remercie Jean-Marc Garot, qui était chef du Centre d'Études de la Navigation Aérienne au début de ma thèse et qui m'a encouragé à me diriger vers la recherche, ainsi que son successeur, Alain Printemps et son adjoint, Dominique Colin de Verdière, pour la confiance qu'il m'ont accordée et pour la liberté qu'ils m'ont laissée au long de ces années.

Parmi les membres du CENA je remercie aussi Philippe Kerlirzin, chef de la division RFM (j'en profite pour saluer toute la division) et Murielle Lasbleiz, qui ont facilité toutes mes démarches administratives. Philippe Kerlirzin m'a de plus apporté une aide précieuse en relisant très attentivement cette thèse.

Je tiens à remercier vivement MM. Joseph Noailles et Philippe Codognet, qui m'ont fait l'honneur d'être rapporteurs sur ces travaux, ainsi que MM. Yves Duthen et Jin-Kao Hao, qui ont accepté de faire partie du jury, aux côtés de Jean-Marc Alliot, Marc Schoenauer et Philippe Kerlirzin.

Pendant que je préparais ma thèse, j'ai partagé mon temps entre deux laboratoires : le Laboratoire d'optimisation Globale, du CENA et de l'ENAC, à Toulouse, et le Centre de Mathématiques Appliquées de l'École Polytechnique, à Palaiseau.

À Toulouse, j'ai été acceuilli chaleureusement au sein d'une équipe très soudée. Nicolas Durand m'a apporté, tout au long de ma thèse, une aide très importante. Je lui dois beaucoup. Je tiens à remercier aussi tous ceux qui ont su rendre agréables mes séjours toulousains : Sofiane Oussedik (qui, avec Hélène, est capable de rendre convivial un repas au Flunch, ce qui n'est pas rien), Isabelle Alliot ("Allo Isabelle ? - Salut Fred, ça va ? - Euh.. non.. - Bon, je te le passe.." mais aussi des soirées très agréables, des cocktails délicieux), Pascal Brisset (mon voisin de bureau toulousain, toujours prêt à me faire profiter de ses compétences en CAML), Yann Lefablec (merci pour le nez rouge), Stéphane Puechmorel, Daniel Delahaye, Géraud Granger (qui a toujours refusé de croire que j'avais des raisons de désespérer), Jean-François Bosc, Éric Blond, Nathalie Lenoir, Christophe Bontemps, Nicole Vaissière et tous ceux qui font du LOG et du Département Mathématiques et Informatique de l'ENAC un lieu de travail si agréable.

À Palaiseau, j'ai eu la chance de passer plusieurs années au CMAP. Trois directeurs successifs ont bien voulu m'y acceuillir et je les en remercie : MM. Raviart, Nédélec et Giovangigli. Ils ont su maintenir au CMAP ce mélange si précieux d'exigence scientifique et de bonne humeur, qui fait la qualité et le charme de ce laboratoire unique.

L'excellence des conditions de travail et de vie au CMAP est aussi due à l'équipe exceptionnelle des secrétaires : Geo Boléat (tu vois Geo, je t'obéis, je t'appelle Geo), Jeanne Bailleul, Liliane Doaré et Nathalie Limonta. Je voudrais remercier tous les membres du CMAP pour les années que j'ai passées parmi eux. Ils sont nombreux, je ne peux pas les nommer tous, mais tous contribuent à faire du CMAP ce qu'il est.

Je salue le professeur Neveu, qui a quitté le laboratoire. Sa gentillesse et sa simplicité n'ont pu que renforcer le respect que j'éprouvais pour lui depuis que j'avais suivi ses cours. Que sa retraite soit longue et heureuse.

Nombreux sont ceux qui n'ont pas mesuré leur temps pour m'apporter leur aide, notamment en s'occupant activement de l'informatique au CMAP : Pedro Alexandre Ferreira (Queria escrever uma frasinha em português para ti, Pedro, mas não posso mostrá-ta para a emendares, portanto desculpa as faltas, por favor : agradeço o meu amigo Pedro, que me ajudou a conhecer e a amar a língua e a cultura portuguesas), Aldjia Mazari, François Jouve, la moustache de François Jouve, Pascale Rich-Hennion (qui a partagé un bureau avec moi), Jérôme Kalifa (avec qui il est très agréable de discuter, vraiment).

Ceux qui connaissent Robert Brizzi comprendront le plaisir que j'éprouve à l'appeler ici mon ami. J'ai pu apprécier aussi de très agréables et très enrichissantes discussions avec d'autres chercheurs, à Palaiseau, notamment avec Michèle Sebag, Toufic Abboud au beau chapeau, Éric Bonnetier, Kamel Hamdache, François James, Habib Ammari, Sylvie Mas-Gallic, Josselin Garnier, Carl Graham, Brigitte Lucquin et Rama Cont, mon nouveau voisin de bureau.

De nombreux thésards ont fréquenté le CMAP en même temps que moi, certains l'ont déjà quitté, d'autres resteront. Je pense d'abord à ceux que j'y ai trouvés en arrivant : Couro Kane et Iléana Christodorescu bien sûr, mais aussi Vincent Mathis, Marc Massot, Komla Domelevo, Nabil Bedjaoui et mes premiers voisins de bureau, Mister Zhou et François Vialle-Béreux. D'autres ont commencé leur thèse en même temps que moi : Natacha Vialle, Florian Méhats, Alessandro Fadda, Mourad Laouadi, Yunzhi Zhang. Une pensée enfin pour les jeunes, à qui je souhaite bonne chance : Annalisa Ambroso (qui n'a rien perdu de son charme et de sa gentillesse en devenant Mme Biroli), Chiraz Latiri, Rita Benhaddouch, Maureen Clerc, les acteurs Erik Burman, Paul Seignourel et Kamal Abboud (au-delà d'eux et au-delà de Geo, Robert, Aldjia, Brigitte et François que j'ai déjà nommés, je salue l'ensemble de l'atelier théâtre), Christophe Bernard, Jonas Ribbe, Kais Ammari. Que ceux que j'ai oubliés me pardonnent.

J'ai voulu limiter ces pages de remerciements aux gens que j'ai connus dans un cadre *professionnel*. D'autres, ailleurs, mes parents, mes amis, m'ont entouré et aidé à travers les années que j'ai consacrées à cette thèse. Je pense à eux aussi.

# Table des matières

# Introduction

L'organisation du trafic aérien et de son contrôle doit répondre à un double objectif de sécurité et d'efficacité. Concernant la sécurité, un rôle important du contrôle aérien est d'assurer la séparation des avions : on dit que deux avions sont *séparés* quand la distance entre eux est suffisante pour exclure tout risque de collision. Si ce n'est pas le cas, on dit qu'ils sont en *conflit*.

On distingue différents types de contrôle aérien, dont le *contrôle d'approche* et le *contrôle en route*, correspondant à différentes étapes du vol d'un avion. Le contrôle d'approche s'applique dans les zones entourant les aéroports d'arrivée et de départ. Le contrôle en route s'applique entre ces zones. Tant qu'un avion est dans la zone entourant son aéroport de départ, et une fois qu'il a atteint celle qui entoure son aéroport de destination, il suit des cheminements bien déterminés et il est contrôlé, depuis le sol, par des contrôleurs. La densité du trafic autour des aéroports rend nécessaire le contrôle, pour assurer la sécurité des avions. Le fait que les avions suivent des cheminements pré-définis facilite la tâche des contrôleurs.

Sur la portion de sa trajectoire située entre ces deux zones entourant des aéroports, l'avion suit aussi des cheminements bien déterminés, le long de couloirs aériens, de balise en balise. Il est aussi contrôlé, du sol, par des contrôleurs. On pourrait croire alors qu'il n'y a aucune différence entre ces trois étapes du vol d'un avion, que l'espace aérien qu'il traverse est homogène. Il y a pourtant une différence de taille : en dehors des zones entourant les aéroports, et surtout dans l'espace aérien supérieur, la densité du trafic aérien est beaucoup plus faible qu'autour des aéroports. La probabilité d'une collision entre deux avions y est donc, elle aussi, très faible.

Il ne s'agit pas pour autant de laisser un avion voler dans ces zones sans tenir compte de la présence des autres avions. Bien qu'une collision soit peu probable, son éventualité ne pourrait pas alors être totalement exclue, et c'est un risque que l'on préférera ne pas courir. Cependant, la question peut se poser de savoir si l'organisation du trafic aérien et de son contrôle est adaptée à ces zones à faible densité de trafic, s'il y répond bien à son second objectif, l'efficacité. Sous sa forme actuelle, le contrôle *en route* est coûteux, du moins de l'avis des compagnies aériennes. Coûteux à plusieurs titres : il représente une part des redevances que paient les compagnies aériennes aux organismes en charge du contrôle dans les pays dont les avions traversent l'espace aérien ; le fait que les avions doivent suivre des couloirs aériens allonge leur route ; les manœuvres imposées par les contrôleurs entraînent des retards.

On envisage ainsi de modifier l'organisation du trafic aérien, dans les zones où il est le moins dense, en laissant suivre aux avions des routes directes, plus courtes que celles que leur imposent les couloirs aériens. Cependant, cela crée de nouveaux problèmes. La gestion du trafic par un contrôleur est une tâche très complexe, dont il parvient à s'acquitter en apprenant à apporter à un certain nombre de situations habituelles les réponses adaptées. C'est l'utilisation par les avions des balises et des couloirs aériens, qui permet au contrôleur de ramener ainsi la complexité de son travail à un niveau acceptable pour un agent humain[1]. Si les avions sont libres de suivre les routes qu'ils souhaitent suivre, ils ne se présenteront plus au contrôleur sous formes de motifs reconnus de lui. Même dans l'espace aérien supérieur, où la densité de trafic est plus faible, le passage aux routes directes risque de n'être pas possible dans le cadre du contrôle aérien tel qu'il est assuré aujourd'hui, ou du moins d'entraîner des problèmes de saturation, et donc une augmentation, au mieux, des retards, au pire, du nombre d'incidents.

L'automatisation, ou la délégation à bord, du contrôle aérien dans les zones où les avions auraient la liberté de choisir leurs routes (on parle alors de *free routes*), offre une alternative qui pourrait permettre de tirer vraiment les bénéfices de ces évolutions du trafic.

L'évolution des moyens techniques disponibles permet d'envisager des modifications profondes du contrôle aérien, à travers son automatisation et l'utilisation éventuelle de systèmes embarqués, grâce auxquels chaque avion serait responsable de la route qu'il suit, et du fait qu'il reste séparé des autres avions. La position d'un avion peut être connue très précisément grâce au positionnement par satellite (GPS : Global Positionning System). La trajectoire prévue peut-être suivie très fidèlement grâce au gestionnaire de vol (FMS : Flight Management System). Des progrès importants ont été fait également en matière de communication : l'ADS-B (Automatic Dependance Surveillance-Broadcast) permet d'envisager de nouveaux modes embarqués de surveillance de l'espace et de diffusion d'informations de vol.

On appelle *système d'évitement automatique*, tout système totalement ou partiellement automatisé permettant d'assurer que les avions restent toujours séparés. On distingue deux approches de tels systèmes : ils peuvent être *centralisés* (les trajectoires des avions étant modifiées de manière centralisée, afin qu'ils restent séparés au prix d'un retard minimal), ou *embarqués*, (la trajectoire de chaque avion étant modifiée au niveau de l'avion lui-même). Des projets d'automatisation du contrôle sont menés actuellement, au sein de différents organismes. Certains selon le principe d'un système d'évitement centralisé (ARC2000 en Europe, AERA aux États-Unis), d'autres selon le principe d'un système d'évitement embarqué (le projet européen FREER). Ces projets sont décrits dans la section 1.4.

Un système d'évitement automatique devra remplir deux fonctions principales : il devra assurer une pré-détection des conflits potentiels, et proposer aux avions impliqués dans ces conflits des trajectoires d'évitement. C'est cette génération de trajectoires sans conflit qui est l'objectif des méthodes présentées ici.

Après avoir exposé le contexte actuel du trafic aérien, ainsi que les évolutions prévisibles de ce contexte, et les différents projets d'automatisation du contrôle aérien existants (chapitre 1), nous présentons dans ce rapport plusieurs méthodes permettant de générer de manière automatique des trajectoires sans conflits. Ces méthodes se répartissent sur deux axes différents. Dans les chapitres 2 et 3, les trajectoires générées sont des trajectoires normalisées, pouvant être définies par certaines manœuvres pré-déterminées et par les instants auxquels ces manœuvres prennent place. Dans les chapitres 5 et 6, les trajectoires sont obtenues de manière réactive, c'est-à-dire au moyen d'une commande continue, réagissant à chaque instant à la situation.

[1]Ce travail, malgré cette réduction de complexité, reste difficile, les *situations habituelles* que nous avons évoquées sont nombreuses et diverses et pour être pleinement opérationnel sur une zone de l'espace aérien, un contrôleur doit y avoir été formé pendant plusieurs années.

Dans le chapitre 2, nous présentons une méthode de génération de trajectoire d'évitement, basée sur un algorithme de type *Branch and Bound* utilisant des méthodes d'intervalles. Les trajectoires générées sont des trajectoires par point tournant définies par des instants de manœuvres et des sens de déviation des avions. Ce sont ces instants de manœuvres et ces sens de déviation qui seront optimisés au moyen de méthodes d'intervalles. Deux versions différentes de cette méthode sont décrites et testées sur des cas-tests de conflit. Chacune de ces deux versions correspond à un des deux types de système d'évitement, centralisé ou embarqué. L'approche *globale* de cette méthode génère, pour un conflit impliquant $n$ avions, les $n$ trajectoires simultanément, en minimisant le retard entraîné par les déviations dues à l'évitement. Elle correspond à l'approche centralisée d'un système d'évitement. L'approche séquentielle consiste à générer successivement les $n$ trajectoires des avions, selon un ordre de priorité, les $i-1$ trajectoires déjà générées imposant des contraintes pour la génération de la trajectoire du $i$-ème avion. Cette approche peut être utilisée dans une forme *séquentielle* d'un système d'évitement embarqué. Les résultats obtenus sur des cas tests de conflit montrent que de telles méthodes permettent en effet de générer des trajectoires sans conflits, mais mettent aussi en évidence leurs limites : l'approche globale est très coûteuse en temps de calcul, et l'efficacité de l'approche séquentielle dépend très fortement de l'ordre dans lequel les trajectoires des avions sont générées. De plus, cette méthode, telle qu'elle est présentée dans le chapitre 2, ne peut pas s'appliquer pour générer tous les types de trajectoires d'évitement normalisées que l'on souhaiterait pouvoir générer, et surtout ne permet pas de prendre en compte des avions dont la vitesse n'est pas constante (c'est toujours le cas des avions en montée ou en descente).

La méthode de génération de trajectoires d'évitement présentée dans le chapitre 3 utilise un algorithme de recherche de chemin de coût minimal dans un arbre à coût additif, l'algorithme $A^*$, pour générer des trajectoires d'évitement. Les trajectoires d'évitement normalisées sont adaptées pour pouvoir être traduites en termes de choix successifs, ce qui permet de représenter l'ensemble des trajectoires possibles par un arbre. Les trajectoires générées sont des trajectoires d'évitement par point tournant avec retour direct vers la destination, mais, contrairement à celle introduite dans le chapitre 2, la méthode est très facilement adaptable à la génération d'autres types de trajectoires d'évitement, ainsi qu'à la prise en compte d'avions dont la vitesse n'est pas constante. Comme dans le chapitre 2, deux versions de cette méthode sont présentées, l'une correspondant à une approche globale et l'autre à une approche séquentielle de la génération des trajectoires. Les résultats obtenus sur des cas tests sont présentés. L'algorithme $A^*$ permet de générer des trajectoires de bonne qualité, tant en terme de retard que de respect des contraintes de séparation. L'approche globale est cependant très coûteuse en mémoire. L'approche séquentielle donne des résultats satisfaisants, avec toujours la forte influence de l'ordre dans lequel les trajectoires sont générées. Cela est dû au fait de générer ces trajectoires les unes après les autres, et non à l'algorithme utilisé pour la génération des trajectoires. Facilement adaptable à la génération de trajectoires diverses et à des vitesses variables, la génération de trajectoires d'évitement au moyen de l'algorithme $A^*$ peut être utilisée dans le cadre d'un étude plus réaliste de systèmes d'évitement embarqués. Un bref résumé de travaux, faisant suite à ceux qui sont exposés ici, et étudiant une telle adaptation de ces méthodes, est donné à la fin du chapitre 3.

Les méthodes présentées dans la suite de ce rapport sont foncièrement différentes. En effet, leur but est de générer les trajectoires des avions de manière réactive. À tout instant le cap de chaque avion est modifié en tenant compte de sa situation et de sa connaissance des situations des avions risquant d'entrer en conflit avec lui. Il s'agit donc d'identifier une fonction, donnant, à partir de ces données, la valeur de la modification de cap la plus efficace. Il faut de plus que le système soit utilisable sur l'ensemble des cas de conflits pouvant se présenter. Les réseaux de neurones, qui sont des approximateurs universels et qui permettent en général l'obtention de bonnes capacités de généralisation, sont utilisés pour calculer cette modification de cap. On ne dispose cependant pas de couples *entrée / sortie* permettant d'opérer une régression et de mener l'apprentissage des réseaux de neurones au moyen de méthodes classiques telles que la *rétro-propagation du gradient*. Cet apprentissage est donc réalisé au moyen d'un algorithme génétique, un algorithme d'optimisation globale stochastique présenté dans le chapitre 4.

Les réseaux de neurones ont réalisé leur apprentissage sur des configurations de conflit formant une *base d'apprentissage*. Le problème principal de cette approche est d'obtenir des réseaux de neurones présentant de bonnes capacités de généralisation, c'est-à-dire générant des trajectoires de bonne qualité sur des configurations n'appartenant pas à leur base d'apprentissage. Dans ce but, trois modes d'apprentissage sont utilisés et comparés. Les deux premiers utilisent un algorithme génétique (une présentation générale de ce type d'algorithme est donnée dans le chapitre 4) et sont présentés dans le chapitre 5. L'un consiste à utiliser une base d'apprentissage fixe, constituée d'un certain nombre de configurations de conflit, qui demeurent les mêmes au cours de l'apprentissage. L'autre mode d'apprentissage présenté dans le chapitre 5 consiste à renouveler aléatoirement les configurations de conflit présentes dans la base d'apprentissage, et utilise, pour évaluer la qualité des réseaux de neurones des notions de *fiabilité* et de *mémoire*. Ces deux premiers modes d'apprentissage sont testés et comparés sur des conflits impliquant 2 avions, puis sur des conflits impliquant 3 avions.

Le troisième mode d'apprentissage est présenté dans le chapitre 6. Il est basé sur le principe de la coévolution. Il s'agit de faire évoluer, au moyen d'un algorithme génétique, non seulement des réseaux de neurones, mais aussi des configurations de conflit sur lesquelles sont évalués ces réseaux de neurones. Le chapitre 6 présente une première approche de cette méthode, appliquée à des conflits impliquant 2 avions. Les résultats obtenus sont exposés et discutés. Des trois modes d'apprentissage testés dans les chapitres 5 et 6 (base d'apprentissage fixe, base d'apprentissage renouvelée, coévolution), celui qui permet d'obtenir les meilleurs résultats reste l'apprentissage sur une base d'apprentissage renouvelée, avec utilisation des notions de fiabilité et de mémoire.

À partir des résultats obtenus avec les différentes méthodes présentées ici, nous concluons que les approches globales des méthodes d'intervalles et de l'algorithme $A^*$ ne sont pas adaptées à la génération de trajectoires sans conflit dans le cadre d'un système d'évitement (problèmes de temps de calcul pour les méthodes d'intervalles, d'encombrement mémoire pour $A^*$). En revanche, leurs approches séquentielles sont utilisables dans le cadre d'un système d'évitement embarqué utilisant un ordre de priorité entre les avions. La plus grande souplesse de l'algorithme $A^*$ le rend bien plus intéressant que les méthodes d'intervalles. La génération de trajectoires d'évitement, de manière réactive, par des réseaux de neurones, pourrait être utilisée par un système d'évitement réactif. À ce titre, les résultats obtenus sont très encourageants, mais d'importantes recherches seraient encore nécessaires pour obtenir, par cette méthode, des résultats pleinement satisfaisants, en termes de fiabilité, sur des conflits impliquant plus de deux avions.

Les travaux présentés dans ce rapport ont été effectués au Centre de Mathématiques Appliquées de l'Ecole Polytechnique (Palaiseau) et au Laboratoire d'Optimisation Globale de l'Ecole Nationale de l'Aviation Civile et du Centre d'Etudes de la Navigation Aérienne (Toulouse).

# Chapitre 1

# L'évitement aérien

## 1.1 Définitions

Le contenu de ce rapport pourrait se décrire de la manière suivante : les différentes méthodes présentées ici visent à assurer la *séparation* des aéronefs, dans les *zones de l'espace aérien* correspondant actuellement au contrôle *en route*, que ces avions utilisent des *couloirs aériens* ou suivent des *routes directes*. Ainsi, quand un *conflit* est susceptible d'avoir lieu, entre deux avions, ou même plus (il s'agit alors en fait d'un *cluster*), elles doivent fournir pour ces avions des trajectoires sans conflit. Elles peuvent être vues comme des étapes préliminaires vers d'éventuels futurs *systèmes d'évitement automatiques*. Ces systèmes d'évitement pourront être *centralisés*, ou *embarqués*.

Les mots en italique dans ce qui précède appellent des définitions.

**Définition 1.1 – Zones de l'espace aérien –** *L'espace aérien est divisé en* zones *différentes qui correspondent à différentes utilisations possibles de cet espace. Elles ont une étendue spatiale bien définie, sont réparties en différentes* classes *selon les règlementations qui s'y appliquent. Certaines de ces zones entourent les aéroports, d'autres sont réservées aux appareils militaires, ou sont interdites à tous les aéronefs, pour des raisons de sécurité, ou de respect de l'environnement.*

**Définition 1.2 – Contrôle en route –** *Le* contrôle en route *prend place à l'extérieur des zones entourant les aéroports. Ce contrôle est actuellement effectué depuis des* centres de contrôle*, au sol. L'espace aérien correspondant est divisé en* secteurs de contrôle*. Une équipe de contrôleurs (généralement composée de deux contrôleurs) assure le contrôle aérien à l'intérieur de chaque secteur.*

Dans le cadre de l'organisation actuelle du trafic aérien, les avions, entre la zone entourant leur aéroport de départ et celle entourant leur aéroport de destination, ne suivent pas des *routes directes*, mais empruntent des *couloirs aériens*.

**Définition 1.3 – Couloirs aériens –** *Les* couloirs aériens*, souvent désignés par le terme anglais* airways*, sont des tubes de section rectangulaire qui entourent des segments de droites joignant entre elles des* balises*. La route que suit un avion est déterminée par les balises successives par lesquelles elle passe. Le* plan de vol *d'un avion contient la liste de ces balises.*

**Définition 1.4 – Routes directes –** *La* route directe *d'un avion est le chemin le plus court, ou le plus avantageux pour l'avion (compte tenu des vents, par exemple) entre le point de sortie de la zone qui entoure l'aéroport de départ et le point d'entrée de celle qui entoure l'aéroport d'arrivée.*

On envisage, dans le cadre des évolutions envisageables de l'organisation du trafic aérien de laisser les avions, dans certaines zones de l'espace aérien (celles ou le trafic est moins dense), libres de choisir leur route. Ces routes choisies par les avions sont appelées *free routes*[1]. Les avions pourront alors, s'ils le souhaitent (c'est-à-dire si les compagnies aériennes le souhaitent), emprunter des routes directes (définition 1.4).

La sectorisation de l'espace aérien, une fois les couloirs aériens fixés, est un problème difficile et intéressant. En effet, de nombreuses contraintes existent : la charge de travail correspondant à chaque secteur doit rester raisonnable pour que la sécurité soit assurée, la géométrie des secteurs et celle des couloirs aériens doit tenir compte des besoins de coordination entre contrôleurs au moment du passage d'un avion d'un réseau à l'autre. On trouvera dans [Del95] des méthodes permettant l'optimisation des secteurs de contrôles tenant compte de ces contraintes. L'optimisation de l'utilisation des couloirs aériens, c'est-à-dire le choix *a priori* de routes suivant différents couloirs aériens entre un aéroport de départ et un aéroport de destination, peut être utilisée, pour réduire la congestion de l'espace aérien [OD98].

C'est souvent aux environs des balises, points de passage obligés de nombreux avions, qu'apparaissent les conflits.

**Définition 1.5 – Séparation –** *Une distance horizontale, exprimée en milles nautiques (NM), appelée* norme de séparation horizontale*, et une distance verticale, exprimée en pieds (ft), appelée* norme de séparation verticale*, sont définies. On dit que deux avions sont* séparés *quand la distance entre leurs projection sur un plan horizontal est supérieure à la norme de séparation horizontale OU quand la différence entre leurs altitudes est supérieure à la norme de séparation verticale.*

Les valeurs de ces normes de séparation varient selon la classe de la zone de l'espace aérien dans laquelle on se place. Dans les méthodes présentées dans la suite de ce rapport, nous n'utiliserons que la norme de séparation horizontale, puisque nous supposerons les avions volent dans un plan horizontal. Dans le cas du trafic *en route* cette norme dépend de plusieurs facteurs, mais est souvent proche de 5 milles nautiques.

**Définition 1.6 – Conflit élémentaire –** *On dit que deux avions sont en* conflit *dès qu'ils ne sont plus* séparés *(définition 1.5). Pour une durée $t_f$ fixée, on dira que deux avions sont en conflit* potentiel *si la probabilité qu'un conflit ait lieu entre eux à l'intérieur de cette durée est non nulle. La* résolution de conflit *consiste à modifier les trajectoires des avions de manière à supprimer tout conflit potentiel entre eux.*

**Définition 1.7 – Cluster –** *Un* cluster *est la fermeture transitive d'avions en conflit potentiel. Si un conflit potentiel (pour un horizon temporel $t_f$ donné) existe entre un avion $a_1$ et un avion $a_2$, et que cet avion $a_2$ est lui même impliqué dans des conflits potentiels avec les avions $a_3$ et $a_4$,*

[1] Il semble malheureusement qu'aucun terme français correspondant ne se soit imposé

*alors les avions $a_1$, $a_2$, $a_3$ et $a_4$ forment un cluster. Dans toute la suite, pour plus de simplicité, on appellera* conflit à $n$ avions *un cluster contenant $n$ avions.*

Toute tentative de résolution de conflit (automatique ou non) doit tenir compte de cette notion de *clusters*. En effet, en reprenant l'exemple donné dans la définition 1.7, la résolution séparée du conflit entre $a_1$ et $a_2$ serait inutile, puisque la résolution du conflit entre $a_2$ et $a_3$, en modifiant la trajectoire de l'avion $a_2$ risquerait de faire apparaître de nouveau un conflit entre $a_1$ et $a_2$.

Les méthodes présentées ici ont pour but d'assurer une telle *résolution de conflits*, c'est-à-dire de fournir des trajectoires sans conflit à des avions impliqués dans des conflits potentiels. Dans le cadre d'une application au trafic réel de telles méthodes, la *pré-détection de conflit* jouerait un rôle très important. Par pré-détection des conflits, on entend une évaluation de la probabilité que deux avions soient en conflit à l'intérieur d'une durée $t_f$.

**Définition 1.8 – Système d'évitement automatique –** *On appellera* système d'évitement automatique *tout système visant à automatiser la résolution de conflit, que cette automatisation soit partielle (considérée comme une aide au contrôle) ou totale.*

Les définitions qui suivent correspondent à différentes approches de l'évitement aérien. Les différences entre ces approches sont réelles et généralement distinguées. Cependant les termes qui les désignent ne sont pas figés. On pourra trouver ailleurs les termes que nous employons ici (*évitement centralisé, supervisé, embarqué, distribué, réactif*) utilisés pour désignés des notions différentes, ou les notions que nous désignons par ces termes désignées par d'autres termes. Les définitions que nous donnons ici précisent le sens qui devra être donné à ces termes *dans la suite de ce rapport*.

**Définition 1.9 – Système d'évitement centralisé –** *On dira qu'un système d'évitement est* centralisé*, ou* supervisé*, quand les trajectoires sans conflit de tous les avions d'une zone donnée de l'espace aérien sont générées par une entité unique (que cette génération soit automatisée, ou assurée par un centre de contrôle, comme c'est le cas actuellement). Ces trajectoires sans conflit sont alors imposées aux avions.*

**Définition 1.10 – Système d'évitement embarqué –** *On dira qu'un système d'évitement est* embarqué *(ou éventuellement* distribué*) lorsque la trajectoire d'évitement de chaque avion est générée individuellement (dans le cadre d'une application d'une telle méthode, elle serait le plus souvent générée à bord de l'avion lui-même, d'où le terme* embarqué*). On parle d'évitement* réactif *quand les actions d'évitement sont le résultat d'une commande continue, ou de commandes successives mais très rapprochées dans le temps (comme c'est le cas pour la méthode présentée dans le chapitre 5).*

## 1.2 Contexte de l'évitement aérien

Les méthodes présentées ici ont comme point commun de viser à résoudre des conflits pouvant survenir dans une fenêtre temporelle allant de 5 à 20 minutes, et de maintenir les avions séparés sur une vingtaine de minutes. C'est un horizon temporel comparable à celui auquel est actuellement réalisé le *contrôle tactique* dont nous allons indiquer la place dans le cadre actuel du contrôle aérien.

Le contrôle est actuellement assuré, depuis le sol, par des contrôleurs aériens. Nous ne nous intéresserons ici qu'au trafic en route. L'espace aérien dans les zones réservées à ce type de contrôle est subdivisé en secteurs de contrôle. À l'intérieur de chacun de ceux-ci, des contrôleurs maintiennent les avions séparés les uns des autres, indiquent s'il le faut aux pilotes des trajectoires d'évitement que ceux-ci doivent suivre.

Mais la séparation des avions déjà en vol n'est pas l'unique rôle du contrôle aérien. Celui-ci fonctionne à travers plusieurs *filtres* successifs, correspondant chacun à des horizons temporels différents. On distingue généralement 5 niveaux différents :

1. **Organisation à long terme :** Elle est assurée par des décisions concernant l'organisation ou l'utilisation de l'espace aérien, de l'orientation générale du trafic. Il s'agit d'organiser le trafic, de façon macroscopique, à long et moyen terme (délai supérieur à six mois).
2. **Pré-régulation, ou organisation à court terme :** Il s'agit de l'organisation du trafic du lendemain, ou du surlendemain. En effet, on connaît alors :
   - Les effectifs qui seront présents dans les différents centres de contrôle et ainsi la capacité de ceux-ci.
   - les plans de vol déjà connus (c'est-à-dire les heures de départ et les routes, en termes de balises successives et de niveaux de vol[2]).
   - Les données décrivant le trafic des semaines et des années précédentes, permettant de prévoir l'importance du trafic.

   C'est dans le cadre de ce filtre que sont gérés les créneaux de décollage, c'est-à-dire des fenêtres temporelles affectées au décollage des différents vols. Cette pré-régulation est, depuis 1995, assurée au niveau européen.
3. **Régulation en temps réel :** Il s'agit de réguler les différents flux selon les événements de la journée. Ainsi, si le trafic transatlantique (correspondant à des arrivées prévisibles plusieurs heures à l'avance) n'est pas conforme dans certains centres aux prévisions sur lesquels se basait la pré-régulation, on peut modifier l'organisation du trafic non transatlantique. Ce filtre permet aussi de prendre en compte la météo, ou les retards de certains appareils.
4. **Contrôle tactique :** C'est la partie du contrôle qui est effectuée par le contrôleur aérien dans son secteur. Le temps de présence d'un avion dans un secteur est de l'ordre de q15 minutes. Les contrôleurs disposent des plans de vol des avions quelques minutes avant l'entrée de ceux-ci dans leur secteur. La détection des conflits et leur résolution n'est pas automatisée. Les contrôleurs se basent sur leur expérience pour repérer les conflits potentiels et donner aux avions des trajectoires d'évitement. Ces trajectoires sont décrites par quatre types d'ordre :
   - Des modifications de cap.
   - Des paliers intermédiaires pour les avions en montée et en descente.
   - Des changements de niveaux de vol.
   - Des modifications des vitesses des avions (surtout pour des avions en descente).

[2]un niveau de vol correspond à une altitude donnée.

La capacité des secteurs est limitée. L'augmentation du trafic risque d'entraîner la saturation de certains secteurs de contrôle. Augmenter le nombre des contrôleurs dans chaque secteur ne résoudrait pas le problème, la coordination des actions des différents contrôleurs d'un même secteur deviendrait vite difficile. Diminuer la taille des secteurs entraînerait une augmentation de la charge de travail due à la communication entre contrôleurs au moment du passage d'un avion d'un secteur à l'autre.

Par leurs horizons temporels, les méthodes présentées ici s'apparentent donc au contrôle tactique. Cependant de telles méthodes de génération de trajectoires sans conflit sont envisageables dans le cadre d'une organisation différente du trafic aérien (voir section 1.3), décrites dans la section suivante. C'est la mise en place dans certaines zones, des *free routes* et du *free flight* qui justifierait et rendrait possible l'emploi de systèmes d'évitement automatisés. Cela nécessiterait bien entendu d'apporter des modification aux autres *filtres* décrits ici. De telles modifications sont envisagées dans [Eur97].

**5. Évitement d'urgence :** C'est le filtre ultime du contrôle aérien. Il n'intervient qu'en cas de défaillance ou d'échec des filtres précédents. Il est actuellement assuré par deux dispositifs :

- Le *filet de sauvegarde* prédit les trajectoires des avions à court terme (quelques minutes) à l'aide d'algorithmes de poursuite utilisant les positions radar. Si un conflit est repéré, une alarme se déclenche. Aucune solution n'est cependant proposée.

- Le TCAS (Trafic alert and Collision Avoidance System) est un système anti-collision embarqué, obligatoire aux États Unis sur tous les avions transportant plus de 30 passagers. Une prédiction de conflit[3] est assurée à très court terme (25 à 40 secondes). Le système donne directement un ordre de résolution au pilote, le délai avant une éventuelle collision étant trop court pour qu'un contrôleur ait le temps d'analyser la situation et de transmettre une solution au pilote. Les consignes proposées consistent à modifier le niveau de vol des avions impliqués. Un des deux avions, prioritaire[4], choisit d'abord sa manœuvre, l'autre avion agit en fonction de cette manœuvre.

## 1.3 Cadres prévisionnels de l'évolution du contrôle du trafic aérien

L'organisation du trafic aérien et de son contrôle telle qu'elle est présentée dans la section 1.2 est appelée à évoluer. En effet, l'augmentation du trafic, l'évolution de sa répartition nécessitent une réorganisation de l'utilisation de l'espace aérien. De plus, les compagnies aérienne demandent la possibilité d'utiliser des routes directes (définition 1.4), ce qui pourrait nécessiter un transfert de responsabilités du contrôleur vers le pilote. Un tel transfert entraînerait en effet une diminution du coût du contrôle aérien, supporté par les compagnies, et permettrait éventuellement aux avions, en choisissant leur route, de supprimer des détours imposés par les contrôleurs. Enfin, de nouvelles technologies (dans le domaine des transmissions, des mesures des positions et des vitesses, de l'informatisation des avions) permettent d'envisager l'utilisation de nouvelles techniques.

Parmi les nouveaux concepts avancés pour l'organisation future du trafic aérien, on trouve le

[3]Il ne s'agit plus là de conflit au sens de la définition 1.6, mais presque de collision.

[4]L'ordre de priorité est donné par le numéro CAUTRA des avions, un numéro (arbitraire) affecté à tout avion contrôlé

*free flight*[5], que nous allons présenter dans cette section. En effet, ce concept de *free flight* peut impliquer l'utilisation de systèmes d'évitement embarqués, auxquelles correspondent certaines des méthodes présentées dans ce rapport.

On distingue deux catégories de vols, les vols *IFR* et les vols *VFR*.

Le sigle IFR signifie en anglais *Instrument Flight Rule* (vol aux instruments). Les avions, lorsqu'ils volent en IFR, utilisent des instruments de bord pour connaître leur position, et sont obligatoirement contrôlés. Les vols commerciaux sont des vols IFR. L'organisation du contrôle aérien décrite dans la section 1.2 concerne les avions volant en IFR.

Le sigle VFR signifie *Visual Flight Rule* (vol à vue). Les avions volant en VFR doivent assurer eux-mêmes, à vue, qu'ils restent séparés des autres avions. En dehors des zones entourant les aéroports, ils ne sont pas contrôlés. Les vols VFR sont, bien entendu, impossibles par mauvaises conditions de visibilité. Le contrôle aérien confère un plus haut niveau de sécurité aux vols IFR qu'aux vols VFR. Cependant, les vols VFR permettent une plus grande flexibilité, et laissent aux pilotes une plus grande latitude dans leurs choix.

Le principe général du *free flight* est d'allier à la fiabilité des vols IFR la flexibilité des vols VFR.

Dans un rapport de la RTCA[6] [RTC95], on en trouve la définition suivante :

**Définition 1.11 – free flight –** *A safe and efficient flight operating capability under instrument flight rule (IFR) in which the operators have the freedom to select their path and speed in real time. Air traffic restrictions are only imposed to ensure separation, to preclude exceeding airport capacity, to prevent unauthorized flight through special use airspace, and to ensure safety of flight. Restrictions are limited in extent and duration to correct the identified problem. Any activity which removes restrictions represents a move towards free flight.*

**– free flight –** *Une modalité de vol sûre et efficace sous les règles de vol aux instruments (IFR), dans laquelle les usagers sont libres de choisir leur route et leur vitesse en temps réel. Des restrictions ne sont imposées au trafic aérien que dans le but de prévenir le dépassement des capacités des aéroports, d'empêcher tout vol non autorisé dans un espace aérien à accès réglementé, et d'assurer la sécurité des vols. Les restrictions sont limitées en importance et en durée à ce qui est nécessaire pour corriger le problème qui a été identifié. Toute activité menant à la suppression de restrictions est un pas en direction du free flight.*

Cette définition du *free flight*, réaliste, n'exclut pas le recours à des systèmes d'évitement supervisés. Elle laisse la responsabilité d'assurer la séparation aux contrôleurs, en tout cas tant que les systèmes de séparations embarqués n'auront pas fait la preuve de leur efficacité et de leur fiabilité. De tels systèmes, cependant, mèneraient à la *suppression* de certaines des *restrictions* ci-dessus, et donc constituent *un pas en direction du free flight*.

La mise en application de ce principe entraînerait, par exemple, la possibilité pour les compagnies de choisir leur route, donc notamment, de faire suivre à leurs avions des routes directes

[5] Il semble malheureusement que ce terme anglais soit le seul utilisé pour désigner ce concept, même dans les milieux francophones.

[6] la RTCA est une agence non gouvernementale américaine, dont le but est d'aider au développement de nouvelles technologies et de nouveaux systèmes d'organisation destinés au transport aérien.

(définition 1.4). C'est ce qu'on appelle le *free routing*. Il permettrait aux avions en vol de modifier leur route ou leur niveau de vol de leur propre initiative. Un changement des conditions météorologiques peut par exemple rendre intéressante de telles modifications.

Eurocontrol adopte une définition plus radicale du *free flight* [Eur97] :

**Définition 1.12 – free flight –** *Freedom for the users to exercise the responsability of separation from other traffic and to effect any trajectory change in any dimension.*

**– free flight –** *La liberté pour les agents d'exercer la responsabilité de leur séparation avec le reste du trafic aérien, et d'opérer tout changement de trajectoire, aussi bien dans le plan horizontal que dans le plan vertical.*

Dans [Eur97] sont définies des zones de l'espace aérien, appelées FFAS[7], où s'appliquera ce concept de *free flight*. Le transfert de la responsabilité de la séparation du contrôleur vers le pilote est prévu, dans ces zones, à l'horizon 2015. Les FFAS seraient réservées aux avions *suffisamment équipés* pour endosser cette responsabilité. L'accès des appareils moins équipés aux FFAS serait soumis à une autorisation préalable. Le FFAS comprendra l'espace aérien dans lequel le trafic est le moins dense (espace aérien supérieur[8]), situé au dessus de zones (où le trafic restera contrôlé, ou espace aérien situé au dessus des océans).

Il est à noter cependant que cette conception radicale du *free flight* ne sera envisageable que quand les systèmes d'évitement embarqués disponibles auront fait la preuve de leur robustesse et de leur fiabilité. La date de 2015 (bien proche, compte tenu du temps que nécessitera la mise au point, et surtout la validation et la certification de tels systèmes) n'est qu'indicative, bien entendu.

## 1.4 Projets d'automatisation existant

### 1.4.1 Le projet américain AERA

Le projet américain AERA[9], mené par la MITRE, a été financé par la FAA[10].

La première étape du projet, AERA 1, visait à prévoir les trajectoires des avions, à assurer une pré-détection d'éventuels problèmes, tels que des conflits potentiel, mais ne contenait pas de système permettant la résolution des conflits.

La deuxième étape, AERA 2 [Cel90, SPSS83], fournissait une aide au contrôle, en proposant des listes de *résolutions recommandées par l'ordinateur*, mais laissait au contrôleur la responsabilité d'assurer la séparation des avions.

La troisième phase, AERA 3 [Nie89b, PA91], se compose de filtres hiérarchiquement organisés. Schématiquement, le but de chacun de ces filtres est de simplifier la situation, pour permettre au filtre suivant d'opérer, jusqu'au dernier de ces filtres, qui réalise la résolution de conflits

[7]Free Flight Air Space.
[8]Par exemple au-dessus du niveau 320, correspondant à 32 000 pieds, environ 10 000 mètres.
[9]Automated En-Route Air Traffic Control.
[10]Federal Aviation Administration.

impliquant deux avions :

1. Au niveau national, l'ATMS (Air Trafic Management System) garde son rôle actuel de gestion des flux (pré-régulation et régulation). Son but est de rendre ces flux compatibles avec les filtres suivants.
2. L'AMPF (Airspace Manager Planning Function) a un horizon temporel de 90 minutes. Il n'est pas prévu dans le projet AERA 3 de l'automatiser. Son but est de gérer les flux de manière locale, en évitant les surcharges de trafic dans certaines zones.
3. Le MOM (Manoeuver Option Management) [Nie89a] a un temps d'anticipation de 5 à 30 minutes. Il vise à détecter les conflits potentiels (définition 1.6) et à simplifier les clusters (définition 1.7), c'est-à-dire à modifier les trajectoires des avions de manière à ce qu'il n'y ait plus que des conflits élémentaires (n'impliquant que deux avions). L'automatisation complète de ce filtre est envisagée, mais n'est pas décrite en détail dans les documents disponibles sur le projet AERA 3.
4. L'ASF (Automated Separation Function), est un système d'évitement automatisé. Il utilise un algorithme baptisé *Gentle Strict*. Il cherche à assurer la séparation des avions en maintenant un des deux avions sur sa trajectoire (*strict*), et en donnant à l'autre (*gentle*), une trajectoire d'évitement par *offset* (c'est à dire en le décalant, en lui faisant suivre, pendant un certain temps, une trajectoire parallèle à sa trajectoire d'origine).

   Le problème à résoudre se subdivise en 4 cas (quel avion dévier, et dans quel sens ?) se ramenant chacun à un problème linéaire. Les résultats obtenus, bons en général, deviennent mauvais pour des valeurs petites (inférieures à $0,85$) du rapport entre les vitesses des avions, et des angles faibles (inférieurs à $45°$) entre les trajectoires des avions.

Le projet AERA 3 semble être aujourd'hui arrêté. Le principe de son organisation en filtres hiérarchisés et les modélisations des trajectoires d'évitement qu'il propose demeurent cependant intéressants.

### 1.4.2 Le projet européen ARC 2000

Le projet ARC 2000 a été mené par Eurocontrol, organisme européen en charge de la recherche et du développement de nouveaux systèmes de contrôle aérien.

Le projet a connu d'importantes évolutions [$\text{K}^+$89, FMT93]. Un point commun à toutes ses versions successives est l'utilisation du concept de *tubes 4D*. La trajectoire de chaque avion est représentée par une courbe de $\mathbb{R}^4$ (les trois coordonnées d'espace plus le temps), qui elle-même donne un *tube* en tenant compte des imprécisions de mesure des positions et de tenue des trajectoires. Dans les premières versions, la trajectoire du premier avion entrant dans le système était fixe et, en cas de conflit prévu un algorithme d'optimisation locale (type *gradient*) modifiait la trajectoire des nouveaux venus pour supprimer les conflits en minimisant les retards.

Ce principe, consistant à générer les trajectoires des avions les unes après les autres, est proche du principe des versions séquentielles des méthodes que nous décrivons dans les chapitres 2 et 3, même si la mise en œuvre de la version séquentielle de la méthode d'évitement utilisant l'algorithme $A^*$ sur des cas de conflits issus du trafic réel apporte une modification importante de ce principe (voir sections 2.8.1 et 3.9).

Les résultats obtenus souffrent des mêmes inconvénients que ceux que nous décrivons dans les sections 2.9.3 et 3.7.2 : non optimalité globale de la solution obtenue, et surtout, influence très forte de l'ordre dans lequel les trajectoires des avions sont générées. Ces défauts étaient encore accentués dans les premières versions du projet par le fait que les trajectoires des avions devaient être générées de l'origine à la destination de manière à ce qu'il n'y ait pas conflit, et ensuite modifiées en cas de non respect par les avions des trajectoires prévues.

Les dernières version du projet semblent avoir renoncé à cette génération de trajectoires entières sans conflit, pour se diriger vers la génération pour les avions de trajectoires sans conflit pour une durée de 20 à 30 minutes seulement.

ARC 2000 est le seul des projets décrits dans ce chapitre qui ait été (partiellement) testé sur le trafic réel. Il a de plus permis de développer des outils d'aide à la décision, tels que HIPS (Highly Interactive Problem Solver) [MG94] qui, sans chercher à résoudre les conflits de manière automatique, apporte aux contrôleurs une aide à la représentation des trajectoires et des conflits.

### 1.4.3 Le projet SAINTEX

Le projet SAINTEX [AL92], mené au sein du CENA[11], utilise conjointement deux approches différentes : d'une part une approche de type "système expert" pour la détection et la résolution des conflits, ceux-ci, une fois détectés, étant résolus par l'application de règles empiriques préétablies par des contrôleurs, et d'autre part, une méthode de génération automatisée de trajectoires sans conflit. Dans ce but, le système SAINTEX génère pour chaque avion une trajectoire optimale, qu'il modifie suivant certaines règles, si un conflit est détecté. Le système génère ainsi des trajectoires, et les teste pendant une durée déterminée (une minute), à l'issue de laquelle la *meilleure* trajectoire sans conflit est retenue pour chaque avion. Ce système, destiné à permettre une semi-automatisation du contrôle en route de nuit (le trafic nocturne étant à la fois moins dense et moins contraint que le trafic diurne), offre l'avantage de fournir des trajectoires pour chaque avion en un temps suffisamment court pour permettre son utilisation pour la résolution de conflits réels, mais ne permet pas d'atteindre un véritable optimum global.

### 1.4.4 Techniques d'évitement réactif

Karim Zeghal [Zeg93, Zeg94] a une approche originale du problème, aboutissant à une solution distribuée du problème de la séparation des avions en route. Le principe de sa méthode est l'application aux avions de *forces*. Cette approche offre l'avantage de ne nécessiter ni communication ni négociation entre les avions, et d'être très robuste aux perturbations. Elle laisse en revanche de côté l'optimisation des trajectoires et implique de pouvoir modifier à tout instant la direction de chaque avion. De plus, elle supporte mal une augmentation de la densité de trafic [Bos97].

Les *forces* utilisées sont de trois types :

- Les forces attractives. Elles sont dirigées vers la destination des avions.
- Les forces répulsives. Elles sont définies à partir d'un potentiel de danger, causé par un autre avion ou une zone interdite. Dans le cas d'un autre avion, par exemple, le potentiel de

[11] Centre d'Etude de la Navigation Aérienne.

danger pourra être inversement proportionnel à la distance. Les forces répulsives, perpendiculaires aux lignes équipotentielles, dirigées vers les potentiels décroissant, entraînent un éloignement rapide des obstacles proches.

- Les forces de glissement sont tangentes à l'équipotentielle passant par l'avion, deux directions sont possibles : on choisit celle qui contribue à rapprocher l'avion de sa destination. Ces forces de glissement permettent le contournement des obstacles. Il est possible de définir ces forces de manière à ce que deux avions en conflit potentiel l'un avec l'autre aient des forces de glissement complémentaires. Cela introduit une coordination d'action sans concertation et permet d'obtenir des trajectoires d'évitement assez efficaces.

À partir des forces s'appliquant à un avion, sa trajectoire est modifiée en suivant la direction donnée par une combinaison de ces forces (pour un conflit impliquant deux avions, Karim Zeghal propose d'additionner simplement ces forces, pour un conflit plus complexe, le dosage de ces forces peut être un problème délicat).

Depuis 1994, ces idées sont développées au sein du CENA, selon trois axes :

**Système purement embarqué :** La mise en œuvre de cette méthode ne nécessite pour chaque avion que des données locales. Cela rend possible son utilisation dans le cadre d'un système purement embarqué (voir définition 1.10). L'avantage de cette méthode serait alors sa robustesse : la panne d'un seul avion ne remettrait pas en cause le système dans sa globalité. Les conflits seraient détectés et résolus avec un horizon temporel de 4 à 10 minutes. Il serait sans doute nécessaire de mettre en place, comme dans le projet AERA (section 1.4.1), un filtre supérieur pour éviter la saturation de la méthode.

**Aide aux contrôleurs :** Les trajectoires calculées ne seraient pas transmise directement aux pilotes, mais proposées, comme aide à la décision, aux contrôleurs.

**Système intermédiaire :** Les trajectoires proposées sont suivies directement par certains avions, celles des autres avions sont proposées aux contrôleurs. La charge de travail des contrôleurs est réduite.

Les travaux de Karim Zeghal, très intéressants par l'originalité de la méthode proposée, ont comme atout majeur leur robustesse. Des inconvénients subsistent encore :

- Les trajectoires obtenues, dans lesquels le cap des avions varie constamment, sont difficiles à suivre pour les pilotes (et seraient pénibles pour les passagers). Une simplification des trajectoires obtenues serait nécessaire.
- La généralisation de la méthode à des conflits impliquant plus de deux avions, dans le cadre d'une augmentation de la densité du trafic, a été étudiée par J.-F. Bosc [Bos97]. Les résultats obtenus montrent que les limites du système sont vite atteintes.

### 1.4.5 FREER

Le projet FREER [DHN97] est mené au Centre Expérimental d'Eurocontrol à Brétigny. Il vise à développer un système permettant aux avions de voler de manière autonome, dans les zones à faible densité.

Ce projet répond à la définition qu'Eurocontrol donne du *free flight* (voir section 1.2).

Les avions choisissent leurs trajectoires d'évitement selon un ordre obtenu à partir d'extension des règles de l'air (telles que la priorité à droite). Les règles de l'air en elles mêmes ne permettent pas de définir d'ordre de priorité entre avions utilisables en pratique (elles ne sont pas transitives, par exemple). Les règles de l'air étendues développées dans le cadre du projet FREER devraient permettre de définir un tel ordre, mais ne sont pas encore réellement diffusées.

La génération de trajectoires sans conflit s'effectue au moyen de différents algorithmes. Le seul qui ait été testé sur des échantillons de trafic est une adaptation de la méthode développée par Karim Zeghal.

### 1.4.6 Les projets du LOG

Plusieurs axes de recherche ont été suivis au sein du Laboratoire d'Optimisation Globale[12] de Toulouse.

Un outil de simulation du trafic [ABDM96, Bos97] a été développé. Il gère les vols d'une journée complète sur l'espace aérien français, et permet de tester, sur des données issues du trafic réel, des systèmes automatisés de pré-détection ou de résolution de conflits.

Nicolas Durand [Dur96] a étudié la possibilité d'aborder le problème de l'évitement aérien comme un problème de commande optimale. Une étude théorique du problème de la génération de trajectoires d'évitement montre que ce problème est très fortement combinatoire. L'application d'une méthode de type gradient permet de confirmer ces résultats théoriques, et de constater l'inefficacité des méthodes locales. Après avoir modélisé des trajectoires d'évitement tenant compte des contraintes opérationnelles, Nicolas Durand a donc utilisé pour générer des trajectoires d'évitement optimales (sans conflit et avec un allongement minimal) un algorithme génétique (voir chapitre 4).

Cette méthode permet de résoudre des conflits impliquant de très nombreux avions (plusieurs dizaines). Les résultats obtenus sur le simulateur de trafic CATS sont excellents (résolution de tous les conflits sur des journées de trafic très chargées).

[12] Laboratoire commun à l'Ecole Nationale de l'Aviation Civile et au Centre d'Etudes de la Navigation Aérienne

# Chapitre 2

# Génération de trajectoires d'évitement par des méthodes d'intervalles

## 2.1 Introduction

Le problème de l'évitement aérien est un problème d'optimisation (minimiser les retards) sous contraintes (les avions devant rester séparés). Parmi les méthodes d'optimisation globale existantes, il nous a semblé intéressant d'appliquer à ce problème un algorithme de type *Branch and Bound* utilisant *l'analyse d'intervalles*. Cette méthode d'optimisation globale nous a semblé indiquée pour résoudre le problème de l'évitement aérien pour les raisons suivantes :

- Elles sont adaptées à l'optimisation de fonctions de $\mathbb{R}^p$ dans $\mathbb{R}$, puisqu'elles utilisent l'ensemble des intervalles sur l'ensemble de départ de la fonction à optimiser, et que l'ensemble des intervalles sur $\mathbb{R}^n$ se définit naturellement. Or, on cherche à proposer aux avions des trajectoires normalisées, définies par des instants auxquels auront lieu des virages prédéterminés, de manière à ce que ces trajectoires soient faciles à transmettre et à suivre. Si la trajectoire est définie par une suite de $p$ instants, c'est-à-dire de réels, elle peut être définie par un point de $\mathbb{R}^p$. Les trajectoires que nous utiliserons ne seront pas totalement définies par des points de $\mathbb{R}^p$, puisque les directions dans lesquelles s'effectuent les virages définissant les trajectoires peuvent varier. Les algorithmes que nous présentons manipulerons donc des objets qui ne seront pas des intervalles de $\mathbb{R}^p$, mais qui pourront s'y ramener (section 2.8.2).
- Par leur mode de fonctionnement, les algorithmes d'optimisation de type *branch and bound* utilisant les méthodes d'intervalles, appliqués à l'optimisation sous contrainte, peuvent permettre d'obtenir, avant la fin du déroulement de l'algorithme, des sous-ensembles (en fait des intervalles) de l'ensemble de départ sur lesquels les contraintes sont respectées. Cela correspond bien au fait que dans le problème de l'évitement aérien la priorité est de pouvoir fournir des trajectoires sans conflit, la minimisation des retards, bien qu'importante, pouvant n'être obtenue et affinée qu'en second lieu, si on a le temps, par exemple, de laisser l'algorithme converger. Pouvoir donner rapidement une solution admissible, puis des solutions de qualité croissante, est donc, dans cette application, un avantage important.
- Tout système d'évitement automatique, pour pouvoir être appliqué au trafic réel, doit permettre de contrôler les incertitudes : les données disponibles pour les avions (position, vi-

tesses, cap) ne sont jamais connues exactement, de même le suivi des routes proposées n'est jamais précis. Les conséquences de ces imprécisions sur les séparations effectives qui correspondront aux trajectoires proposées doivent pouvoir être contrôlée. L'analyse d'intervalles offre un cadre mathématique du contrôle des erreurs et des imprécisions. La méthode que nous décrivons ici permet ainsi de gérer l'influence de l'incertitude sur les instants auxquels prendront place les manœuvres des avions.

Après une présentation rapide de l'analyse d'intervalles, nous décrivons des algorithmes d'optimisation l'utilisant (sections 2.2 et 2.3). Nous présentons ensuite la modélisation des trajectoires d'évitement que nous utilisons. Cette modélisation, appelée *évitement par point tournant*, permet de définir la trajectoire d'évitement de chacun des avions au moyen seulement d'un sens de déviation, et de deux *instants de manœuvres*, $t_0$ et $t_1$, auxquels l'avion changera de cap.

L'utilisation de méthodes d'intervalles reposera sur le fait qu'on peut calculer l'ensemble des positions possibles de l'avion à tout instant $t$ donné lorsque les *instants de manœuvre* $t_0$ et $t_1$ ne sont plus définis précisément mais peuvent prendre n'importe quelles valeurs à l'intérieur d'intervalles, appelées *intervalles de manœuvres* (section 2.5). On pourra ainsi déterminer les conséquences de l'utilisation des intervalles de manœuvre sur le respect par les avions des contraintes de séparation, ainsi que sur leurs retards (section 2.7).

Ceci permet d'appliquer au problème de l'évitement des algorithmes d'optimisation utilisant des méthodes d'intervalles. Nous présentons deux algorithmes reprenant le principe des algorithmes présentés dans la section 2.3, adaptés au problème de l'évitement.

Ces deux algorithmes répondent à deux manières différentes d'aborder le problème de l'évitement aérien. L'un deux correspond à un système d'évitement *centralisé* (définition 1.9), dans lequel les trajectoires de tous les avions sont générées simultanément, en cherchant à minimiser la somme des retards entraînés par les déviations dues à l'évitement. Nous dirons que cet algorithme réalise *l'approche globale* du problème. L'autre algorithme que nous présentons correspond à une manière d'envisager un système d'évitement *distribué* (définition 1.10), dans lequel chaque avion génère sa propre trajectoire. Les trajectoires sont alors générées successivement, après qu'un ordre de priorité a été fixé entre les avions. Cet algorithme réalise l'approche *séquentielle* du problème (section 2.8). Ce mode de séquencement est du même type que celui qui est utilisé par le TCAS (voir section 1.2) : dans le cadre du TCAS en effet, un des deux avions choisit d'abord une action d'évitement, et l'autre avion agit en fonction de ce que fait le premier.

Ces deux approches sont appliquées à des cas-tests de conflits, les résultats obtenus sont présentés et discutés dans la section 2.9.

## 2.2 L'analyse d'intervalles

### 2.2.1 Présentation

L'analyse d'intervalles a été originellement développée en 1966 par Moore [Moo66] comme un outil permettant de prendre en compte et contrôler les erreurs d'arrondis dans les calculs numériques effectués sur ordinateurs.

Elle a été ensuite utilisée dans le cadre d'algorithmes d'optimisation globale (voir section 2.3).

L'analyse d'intervalles est une généralisation de l'analyse réelle. Dans le cadre de l'analyse d'intervalles, les intervalles de $\mathbb{R}$ (ou de $\mathbb{R}^n$) remplacent les nombres réels (ou les éléments de $\mathbb{R}^n$), l'arithmétique sur $\mathbb{R}$ (ou sur $\mathbb{R}^n$) est remplacée par une arithmétique sur les intervalles de $\mathbb{R}$ (ou de $\mathbb{R}^n$), l'arithmétique d'intervalles.

Nous donnons ici une présentation rapide de l'analyse d'intervalles. Pour plus de détail, on pourra se reporter, par exemple, à [RR95] et [Han92].

Dans toute la suite, nous appellerons $I(\mathbb{R})$, respectivement $I(\mathbb{R}^n)$, l'ensemble des intervalles de $\mathbb{R}$, respectivement $\mathbb{R}^n$, les intervalles de $\mathbb{R}^n$ étant définis comme des produits de $n$ intervalles de $\mathbb{R}$. Les fonctions de l'analyse d'intervalles seront donc des fonctions de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^m)$. Nous verrons (section 2.2.3) comment certaines de ces fonctions peuvent être considérées comme des généralisations de fonctions de $\mathbb{R}^n$ dans $\mathbb{R}^m$.

Les opérations que nous allons décrire dans la section suivante sont des traductions, en terme d'intervalles de $I(\mathbb{R})$ et $I(\mathbb{R}^n)$, des opérations arithmétiques sur $\mathbb{R}$ et sur $\mathbb{R}^n$.

On note $x_{max}$ la borne supérieure de l'intervalle $X$ de $I(\mathbb{R})$ et $x_{min}$ sa borne inférieure.

**Définition 2.1 – Largeur d'un intervalle –** *On définit la largeur $l(X)$ d'un intervalle $X \in I(\mathbb{R})$ de la manière suivante, pour $X = [x_{min}, x_{max}]$ :*

$$l(X) = x_{max} - x_{min}$$

*Et pour $X \in I(\mathbb{R}^n)$, $X = X_1 \times \,..\, \times X_n$ :*

$$l(X) = \max_{i \in \{1..n\}} (l(X_i))$$

**Définition 2.2 – Point milieu d'un intervalle –** *On définit le point milieu $l(X)$ d'un intervalle $X \in I(\mathbb{R})$ de la manière suivante, pour $X = [x_{min}, x_{max}]$ :*

$$m(X) = \frac{x_{max} + x_{min}}{2}$$

*Et pour $X \in I(\mathbb{R}^n)$, $X = X_1 \times \,..\, \times X_n$ :*

$$m(X) = (m(X_1), .., m(X_n))$$

**Définition 2.3 – Intervalle dégénéré –** *On dit qu'un intervalle est dégénéré quand sa largeur est nulle.*

Si $x \in \mathbb{R}^n$, l'intervalle dégénéré $([x_1, x_1], .., [x_n, x_n])$ de $I(\mathbb{R}^n)$ est assimilé au vecteur de $\mathbb{R}^n$ $(x_1, .., x_n)$ ; dans toute la suite de ce chapitre, l'intervalle dégénéré de $I(\mathbb{R}^n)$ $(X_1, ., X_n)$, où $X_1 = [x_1, x_1]..X_n = [x_n, x_n]$, sera noté $(x_1, .., x_n)$.

Pour nous, l'intérêt des méthodes d'intervalles reposera principalement sur l'exploitation des deux caractéristiques suivantes :

- Les méthodes d'intervalles permettent d'utiliser des méthodes d'optimisation globales spécifiques.
- Elles permettent aussi la prise en compte des erreurs et des incertitudes. Il ne s'agit pas uniquement d'erreurs d'arrondis dues aux outils informatiques, mais aussi d'incertitudes inhérentes au problème même que l'on se propose de traiter. Ainsi, les méthodes d'intervalles permettront de considérer que les virages des avions dans le cadre de leurs trajectoires d'évitement n'auront pas lieu à des instants précis, mais prendront place dans des intervalles de temps.

Le fait de travailler avec des intervalles, et non plus des points, oblige à définir des opérations arithmétiques sur les intervalles, correspondant à celles qui existent sur $\mathbb{R}^n$, et des fonctions de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^n)$, traduisant en termes d'intervalles les fonctions de $\mathbb{R}^n$ dans $\mathbb{R}^n$.

### 2.2.2 Opérations arithmétiques sur $I(\mathbb{R})$ et $I(\mathbb{R}^n)$

Les opérations arithmétiques, telles que l'addition, la soustraction, la multiplication, la division sont définies sur $I(\mathbb{R})$ et $I(\mathbb{R}^n)$.

Nous allons présenter rapidement les opérations arithmétiques sur $I(\mathbb{R})$. Les opérations équivalentes sur $I(\mathbb{R}^n)$ s'en déduisent naturellement.

Si $X$ et $Y$ sont deux intervalles de $I(\mathbb{R})$, et *op* est une opération arithmétique de $\mathbb{R}$, on définit $X_1 \; op \; X_2$ de la manière suivante :

$$X_1 \; op \; X_2 \quad = \quad \{y = x_1 \; op \; x_2; \; x_1 \; \in X_1 \; et \; x_2 \; \in X_2\} \tag{2.1}$$

Cette définition s'applique directement pour l'addition, la soustraction et la multiplication. La division est ainsi définie pour $0 \notin X_2$.

Moore [Moo66] donne des règles de constructions équivalentes à ces définitions, qui permettent de construire les intervalles images par les opérations arithmétiques de deux intervalles donnés, $[a, b]$ et $[c, d]$. On a, par exemple, pour la soustraction :

$$[a, b] - [c, d] \quad = \quad [a - d, b - c] \tag{2.2}$$

et pour la multiplication :

$$[a, b] * [c, d] \quad = \quad [min(ac, ad, bc, bd), max(ac, ad, bc, bd)] \tag{2.3}$$

Des définitions semblables existent pour la division, selon la position de 0 par rapport à l'intervalle $[c, d]$.

Hanson [Han68] et Kahan [Kah68] ont introduit, simultanément, mais indépendamment, une extension de l'arithmétique d'intervalles (l'arithmétique d'intervalles *étendue*, dans laquelle une borne (ou deux) d'un intervalle de $\mathbb{R}$ peut être $+\infty$ ou $-\infty$, ce qui permet, entre autre, par exemple, de définir la division par un intervalle contenant 0.

On définit de même à partir de (2.1) l'équivalent sur $I(\mathbb{R})$ des puissances entières sur $\mathbb{R}$. Pour $n \in \mathbb{N}$ et $X \in I(\mathbb{R})$, on définit :

$$X^n = \{y = x^n \ tq \ x \ \in X\}$$

Les opérations arithmétiques sur $I(\mathbb{R}^n)$ sont construites à partir des opérations sur $I(\mathbb{R})$. Ainsi, pour l'addition sur $I(\mathbb{R}^2)$, si on a :

$$\begin{array}{rcl} A \in I(\mathbb{R}^2),\ A & = & A_1 \times A_2,\ A_1 \in I(\mathbb{R}),\ A_2 \in I(\mathbb{R}) \\ B \in I(\mathbb{R}^2),\ B & = & B_1 \times B_2,\ B_1 \in I(\mathbb{R}),\ B_2 \in I(\mathbb{R}) \end{array}$$

la somme des intervalles A et B est définie de la manière suivante :

$$A + B = (A_1 + B_1) \times (A_2 + B_2)$$

L'arithmétique d'intervalles, avec des opérations ainsi définies, pose des *problèmes de dépendance*, c'est-à-dire que les traductions en termes d'arithmétique d'intervalles de deux expressions arithmétique équivalentes sur $\mathbb{R}$ peuvent ne plus être équivalentes sur $I(\mathbb{R})$. La *traduction* en termes d'intervalles d'une expression *dépend* de la forme de cette expression.

Ainsi pour tout $x \in \mathbb{R}$ on a : $2\ x - x = x$ La traduction en terme d'intervalles de cette égalité n'est pas vraie sur tout $I(\mathbb{R})$. Par exemple, pour $A = [0, 1]$, on a :

$$\begin{array}{rcl} 2\ A - A & = & \{y \in \mathbb{R} \ tq \ y = 2\ x_1 - x_2,\ x_1 \in [0,1] \ et \ x_2 \ \in [0,1]\} \\ & = & [-1, 2] \ \neq A \end{array}$$

Cela entraîne qu'une expression arithmétique sur $\mathbb{R}$ et $\mathbb{R}^n$ n'a pas de correspondant unique sur $I(\mathbb{R})$ et $I(\mathbb{R}^n)$. Ainsi, il n'y aura pas de correspondance univoque entre les fonctions de $\mathbb{R}^n$ dans $\mathbb{R}^m$ et celle de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^m)$.

Nous présentons dans la section suivante des définitions concernant les fonctions de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^m)$, obtenues à partir de fonctions de $\mathbb{R}^n$ dans $\mathbb{R}^m$, qui prennent en compte ces problèmes de dépendance.

### 2.2.3 Fonctions de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^m)$

Deux définitions concernant les fonctions de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^m)$ sont particulièrement importantes : la définition d'une extension sur $I(\mathbb{R}^n)$ d'une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^m$, et celle d'une fonction d'inclusion sur $I(\mathbb{R}^n)$ pour une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^m$.

**Définition 2.4 – Extension d'une fonction de $\mathbb{R}^n$ –** *Une fonction $F$ de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^m)$ est appelée une extension sur les intervalles d'une fonction $f$ de $\mathbb{R}^n$ dans $\mathbb{R}^m$ si son image sur tout* *intervalle dégénéré* $(x_1, .., x_n)$ *de* $I(\mathbb{R}^n)$ *est l'intervalle dégénéré de* $I(\mathbb{R}^m)$ $f(x_1, .., x_n)$, *c'estàdire si :*

$$\forall (x_1, ..x_n) \in I(\mathbb{R}^n),\ F\left([x_1, x_1], .., [x_n, x_n]\right) = f(x_1, .., x_n)$$

**Définition 2.5 – Fonction d'inclusion –** *Pour un sous-ensemble* $D \subseteq \mathbb{R}^n$, *et une fonction f de* $D$ *dans* $\mathbb{R}^n$, *on note* $I(D)$ *l'ensemble des intervalles inclus dans D, et, pour* $Y \in I(D)$, *on note* $Im_f(Y) = \{f(x)\ :\ x \in Y\}$, *l'image de* $f$ *sur* $Y$.

*Une fonction* $F$ *de* $I(D)$ *dans* $I$ *est appelée* fonction d'inclusion pour $f$ *sur* $D$ *si on a pour tout* $Y \in I(D)$ *l'inclusion :*

$$Im_f(Y) \subseteq F(Y)$$

Concernant les fonctions définies ci-dessus, après avoir donnée la définition suivante :

**Définition 2.6 – Monotonie pour l'inclusion –** *Une fonction* $F$ *de* $I(\mathbb{R}^n)$ *dans* $I(\mathbb{R}^m)$ *est dite monotone pour l'inclusion si :*

$$\begin{array}{ll} \forall & (X_1, ..X_n) \in I(\mathbb{R}^n) \\ \forall & (Y_1, ..Y_n) \in I(\mathbb{R}^n) \\ X_i & \subset Y_i,\ i = 1..n \implies F(X_1, ..X_n) \subset F(Y_1, ..Y_n) \end{array}$$

On a le théorème :

**Théorème 2.1 –** *Soit* $F$ *une fonction de* $I(\mathbb{R}^n)$ *dans* $I(\mathbb{R}^m)$, *et* $f$ *une fonction de* $\mathbb{R}^n$ *dans* $\mathbb{R}^m$. *Si* $F$ *est une extension de* $f$ *sur* $I(\mathbb{R}^n)$, *et si de plus* $F$ *est monotone pour l'inclusion, alors* $F$ *est une fonction d'inclusion de* $f$ *sur* $I(\mathbb{R}^n)$.

dont la démonstration découle immédiatement des définitions 2.5 et 2.6.

À une fonction $f$ de $\mathbb{R}^n$ dans $\mathbb{R}^m$ ne correspond pas une unique fonction d'inclusion pour $f$ sur $I(\mathbb{R}^n)$, notamment à cause des problèmes de dépendance dont il a été question dans la section précédente.

Les différentes fonctions d'inclusion pour $f$ peuvent, sur des pavés $(X_1, .., X_n)$ donnés, fournir des encadrement de $f(X_1, .., X_n)$ de précisions très différentes. Une mesure de cette précision est donnée dans [Moo66] :

**Définition 2.7 – Ordre d'une fonction d'inclusion –** *On considère une fonction* $f$ *de* $D \subset \mathbb{R}^n$ *dans* $\mathbb{R}^m$, *et* $F$ *une fonction d'inclusion pour* $f$ *sur* $I(D)$. *On dit que* $F$ *est d'ordre* $\alpha$, *ou d'ordre de convergence* $\alpha$ *s'il existe un réel c tel que l'on ait :*

$$l\left(F(X)\right) - l\left(Im_f(X)\right)\ \leq\ c\ \left(l(X)\right)^{\alpha},\ \forall X \in I(D)$$

On trouve dans [RR84] un autre critère de qualité concernant les fonctions d'inclusion :

**Définition 2.8 – Fonction d'inclusion $K$-lipschitzienne –** *Avec les mêmes notations que pour la définition précédente (2.7), On dit que $F$ est $K$-lipschitzienne s'il existe un réel $K$ tel que l'on ait :*

$$l\left(F(X)\right) \leq Kl(X),\ \forall X \in I(D)$$

On trouvera plusieurs théorèmes concernant ces propriétés et permettant de construire des fonctions d'inclusion lipschitziennes d'ordre de convergence donné pour une fonction $f$ de $\mathbb{R}^n$ dans $\mathbb{R}^m$, dans [RR95] et [Han92].

## 2.3 Méthodes d'optimisation globale

Nous avons appliqué au problème de l'évitement automatique un algorithme de type *Branch and Bound*. Nous allons décrire le principe général d'un algorithme de type *Branch and Bound* basé sur des méthodes d'intervalles dans cette section, tel qu'il est présenté dans [RR95]. Nous présenterons plus en détail la version de cet algorithme que nous avons mis au point et utilisé après avoir, dans les sections suivantes, présenté le cadre dans lequel cet algorithme est appliqué.

Le problème de l'évitement que nous nous attachons à résoudre est, comme nous allons le voir dans la section 2.4, un problème d'optimisation sous contraintes : il s'agit de fournir des trajectoires aux avions permettant de minimiser leurs retards tout en maintenant entre eux une distance suffisante (il s'agit de *contraintes de séparations*).

Nous allons tout d'abord décrire le principe de base d'un algorithme de type Branch and Bound utilisant l'analyse d'intervalles pour la résolution d'un problème d'optimisation sans contraintes.

Nous supposerons que le but de l'algorithme est de minimiser une fonction de $D \in I(\mathbb{R}^n)$ dans $\mathbb{R}$).

L'algorithme utilisera :

- Une séquence ordonnée d'intervalles (qui sera en fait une file de priorité) notée $\mathcal{F}_{int}$, dans laquelle les intervalles seront ordonnés selon un critère qui pourra dépendre du problème à résoudre.
- Une fonction d'inclusion de $f$ sur $D$, notée $F$ (voir définition 2.5). Pour un intervalle $X \in D$, on notera : $F(X) = [f_{min}(X), f_{max}(X)]$ l'image de $X$ par $F$. Dans le cadre de l'algorithme d'optimisation, pour un intervalle $X \in \mathcal{F}_{int}$, $F(X)$ sera appelé l'intervalle image de $X$.
- Un *estimateur* de chaque intervalle manipulé par l'algorithme : c'est une valeur de la fonction à optimiser dont on sait qu'elle est atteinte sur l'intervalle $X$. Dans le cas de la minimisation, un *estimateur* de $f$ sur $X$ est un réel noté $f_{est}(X)$ tel qu'il existe $x \in X$ tel que $f_{est}(X) \leq f(x)$. Ainsi, si $F(X) = [f_{min}(X), f_{max}(X)]$ est l'image de $X$ par $F$, on aura

l'encadrement suivant de la plus petite valeur de $f$ atteinte sur $X$ :

$$f_{min}(X) \leq \min_{x \in X}(f(x)) \leq f_{est}(X)$$

- Une fonction de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^n) \times I(\mathbb{R}^n)$ permettant de *couper* un intervalle en deux intervalles.

  En dimension 1, cette fonction sera par exemple celle qui à l'intervalle $[a, b]$ associe les d'intervalles $[a, c]$ et $[c, b]$, où $c$ est le milieu de $[a, b]$.

  En dimension $n > 1$, on prendra la fonction qui à l'intervalle $X = X_1 \times .. \times X_n$, avec $X_i \in I(\mathbb{R})$ associe $X' = X_1 \times .. \times X_{i_0}^{'} \times .. \times X_n$ et $X'' = X_1 \times .. \times X_{i_0}^{''} \times .. \times X_n$, où $X_{i_0}^{'}$ et $X_{i_0}^{''}$ sont obtenus en coupant en deux de la manière décrite ci-dessus pour la dimension 1 l'intervalle $X_{i_0} \in I(\mathbb{R})$, qui est une composante de $X$ de largeur maximale : $i_0$ tel que $l(X_{i_0}) = \max_{i=1..n}(l(X_i))$.
- Une fonction d'insertion de $D$ dans $\mathbb{R}$, notée $f_{ins}$, donnant l'ordre selon lequel un intervalle sera inséré dans $\mathcal{F}_{int}$. On pourra, par exemple, simplement insérer systématiquement les intervalles à la fin de $\mathcal{F}_{int}$, mais l'ordre dans lequel on les insérera pourra avoir des conséquences importantes sur la rapidité de l'algorithme, puisque celui-ci traitera ensuite systématiquement le premier élément de $\mathcal{F}_{int}$.
- Un critère selon lequel on fera ou non passer un intervalle $X$ de $\mathcal{F}_{int}$ dans une autre file, qui sera appelée *file-résultat* et notée $\mathcal{F}_{res}$. Ce critère sera généralement le fait que la taille de l'intervalle $X$ soit inférieure à une valeur donnée, et que la borne inférieure de $F(X)$ soit inférieure à la plus petite valeur de la fonction $f$ calculée par l'algorithme (le plus petit estimateur déjà rencontré par l'algorithme, qui sera noté $f_{est}^*$). Ce critère sera noté $c_r$.

L'algorithme suit les étapes suivantes :

1. $\mathcal{F}_{int} \leftarrow \{D\}$
2. *Calcul de $F(D)$ et d'un estimateur $f_{est}(X)$ de cet intervalle.*
3. $f_{est}^* \leftarrow f_{est}(X)$
4. *Tant que $\mathcal{F}_{int} \neq \emptyset$, on répète les étapes suivantes :*
   (a) *On extrait $X$ le premier élément de la file $\mathcal{F}_{int}$.*
   (b) *Si $X$ satisfait le critère $c_r$, on extrait $X$ de $\mathcal{F}_{int}$ pour l'insérer dans $\mathcal{F}_{res}$. Les intervalles sont insérés dans $\mathcal{F}_{res}$ selon un ordre croissant sur leurs estimateurs.*
   (c) *Si $X$ ne satisfait le critère $c_r$, on coupe $X$ en deux intervalles $X_1$ et $X_2$. Les trois étapes suivantes seront menées pour $j = 1$ et $j = 2$.*
      i. *calcul d'un estimateur $f_{est}(X_j)$ et de l'intervalle image $F(X_j)$.*
      ii. *Si $f_{est}(X_j) < f_{est}^*$ $f_{est}^* \leftarrow f_{est}(X_j)$ et on insère $X_j$ dans la file $\mathcal{F}_{int}$.*
      iii. *Si $f_{min}(X_j) \leq f_{est}^*$, on insère $X_j$ dans la file $\mathcal{F}_{int}$.*
5. *Quand $f = \emptyset$, le résultat de l'algorithme est le premier élément de $\mathcal{F}_{res}$ (l'élément de $\mathcal{F}_{res}$ d'estimateur minimal).*

On voit que la fonction $F$, qui est une fonction d'inclusion pour la fonction $f$ que l'on cherche à minimiser, joue un rôle très important : en effet, à l'étape $(4.a)$ de l'algorithme, on insère dans $\mathcal{F}_{int}$ les intervalles susceptibles d'apporter une amélioration par rapport à la meilleure valeur de la fonction calculée (c'est-à-dire le meilleur estimateur rencontré, que nous noterons $est_0$).

On sait que la borne inférieure de $F(X)$ fournit un minorant des valeurs de la fonction $f$ sur l'intervalle $X$. C'est pourquoi l'intervalle $X$ n'est inséré dans $\mathcal{F}_{int}$ que si la borne inférieure de $F(X)$ est plus petite que $est_0$.

Supposons que l'on se trouve dans ce dernier cas de figure.

Si la borne inférieure de $F(X)$ est égale à la borne inférieure de l'image de $X$ par $f$, $X$ contiendra en effet un point en lequel la fonction $f$ sera plus petite que $est_0$.

Si, en revanche, la borne inférieure de $F(X)$ est beaucoup plus petite que celle de l'image de $X$ par $f$, il est possible que cette dernière soit en fait plus grande que $est_0$. Dans ce cas, l'intervalle $X$ est conservé dans $\mathcal{F}_{int}$ alors qu'il ne contient aucun point sur lequel la fonction $f$ prend une valeur plus petite que $est_0$.

Ainsi, on conservera d'autant moins d'intervalles inutiles dans $\mathcal{F}_{int}$ que les bornes inférieures des images intervalles par la fonction $F$ seront proches de celles de leurs images par la fonction $f$.

Le traitement des contraintes par un algorithme d'optimisation basé sur les méthodes d'intervalles utilise la définition suivante :

**Définition 2.9 – Intervalles admissibles, non admissibles, et indéterminés –** *on distingue trois types d'intervalles, selon la manière dont sont respectées les contraintes :*

- *un intervalle est dit admissible quand les contraintes sont respectées en tout point de cet intervalle.*
- *un intervalle est dit non admissible quand les contraintes ne sont respectées en aucun point de cet intervalle.*
- *un intervalle est dit indéterminé quand il n'est ni admissible, ni non admissible, au sens des deux définitions ci-dessus.*

On suppose que l'espace de départ, $D$, est un intervalle de $I(\mathbb{R}^n)$.

Nous allons présenter un algorithme permettant de déterminer les parties de $D$ satisfaisant les contraintes. Cet algorithme est présenté plus en détail et discuté dans [RR95]. Il manipule une file de priorité contenant des intervalles indéterminés ordonnés par largeur croissante et notée $\mathcal{F}_{ind}$ et une liste d'intervalles admissibles, appelée liste solution et notée $\mathcal{L}_s$.

L'algorithme s'arrête quand un critère d'arrêt, portant sur les éléments de $\mathcal{F}_{ind}$, est satisfait. Ce critère, noté $c_a$, peut être par exemple le fait que tous les éléments de $\mathcal{F}_{ind}$ ont une largeur inférieur à un réel $\epsilon > 0$, ou que la somme des largeurs des éléments de $\mathcal{F}_{ind}$ est inférieure à un $\epsilon > 0$.

On suppose que l'intervalle de départ $D$ est indéterminé, et de taille supérieure à $\epsilon$.

1. $\mathcal{F}_{ind} \leftarrow D$ , $l_s \leftarrow \emptyset$
2. *On extrait $X$ le premier élément de la liste $\mathcal{F}_{ind}$.*
3. *On coupe $X$ en deux intervalles $X_1$ et $X_2$, on répète les étapes suivantes pour $j = 1$ et $j = 2$ :*
   (a) *si $X_j$ est admissible, on l'insère dans $\mathcal{L}_s$.*
   (b) *si $X_j$ est indéterminé, on l'insère dans $\mathcal{F}_{ind}$. Les intervalles sont insérés dans $\mathcal{F}_{ind}$ par ordre de largeurs croissantes.*

*(c) si $X_j$ est non admissible, il ne sera pas conservé par l'algorithme (il n'est conservé dans aucune file).*

*4. si le critère d'arrêt $c_a$ est vérifié, fin de l'algorithme, sinon, on revient à (2).*

On note $D_f$ la sous partie de $D$ sur laquelle les contraintes sont respectées. La liste $\mathcal{L}_s$ et la file $\mathcal{F}_{ind}$ obtenues sont telles que :

$$\bigcup_{X_i \in \mathcal{L}_s} X_i \subset D_f \subset \left( \bigcup_{X_i \in \mathcal{L}_s} X_i \right) \bigcup \left( \bigcup_{X_j \in \mathcal{F}_{ind}} X_j \right) \tag{2.4}$$

Cet algorithme permet donc d'approcher $D_f$ de $D$ par l'union des intervalles de $\mathcal{L}_s$, avec une précision dépendant du critère $c_a$. La liste $\mathcal{L}_s$ et la file $\mathcal{F}_{ind}$ pouvant servir de point de départ à l'algorithme d'optimisation décrit ci-dessus. Selon la nature des contraintes, différents algorithmes de type *branch and bound* utilisant les méthodes d'intervalles permettant de traiter les problèmes d'optimisation sous contraintes existent (voir [RR95]). Ils reprennent le plus souvent les principes des deux algorithmes décrits dans cette section, en les intégrant l'un à l'autre.

Le problème de l'évitement se caractérisera, comme on le verra dans la suite de ce chapitre, par le fait que l'optimisation de la fonction, c'est-à-dire la minimisation des retards des avions, à l'intérieur d'un intervalle admissible, ne posera pas de problème. Le but de l'utilisation de méthodes d'optimisation basées sur l'analyse d'intervalles sera d'obtenir de tels intervalles, et d'obtenir ceux qui mèneront au retard minimal.

C'est pourquoi l'algorithme que nous utilisons s'inspire de l'algorithme d'optimisation sans contraintes décrit dans cette section, mais reprend aussi certaines caractéristiques du second algorithme présenté dans cette section. La fonction à optimiser tient compte des contraintes, de manière à traiter différemment les intervalles admissibles, non admissibles et indéterminés.

## 2.4 Problème à résoudre

Le problème que nous nous proposons de résoudre ici est un problème classique d'évitement : minimiser les retards des avions (retards entraînés par les altérations de leurs trajectoires dues à l'évitement), sous la contrainte qu'il n'y ait aucun conflit entre les avions, c'est-à-dire que les avions restent toujours séparés deux à deux .

Rappelons la définition 1.5 du chapitre 1 :

**Définition 2.10 – Séparations –** *On définit une distance horizontale exprimée en milles nautiques (NM), appelée séparation horizontale, et une distance verticale, exprimée en pieds (ft), appelée séparation verticale. On dit que deux avions sont séparés si la distance entre leurs projections dans un plan horizontal est supérieure à la norme de séparation horizontale OU si la différence entre leurs altitudes est supérieure à la norme de séparation verticale.*

Dans la suite de ce chapitre nous dirons que deux avions sont séparés par une distance *d*, si, sur toute la durée de l'évitement, la distance entre eux reste toujours supérieure ou égale à cette distance *d*. On dira qu'ils restent séparés, sans préciser de quelle distance, quand ils sont séparés au sens de la définition ci-dessus, c'est-à-dire que la distance entre eux restera supérieure ou égale à la norme de séparation.

Nous nous limiterons ici aux avions volant dans un plan horizontal, et nous n'autoriserons que des manœuvres qui maintiennent l'avion dans ce plan horizontal (déviations à droite ou à gauche). Nous ne prendrons donc en compte que la norme de séparation horizontale, et non la norme de séparation verticale.

Nous considérerons que les avions volent à une vitesse constante.

Nous n'introduisons pas d'incertitude sur les positions des avions. Nous supposons que la norme de séparation peut être augmentée, dès le début de l'évitement, pour tenir compte de l'augmentation avec le temps de l'incertitude sur les positions réelles des avions. Ceci est raisonnable compte tenu de la courte durée des trajectoires d'évitement à proposer.

Ces hypothèses étant posées, il reste à vérifier que les modélisations envisagées pour les trajectoires d'évitement permettront l'utilisation des méthodes d'intervalles.

Le principe de l'utilisation des méthodes d'intervalles que nous allons présenter dans les prochaines sections repose sur les points suivants :

- Les trajectoires d'évitement possibles pour un avion éventuellement impliqué dans un conflit sont définies par certaines manœuvres, réalisées à des instants donnés. Dans le cadre de l'utilisation des méthodes d'intervalles, ces instants ne seront pas définis comme des réels, mais comme des intervalles, représentant les intervalles de temps dans lesquels prendront place les manœuvres d'évitement.
- On pourra, au vu des intervalles dans lesquels prendront place les manœuvres d'évitement, savoir si les avions seront ou non séparés durant la durée de l'évitement. Cela sera possible en déterminant, pour un instant $t$ donné, l'ensemble des positions que pourra occuper chaque avion en fonction des intervalles de temps dans lesquels prendront place ses manœuvres, puis en discrétisant le temps sur lequel sera conduit l'evitement, et en vérifiant, pour chaque pas de temps, si les positions possibles pour les avions susceptibles d'être impliqués dans un conflit sont telles que leur séparation sera, ou non, assurée.

  Il faut bien entendu garder présent à l'esprit le fait que la séparation entre les avions n'est vérifiée que sur les pas de temps. On est amené à tenir compte de l'espacement entre les pas de temps en majorant la norme de séparation utilisée. Il faut que cet espacement soit suffisamment petit si l'on veut être assuré à tout moment d'une séparation suffisante, en utilisant dans le cadre de l'algorithme une norme de séparation majorée qui ne soit pas trop grande. L'utilisation d'une norme majorée trop grande aura des conséquences négatives sur les résultats de l'optimisation (voir section 2.8.6).

La première étape vers l'utilisation de telles méthodes d'intervalles est de calculer les positions susceptibles d'être occupées par un avion à un instant donné quand ses manœuvres ne prennent pas place à des instants ponctuels, mais à l'intérieur d'intervalle de temps.

C'est l'objet des sections suivantes (2.5 à 2.7). Nous verrons ensuite quelle connaissance on peut avoir de l'influence de l'introduction des intervalles sur la séparation des avions (section 2.8.3), et nous présenterons ensuite l'algorithme que nous utilisons, et les résultats que nous avons obtenus.

Nous étudierons dans ce chapitre deux modélisations pour les trajectoires d'évitement : l'évitement par offset et l'évitement par point tournant. Ces trajectoires sont définies par les instants auxquels prendront place certaines manœuvres déterminées. Nous ne présentons que les résultats obtenus avec la modélisation par point tournant. En effet, la modélisation par offset, nous le verrons dans la section 2.9.1 ne se prête pas directement à l'utilisation des méthodes d'intervalles telles qu'elles sont décrites ici.

## 2.5 Évitement par point tournant

### 2.5.1 Modélisation des trajectoires d'évitement

La trajectoire d'évitement par point tournant que nous utiliserons ici est une trajectoire d'évitement normalisée, illustrée par la figure 2.1, qui comporte quatre étapes :

1. Jusqu'à un temps $t_0$, l'avion reste sur sa trajectoire d'origine.
2. Au temps $t_0$, l'avion quitte sa trajectoire, vers la droite ou vers la gauche, mais avec un angle $\alpha$ fixé. Il poursuit ensuite sa trajectoire déviée avec un cap constant jusqu'à un temps $t_1$.
3. Au temps $t_1$, l'avion prend un cap de retour tel que sa trajectoire de retour fasse avec sa trajectoire d'origine un angle égal à l'angle de déviation $\alpha$ pris par l'avion au temps $t_0$. L'utilisation que nous présentons ici de cette modélisation de l'évitement reste valable si cet angle de retour a une valeur différente de celle de l'angle selon lequel l'avion a quitté sa trajectoire. Il est nécessaire cependant que ces deux angles aient des valeurs fixées : cette méthode repose sur le fait que l'ensemble des positions possibles d'un avion, quand ces temps de manœuvres prennent leurs valeurs dans des intervalles, ont des formes simples (section 2.5.3). Cela n'est plus le cas si l'angle de retour vers la trajectoire d'origine n'est plus constant. Dans toute la suite, ces deux angles seront égaux, ce qui permettra d'obtenir des expressions mathématiques plus simples pour les retards et les positions des avions.
4. Il rejoint sa trajectoire d'origine à un temps $t_2$, reprend son cap d'origine et poursuit ensuite sa route sur sa trajectoire d'origine.

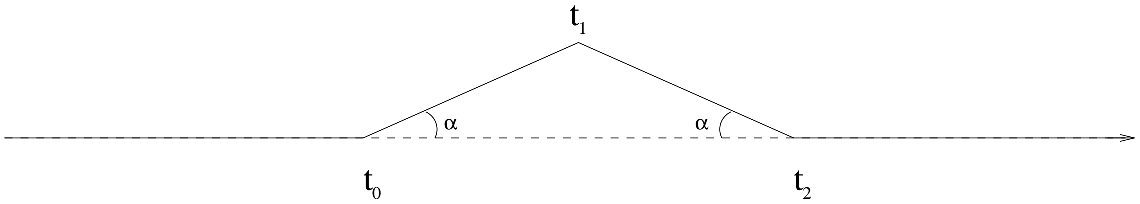



FIG. 2.1: *Modélisation d'une trajectoire d'évitement par point tournant.*

Notons $p(t)$ la position de l'avion au temps $t$. Par construction de la trajectoire décrite ci-dessus, le triangle $p(t_0)p(t_1)p(t_2)$ est isocèle. La vitesse de l'avion étant constante, on a donc

$$t_2 - t_1 = t_1 - t_0$$

ce qui donne :

$$t_2 = 2t_1 - t_0 \tag{2.5}$$

Ainsi définie, la trajectoire d'évitement d'un avion, l'angle $\alpha$ étant fixé, est donc entièrement déterminée par, le sens de la déviation (à droite ou à gauche) et les instants $t_0$ et $t_1$.

**Définition 2.11 – Instants de manœuvres –** *Dans le cadre de l'évitement par point tournant, on appellera instants de manœuvres d'un avion les instants $t_0$ et $t_1$ définis ci-dessus.*

Ce sera sur ces variables que sera conduite l'optimisation.

On note que nous avons pris comme contrainte sur $t_1$ : $t_1 \geq t_0$. Le cas limite $t_1 = t_0$ correspond en fait à un trajectoire non déviée.

Comme nous le verrons dans la section 2.8.2, seuls $t_0$ et $t_1$ seront remplacés par des intervalles, l'angle $\alpha$ prendra des valeurs discrètes, selon le sens de la déviation. Le choix du sens de déviation sera intégré à l'algorithme d'optimisation.

### 2.5.2 Position de l'avion à un temps *t*

Comme nous l'avons vu, la trajectoire d'évitement d'un avion est définie par le sens de sa déviation et les valeurs de $t_0$ et $t_1$. Nous considérons dans cette section un avion dont le sens de déviation est fixé. On cherche à connaître la position $p(t)$ à un temps $t$ donné. Pour des valeurs de $t_1$ et $t_0$ données, cette position est entièrement déterminée.

On considère un repère orthonormé dans le plan, tel que l'avion soit sur l'origine $O$ de ce repère au début de l'évitement. On suppose que l'origine des temps ($t = 0$) correspond au début de l'évitement. Rappelons que ce que nous appelons le début de l'évitement n'est pas le début de la trajectoire déviée, qui correspond, lui à l'instant $t_0$. Le début de l'évitement est le moment à partir duquel la déviation peut avoir lieu. C'est donc une borne inférieure pour $t_0$. On définit aussi une borne supérieure sur les instants auxquels pourront avoir lieu les manœuvres de l'avion. Cette borne supérieure sera, dans toute la suite, notée $t_f$. On aura donc :

$$0 \leq t_0 \leq t_1 \leq t_f$$

On rappelle que l'avion est de retour sur sa trajectoire d'origine au temps $t_2$ :

$$t_2 = 2t_1 - t_0.$$

La vitesse est notée $v$.

Pour $t_0$ et $t_1$ fixés, la position $p(t)$ de l'avion à l'instant $t$ dépend de la position de $t$ relativement à $t_0$, $t_1$, et $t_2$. Dans le repère décrit ci-dessus, on note : $p(t) = (x(t), y(t))$. On distingue quatre cas, correspondant aux quatre étapes de la trajectoire d'évitement :

1. $t \leq t_0$. On a alors :

$$\begin{cases} x(t) = vt \\ y(t) = 0 \end{cases} \tag{2.6}$$

   L'avion n'a pas encore été dévié.

2. $t_0 \leq t \leq t_1$. On a alors :

$$\begin{cases} x(t) = v\left(t_0(1 - \cos\alpha) + t\cos\alpha\right) \\ y(t) = (t - t_0)\sin\alpha \end{cases} \tag{2.7}$$

   L'avion a été dévié et s'écarte de sa trajectoire d'origine, avec l'angle $\alpha$.

3. $t_1 \leq t \leq t_2$. On a alors :

$$\begin{cases} x(t) = v\,(t_0(1-\cos\alpha) + t\cos\alpha) \\ y(t) = (2t_1 - t_0 - t)\sin\alpha \end{cases} \tag{2.8}$$

   L'avion revient sur sa trajectoire d'origine, avec un angle de retour égal à $\alpha$.

4. $t \geq t_2$. On a alors :

$$\begin{cases} x(t) = v\,(t - 2(t1 - t_0)(1-\cos\alpha)) \\ y(t) = 0 \end{cases} \tag{2.9}$$

   L'avion est revenu sur sa trajectoire d'origine.

Une fois l'avion revenu sur sa trajectoire d'origine, on peut calculer le retard entraîné par la trajectoire d'évitement définie par les temps de manœuvres $t_0$ et $t_1$.

Ce retard, noté $r(t_0, t_1)$, est égal au temps mis par l'avion pour aller du point où il quitte sa trajectoire d'origine (au temps $t_0$), au point où il la rejoint (au temps $t_2$), auquel on soustrait le temps qu'il aurait mis pour joindre ces deux points en ligne droite $(x(t_2) - x(t_0)/v)$. on obtient donc, à partir de 2.6 et 2.9 :

$$r(t_0, t_1) = 2(t_1 - t_0)(1 - \cos\alpha) \tag{2.10}$$

### 2.5.3 Conséquences de l'introduction des intervalles

On considère maintenant que l'on a $t_0 \in T_0$ et $t_1 \in T_1$, où $t_0$ et $T_1$ sont deux intervalles de $I(\mathbb{R})$ ($T_0 = [t_{0min}, t_{0max}]$ et $T_1 = [t_{1min}, t_{1max}]$), c'est-à-dire que les instants de manœuvres ne sont plus définis comme des instants ponctuels : on définit deux intervalles de temps ($T_0$ et $T_1$), à l'intérieur desquels prendront place les deux premières manœuvres de la trajectoire d'évitement (on aura $t_0 \in T_0$ et $t_1 \in T_1$). Nous allons calculer l'ensemble des positions possibles d'un avion à un instant $t$ quand ses instants de manœuvres sont donnés par de tels intervalles. Le fait de pouvoir calculer cet ensemble est en effet à la base de la méthode présentée ici (voir section 2.8.3).

**Définition 2.12 – Intervalles de manœuvres –** *Dans le cadre de l'évitement par point tournant, on appellera intervalles de manœuvres d'un avion les intervalles $T_0$ et $T_1$ définis ci-dessus.*

**Définition 2.13 – Positions possibles pour un avion à un instant $t$ –** *À un instant $t$, et pour des intervalles de manœuvres $T_0$ et $T_1$ donnés, on appelle position possible pour l'avion tout point $p$ du plan tel qu'il existe $t_0 \in T_0$, $t_1 \in T_1$, avec $t_0 \leq t_1$, définissant une trajectoire d'évitement telle que si l'avion suit cette trajectoire, il se trouve en $p$ à l'instant $t$. L'ensemble des positions possibles à un instant $t$ pour l'avion considéré est noté $P(t)$.*

Si les trajectoires d'évitement sont décrites par des intervalles de manœuvres, plusieurs des cas énumérés dans la section 2.5.2 peuvent être possibles simultanément, selon la position de $t$ relativement à $T_0$ et $T_1$, et de plus les valeurs de $t_0$ et $t_1$ apparaissant dans les équations ci-dessus ne sont plus uniques.

Comme on doit pouvoir avoir $t_0 \leq t_1$, nous considérerons, dans cette section, qu'on a toujours $t_{0min} \leq t_{1min}$ et $t_{0max} \leq t_{1max}$.

Pour savoir quelle équation utiliser, parmi les équations 2.6 à 2.9, pour calculer l'ensemble des positions possibles de l'avion au temps $t$, il faut déterminer à quelle étape de la trajectoire d'évitement peut se trouver l'avion au temps $t$, en fonction des intervalles $T_0$ et $T_1$.

**Première étape :** l'avion peut être en première étape de l'évitement (c'est-à-dire ne pas avoir encore quitté sa trajectoire d'origine) si et seulement s'il peut exister un $t_0$ tel que $t \leq t_0$, c'est-à-dire si et seulement si :

$$t \leq t_{0max} \tag{2.11}$$

**Deuxième étape :** l'avion peut être à la deuxième étape de l'évitement (c'est-à-dire être en train de s'éloigner de sa trajectoire d'origine), si et seulement s'il existe des valeurs de $t_0$ et de $t_1$ telles que $t_0 \leq t \leq t_1$, c'est-à-dire si et seulement si :

$$t_{0min} \leq t \leq t_{1max} \tag{2.12}$$

**Troisième étape :** l'avion peut être à la troisième étape de l'évitement (c'est-à-dire être en train de revenir vers sa trajectoire d'origine), si et seulement s'il existe des valeurs de $t_0$ et de $t_1$ telles que $t_1 \leq t \leq t_2$, où $t_2 = 2t_1 - t_0$, c'est-à-dire si et seulement si :

$$t_{1min} \leq t \leq 2t_{1max} - 2t_{0min} \tag{2.13}$$

En effet, $2t_{1max} - t_{0min}$ est la plus grande valeur possible pour $t_2$.

**Quatrième étape :** l'avion peut être à la quatrième étape de sa trajectoire d'évitement (c'est-à-dire être revenu sur sa trajectoire d'origine), si et seulement si il peut exister des valeurs de $t_0$ et $t_1$ telles que $t_2 \leq t$, avec $t_2 = 2t_1 - t_0$.

Si $t_{0max} \geq t_{1min}$, on peut avoir $t_0 = t_1 = t_2$, ce qui correspond au cas où l'avion n'est pas dévié. La plus petite valeur possible pour $t_2$ est $t_{1min}$.

Si $t_{0max} < t_{1min}$, la plus petite valeur possible pour $t_2$ est $2t_{1min} - t_{0max}$.

On peut résumer ces deux cas en disant que l'avion peut être à la quatrième étape de l'évitement si et seulement si :

$$t \geq \max(t_{1min}, 2t_{1min} - t_{0max}) \tag{2.14}$$

Selon la position de $t$ par rapport aux intervalles $T_1$ et $T_0$, l'ensemble $P(t)$ des positions possibles de l'avion à l'instant $t$ peut correspondre à une ou plusieurs des étapes décrites ci-dessus.

Dans les sections suivantes, on supposera que l'instant $t$ est fixé, ainsi que les deux intervalles $T_0 = [t_{0min}, t_{0max}]$, et $T_1 = [t_{1min}, t_{1max}]$.

Nous détaillons dans les sections 2.5.4 à 2.5.7 la construction de l'ensemble $P(t)$, qui est résumée dans la section 2.5.8.

### 2.5.4 Prise en compte de la première étape

On note $P_1(t)$ l'ensemble des positions possibles pour l'avion au temps $t$ correspondant à la première étape de l'évitement (l'avion n'a pas encore quitté sa trajectoire d'origine). Il faut prendre en compte les positions possibles de l'avion correspondant à la première étape de l'évitement si on a $t \leq t_{0max}$. Si l'avion est encore à la première étape, l'ensemble des positions possibles pour lui est alors réduit à un point dont les coordonnées sont données par les équations 2.6 :

$$\begin{cases} x(t) = vt \\ y(t) = 0 \end{cases}$$

Ce point sera noté $B_1(t)$. On a : $P_1(t) = \{B_1(t)\}$ si $t \leq t_{0max}$, et $P_1(t) = \emptyset$ sinon. Si on a de plus $t \leq t_{0min}$, on sait que l'avion ne peut être qu'en première étape, c'est alors tout l'ensemble de ses positions possibles qui est réduit à un point, et on connaît précisément la position de l'avion.

### 2.5.5 Prise en compte de la deuxième étape

On note $P_2(t)$ l'ensemble des positions possibles pour l'avion au temps $t$ correspondant à la deuxième étape de l'évitement. Comme on l'a vu, l'avion peut être à la deuxième étape de l'évitement si et seulement si on a $t_{0min} \leq t \leq t_{1max}$ La possibilité pour l'avion d'être en deuxième étape se traduit par le fait que les coordonnées des positions de l'avion sont données par les équations 2.7 :

$$\begin{cases} x(t) = v(t_0(1 - \cos\alpha) + t\cos\alpha) \\ y(t) = v(t - t_0)\sin\alpha \end{cases}$$

avec les contraintes suivantes sur $t_0$ :

$$\begin{cases} t_{0min} \leq t_0 \leq t_{0max} \\ t_0 \leq t \end{cases} \tag{2.15}$$

On note $t'_{0max} = \min(t, t_{0max})$ ; l'ensemble des positions possibles de l'avion correspondant à la deuxième étape de l'évitement est alors un segment de droite, dont les extrémités sont le point de coordonnées :

$$\begin{cases} x_{A_2}(t) = v(t_{0min}(1 - \cos\alpha) + t\cos\alpha) \\ y_{A_2}(t) = v(t - t_{0min})\sin\alpha \end{cases} \tag{2.16}$$

que nous noterons $A_2(t)$, et le point de coordonnées :

$$\begin{cases} x_{B_2}(t) = v(t'_{0max}(1 - \cos\alpha) + t\cos\alpha) \\ y_{B_2}(t) = v(t - t'_{0max})\sin\alpha \end{cases} \tag{2.17}$$

que nous noterons $B_2(t)$.

On a donc $P_2(t) = [A_2(t), B_2(t)]$ si $t_{0min} \leq t \leq t_{1max}$, et $P_2(t) = \emptyset$ sinon.

On note que si on a $t < t_{0max}$, alors $t'_{0max} = t$, et donc $B_2 = B_1$, où $B_1$ est la position correspondant au cas où l'avion est toujours à la première étape. Dans ce cas, le segment de droite décrit ci-dessus est l'ensemble des positions possibles de l'avion correspondant aux deux premières étapes de l'évitement.

### 2.5.6 Prise en compte de la troisième étape

On note $P_3(t)$ l'ensemble des positions possibles pour l'avion au temps $t$ correspondant à la troisième étape de l'évitement. Il est possible que l'avion soit en troisième étape si et seulement si on a :

$$t_{1min} \leq t \leq 2t_{1max} - 2t_{0min}$$

Les positions possibles correspondant à cette étape sont alors données par les équations 2.8 :

$$\begin{cases} x(t) = v(t_0(1 - \cos\alpha) + t\cos\alpha) \\ y(t) = (2t_1 - t_0 - t)\sin\alpha \end{cases}$$

sous les conditions :

$$\begin{cases} t_{0min} \leq t_0 \leq t_{0max} \\ t_{1min} \leq t_1 \leq t_{1max} \\ t_0 \leq t_1 \leq t \\ t \leq 2t_1 - t_0 \end{cases} \tag{2.18}$$

Ces contraintes sont redondantes : les conditions $t \geq t_1$ (traduisant le fait que l'avion est à la troisième étape de l'évitement) et $t \leq 2t_1 - t_0$, suffisent pour assurer $t_0 \leq t_1$.

Dans toute la suite, on notera $t^{'}_{0max} = \min(t_{0max}, t)$ et $t^{'}_{1max} = \min(t_{1max}, t)$.

Avec les notations $t^{'}_{0max} = \min(t_{0max}, t)$ et $t^{'}_{1max} = \min(t_{1max}, t)$, les contraintes 2.18 se ramènent alors aux conditions suivantes :

$$\begin{cases} t_{0min} \leq t_0 \leq t^{'}_{0max} \\ t_{1min} \leq t_1 \leq t^{'}_{1max} \\ t \leq 2t_1 - t_0 \end{cases} \tag{2.19}$$

La contrainte $t \leq 2t_1 - t_0$ correspond en fait simplement à $y \geq 0$. L'ensemble des positions possibles de l'avion correspondant à la troisième étape est donc un parallélogramme tronqué, plus précisément l'intersection d'un parallélogramme et du demi-plan d'équation $y = 0$.

On note $A_3$, $B_3$, $C_3$ et $D_3$ les sommets de ce parallélogramme (voir figure 2.2 (a))

Les coordonnées du point $A_3$ sont :

$$\begin{cases} x_{A_3}(t) = v(t_{0min}(1 - \cos\alpha) + t\cos\alpha) \\ y_{A_3}(t) = v(2t^{'}_{1max} - t_{0min} - t)\sin\alpha \end{cases} \tag{2.20}$$

Le point $A_3$ correspond au cas où l'avion a quitté sa trajectoire le plus tôt possible, c'est-à-dire au temps $t_{0min}$, et a amorcé son retour vers sa trajectoire d'origine le plus tard possible, c'est-à-dire au temps $t_{1max}$ dans le cas au $t \geq t_{1max}$, ou même n'a pas encore amorcé son retour, dans le cas où $t \leq t_{1max}$ (on a alors $t^{'}_{1max} = t$).

Les coordonnées du point $B_3$ sont :

$$\begin{cases} x_{B_3}(t) = v\left(t^{'}_{0max}(1 - \cos\alpha) + t\cos\alpha\right) \\ y_{B_3}(t) = v(2t^{'}_{1max} - t^{'}_{0max} - t)\sin\alpha \end{cases} \tag{2.21}$$

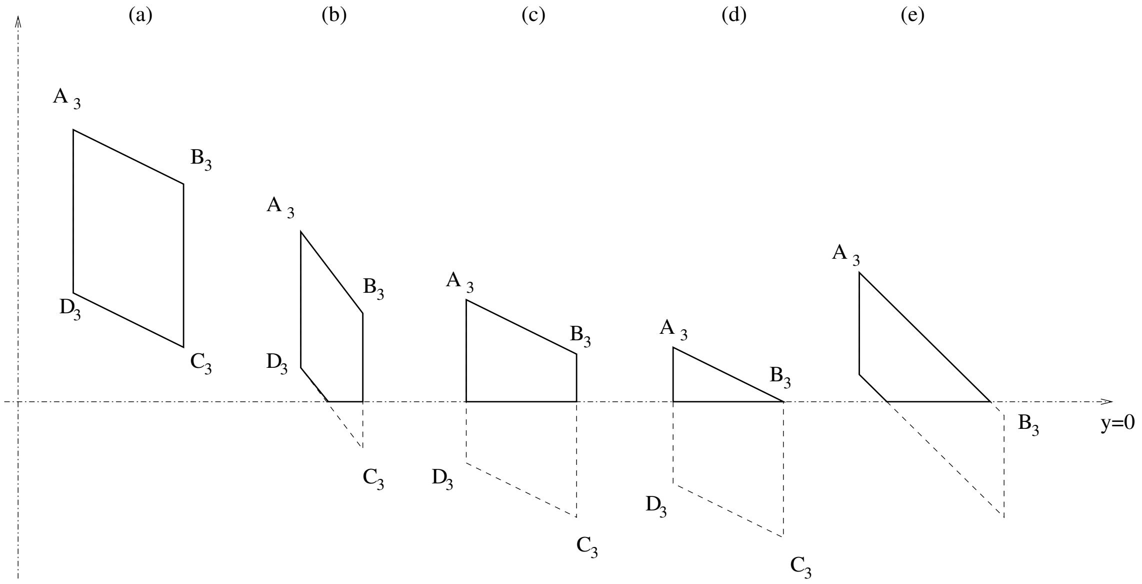



FIG. 2.2: *Prise en compte de la troisième étape de l'évitement*

On peut avoir $y_{B_3}(t) = 0$ dans le cas où $t_{1min}t \geq t_{0max}$ (on a alors $B_3(t) = B_1(t)$ : l'avion n'a pas encore quitté sa trajectoire d'origine), ou dans le cas où $t = 2t_{1max} - t_{0min}$.

Si $y_{B_3}(t) > 0$, le point $B_3$ correspond au cas où l'avion a quitté sa trajectoire le plus tard possible (c'est-à-dire au temps $t_{0max}$, puisque on a alors $t \geq t_{0max}$) et a amorcé son retour vers sa trajectoire d'origine le plus tard possible (c'est-à-dire en $t_{1max}$ si $t \geq t_{1max}$) ou n'a pas encore amorcé son retour vers sa trajectoire d'origine (si $t \leq t_{1max}$).

Les coordonnées du point $C_3$ sont :

$$\left\{ \begin{array}{l} x_{C_3}(t) = v(t'_{0max}(1 - \cos\alpha) + t\cos\alpha) \\ y_{C_3}(t) = v(2t'_{1min} - t'_{0max} - t)\sin\alpha \end{array} \right. \tag{2.22}$$

Si $y_{C_3}(t) \geq 0$, le point $C_3$ correspond au cas où l'avion a quitté sa trajectoire le plus tard possible, c'est-à-dire au temps $t_{0max}$, et a amorcé son retour vers sa trajectoire d'origine le plus tôt possible, c'est-à-dire au temps $t_{1min}$, puisque l'on a $t_0 \leq t_{1min} \leq t$.

Les coordonnées du point $D_3$ sont :

$$\left\{ \begin{array}{l} x_{D_3}(t) = v\,(t_{0min}(1 - \cos\alpha) + t\cos\alpha) \\ y_{D_3}(t) = v(2t'_{1min} - t_{0min} - t)\sin\alpha \end{array} \right. \tag{2.23}$$

Si $y_{D_3}(t) \geq 0$, le point $D_3$ correspond au cas où l'avion a quitté sa trajectoire le plus tôt possible, et a amorcé son retour vers sa trajectoire d'origine le plus tôt possible.

Dans le cas où on n'a pas $t_{1min} \leq t \leq 2t_{1max} - 2t_{0min}$, l'avion ne peut pas être à la troisième étape de l'évitement et donc $P_3(t) = \emptyset$.

Dans le cas où on a $t_{1min} \leq t \leq 2t_{1max} - 2t_{0min}$, l'ensemble des $P_3(t)$ positions possibles correspondant à la troisième étape de l'évitement est un parallélogramme si $y_{D_3}(t) \geq 0$, c'est-à-dire si $t \leq 2t_{1min} - t'_{0max}$.

Si cette condition n'est pas remplie, $P_3(t)$ est l'intersection du parallélogramme $A_3\ B_3\ C_3\ D_3$ et du demi-plan d'équation $y \geq 0$, qu'il est alors facile de calculer.

Les différentes formes possibles pour l'ensemble des points correspondant à la troisième étape de l'évitement sont représentées sur la figure 2.2.

On remarque que si on a $t\ \leq t_{1max}$, alors on a $A_3(t) = A_2(t)$ et $B_3(t) = B_2(t)$. L'ensemble des positions possibles correspondant à la troisième étape contient donc, sur sa limite (le segment $[A_3, B_3]$), les positions correspondant à la deuxième étape de l'évitement, et même, dans le cas où $t \leq t_{0max}$ le point correspondant à la première étape de l'évitement (alors égal à $B_3$).

### 2.5.7 Prise en compte de la dernière étape de l'évitement

On note $P_4(t)$ l'ensemble des positions possibles pour l'avion $t$ correspondant à la quatrième et dernière étape de l'évitement (l'avion est revenu sur sa trajectoire d'origine). Au temps $t$, l'avion peut être à la quatrième étape de l'évitement si et seulement si $t \geq max(t_{1min}, 2t_{1min} - t_{0max})$.

Si on a $2t_{1max} - t_{0min} - t \leq 0$, toutes les position possibles de l'avion correspondent à la dernière étape de la trajectoire de l'évitement (l'avion est revenu sur sa trajectoire d'origine). L'ensemble des positions possibles de l'avion est décrit par les équations 2.9 :

$$\left\{ \begin{array}{l} x(t) = v\,(t - 2(t1 - t_0)(1 - \cos\alpha)) \\ y(t) = 0 \end{array} \right.$$

avec les conditions :

$$\left\{ \begin{array}{l} t_{0min} \leq t_0 \leq t_{0max} \\ t_{1min} \leq t_1 \leq t_{1max} \\ t_0 \leq t_1 \leq t \\ t \geq 2t_1 - t_0 \end{array} \right. \tag{2.24}$$

Il s'agit d'un segment de droite, que nous notons $[EF]$.

Les coordonnées de $E$ correspondent au maximum possible de $t_1 - t_0$, c'est-à-dire au cas où $t_1 = t_{1max}$ et $t_0 = t_{0min}$ (le fait que $t > t_{1max}$ est assuré par $2t_{1max} - t_{0min} - t < 0$) :

$$\left\{ \begin{array}{l} x_E(t) = v\,(t - 2(t_{1max} - t_{0min})(1 - \cos\alpha)) \\ y_E(t) = 0 \end{array} \right. \tag{2.25}$$

Les coordonnées de F correspondent au minimum possible pour $t_1 - t_0$. Ce minimum est égal à 0 si $t_{1min} \leq t_{0max}$ et $t_{1min} - t_{0max}$ si $t_{1min} > t_{0max}$. On a donc les deux cas suivants :
Si $t_{1min} \leq t_{0max}$ :

$$\left\{ \begin{array}{l} x_F(t) = vt \\ y_F(t) = 0 \end{array} \right. \tag{2.26}$$

Si $t_{1min} > t_{0max}$ :

$$\left\{ \begin{array}{l} x_F(t) = v\,(t - 2(t_{1min} - t_{0max})(1 - \cos\alpha)) \\ y_F(t) = 0 \end{array} \right. \tag{2.27}$$

Dans le cas où on a $t \geq max(t_{1min}, 2t_{1min} - t_{0max})$ mais $2t_{1max} - t_{0min} - t > 0$, cela signifie que l'avion peut être dans des positions correspondant ła troisième étape de la trajectoire d'évitement (l'avion revient vers sa trajectoire d'origine), mais aussi dans des positions correspondant à la dernière étape (l'avion est revenu sur sa trajectoire d'origine).

$P_4(t)$, l'ensemble des positions possible de l'avion correspondant à la quatrième étape de l'évitement est encore un segment, noté [EF]. Les coordonnées de F sont alors encore données par les équations (2.26) ou (2.27) et le point $E$ est dans ce cas à l'intérieur de l'ensemble $P_3(t)$ décrit dans la section précédentes. Il est donc inutile de calculer ses coordonnées ; En effet, l'ensemble $P_3(t)$ est alors un parallélogramme tronqué dont une arête est portée par la droite d'équation $y = 0$. On obtient les points de $P_4(t)$ qui ne sont pas dans $P_3(t)$ en prolongeant cette arête jusqu'au point $F$ dont les coordonnées sont données ci-dessus.

Si $t < max(t_{1min}, 2t_{1min} - t_{0max})$, $P_4(t) = \emptyset$.

### 2.5.8 Construction de l'ensemble $P(t)$

L'ensemble $P(t)$ est la réunion des ensembles $P_i(t)$ définis dans les sections précédentes. On peut résumer la marche à suivre pour le construire de la manière suivante :

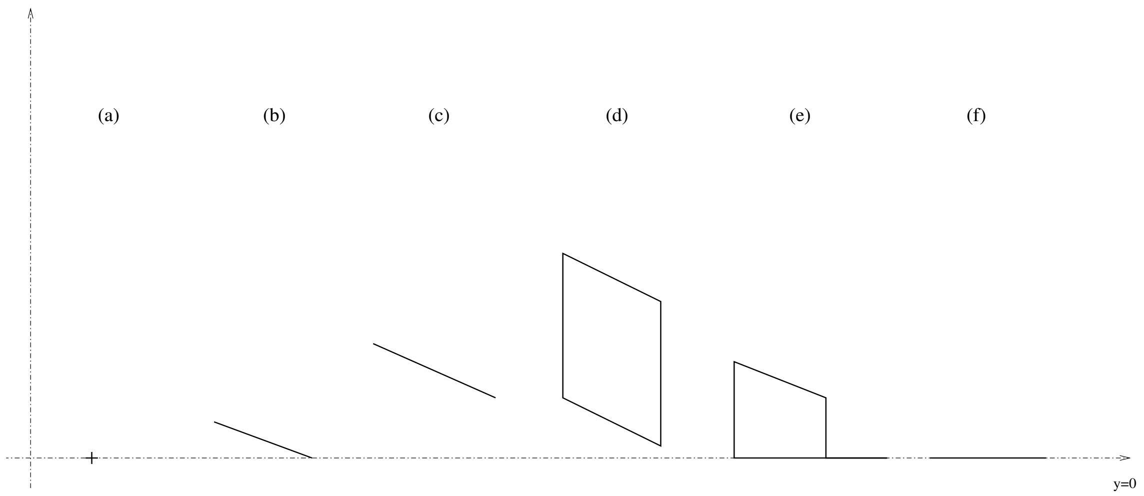



FIG. 2.3: *Différentes formes possibles pour l'ensemble P(t)*

1. Si $t \leq t_{0min}$, on obtient un point, correspondant à la position de l'avion non dévié. C'est le cas (a) de la figure 2.3.
2. Si $t_{0min} < t \leq t_{1min}$, on obtient un segment de droite, correspondant aux positions possibles de l'avion en deuxième étape de l'évitement (l'avion s'écarte de sa trajectoire d'origine). C'est le cas (c) de la figure 2.3. Notons que si $t \leq t_{0max}$, l'avion peut ne pas avoir été dévié, mais le point qu'il occuperait alors est contenu dans le segment de droite correspondant à la deuxième étape de l'évitement. C'est le point limite obtenu pour $t_0 = t$, qui correspond à l'intersection du segment de droite obtenu en section 2.5.5 et de la droite $(y = 0)$, on est alors dans le cas (b) de la figure 2.3.

3. si $t \geq t_{1min}$, il faut tenir compte de la possibilité qu'a l'avion d'être en troisième étape (l'avion revient vers sa trajectoire d'origine). Il faut donc déterminer, selon la méthode décrite dans les section précédentes, l'ensemble $P_3(t)$ (positions possibles correspondant à la troisième étape de l'évitement), et éventuellement, l'ensemble $P_4(t)$ (positions possibles correspondant à la quatrième étape).

   Si on a $t \leq t_{0max}$, l'avion peut être encore en première étape (avion non dévié), mais ce cas est un point limite de $P_3(t)$ : c'est le point $B_3$ (et on a dans ce cas $y_{B_3} = 0$). De même, si $t \leq t_{1max}$, l'avion peut être encore en deuxième étape (il s'écarte de sa trajectoire), mais ce cas est aussi un cas limite de $P_3(t)$ : c'est le segment $[A_3 B_3]$. Donc dans le cas $t \geq t_{1min}$, les calculs présentés dans sections 2.5.6 et suivantes permettent de déterminer totalement l'ensemble des positions possibles de l'avion.

   Le cas (d) de la figure 2.3 montre la situation dans laquelle l'avion ne peut être qu'en troisième étape, tandis que dans le cas (e) l'avion peut être en troisième ou en quatrième étape.

4. si $t \geq 2t_{1max} - t_{0min}$, l'avion ne peut-être qu'à la quatrième étape de l'évitement, $P(t)$ est un segment de droite. C'est le cas (f) de la figure 2.3.

Les conséquences, sur les distances entre avions, de l'utilisation d'intervalles de manœuvres au lieu d'instants de manœuvres seront étudiées dans la section 2.7.2. Ses conséquences sur les retards des avions, dans la section 2.7.3.

## 2.6 Évitement par offset

### 2.6.1 Modélisation de la trajectoire d'évitement

La trajectoire d'évitement par offset comporte les étapes suivantes :

1. Jusqu'à un temps $t_0$, l'avion reste sur sa trajectoire d'origine.
2. Au temps $t_0$, l'avion quitte sa trajectoire d'origine en effectuant un virage vers la gauche ou vers la droite, la valeur absolue de son changement de cap étant toujours égale à une même valeur, fixée, notée $\alpha$. Il continue de s'éloigner de sa trajectoire d'origine jusqu'au temps $t_1$.
3. Au temps $t_1$, il reprend son cap d'origine, et poursuit sa route sur une nouvelle trajectoire, parallèle à sa trajectoire d'origine.

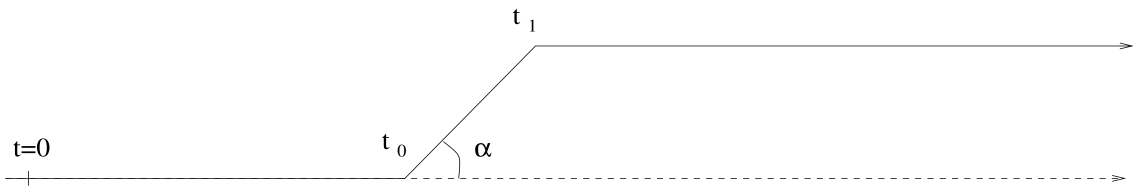



FIG. 2.4: *Modélisation d'une trajectoire d'évitement par offset*

Un exemple de trajectoire d'évitement par offset est donné sur la figure 2.4.

Ainsi définie, la trajectoire d'évitement d'un avion sera entièrement déterminée par le sens de la déviation (à droite ou à gauche), la valeur de $t_0$ et la valeur de $t_1$, avec $t_1 \geq t_0$.

Notons tout d'abord que les trajectoires d'évitement par offset sont déterminées par les mêmes types de données que les trajectoires d'évitement par point tournant présentées dans la section précédente.

Ceci permettra d'utiliser le même algorithme avec les deux modélisations. L'algorithme gardera la même structure, seul le calcul à chaque pas de temps de l'ensemble $P(t)$ devra être modifié pour passer d'une modélisation à une autre.

De même que pour la modélisation par point tournant, on supposera que $t_0 \leq t_1$, $t_0 = t_1$ correspondant à une trajectoire non déviée.

### 2.6.2 Position de l'avion à un temps *t*

Nous supposons que le sens de déviation est fixé. On cherche dans un premier temps, pour des valeurs fixées de $t_0$ et de $t_1$, la position $p(t)$ au temps $t$ de l'avion s'il suit la trajectoire d'évitement par offset déterminée par $t_0$, $t_1$ et le sens de déviation en $t_0$.

On considère un repère orthonormé dans le plan, tel que l'avion soit sur l'origine $O$ de ce repère au début de l'évitement. On suppose l'origine des temps ($t = 0$) correspond au début de l'évitement, et que les manœuvres des avions doivent avoir lieu avant un instant $t_f$. On a donc, comme dans le cas de l'évitement par point tournant :

$$0 \leq t_0 \leq t_1 \tag{2.28}$$

La vitesse est notée $v$. Dans le repère décrit ci-dessus, on note : $p(t) = (x(t), y(t))$. On distingue trois cas, correspondant aux trois étapes de la trajectoire d'évitement :

1. $t \leq t_0$. L'avion n'a pas encore quitté sa trajectoire d'origine, et on a alors :

$$\begin{cases} x(t) = vt \\ y(t) = 0 \end{cases} \tag{2.29}$$

2. $t_0 \leq t \leq t_1$. On a alors :

$$\begin{cases} x(t) = v\,(t_0(1 - \cos\alpha) + t\cos\alpha) \\ y(t) = v(t - t_0)\sin\alpha \end{cases} \tag{2.30}$$

   L'avion a été dévié, et s'écarte de sa trajectoire d'origine, avec l'angle $\alpha$.

3. $t \geq t_1$. L'avion a repris son cap d'origine et suit une trajectoire parallèle à sa trajectoire d'origine. Ses coordonnées sont alors les suivantes :

$$\begin{cases} x(t) = v\,(t - (t_1 - t_0)(1 - \cos\alpha)) \\ y(t) = v(t_1 - t_0)\sin\alpha \end{cases} \tag{2.31}$$

Dans le cas de l'évitement par offset, il n'est plus possible de définir le retard, comme nous l'avons fait dans le cas de l'évitement par point tournant, comme la différence entre le temps que l'avion met pour atteindre, après l'évitement, un point de sa trajectoire d'origine et le temps qu'il aurait mis à atteindre ce point en l'absence de déviation. En effet, à la fin de l'évitement, l'avion ne revient pas sur sa trajectoire d'origine, mais reste sur une trajectoire parallèle à celle-ci.

On considère alors une ligne virtuelle perpendiculaire à la trajectoire d'origine de l'avion, qui jouera en quelque sorte le rôle d'une ligne d'arrivée. Le retard est défini comme le temps supplémentaire mis par l'avion pour atteindre un point donné de la trajectoire sur laquelle il se trouve à partir de la troisième étape de l'évitement, par rapport au temps qu'il aurait mis pour atteindre le projeté de ce point sur sa trajectoire d'origine suivant un direction perpendiculaire à sa direction d'origine.

Rappelons que nous considérons ici que la trajectoire d'origine de l'avion est porté par la droite d'équation $y = 0$. Le retard de l'avion est alors la différence entre le temps mis par l'avion pour atteindre après sa déviation un point d'une abscisse donnée, et le temps qu'il aurait mis pour atteindre le point de même abscisse et d'ordonnée égale à 0 en l'absence de déviation.

On obtient à partir des équations 2.29 et 2.31 l'expression suivante pour le retard, en fonction de $t_0$ et $t_1$ :

$$r(t_0, t_1) = (t_1 - t_0)(1 - \cos\alpha) \tag{2.32}$$

Ainsi que nous l'avons fait pour l'évitement par point tournant, nous allons maintenant définir l'ensemble des positions possibles de l'avion à l'instant $t$ pour $t_0 \in T_0$ et $t_1 \in T_1$, où $T_0$ et $T_1$ sont des intervalles. Cet ensemble sera noté $P(t)$. Comme nous allons le voir, $P(t)$ sera beaucoup plus simple pour l'évitement par offset qu'il ne l'était pour l'évitement par point tournant.

### 2.6.3 Conséquences de l'introduction des intervalles

De même que pour la modélisation par point tournant, on considère maintenant qu'on a $t_0 \in T_0$ et $t_1 \in T_1$, avec $T_0 = [t_{0min}, t_{0max}]$ et $T_1 = [t_{1min}, t_{1max}]$.

On peut établir, en fonction de $t$, $T_0$ et $T_1$, les conditions nécessaires et suffisantes pour que l'avion puisse être à une étape donnée de la trajectoire dévitement par offset.

**Première étape :** il est possible que l'avion soit à l'instant $t$ à la première étape si et seulement s'il existe $t_0 \in T_0$ tel que $t \leq t_0$, c'est-à-dire si et seulement si :

$$t \leq t_{0max} \tag{2.33}$$

**Deuxième étape :** il est possible qu'il soit à la deuxième étape si et seulement s'il existe $t_0 \in T_0$ tel que $t \leq t_0$, et $t_1 \in T_1$ tel que $t \geq t_1$, c'est-à-dire si et seulement si :

$$t_{0min} \leq t \leq t_{1max} \tag{2.34}$$

**Troisième étape :** il est possible qu'il soit à la troisième étape si et seulement s'il existe $t_1 \in T_1$ tel que $t \geq t_1$, c'est-à-dire si et seulement si :

$$t \geq t_{1min} \tag{2.35}$$

### 2.6.4 Construction de l'ensemble $P(t)$

Les deux premières étapes de l'évitement par offset sont en fait identiques aux deux premières étapes de l'évitement par point tournant. Si on note $P_i(t)$ l'ensemble des positions possibles de l'avion correspondant à l'étape ($i$), les ensembles $P_1(t)$ et $P_2(t)$ sont donc les mêmes que ceux qui ont été définis dans les sections 2.5.4 et 2.5.5.

L'ensemble $P_3(t)$ est l'ensemble des points $p(t)$ dont les coordonnées $(x(t), y(t))$ sont données par les équations 2.31 :

$$\begin{cases} x(t) = v\left(t - (t_1 - t_0)(1 - \cos\alpha)\right) \\ y(t) = v(t_1 - t_0)\sin\alpha \end{cases} \tag{2.36}$$

avec les conditions :

$$\begin{cases} t_{0min} \leq t_0 \leq t_{0max} \\ t_{1min} \leq t_1 \leq t_{1max} \\ t_0 \leq t_1 \leq t \end{cases} \tag{2.37}$$

On définit $t'_{1max} = \min(t_{1max}, t)$ et $t'_{0,1} = \max(0, t_{1min} - t_{0max})$. Si on note $\tau = t_1 - t_0$, les équations 2.36 et les conditions 2.37 deviennent :

$$\begin{cases} x(t) = v\left(t - \tau(1 - \cos\alpha)\right) \\ y(t) = v\tau\sin\alpha \end{cases} \tag{2.38}$$

avec la condition :

$$\left\{ t'_{0,1} \leq \tau \leq t'_{1max} - t_{0min} \right. \tag{2.39}$$

Les équations et contraintes ci-dessus impliquent que dans le cas $t_{0min} \leq t \leq t_{1max}$, $P_3(t)$ est un segment de droite. On note $A_3$ et $B_3$ les extrémités de ce segment ($A_3$ correspondant au point d'abscisse minimale).

Les coordonnées de $A_3$ sont :

$$\begin{cases} x_{A_3}(t) = v(t - (t'_{1max} - t_{0min})(1 - \cos\alpha)) \\ y_{A_3}(t) = v(t'_{1max} - t_{0min})\sin\alpha \end{cases} \tag{2.40}$$

et celles de $B_3$ sont :

$$\begin{cases} x_{B_3}(t) = v(t - t'_{0,1}(1 - \cos\alpha)) \\ y_{B_3}(t) = vt'_{0,1}\sin\alpha \end{cases} \tag{2.41}$$

On a donc $P_3(t) = [A_3(t), B_3(t)]$ si $t_{0min} \leq t \leq t_{1max}$ et $P_3(t) = \emptyset$ sinon.

On peut vérifier que si l'on a $t_{1min} \leq t \leq t_{1max}$, on a obtient le même segment de droite que pour la deuxième étape ($A_3(t) = A_2(t)$ et $B_3(t) = B_2(t)$) (voir section 2.5.5).

L'ensemble $P(t)$ est donc réduit à un point si $t \leq t_{0min}$ et est un segment de droite sinon. Ce segment de droite, pour $\alpha$ positif est inclus dans le demi-plan d'équation $y \geq 0$. Il touche la droite d'équation $y = 0$ (c'est-à-dire qu'il est possible que l'avion ne soit jamais dévié de sa trajectoire d'origine) si $t_{0max} \geq t_{1min}$.

## 2.7 Effet des intervalles de manœuvres sur l'évitement

### 2.7.1 Caractéristiques des intervalles de manœuvres

Utilisées dans le cadre de méthodes d'intervalles, les deux modélisations qui viennent d'être étudiées sont telles que l'algorithme d'optimisation manipulera des données identiques dans les deux cas :

- les intervalles de manœuvres $T_0$ et $T_1$, avec pour les deux modélisations de l'évitement les contraintes suivantes sur les bornes de ces intervalles :

$$0 \leq t_{0min} \leq t_{0max} \leq t_f \tag{2.42}$$

$$0 \leq t_{1min} \leq t_{1max} \leq t_f \tag{2.43}$$

$$t_{0min} \leq t_{1max} \tag{2.44}$$

  L'inéquation 2.44 traduit le fait qu'il doit pouvoir exister $t_0 \in T_0$ et $t_1 \in T_1$ tels que $t_0 \leq t_1$. Nous avons jusqu'ici supposé qu'on avait $t_{0min} \leq t_{1min}$ et $t_{0max} \leq t_{1max}$. Ces conditions ne sont pas nécessaires pour assurer que les intervalles $T_0$ et $T_1$ contiennent des manœuvres possibles. Cependant, dans les définitions des ensembles de positions possibles des avions en fonction de leurs intervalles de manœuvres, la condition $t_0 \leq t_1$ intervient toujours. Ceci implique que si $T_0$ et $T_1$ sont tels que $t_{1min} < t_{0min}$ ou $t_{0max} > t_{1max}$, certaines parties des intervalles de manœuvres sont inutiles. Or l'algorithme visant, comme nous le verrons, à diminuer la taille des intervalles de manœuvres, ceux-ci seront alors systématiquement tronqués.
- le sens de déviation, qui se traduit par $\alpha < 0$ ou $\alpha > 0$. La valeur absolue de $\alpha$ est fixée. On rappelle que le cas dans lequel un avion n'est pas dévié ne sera pas traduit par $\alpha = 0$, mais par $t_0 = t_1$. En terme d'intervalles de manœuvres, la possibilité que l'avion ne soit pas dévié est donnée par $t_{0max} \geq t_{1min}$. On est sûr que l'avion ne sera pas dévié uniquement si on a :

$$t_{0min} = t_{0max} = t_{1min} = t_{1max}$$

### 2.7.2 Distances entre avions

Le premier but d'un système d'évitement, qu'il soit ou non automatique, est d'assurer la séparation des avions, c'est-à-dire d'assurer que, pendant toute la durée de l'évitement, les avions susceptibles d'être impliqués dans un conflit soient séparés deux à deux (voir la définition 2.10).

Comme nous nous limitons dans ce chapitre aux avions volant dans un plan horizontal, nous ne tiendrons compte que de la norme de séparation horizontale, que nous noterons $n_h$. La distance entre deux points $p$ et $q$ du plan sera notée $d(p, q)$.

Les trajectoires d'évitement utilisées correspondent à une des deux modélisations décrites ci-dessus, par offset ou par point tournant. Dans la suite, nous utiliserons la modélisation par point tournant, en signalant les quelques points à modifier pour utiliser une modélisation par offset.

Considérons tout d'abord deux avions, notés $a^i$ et $a^j$, susceptibles d'être impliqués dans un conflit que nous nous proposons de résoudre.

Pour chacun de ces avions, les sens de déviation sont fixés. On note $\alpha^i$ (resp. $\alpha^j$) l'angle de déviation de l'avion $a^i$ (resp. $a^j$). On a $|\alpha^i| = |\alpha^j| = \alpha$. Le signe de l'angle de déviation d'un avion donne son sens de déviation.

Supposons, dans un premier temps, que les trajectoires d'évitement des avions $a^i$ et $a^j$ sont définies par des temps de manœuvres, et non pas des intervalles de manœuvres.

Dans ce cas, les positions des avions $a^i$ et $a^j$ à un temps $t \in [0, t_f]$ sont, comme on l'a vu, parfaitement définies. On note $p^i(t)$ et $p^j(t)$ ces positions, et $\delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j)$ la distance entre elles à l'instant $t$ : $\delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j) = d(p^i(t), p^j(t))$.

La condition de séparation entre les avions $a^i$ et $a^j$ s'écrirait alors :

$$\delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j) \geq n_h, \ \forall t \in [0, t_f] \tag{2.45}$$

En notant :

$$\delta_{inf}^{i,j}(t_0^i, t_1^i, t_0^j, t_1^j) = \inf_{t \in [0, t_f]} \left( \delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j) \right) \tag{2.46}$$

la condition 2.45 devient :

$$\delta_{inf}^{i,j}(t_0^i, t_1^i, t_0^j, t_1^j) \geq n_h \tag{2.47}$$

Nous définissons maintenant, à partir des $\delta_{inf}^{i,j}(t_0^i, t_1^i, t_0^j, t_1^j)$, le minimum des distances entre avions au cours de l'évitement si la trajectoire de chaque avion $a^i$ est définie par les temps de manœuvres $t_0^i \in T_0^i$ et $t_1^i \in T_1^i$. Ce minimum, noté $\delta_{min}(t_0^1, t_1^1, .., t_0^n, t_1^n)$ peut se définir de la manière suivante :

$$\delta_{min}(t_0^1, t_1^1, .., t_0^n, t_1^n) = \min_{\substack{i \in \{1..n\} \\ j \in \{1..n\},\, j<i}} \left( \inf_{t \in [0, t_f]} \left( \delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j) \right) \right) \tag{2.48}$$

Les avions sont deux à deux séparés sur la durée de l'évitement si et seulement si on a :

$$\delta_{min}(t_0^1, t_1^1, .., t_0^n, t_1^n) \geq n_h \tag{2.49}$$

Nous cherchons l'équivalent de la condition de séparation 2.49 quand les trajectoires des $n$ avions ne sont plus définies par des temps de manœuvres, mais par des intervalles de manœuvres.

Nous supposons toujours que les sens de déviation sont fixés.

Nous notons $T = (T_0^1, T_1^1, .., T_0^n, T_1^n)$ l'intervalle de $I(\mathbb{R}^{2n})$ qui donne les intervalles de manœuvres des avions.

A partir d'un équivalent en terme d'intervalles de l'équation 2.49 nous cherchons un critère permettant de savoir si un intervalle $T$, pour des sens de déviation donnés, est admissible, non admissible, ou indéterminé, au regard des contraintes de séparation (voir définition 2.9 de la section 2.3).

Deux grandeurs intéressantes pour juger de la qualité de l'intervalle $T$ pour la résolution du problème de l'évitement sont les bornes de $\delta_{min}(t_0^1, t_1^1, .., t_0^n, t_1^n)$ quand les valeurs de $t_0^i$ et $t_1^i$, pour $i \in \{1..n\}$, parcourent toutes les valeurs possibles pour elles dans les intervalles $T_0^i$. Rappelons qu'on doit avoir, pour tout $i$, $t_0^i \in T_0^i$, $t_1^i \in T_1^i$ et $t_0^i \leq t_1^i$.

Ces bornes sont notées $\delta_{min}^{inf}$ et $\delta_{min}^{sup}$ :

$$\delta_{min}^{inf}(T) = \inf_{\substack{t_0^1 \in T_0^1,\, t_1^1 \in T_1^1,\, t_0^1 \leq t_1^1 \\ \vdots \\ t_0^n \in T_0^n,\, t_1^n \in T_1^n,\, t_0^n \leq t_1^n}} \left( \min_{\substack{i \in \{1..n\} \\ j \in \{1..n\},\, j<i}} \left( \inf_{t \in [0,t_f]} \left( \delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j) \right) \right) \right) \tag{2.50}$$

et

$$\delta_{min}^{sup}(T) = \sup_{\substack{t_0^1 \in T_0^1,\, t_1^1 \in T_1^1,\, t_0^1 \leq t_1^1 \\ \vdots \\ t_0^n \in T_0^n,\, t_1^n \in T_1^n,\, t_0^n \leq t_1^n}} \left( \min_{\substack{i \in \{1..n\} \\ j \in \{1..n\},\, j<i}} \left( \inf_{t \in [0,t_f]} \left( \delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j) \right) \right) \right) \tag{2.51}$$

Par définition, $\delta_{min}^{inf}(T)$ est la plus mauvaise séparation possible au cours de l'évitement, pour les sens de déviations fixés pour les avions, si les trajectoires d'évitement possibles sont données par l'intervalle $T$ : on sait que deux avions ne s'approcheront jamais l'un de l'autre d'une distance inférieure à $\delta_{min}^{inf}(T)$. On sait en revanche que $T$ et les sens de déviation des avions rendent possible le fait que deux avions s'approchent l'un de l'autre jusqu'à ce que la distance entre eux soit arbitrairement proche de $\delta_{min}^{inf}(T)$.

De même, $\delta_{min}^{sup}(T)$ est la meilleure séparation possible au cours de l'évitement : il y aura toujours deux avions qui s'approcheront l'un de l'autre, au cours de l'évitement, d'une distance inférieure à $\delta_{min}^{sup}(T)$. Mais l'intervalle $T$ et les sens de déviation rendent possible le fait que les distances entre avions restent toujours, au cours de l'évitement, supérieures à une valeur arbitrairement proche de $\delta_{min}^{sup}(T)$.

Selon les positions de $\delta_{min}^{inf}(T)$ et $\delta_{min}^{sup}(T)$ par rapport à la borne de séparation $n_h$, $T$ sera donc admissible, non admissible ou indéterminé au sens des contraintes de séparation (définition 2.9) :

**Si $\boldsymbol{\delta_{min}^{inf}(T) \geq n_h}$ :** l'intervalle $T$ est admissible. En effet, au cours de l'évitement deux avions ne s'approcheront jamais l'un de l'autre d'une distance inférieure à $n_h$. Quelles que soient les trajectoires des avions, parmi celles qui sont décrites par l'intervalle $T$, tous les avions seront deux à deux séparés sur toute la durée de l'évitement.

**Si $\boldsymbol{\delta_{min}^{sup}(T) < n_h}$ :** l'intervalle $T$ est non admissible. Au cours de l'évitement, il arrivera toujours que deux avions s'approchent l'un de l'autre jusqu'à une distance inférieure à $\delta_{min}^{sup}(T)$, donc à $n_h$.

**Si $\boldsymbol{\delta_{min}^{inf}(T) < n_h < \delta_{min}^{sup}(T)}$ :** L'intervalle $T$ est indéterminé. Il n'est pas admissible, puisqu'il est possible que deux avions s'approchent l'un de l'autre jusqu'à une distance arbitrairement proche de $\delta_{min}^{inf}(T)$, donc jusqu'à une distance inférieure à $n_h$. Il n'est pas non admissible, puisqu'il est possible que les distances entre avions restent toujours supérieures à une distance arbitrairement proche de $\delta_{min}^{sup}(T)$.

Ces critères ne sont cependant pas directement utilisables par l'algorithme. Cependant, nous avons vu que l'on pouvait avoir une connaissance de l'effet des intervalles de manœuvres à un instant $t$ donné. Pour exploiter cette connaissance, nous serons amené à discrétiser l'intervalle de temps sur lequel est menée l'optimisation, et à minimiser sur l'ensemble des pas de temps, les séparations maximale et minimale entre avions correspondant aux intervalles de manœuvres à chaque pas de temps. Ces séparations minimale et maximale sont obtenues grâce aux ensembles des positions possibles des avions à chaque pas de temps calculés selon la méthode décrite dans les sections 2.5 et 2.6.

L'utilisation des critères donnés dans cette section nécessiterait de prendre un minimum sur le temps et sur les couples d'avions, puis de prendre des maximum ou minimum sur les intervalles de manœuvres, ce qui est en pratique impossible à traduire de manière numérique sans perdre l'information globale portée par les intervalles de manœuvres.

On pourra ainsi obtenir une valeur approchée pour $\delta_{min}^{inf}(T)$, alors qu'on n'obtiendra qu'un majorant de $\delta_{min}^{sup}(T)$.

### 2.7.3 Retards

Nous avons calculé, pour les deux modélisations de l'évitement que nous avons envisagées, le retard subi par l'avion $a^i$ s'il suit une trajectoire d'évitement définie par les temps de manœuvres $t_0^i$ et $t_1^i$ : Pour l'évitement par point tournant, on a l'expression suivante du retard, noté $r_{pt}^i(t_0, t_1)$ (voir équation 2.10, section 2.5.2) :

$$r_{pt}^i(t_0^i, t_1^i) = 2(t_1^i - t_0^i)(1 - \cos\alpha) \tag{2.52}$$

et pour l'évitement par offset, l'équation 2.32 de la section 2.6.2 donne l'expression suivante pour le retard, noté $r_{off}^i(t_0, t_1)$ :

$$r_{off}^i(t_0^i, t_1^i) = (t_1^i - t_0^i)(1 - \cos\alpha) \tag{2.53}$$

On remarque que le retard d'un avion, pour les mêmes valeurs de $t_0^i$ et $t_1^i$, est le même dans les deux cas, à un facteur 2 près. On notera dans la suite $r^i(t_0^i, t_1^i)$ le retard donné par l'équation 2.53. C'est cette valeur que l'algorithme s'attachera à minimiser quelle que soit la modélisation employée.

Quand on remplace les temps de manœuvres $t_0^i$ et $t_1^i$ par les intervalles de manœuvres $T_0^i$ et $T_1^i$, on cherche l'ensemble des retards possibles pour un avion dont la trajectoire d'évitement est définie par les intervalles $T_0^i$ et $T_1^i$.

On ne peut pas appliquer telle quelle la formule 2.2 donnée dans la section 2.2.2, c'est-à-dire remplacer, dans les équations 2.52 et 2.53, $t_1^i - t_0^i$, par $T_1^i - T_0^i$, au sens de la soustraction sur l'ensemble $I(\mathbb{R})$ des intervalles de $\mathbb{R}$. En effet, il faut tenir compte de la contrainte supplémentaire sur $t_0^i$ et $t_1^i$ :

$$t_0^i \leq t_1^i$$

On doit donc remplacer, dans l'équation 2.53, $t_1^i - t_0^i$ par l'intervalle $T_{0,1}^i$ défini de la manière suivante :

$$T_{0,1}^i = \{y = t_1^i - t_0^i;\ t_0^i \in T_0^i,\ t_1^i \in T_1^i\ et\ t_1^i \geq t_0\}$$

ce qui donne

$$T_{0,1}^i = [\max(0, t_{0min}^i - t_{0max}^i), t_{1max}^i - t_{0min}^i] \qquad (2.54)$$

Donc pour des intervalles de manœuvres $T_0^i$ et $T_1^i$, avec les deux modélisations envisagées, l'ensemble des retards possibles est un intervalle.

On note

$$r_{min}^i(T_0^i, T_1^i) = \max\left(0, (t_{1min}^i - t_{0max}^i)(1 - \cos\alpha)\right)$$

et

$$r_{max}^i(T_0^i, T_1^i) = (t_{1max}^i - t_{0min}^i)(1 - \cos\alpha)$$

L'ensemble des valeurs possibles de $r^i(t_0^i, t_1^i)$ est donc donné par l'intervalle $R^i(T_0^i, T^i 1)$ :

$$R^i(T_0^i, T_1^i) = [r_{min}^i(T_0^i, T_1^i), r_{max}^i(T_0^i, T_1^i)] \qquad (2.55)$$

.

## 2.8 Application de l'algorithme d'optimisation

Nous pouvons maintenant décrire l'algorithme d'optimisation utilisé.

Dans toute la suite, sauf précision contraire, la modélisation utilisée pour les trajectoires d'évitement sera la modélisation par point tournant.

Nous avons utilisé deux approches différentes du problème, l'approche *globale* et l'approche *séquentielle*. L'approche globale correspond à un système de contrôle *centralisé*, dans le cadre duquel les trajectoires de tous les avions sont générées de manière centralisée, tandis que l'approche séquentielle correspond à un système de contrôle *distribué*, ou *embarqué*, dans le cadre duquel chaque avion génère sa propre trajectoire d'évitement (voir section 1.1).

**Approche globale :** On optimise simultanément les trajectoires de tous les avions, de manière à obtenir un retard total (égal à la somme des retards de tous les avions) minimal, tout en assurant la séparation des avions.

**Approche séquentielle :** On cherchera toujours à obtenir $n$ trajectoires telles que les avions, s'ils suivent ces trajectoires, atteindront leurs destinations sans conflit. Cependant ces trajectoires seront générées les unes après les autres, en suivant les étapes suivantes :

1. On établit entre les avions un ordre de priorité. Une méthode permettant d'établir cet ordre de priorité est décrite dans la section 2.8.1. Supposons, par exemple, que l'ordre de priorité corresponde à la numérotation de avions ($a^1$ prioritaire, puis $a^2$, puis $a^3$, etc).

2. L'avion $a_1$, le plus prioritaire, n'est pas dévié.
3. On génère successivement des trajectoires pour les avions $a^2$ à $a^n$. Au moment de générer la trajectoire de l'avion $a^i$, les trajectoires générées pour les avions précédant $a^i$ dans l'ordre de priorité sont considérées comme définitivement fixées, et la trajectoire de l'avion $a^i$ doit être telle que :
   - elle assure que l'avion $a^i$ atteint sa destination en restant séparé de tous les avions $a^j$ pour $j < i$ ;
   - l'allongement de la trajectoire de l'avion $a^i$ est minimal, compte tenu de ces contraintes.

Au lieu de résoudre un seul problème d'optimisation sur un espace de recherche de grande dimension, on en résout successivement $n-1$ sur des espaces de dimensions beaucoup plus petites, comme nous le verrons dans la section 2.8.2.

Cependant, chacun de ces $n-1$ problèmes d'optimisation reste un problème difficile d'optimisation globale. Au moment de la génération de la trajectoire d'un avion, celles des avions qui le précèdent dans l'ordre de priorité imposent des contraintes pouvant être très sévères. Dans le cas de l'approche séquentielle, le plus difficile sera le plus souvent d'obtenir des instants de manœuvres définissant des trajectoires sans conflit. De plus l'ensemble des trajectoires possibles n'est pas connexe. Dans le cas de la génération d'une trajectoire pour un avion, quand les contraintes sont données par les trajectoires de $p$ avions, le nombre de composantes connexes de cet ensemble peut aller jusqu'à $2^{p+1}$ (deux sens de déviation possibles pour l'avion, et deux façons de croiser les autres avions, en passant devant ou en passant derrière). Ce problème est traité plus en détail dans [Méd94].

Contrairement à l'approche globale, l'approche séquentielle ne recherche pas d'optimum global pour le problème du conflit : en générant les trajectoires des avions successivement, on peut obtenir un retard total beaucoup plus important qu'en les générant toutes de manière centralisée.

L'approche séquentielle offre deux avantages :
- elle est plus adaptée à un système d'évitement embarqué (définition 1.10).
- elle est beaucoup plus rapide

Mais présente un inconvénient majeur :
- la qualité, l'existence même parfois, des trajectoires d'évitement dépend très fortement de l'ordre de priorité choisi.

Ces points seront détaillés dans la section 2.9.3.

Le fait d'utiliser la méthode globale ou la méthode séquentielle modifie la dimension de l'espace sur lequel est mené la recherche. Cependant, le fonctionnement de l'algorithme de *Branch and Bound* lui-même reste identique dans les deux cas. Dans la suite, nous allons décrire le fonctionnement de la version globale de l'algorithme, en précisant les points qui seront différents dans la version séquentielle.

### 2.8.1 Définition d'un ordre de priorité

Dans le cadre d'un système d'évitement embarqué, un avion ne disposera que de données locales. Nous supposons ici qu'un avion a une connaissance des avions situés à l'intérieur d'un cercle dont il est le centre, et dont le rayon est appelé *rayon de détection*.

Le problème que pose alors la définition d'un ordre de priorité entre avions, quel que soit le critère de priorité choisi, est que cet ordre de priorité doit s'appliquer à un ensemble d'avions, n'ayant pas tous deux à deux d'information l'un sur l'autre : il s'agit de faire émerger un ordre global d'informations locales.

Nous présentons dans cette section une méthode due à Géraud Granger et Jean-Marc Alliot permettant de définir un tel ordre. Cette méthode est présentée plus en détail et utilisée dans le rapport de DEA de Géraud Granger [Gra98], dans le cadre d'une application à des cas de conflit issus du trafic réel de la génération de trajectoires d'évitement par un algorithme $A^*$ décrite dans le chapitre 3 (voir section 3.9).

Le critère selon lequel un avion est plus prioritaire qu'un autre est indépendant de cette méthode. Le critère utilisé par Géraud Granger est basé sur les numéros CAUTRA[1] des avions (un numéro attribué à tout avion présent dans l'espace aérien) et la manœvrabilité d'un avion, notion définie par G. Granger, et qui tient compte des possibilités de manœuvres offertes à l'avion, selon l'étape de la trajectoire d'évitement à laquelle il se trouve (voir section 3.9). Le TCAS, par exemple, utilise le numéro CAUTRA des avions pour générer un ordre de priorité.

Chaque avion a la connaissance des avions en conflit qui sont dans son rayon de détection. Il connaît, pour ces avions et pour ces avions seulement son ordre de priorité, et chaque avion a son propre rayon de détection dans lequel il connaît son ordre de priorité.

Considérons l'exemple de la figure 2.5. L'avion 1 détecte un conflit avec l'avion 2 qui lui-même détecte un conflit avec les avions 1 et 3. L'avion 3 détecte un conflit avec les avions 2 et 4. L'avion 4 détecte un conflit avec l'avion 3.

L'avion 4 sait qu'il est moins prioritaire que l'avion 3 qui lui-même sait qu'il est plus prioritaire que l'avion 4 mais moins que l'avion 2. L'avion 2 sait qu'il est plus prioritaire que l'avion 3 mais moins que l'avion 1. L'avion 1 sait qu'il est plus prioritaire que l'avion 2.

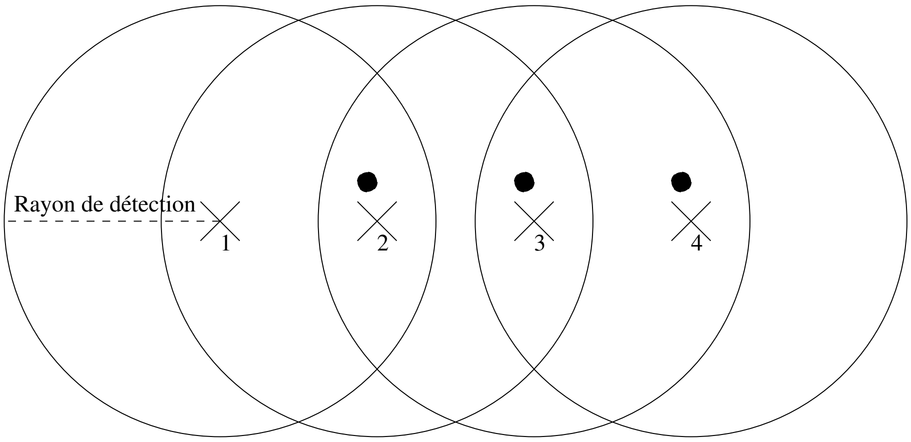



FIG. 2.5: *Visibilité entre* 4 *avions.*

Chaque avion n'a qu'une information partielle et doit attendre les décisions des avions plus prioritaires que lui pour pouvoir agir.

Pour résoudre ce problème, un système de distribution de jetons est utilisé, permettant à chaque

[1]Contrôle AUtomatisé du Trafic Aérien

avion de savoir s'il doit générer une trajectoire d'évitement car il est devenu le plus prioritaire, ou attendre que d'autres avions plus prioritaires fixent leur trajectoire.

Chaque avion distribue un jeton aux avions présents dans son rayon de détection qui sont en conflit avec lui-même ou avec tout autre avion, et qui sont moins prioritaires que lui (les avions n'étant pas en conflit ne reçoivent jamais de jeton). Ainsi, dans notre exemple, l'avion 1 donne un jeton à l'avion 2. L'avion 2 donne un jeton à l'avion 3 et l'avion 3 donne un jeton à l'avion 4.

La règle suivante est ensuite appliquée : dès qu'un avion n'a plus de jetons, il génère une trajectoire d'évitement en tenant compte des contraintes imposé par les trajectoires déjà fixées (celles des avions qui n'ont plus de jetons). Il ne tient pas compte des trajectoires des avions qui ont encore des jetons. Une fois cette opération réalisée, il reprend tous les jetons qu'il a donnés. Cette règle est appliquée jusqu'à ce que plus aucun avion n'ait de jeton.

Dans notre exemple, l'avion 1 n'a pas de jeton et optimise sa trajectoire en ne tenant compte que des avions de son rayon de détection ayant déjà résolu. L'avion 2 ayant un jeton, l'avion 1 n'en tient pas compte et ne modifiera donc pas sa trajectoire. L'avion 1 récupère donc le jeton qu'il a donné a l'avion 2.

On se trouve donc désormais dans la situation suivante :

- L'avion 1 a 0 jeton et a déjà résolu
- L'avion 2 a 0 jeton
- L'avion 3 a 1 jeton
- L'avion 4 a 1 jeton

C'est donc l'avion 2 qui doit désormais optimiser sa trajectoire en tenant compte des avions de son rayon de détection n'ayant pas de jeton, c'est-à-dire ici l'avion 1. L'avion 2 reprend ensuite son jeton à l'avion 3.

On se trouve alors dans la situation suivante :

- L'avion 1 a 0 jeton et a déjà résolu
- L'avion 2 a 0 jeton et a déjà résolu
- L'avion 3 a 0 jeton
- L'avion 4 a 1 jeton

L'avion 3 optimise ensuite sa trajectoire en tenant compte de l'avion 2 et reprend le jeton donné à l'avion 4. Enfin l'avion 4 optimise sa trajectoire en tenant compte de l'avion 3.

Il est important de remarquer que le nombre de jetons que détient chaque avion n'indique pas son temps d'attente. En effet, dans notre exemple, on constate qu'au départ, l'avion 2 et l'avion 4 ont chacun un jeton. Or l'avion 4 doit attendre que l'avion 2 et l'avion 3 aient optimisé leur trajectoire pour pouvoir optimiser la sienne. Un avion peut avoir beaucoup de jetons et pourtant un temps d'attente court. Prenons par exemple le cas de la figure 2.6. Les avions 1, 2 et 3 donnent chacun un jeton à l'avion 4 situé dans leur rayon de détection.

On se trouve alors dans la situation suivante :

- L'avion 1 a 0 jeton
- L'avion 2 a 0 jeton
- L'avion 3 a 0 jeton

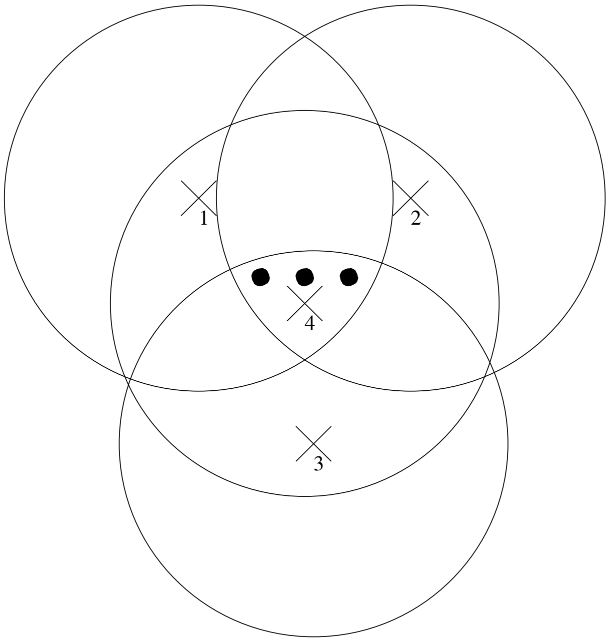



FIG. 2.6: *Répartition des jetons entre 4 avions.*

- L'avion 4 a 3 jetons

Les avions ayant 0 jeton vont optimiser simultanément leur trajectoire en ne tenant compte que des avions dans leur rayon de détection n'ayant pas de jeton. Dans le cas présent, ils n'ont donc pas de contrainte et ne modifient pas leurs trajectoires. Ils reprennent chacun leur jeton à l'avion 4 qui se retrouve alors sans jeton et optimise sa trajectoire en tenant compte des trajectoires des avions 1, 2 et 3.

Cette technique de distribution - résolution -récupération de jetons ne peut pas entraîner de situation de blocage du processus de résolution. On ne peut pas se trouver dans une situation où tous les avions auraient au moins un jeton, ni dans une situation où deux avions en conflit sans jeton devraient résoudre simultanément.

En effet, si l'ordre défini sur les avions est un ordre total, toute paire d'avion peut être comparée, et la transitivité permet de garantir qu'un avion parmi tous les avions en conflit est le plus prioritaire. Il n'a alors pas de jeton. Une fois que cet avion a résolu, il reprend ses jetons. L'avion le plus prioritaire parmi les avions restants ne peut alors plus avoir de jeton et peut donc résoudre. Ce processus se répète jusqu'à ce que tous les avions aient généré leur trajectoire et repris leurs jetons. Lorsque deux avions sont en conflit, l'un donne nécessairement un jeton à l'autre, qu'il est le seul à pouvoir récupérer. Cet autre avion ne peut donc pas se retrouver sans jeton tant que le premier n'a pas résolu. Deux avions en conflit potentiel l'un avec l'autre ne peuvent donc pas se trouver sans jeton simultanément.

Bien entendu, en pratique, il n'est pas question que les avions s'échangent des informations à chaque fois que la méthode décrite ci-dessus prévoit un *échange de jetons*. Ces échanges de jetons constituent simplement une représentation symbolique de la méthode d'attribution d'un rang dans l'ordre de priorité. Cette méthode est suivie par chaque avion, à partir des informations locales dont il dispose (c'est-à-dire les informations concernant les avions situés dans son rayon de détection).

Pour un avion donné, on dira que les avions situés dans son rayon de détection sont les avions qu'il *perçoit*.

Le mode de communication nécessaire peut se ramener à l'émission, par chaque avion, des informations à partir desquelles les autres avions sauront s'il est plus ou moins prioritaire qu'eux (information dépendant du critère choisi pour l'établissement de l'ordre de priorité, dans l'application présentée dans la section 3.9, il s'agira de son numéro CAUTRA et de l'étape de l'évitement à laquelle il est).

A partir des informations émise par les avions qu'il perçoit, l'avion connaît son ordre de priorité par rapport à eux (c'est son nombre de jetons).

Un avion qui n'a plus de jetons choisit sa trajectoire, et modifie les informations qu'il émet, pour signaler qu'il a fixé sa trajectoire et la transmettre. Les avions qui le perçoivent remettent alors à jour leur nombre de jetons, jusqu'à n'en avoir plus aucun, et fixer leur trajectoire à leur tour (on a vu que deux avions en conflit ne pouvaient pas se retrouver sans jeton simultanément).

Cette méthode ne nécessite donc pas de communications *bilatérales*, c'est-à-dire d'un avion à un autre, successives. Il suffit que chaque avion puisse diffuser les informations le concernant.

### 2.8.2 Variables de l'optimisation.

Dans le cadre de l'évitement par point tournant comme dans celui de l'évitement par offset, les trajectoires des avions sont déterminées par leurs temps de manœuvres $t_0$ et $t_1$ et par leurs directions de déviation, données par le signe des angles de déviation $\alpha_i$, dont les valeurs absolues sont toujours égales à $\alpha$ (Rappelons que l'absence de déviation se traduit par $t_1 = t_0$, et non par $\alpha = 0$).

C'est donc sur les temps de manœuvres et les sens de déviation que doit être menée l'optimisation. Les temps de manœuvres sont tels que l'on a $(t_0, t_1) \in [0, t_f]^2$.

Pour un problème à $n$ avions, traité dans le cadre de l'approche globale, l'espace sur lequel sera mené l'optimisation sera donc :

$$E^n_{opt} = ([0, t_f] \times [0, t_f] \times \{\alpha, -\alpha\})^n$$

On cherchera donc à minimiser une fonction $f^n_{opt}$ définie sur un sous ensemble $D^n_{opt}$ de l'ensemble $E^n_{opt}$. L'ensemble $D$ étant l'ensemble des points de $E^n_{opt}$ permettant de définir une trajectoire d'évitement, c'est-à-dire tels que l'on a $t_0 \leq t_1$ :

$$D^n_{opt} = \{\big((t^1_0, t^1_1, \alpha^1), .., (t^n_0, t^n_1, \alpha^n)\big) \mid t^i_0 \leq t^i_1,\ i = 1..n\}$$

Un point de $D^n_{opt}$ définit une trajectoire d'évitement pour chacun des $n$ avions impliqués dans le conflit. On dira qu'un point de $D^n_{opt}$ respecte les contraintes de séparation si, en suivant les trajectoires définies par ce point, les $n$ avions restent séparés deux à deux sur toute la durée de l'évitement.

Le fait qu'un point de $D$ respecte ou non les contraintes de séparation se traduit par la valeur de la fonction $f^n_{opt}$ en ce point :

Pour cela on définit un réel suffisamment grand, noté $G_r$ (nous allons préciser tout de suite ce que nous entendons par *suffisamment grand*).

Considérons un point $t_r = \big((t_0^1, t_1^1, \alpha^1), .., (t_0^n, t_1^n, \alpha^n)\big) \in D$.

- Si $t_r$ respecte les contraintes de séparation, c'est-à-dire définit des trajectoires d'évitement telles que si les avions suivent ces trajectoires, ils restent deux à deux séparés d'une distance supérieure ou égale à la norme de séparation $n_h$, la fonction $f_{opt}^n$ traduit le retard entraîné par ces trajectoires :

$$f_{opt}^n(t_r) = \sum_{i=1}^{n} t_1^i - t_0^i \tag{2.56}$$

- Si $t_r$ ne respecte pas les contraintes de séparation, on veut que la valeur de $f_{opt}^n$ en $(t_r)$ soit plus grande qu'en tous les points de $D_{opt}^n$ qui respectent les contraintes de séparation. On veut de plus que $f_{opt}^n(t_r)$ soit d'autant plus grand que $t_r$ viole plus ces contraintes, c'est-à-dire d'autant plus grand que la distance minimale entre deux avions au cours de l'évitement si les avions suivent les trajectoires définies par $t_r$ est petite. Cette distance minimale est calculée dans la section 2.7.2, c'est $\delta_{min}(t_r)$, définie par l'équation 2.48.

  On prend donc comme valeur de $f_{opt}^n(t_r)$ :

$$f_{opt}^n(t_r) = G_r - \delta_{min}(t_r) \tag{2.57}$$

  En fonction des données du problème, on choisit $G_r$ suffisamment grand pour que les valeurs de $f_{opt}^n$ sur les points ne respectant pas les contraintes de séparation soient plus grandes que ses valeurs sur les points les respectant.

L'utilisation de méthodes d'intervalles nécessite de traduire les variables du problème d'optimisation en termes d'intervalles. En ce qui concerne les temps de manœuvres, cette traduction se basera sur les ensembles des positions possibles des avions à un temps $t$ en fonction de leurs intervalles de manœuvres.

L'angle de déviation d'un avion ne pourra prendre que deux valeurs discrètes, $\alpha$ et $-\alpha$. Le fait de traduire cet angle de déviation, lui aussi, par des intervalles, ne permettrait plus d'obtenir une expression de l'ensemble des positions possibles d'un avion, à un instant $t$, utilisable dans le cadre de la méthode présentée ici.

On procédera donc de la manière suivante.

Comme nous l'avons vu dans la section 2.3, le principe d'un algorithme d'optimisation utilisant des méthodes d'intervalles est de manipuler une file de priorité contenant des intervalles ordonnés selon un critère que nous préciserons, cette file étant notée $f$.

Cette file d'intervalles est en général, au début de l'algorithme, réduite à un élément unique : le domaine sur lequel est menée l'optimisation, qui doit alors être un intervalle de $\mathbb{R}^n$.

Dans le cadre de l'approche globale du problème de l'évitement, pour des sens de déviation fixés, les intervalles manipulés par l'algorithme seraient des intervalles de $\mathbb{R}^{2n}$, correspondant aux intervalles de manœuvres de chacun des $n$ avions dont on cherche à générer des trajectoires d'évitement, c'est-à-dire de la forme $T = T_0^1 \times T_1^1 \times .. \times T_0^n \times T_1^n$. Le domaine sur lequel serait conduite l'optimisation serait $[0, t_f]^{2n}$. On a vu que, quand les sens de déviations ne sont pas fixés, le domaine sur lequel est conduit l'optimisation est : $E_{opt}^n = ([0, t_f] \times [0, t_f] \times \{\alpha, -\alpha\})^n$.

On définit l'ensemble $I^n_{opt}$ :

$$I^n_{opt} = (I([0, t_f]) \times I([0, t_f]) \times \{\alpha, -\alpha\})^n$$

L'ensemble $I([0, t_f])$ étant l'ensembles des intervalles de $\mathbb{R}$ inclus dans $[0, t_f]$.

Pour tenir compte du fait que deux sens de déviation sont possibles pour chaque avion, on initialise la file qui sera utilisée par l'algorithme $f$, non pas avec un seul intervalle $[0, t_f]^{2n}$, mais les $2^n$ éléments de $I^n_{opt}$ de la forme $(([0, t_f], [0, t_f], \alpha^1), .., ([0, t_f], [0, t_f], \alpha^n)$, correspondant chacun à une des $2^n$ combinaisons possibles pour les angles de déviations $\alpha^i$ des $n$ avions. Les éléments de $I^n_{opt}$ contenus dans cette file ne sont pas à strictement parler des intervalles, mais des intervalles associés à $n$ sens de déviation.

Dans toute la suite, nous continuerons cependant à appeler ces objets des intervalles, les sens de déviation correspondant à un élément donné de la file $f$ étant fixés, et l'algorithme ne manipulant pas d'intervalle qui ne soit associé aux angles de déviation correspondants.

On définit, à partir des définitions existant sur les intervalles de $I(\mathbb{R}^n)$ (définitions 2.1 et 2.2), la largeur et le point milieu d'un élément de $I^n_{opt}$ :

**Définition 2.14 – Largeur et point milieu d'un élément de $I^n_{opt}$ –** *On considère $X \in I^n_{opt}$, avec $X = (([t^1_{0min}, t^1_{0max}], [t^1_{1min}, t^1_{1max}], \alpha^1), .., ([t^n_{0min}, t^n_{0max}], [t^n_{1min}, t^n_{1max}], \alpha^n))$. On définit la $l(X)$, la largeur de $X$ de la manière suivante :*

$$l(X) = \max_{\substack{i \in \{1,..,n\} \\ j \in \{0,1\}}} (t^i_{jmax} - t^i_{jmin})$$

*et le point milieu de $X$, noté $m(X)$ est l'élément de $E^n_{opt}$ :*

$$m(X) = \left( \left( \frac{t^1_{0min} + t^1_{0max}}{2}, \frac{t^1_{1min} + t^1_{1max}}{2}, \alpha^1 \right), .., \left( \frac{t^n_{0min} + t^n_{0max}}{2}, \frac{t^n_{1min} + t^n_{1max}}{2}, \alpha^n \right) \right)$$

Dans la suite, pour éviter toute confusion avec la *largeur intérieure* d'un élément de $I^n_{opt}$, que nous définirons dans la section 2.8.6, on pourra employer le mot *taille* pour désigner la largeur d'un élément de $I^n_{opt}$.

Chacun des intervalles utilisés par l'algorithme est généré en divisant en deux un intervalle de la file manipulée par l'algorithme. Un intervalle de $I^n_{opt}$ sera divisé selon la méthode décrite dans la section 2.3 : c'est-à-dire en divisant un des intervalles de manœuvres $T^i_0$ ou $T^i_1$ d'un avion $a^i$, tous les autres intervalles de manœuvres demeurant inchangés, de même que tous les sens de déviation, y compris celui de l'avion dont un des intervalles de manœuvres a été divisé.

Ainsi, quand deux intervalles de $I^n_{opt}$ sont créés à partir d'un intervalle de la file manipulée par l'algorithme, les sens de déviation de tous les avions, pour chacun des deux intervalles générés, sont ceux qui correspondaient à l'intervalle à partir duquel ils ont été générés.

Les intervalles de manœuvres des avions, $T^i_0 = [t^i_{0min}, t^i_{0max}]$ ou $T^i_1 = [t^i_{1min}, t^i_{1max}]$ devront respecter des conditions portant sur leurs bornes (voir section 2.7.1) :

$$t^i_{0min} \leq t^i_{1max} \quad (2.58)$$
$$t^i_{0min} \leq t^i_{1min} \quad (2.59)$$
$$t^i_{0max} \leq t^i_{1max} \quad (2.60)$$

Après avoir divisé un intervalle de la file $f$, en divisant un des deux intervalles de manœuvres d'un avion $a^i$, on doit s'assurer que les conditions ci-dessus sont encore respectées :

- Si la condition 2.58 n'est pas respectée, l'intervalle issu de cette division est supprimé, c'est-à-dire n'est pas inséré dans la file $f$, ni dans aucune des deux *files résultats* qui seront définies dans la suite (section 2.8.5). En effet, aucun point de cet intervalle ne correspond à des temps de manœuvres définissant une trajectoire d'évitement possible pour l'avion $a^i$, c'est-à-dire respectant la condition $t_0 \leq t_1$.

- Si la condition 2.59 n'est pas respectée, on normalise les intervalles de manœuvres en prenant pour borne inférieure de $T^i_1$ : $t^i_{1min} = t^i_{0min}$. En effet les points de $T^i_1$ inférieurs à $t^i_{0min}$ ne permettent pas de satisfaire la condition $t_0 \leq t_1$.

- De même, si la condition 2.60 n'est pas respectée, on prend pour borne supérieure de $T^i_0$ : $t^i_{0max} = t^i_{1max}$.

Le fait de devoir tenir compte des $2^n$ combinaisons possibles pour les sens de déviations des $n$ avions allongera la file $f$ manipulée par l'algorithme dans le cadre de l'approche globale du problème de l'évitement.

Dans le cadre de l'approche séquentielle, on utilise $n$ fois successivement l'algorithme de *branch and bound*, mais pour générer à chaque fois la trajectoire d'un seul avion.

Quand on génère la trajectoire d'évitement d'un avion $a^{i_0}$, les trajectoires des avions $a^j$, pour $j < i_0$, sont déjà déterminées. Leurs sens de déviations et leurs intervalles de manœuvres sont fixés (ces intervalles de manœuvres seront en fait, comme nous le verrons dans la section 2.8.5, des intervalles dégénérés, c'est-à-dire que les intervalles de manœuvres correspondront de fait à des temps de manœuvre uniques). Ces intervalles et ces sens de déviation ne sont plus des variables de l'algorithme, ils sont utilisés à travers les contraintes de séparation qu'ils imposent à l'avion $a^i_0$. Rappelons qu'on ne tient pas compte des avions $a^k$, pour $k > i_0$.

Les intervalles manipulés par l'algorithme, dans sa version séquentielle, contiennent donc les seuls intervalles de manœuvres de l'avion $a^{i_0}$. Ce sont donc des intervalles de $\mathbb{R}^2$, inclus dans $[0, t_f]^2$. À chacun de ces intervalles est associé le sens de déviation de l'avion $a^{i_0}$.

La version séquentielle utilise donc les ensembles $E^1_{opt}$, $D^1_{opt}$, et $I^1_{opt}$, là où la version globale utilise les ensemble $E^n_{opt}$, $D^n_{opt}$, et $I^n_{opt}$.

La file $f$ contient donc, au début de l'algorithme générant la trajectoire de $a^{i_0}$, les deux éléments de $I^1_{opt}$ $([0, t_f], [0, t_f], \alpha)$ et $([0, t_f], [0, t_f], -\alpha)$. Les intervalles de la file $f$ seront divisés en divisant un des deux intervalles de manœuvres de l'avion $a^{i_0}$, en conservant à travers cette division le signe de $a^{i_0}$.

### 2.8.3 Intervalles admissibles, non admissibles et indéterminés

Nous supposerons, d'abord, que nous cherchons à générer les trajectoires de $n$ avions simultanément (approche globale du problème de l'évitement). Comme nous le verrons, le passage à l'approche séquentielle nécessitera peu de modifications.

Dans le cadre d'un algorithme de type *branch and bound* basé sur les méthodes d'intervalles, l'intervalle image d'un intervalle $X$ doit fournir un encadrement des valeurs prises par la fonction à optimiser sur l'intervalle $X$ (voir section 2.3). Compte tenu de l'introduction des contraintes de séparation dans la définition de la fonction $f_{opt}^n$, nous aurons besoin, pour pouvoir calculer un tel encadrement, de savoir si $X$ est admissible, non admissible, ou indéterminé. C'est l'objet de cette section.

Les avions sont notés $a^i$ pour $i = 1..n$. Dans la suite de cette section, les sens de déviation sont fixés. On considère un élément de $I_{opt}^n$, $X = \big((T_0^1, T_1^1, \alpha^1), .., (T_0^n, T_1^n, \alpha^n)\big)$, avec les notations $T_0^i = [t_{0min}^i, t_{0max}^i]$ et $T_1^i = [t_{1min}^i, t_{1max}^i]$ pour $i \in \{1, .., n\}$, qui sera dans cette section confondu avec l'intervalle de $I(\mathbb{R}^{2n})$ $T = (T_0^1, T_1^1, .., T_0^n, T_1^n)$ utilisé dans la section 2.7.2, les sens de déviation étant fixés.

Nous supposerons que les intervalles de manœuvre $T_0^i$ et $T_1^i$ de l'avion $a^i$ satisfont les conditions 2.42 à 2.44 de la section 2.7.1.

Considérons tout d'abord deux avions, $a^i$ et $a^j$.

Les positions des avions à un temps $t$ donné ne sont plus déterminées précisément, par des points. On notera, pour tout $t \in [0, t_f]$, $P^i(t, T_0^i, T_1^i)$ (resp. $P^j(t, T_0^j, T_1^j)$) l'ensemble des positions possibles de l'avion $a^i$ (resp. $a^j$) à l'instant $t$, en fonction de son sens de déviation et de ses intervalles de manœuvres donnés par l'intervalle $T$.

On définit, pour $t \in [0, t_f]$ :

$$\rho_{min}^{i,j}(t, T_0^i, T_1^i, T_0^j, T_1^j) = \min_{\substack{p \in P^i(t, T_0^i, T_1^i) \\ q \in P^j(t, T_0^j, T_1^j)}} (d(p, q)) \tag{2.61}$$

et

$$\rho_{max}^{i,j}(t, T_0^i, T_1^i, T_0^j, T_1^j) = \max_{\substack{p \in P^i(t, T_0^i, T_1^i) \\ q \in P^j(t, T_0^j, T_1^j)}} (d(p, q)) \tag{2.62}$$

D'un point de vue géométrique, ces définitions signifient que $\rho_{min}^{i,j}(t, T_0^i, T_1^i, T_0^j, T_1^j)$ est la distance entre $P^i(t, T_0^i, T_1^i)$ et $P^j(t, T_0^j, T_1^j)$, et que $\rho_{sup}^{i,j}(t, T_0^i, T_1^i, T_0^j, T_1^j)$ est le diamètre de $P^i(t, T_0^i, T_1^i) \cup P^j(t, T_0^j, T_1^j)$.

Par définition de $P^i(t, T_0^i, T_1^i)$ et $P^j(t, T_0^i, T_1^i)$, $\rho_{min}^{i,j}(t, T_0^i, T_1^i, T_0^j, T_1^j)$ est donc le minimum, et $\rho_{max}^{i,j}(t, T_0^i, T_1^i, T_0^j, T_1^j)$ le maximum, de l'ensemble des distances possibles entre les avions $a^i$ et $a^j$ au temps $t$, puisque, à cet instant, $a^i$ (resp. $a^j$) peut occuper n'importe quel point dans $P^i(t, T_0^i, T_1^i)$ (resp. $P^j(t, T_0^j, T_1^j)$).

Les ensembles $P^i(t, T_0^i, T_1^i)$ et $P^j(t, T_0^j, T_1^j)$ étant fermés et bornés, ces minima et ces maxima sont atteints.

On définit ensuite :

$$d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) = \inf_{t\in[0,t_f]}(\rho_{min}^{i,j}\left(t, T_0^i, T_1^i, T_0^j, T_1^j)\right) \quad (2.63)$$

et

$$d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) = \inf_{t\in[0,t_f]}\left(\rho_{max}^{i,j}(t, T_0^i, T_1^i, T_0^j, T_1^j)\right) \quad (2.64)$$

Il découle des définitions 2.61 et 2.63 que $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ est la borne inférieure des distances possibles entre les avions $a^i$ et $a^j$ au cours de l'évitement c'est-à-dire qu'il est possible, pour tout $\epsilon > 0$, qu'à un moment de l'évitement la distance entre les avions $a^i$ et $a^j$ soit inférieure à $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) + \epsilon$, et qu'il n'est possible à aucun moment de l'évitement que la distance entre $a^i$ et $a^j$ soit inférieure à $d_{inf}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$.

Ce qui signifie que :

- Si $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) \geq n_h$, alors on est sûr que les avions $a^i$ et $a^j$ seront séparés sur toute la durée de l'évitement.
- Si $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) < n_h$, alors il est possible que les avions $a^i$ et $a^j$ ne soient pas toujours séparés sur la durée de l'évitement.

La signification de $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ est plus délicate à établir. D'après les définitions 2.62 et 2.64, $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ est la borne inférieure des distances maximales possibles à tout moment de l'évitement entre les avions $a^i$ et $a^j$, c'est-à-dire que pour tout instant $t \in [0, t_f]$, il existe un couple $(t_0^i, t_1^i) \in T_0^i \times T_1^i$ et un couple $(t_0^j, t_1^j) \in T_0^j \times T_1^j$ tels que si $t_0^i$ et $t_1^i$ sont les temps de manœuvres de l'avion $a^i$ et et $t_0^j$ et $t_1^j$ ceux de $a^j$, on a :

$$d\left(p^i(t), p^j(t)\right) \geq d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) \quad (2.65)$$

et que pour tout $\epsilon > 0$, il existe un instant $t \in [0, t_f]$ et un couple $(t_0^j, t_1^j)) \in T_0^j \times T_1^j$ tels que si $t_0^i$ et $t_1^i$ étaient les temps de manœuvres de l'avion $a^i$ et et $t_0^j$ et $t_1^j$ ceux de $a^j$, on aurait, en notant $p^i(t)$ la position qu'aurait alors l'avion $a^i$ et $p^j(t)$ celle qu'aurait alors l'avion $a^j$ au temps $t$ :

$$d_{max}^{i,j} \leq d\left(p^i(t), p^j(t)\right) < d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) + \epsilon \quad (2.66)$$

Ce qui signifie que :

- Si $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) < n_h$, alors on est sûr que les avions $a^i$ et $a^j$ ne seront pas toujours séparés sur la durée de l'évitement.
- Si $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) \geq n_h$, cela n'implique pas qu'il est possible que les avions $a^i$ et $a^j$ soient séparés sur toute la durée de l'évitement. En effet, il faut, pour que les avions soient séparés sur toute la durée du conflit, qu'il existe deux couples $(t_0^i, t_1^i)$ et $(t_0^j, t_1^j)$ de temps de manœuvres pour les avions tels que l'inégalité 2.65 tienne pour tout $t \in [0, t_f]$, ce qui n'est pas le cas, les temps de manœuvres permettant d'obtenir 2.65 dépendant de $t$.

Ces restrictions sur la signification de $d^{i,j}_{max}(T^i_0, T^i_1, T^j_0, T^j_1)$ sont liées au fait que l'utilisation de la méthode présentée ici impose de réaliser les minimisation et maximisation successives dans un ordre différent de celui qui a été employé dans la section 2.7.2.

Ce sont les valeurs de $d^{i,j}_{min}(T^i_0, T^i_1, T^j_0, T^j_1)$ et $d^{i,j}_{max}(T^i_0, T^i_1, T^j_0, T^j_1)$ qui fournissent un équivalent de la condition de séparation 2.46 entre les avions $a^i$ et $a^j$.

- La séparation sera toujours assurée si $n_h \leq d^{i,j}_{min}(T^i_0, T^i_1, T^j_0, T^j_1)$.
- Si $n_h > d^{i,j}_{max}(T^i_0, T^i_1, T^j_0, T^j_1)$, on est sûr qu'un conflit aura lieu entre les avions $a^i$ et $a^j$.
- Si $d^{i,j}_{min}(T^i_0, T^i_1, T^j_0, T^j_1) < n_h \leq d^{i,j}_{max}(T^i_0, T^i_1, T^j_0, T^j_1)$ on sait qu'il existe dans les intervalles de manœuvres des temps de manœuvres qui mènent à un conflit entre $a^i$ et $a^j$, mais on ne sait pas s'il existe des temps de manœuvres permettant à $a^i$ et $a^j$ de ne jamais être en conflit sur la durée de l'évitement.

On définit, à partir des $d^{i,j}_{min}(T^i_0, T^i_1, T^j_0, T^j_1)$ et $d^{i,j}_{max}(T^i_0, T^i_1, T^j_0, T^j_1)$ :

$$d_{min}(T) = \min_{\substack{i \in \{1..n\} \\ j \in \{1..n\},\ j<i}} \left( d^{i,j}_{min}(T^i_0, T^i_1, T^j_0, T^j_1) \right) \tag{2.67}$$

et

$$d_{max}(T) = \min_{\substack{i \in \{1..n\} \\ j \in \{1..n\},\ j<i}} \left( d^{i,j}_{max}(T^i_0, T^i_1, T^j_0, T^j_1) \right) \tag{2.68}$$

**Définition 2.15 – Intervalles de séparation –** *Dans la suite de ce chapitre, on appellera intervalle de séparation l'intervalle $[d_{min}(T), d_{max}(T)]$, où $d_{min}(T)$ et $d_{max}(T)$ sont donnés par les équations ci-dessus.*

D'après les équations 2.63 et 2.61, de la section 2.7.2, $d_{min}$ peut s'écrire :

$$d_{min}(T) = \min_{\substack{i \in \{1..n\} \\ j \in \{1..n\},\ j<i}} \left( \inf_{t \in [0,t_f]} \left( \inf_{\substack{t^1_0 \in T^1_0,\ t^1_1 \in T^1_1,\ t^1_0 \leq t^1_1 \\ \vdots \\ t^n_0 \in T^n_0,\ t^n_1 \in T^n_1,\ t^n_0 \leq t^n_1}} \left( \delta^{i,j}(t, t^i_0, t^i_1, t^j_0, t^j_1) \right) \right) \right) \tag{2.69}$$

Ce qui revient à l'expression de $\delta^{inf}_{min}(T)$ donnée par l'équation 2.50 (on peut inverser l'ordre des minimisations successives). On a donc :

$$d_{min}(T) = \delta^{inf}_{min}(T). \tag{2.70}$$

$d_{min}$ est donc la distance minimale possible entre deux avions au cours de l'évitement si les trajectoires possibles pour les avions sont données par l'intervalle $T$.

De même, d'après les équations 2.64 et 2.62 de la section 2.7.2, $d_{max}$ peut s'écrire :

$$d_{max}(T) = \min_{\substack{i\in\{1..n\}\\ j\in\{1..n\},\, j<i}} \left( \inf_{t\in[0,t_f]} \left( \sup_{\substack{t_0^1\in T_0^1,\, t_1^1\in T_1^1,\, t_0^1\le t_1^1 \\ \vdots \\ t_0^n\in T_0^n,\, t_1^n\in T_1^n,\, t_0^n\le t_1^n}} \left( \delta^{i,j}(t, t_0^i, t_1^i, t_0^j, t_1^j) \right) \right) \right) \tag{2.71}$$

On déduit des équations 2.71 et 2.51 que :

$$d_{max}(T) \geq \delta_{min}^{sup}(T) \tag{2.72}$$

$d_{max}(T)$ peut être sensiblement plus grand que $\delta_{min}^{sup}(T)$ ; il s'agit bien d'un majorant, et non d'une valeur approchée de $\delta_{min}^{sup}(T)$. Cependant, comme nous le verrons dans la suite de cette section, nous pourrons facilement calculer une valeur approchée de $d_{max}(T)$, ce qui n'est pas possible pour $\delta_{min}^{sup}(T)$.

Il est possible en effet de calculer des valeurs approchées des distances $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ et $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ en procédant de la manière suivante :

1. Le temps est discrétisé, c'est-à-dire divisé en un nombre fini de pas de temps de longueur constante (pour plus de simplicité, on emploiera l'expression *pas de temps* pour désigner un instant correspondant à un multiple du pas de temps : on dira ainsi qu'on est *sur un pas de temps*, ou bien *entre deux pas de temps*), on note $N_t$ le nombre de pas de temps. La longueur d'un pas de temps est $l_t = t_f/(N_t - 1)$, et pour $k \in \{0, .., N_t\}$, $t_k = k\ l_t$.
2. Pour chaque pas de temps $t_k$, à partir des ensembles $P^i(t)$ et $P^j(t)$, construit comme dans la section 2.5, on calcule $\rho_{min}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j)$ et $\rho_{max}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j)$ tels qu'il sont définis par 2.61 et 2.62.

   Quand les trajectoires utilisées pour l'évitement sont des trajectoires d'évitement par point tournant, chacun des ensembles $P^i(t_k)$ et $P^j(t_k)$ est, comme on l'a vu dans la section 2.5, soit un point, soit un segment de droite, soit un parallélogramme, soit la réunion d'un parallélogramme tronqué (qui est un polygone convexe du plan) et d'un segment. Dans tout ces cas, $\rho_{min}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j)$ et $\rho_{max}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j)$ sont simples et rapides à calculer.

   Quand on utilise pour les trajectoires d'évitement la modélisation par offset, les ensembles $P^i(t_k)$ et $P^j(t_k)$ sont soit des points soit des segments de droite (voir section 2.6). Les distances $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)(t_k)$ et $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)(t_k)$ sont donc, aussi dans ce cas, très simples et rapides à calculer .
3. On calcule ensuite :

$$dist_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) = \min_{k\in\{0,..,N_t\}} \left( \rho_{min}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j) \right) \tag{2.73}$$

   et

$$dist_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) = \max_{k\in\{0,..,N_t\}} \left( \rho_{max}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j) \right) \tag{2.74}$$

Il faut noter que $dist_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ et $dist_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ ne sont que des approximations de $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ et $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$, obtenues à partir des valeurs sur les pas de temps de $\rho_{min}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j)$ et $\rho_{min}^{i,j}(t_k, T_0^i, T_1^i, T_0^j, T_1^j)$.

**Notation –** *Dans toute la suite, les valeurs exactes des bornes inférieures ou supérieures sur les distances entre avions au cours de l'évitement seront désignées par la lettre "d" portant différents indices, tandis que les valeurs approchées correspondantes, calculées et manipulées par l'algorithme seront désignées par "$dist$", avec les indices correspondant aux valeurs exactes qu'elles approchent.*

On définit les minima de $dist_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ et $dist_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ sur les couples $(i, j)$ :

$$dist_{min}(T) = \min_{\substack{i\in\{1..n\} \\ j\in\{1..n\},\ j<i}} \left(dist_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)\right) \tag{2.75}$$

et

$$dist_{max}(T) = \min_{\substack{i\in\{1..n\} \\ j\in\{1..n\},\ j<i}} \left(dist_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)\right) \tag{2.76}$$

On peut calculer la précision de ces approximations en fonction de l'espacement entre deux pas de temps, noté $e_t$, de la vitesse de l'avion $a^i$, notée $v^i$, et de celle de l'avion $a^j$, notée $v^j$. On calcule d'abord

$$e_d^{i,j} = \frac{(v^i + v^j)e_t}{2} \tag{2.77}$$

$e_d^{i,j}$ est la distance que parcourraient l'un vers l'autre les avions $a^i$ et $a^j$, s'ils volaient droit l'un sur l'autre, pendant une durée égale à la moitié de l'espacement entre les pas de temps. C'est donc une borne supérieure pour l'accroissement ou la diminution de la distance entre ces deux avions sur un demi pas de temps.

Donc, si pour un temps $t \in [0, t_f]$ donné, on note $t_k$ le pas de temps le plus proche de $t$ (on a alors $|t - t_k| \leq e_t/2$), pour toute distance possible $d_{poss}^{i,j}(t)$ entre les avions $a^i$ et $a^j$ au temps $t$, il y aura toujours une distance possible $d_{poss}^{*i,j}(t_k)$ entre les avions sur le pas de temps $t_k$ telle que $|d_{poss}^{i,j}(t) - d_{poss}^{*i,j}(t_k)| \leq e_d^{i,j}$.

On obtient ainsi, d'après les définitions de $dist_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$, $dist_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$, $d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ et $d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j)$ :

$$dist_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) - e_d^{i,j} \leq d_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) \leq dist_{min}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) \tag{2.78}$$

et

$$dist_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) - e_d^{i,j} \leq d_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) \leq dist_{max}^{i,j}(T_0^i, T_1^i, T_0^j, T_1^j) \tag{2.79}$$

Après avoir calculé le maximum des $e_d^{i,j}$ sur tous les couples d'avions impliqués dans le conflit, noté $e_{dmax}$ :

$$e_{dmax} = e_t(\max_{i \in \{1..n\}} v^i) \tag{2.80}$$

On a, d'après les inéquations 2.78 et 2.79 de la section 2.7.2 :

$$dist_{min}(T) - e_{dmax} \leq d_{min}(T) \leq dist_{min}(T) \tag{2.81}$$

et

$$dist_{max}(T) - e_{dmax} \leq d_{max}(T) \leq dist_{max}(T) \tag{2.82}$$

On définit à partir des équations 2.75 et 2.76, l'intervalle de séparation approché :

**Définition 2.16 – Intervalle de séparation approché –** *Dans la suite de ce chapitre, on appellera intervalle de séparation approché l'intervalle* $[dist_{min}(T), dist_{max}(T)]$, *dont les bornes sont données par les équations ci-dessus.*

C'est en fonction des valeurs de $dist_{max}$ et de $dist_{min}$, que $T$ sera considéré, dans le cadre de l'algorithme d'optimisation, comme étant admissible, non admissible ou indéterminé au sens des contraintes de séparation (voir définition 2.9).

Dans ce but nous introduisons une norme de séparation majorée :

**Définition 2.17 – Norme de séparation majorée –** *À partir de la norme de séparation horizontale, notée* $n_h$, *on définit une norme de séparation majorée, notée* $n_h^*$, *par :*

$$n_h^* = n_h + e_{dmax} \tag{2.83}$$

Cette majoration de $e_{dmax}$ de la norme de séparation peut, pour un espacement entre pas de temps suffisamment faible, rester très raisonnable : si l'espacement entre deux pas de temps est de 10 secondes et la vitesse des avions de 400 kt, on obtient une majoration de la norme de séparation de l'ordre de 1 NM.

Les trois cas suivants sont possibles :

**Si $dist_{min} \geq n_h^*$ :** D'après les inéquations 2.81, on a alors : $d_{min} \geq n_h$, c'est-à-dire, comme $d_{min} = \delta_{min}^{inf}(T)$,

$$\delta_{min}^{inf}(T) \geq n_h$$

L'intervalle $T$ assure donc que les avions sont séparés deux à deux sur toute la durée de l'évitement : l'intervalle $T$ est admissible au sens des contraintes de séparation.

**Si $dist_{max} < n_h$ :** D'après les inéquations 2.82, on a :

$$d_{max} < n_h$$

et donc, comme $d_{max} \geq \delta_{min}^{sup}(T)$,

$$\delta_{min}^{sup}(T) < n_h$$

$T$ est donc non admissible au sens des contraintes de séparation.

**Si $dist_{min} < n_h^*$ et $n_h \geq dist_{max}$ :** Comme $dist_{min} < n_h^*$, on n'est pas assuré de respecter la condition $\delta_{min}^{inf}(T) < n_h$, ce qui entraîne que l'intervalle $T$ ne peut pas être considéré comme étant admissible.

On a $dist_{max} \geq n_h$ et $dist_{max} \geq \delta_{min}^{sup}(T)$. On ne sait donc pas si $\delta_{min}^{sup}(T) < n_h$ : on ne sait en fait pas si l'intervalle $T$ est non admissible ou indéterminé.

Cependant, puisqu'il est impossible d'établir, d'après les grandeurs qui seront calculées dans le cadre de l'algorithme d'optimisation, si l'intervalle $T$ est ou non non admissible, il sera considéré comme étant indéterminé.

### 2.8.4 Intervalle image et estimateurs d'un intervalle

Nous pouvons maintenant calculer l'intervalle image, ainsi qu'un estimateur, d'un intervalle $T$.

Un estimateur de $T$ donne une valeur atteinte sur l'intervalle $T$ par la fonction que l'on cherche à optimiser. L'intervalle image d'un intervalle fournit un encadrement des valeurs prises, sur cet intervalle, par cette fonction.

Cet encadrement permet de savoir si l'optimum recherché peut ou non se trouver dans l'intervalle $T$ (voir section 2.3).

Les intervalles images et les estimateurs serviront donc à décider d'insérer ou non chacun des intervalles traités par l'algorithme dans une file d'intervalles, et à ordonner cette file d'intervalles. Les intervalles intéressant au regard du problème à résoudre (tels qu'il soit possible qu'ils contiennent l'optimum cherché) seront insérés dans cette file, les autres seront éliminés (voir section 2.3).

L'ordre des intervalles dans cette file induira l'ordre dans lequel les intervalles seront *explorés* par l'algorithme.

Il s'agit donc de définir les intervalles images et les estimateurs des intervalles de manière à ce que ces intervalles images et ces estimateurs permettent de retenir les intervalles intéressants, en les ordonnant, et d'éliminer les autres.

**Intervalles admissibles :** Les intervalles admissibles sont les intervalles sur lesquels les contraintes de séparation sont toujours satisfaites. Leur intérêt pour le problème de l'évitement est directement lié aux retards qu'ils entraînent.

À partir de l'intervalle $R^i(T_0^i, T_1^i) = [r_{min}^i(T_0^i, T_1^i), r_{max}^i(T_0^i, T_1^i)]$, défini pour chaque avion $a_i$ (voir section 2.7.3), on définit :

$$r_{min}(T) = \sum_{i=1}^{n} \left(r_{min}^i(T_0^i, T_1^i)\right) \tag{2.84}$$

et :

$$r_{max}(T) = \sum_{i=1}^{n} \left(r_{max}^i(T_0^i, T_1^i)\right) \tag{2.85}$$

Les significations de $r_{min}(T)$ et de $r_{max}(T)$ sont claires : ce sont les minimum et maximum possibles de la somme des retards des avions. On connaît dans ce cas la plus petite valeur possible des retards : $r_{min}(T)$, qu'on prend pour estimateur de $T$.

L'intervalle image de $T$ devrait être l'intervalle $R(T) = [r_{min}(T), r_{max}(T)]$. Cependant, on connaît, pour chaque avion $a^i$, les temps de manœuvres correspondant au minimum du retard possible (voir section 2.7.3) :

**Si $t^i_{0max} < t^i_{1min}$ :** le retard minimal pour l'avion $a^i$ est obtenu pour $t^i_0 = t^i_{0max}$ et $t^i_1 = t^i_{1min}$.

**Si $t^i_{0max} \geq t^i_{1min}$ :** le retard minimal (d'ailleurs nul) pour l'avion $a^i$ est obtenu pour les temps de manœuvres $t^i_0$ et $t^i_1$ tels que :

$$t^i_{1min} \leq t^i_0 = t^i_1 \leq t^i_{0max}$$

on prendra dans ce cas :

$$t^i_0 = t^i_1 = t^i_{1min}$$

On peut donc dans ce cas *réduire* l'intervalle $T$ à un intervalle dégénéré, l'intervalle : $T' = (t^1_0, t^1_1, .., t^n_0, t^n_1)$ où les valeurs de $t^i_0$ et $t^i_1$ on été choisies comme il est décrit ci-dessus de manière à entraîner le retard minimal, et l'intervalle image de $T'$ devient donc l'intervalle dégénéré $[r_{min}(T), r_{min}(T)]$.

**Intervalles non admissibles :** Les intervalles non admissibles sont des intervalles sur lesquels les contraintes de séparation ne sont jamais respectées.

Ils ne seront donc pas retenus par l'algorithme, il est inutile de définir leur intervalle image et leur estimateur.

**Intervalles indéterminés :** Nous avons vu dans la section précédente qu'un intervalle considéré comme indéterminé pouvait contenir des points respectant les contraintes de séparation et des points ne les respectant pas.

L'intervalle image d'un intervalle indéterminé $T$ doit fournir un encadrement de l'image de cet intervalle par la fonction $f^n_{opt}$ définie dans la section 2.8.2.

La borne inférieure de cet intervalle image doit fournir un minorant des retards entraînés par les points respectant les contraintes de séparation éventuellement contenus par l'intervalle $T$. On prendra donc, pour cette borne inférieure, $r_{min}(T)$, calculée de la même manière que si $T$ était admissible.

La borne supérieure de cet intervalle image doit fournir un majorant de la valeur de $f^n_{opt}$ qui traduirait la plus mauvaise séparation assurée par les points ne respectant pas les contraintes de séparation éventuellement contenus dans l'intervalle $T$.

Or la plus mauvaise séparation assurée par un point de l'intervalle $T$ est comprise entre $dist_{min} - e_{dmax}$ et $dist_{min}$ (voir section 2.8.3).

La valeur de $f^n_{opt}$ est donc toujours inférieure, en tout point de $T$ ne respectant pas les contraintes de séparation, à $G_r - dist_{min} + e_{dmax}$ (voir équation 2.57).

L'intervalle image de l'intervalle $T$ est donc $[r_{min}(T), G_r - dist_{min} + e_{dmax}]$.

Le calcul d'un estimateur de l'intervalle indéterminé $T$ est plus délicat : en effet, on sait que si $T$ contient un point respectant les contraintes de séparation, alors la valeur de la fonction $f^n_{opt}$ en ce point est égale au retard entraîné par ce point. Et on sait que ce retard est inférieur à $r_{max}(T) = \sum_{i=1}^{n}(r_{max}^i(T_0^i, T_1^i))$ (voir section 2.7.3). S'il était sûr que l'intervalle $T$ contînt au moins un point respectant les contraintes de séparation, $r_{max}(T)$ conviendrait comme estimateur de $T$.

Cependant, il est possible que $T$ ne contienne pas de tel point. On procède donc de la manière suivante :

- On choisit (aléatoirement, de manière uniforme) un point $t_{est}$ dans l'intervalle $T$, c'est-à-dire des temps de manœuvres $t_0^i \in T_0^i$ et $t_1^i \in T_1^i$ pour chaque avion $a^i$ (les sens de déviation sont imposés par $T$).
- On calcule $dist_{min}(t_{est})$, la valeur de $dist_{min}$ sur l'intervalle dégénéré réduit au point $t_{est}$, comme on l'a vu dans la section 2.8.3, c'est-à-dire en discrétisant le temps. Ce calcul est facilité du fait que les positions possibles pour chaque avion à un instant donné sont réduites à un point, puisque les intervalles de manœuvre des avions sont eux mêmes dégénérés. On obtient ainsi une valeur approchée de $d_{min}(t_{est})$, la distance minimale entre deux avions au cours de l'évitement s'ils suivent les trajectoires déterminées par $t_{est}$.
- Si $dist_{min}(t_{est}) \geq n_h^*$, alors le point $t_{est}$ respecte les contraintes de séparation, et on prend comme estimateur $est(t)$ de $T$ la somme des retards entraînés par les trajectoires déterminées par $t_{est}$.
- Si $dist_{min}(t_{est}) < n_h^*$, alors le point $t_{est}$ peut ne pas respecter les contraintes de séparation, et on prend dans ce cas comme valeur pour l'estimateur de $T$ le réel $est(T) = G_r - dist_{min}(t_{est}) + e_{dmax}$, qui est le plus petit majorant connu de la plus petite valeur de $f_{opt}^n$ sur $T$.

Comme nous le verrons dans la section 2.8.5, les intervalles manipulés par l'algorithme sont, à part les intervalles ayant servis à initialiser la file d'intervalles manipulée par l'algorithme, issus de la coupure en deux intervalles d'un intervalle extrait de cette file.

Dans ce cas, il est possible que l'intervalle $T$ contienne le point en lequel a été calculé l'estimateur de l'intervalle dont il est issu. Si l'estimateur de cet intervalle, que nous notons $est_p$ est meilleur (plus petit) que l'estimateur calculé pour $t$ selon la méthode décrite ci-dessus, on prend pour estimateur de $T$ : $est(T) = est_p$.

Cela oblige à conserver en mémoire, à travers la coupure d'un intervalle indéterminé, la valeur de l'estimateur de cet intervalle et le point en lequel il a été calculé.

Cette précaution n'est pas nécessaire au fonctionnement de l'algorithme. En effet, même sans elle, on sera assuré que si l'algorithme rencontre un estimateur de meilleure qualité que ceux qui ont été calculés jusqu'alors, il conservera toujours un intervalle contenant le point où il a été calculé dans la file d'intervalles qu'il manipule, tant qu'un meilleur estimateur n'aura pas été rencontré, puisque la borne inférieure de l'intervalle image d'un tel intervalle sera toujours inférieure ou égale à cet estimateur (section 2.8.5). Nous verrons que dans le cadre de l'élagage, que nous introduirons dans la section 2.8.5, le fait de garder en mémoire le point en lequel a été calculé le meilleur estimateur rencontré sera nécessaire au fonctionnement de l'algorithme.

Cependant, comme nous le verrons, un intervalle indéterminé suffisamment petit dont l'estimateur aura été calculé en un point respectant les contraintes de séparation pourra être réduit à un intervalle dégénéré correspondant à ce point. Cette précaution évite de perdre de tels estimateurs, et de pouvoir, le cas échéant, proposer comme solution le point en lequel a été calculé le meilleur estimateur rencontré par l'algorithme.

Nous avons défini les intervalles images et les estimateurs d'un élément de $I^n_{opt}$ du type $T = ((T^1_0, T^1_1, \alpha^1), .., (T^n_0, T^n_1, \alpha^n))$, dans le cadre de l'approche globale du problème de l'évitement. Dans le cadre de l'approche séquentielle, les intervalles manipulés par l'algorithme pour générer la trajectoire d'un avion $a^{i_0}$ sont des éléments de $I^1_{opt}$, du type $T = (T^{i_0}_0, T^{i_0}_1, \alpha^{i_0})$. On ne prend en compte que le retard de l'avion $a^{i_0}$, et on ne considère, sur la durée de l'évitement, que la distance entre l'avion $a^{i_0}$ et les avions $a^j$, pour $j < i_0$.

Ainsi, la définition de $d_{min}$ (voir équation 2.67) devient :

$$d_{min} = \min_{\substack{j \in \{1..n\} \\ j < i_0}} d^{i,j}_{min} \tag{2.86}$$

et toute les définition de distances qui qui étaient des minima, sur tous les couples $(a^i, a^j)$, de distances entre avions, deviennent ainsi des minima sur les couples $(a^{i_0}, a^j)$, pour $j < i_0$.

À ces modifications près les intervalles images et estimateurs des intervalles se calculent de la même manière dans le cadre de l'approche séquentielle que dans le cadre de l'approche globale du problème de l'évitement.

### 2.8.5 Déroulement de l'algorithme

Nous disposons maintenant de tous les éléments permettant de présenter l'algorithme utilisé sur le problème de l'évitement.

Le principe de l'algorithme que nous utilisons est très proche de l'algorithme d'optimisation sans contraintes présenté dans la section 2.3.

Nous allons d'abord présenter la version de l'algorithme qui correspond à l'approche globale du problème de l'évitement.

L'algorithme manipule une file d'intervalles notée $\mathcal{F}$, initialisée de manière à contenir, au début de l'optimisation, les $2^n$ intervalles de $I^n_{opt}$ décrits dans la section 2.8.2.

Les intervalles de cette file sont successivement extraits, coupés en deux nouveaux intervalles selon le processus décrit dans la section 2.8.2, et pour chacun des deux intervalles ainsi produits, sont calculés un intervalle image et un estimateur, selon la méthode décrite dans la section 2.8.4.

Considérons un de ces deux intervalles. Si la borne inférieure de son intervalle image est supérieure au meilleur estimateur rencontré précédemment, on sait que l'intervalle en question ne peut pas contenir de point meilleur que celui en lequel a été calculé le meilleur estimateur rencontré. Dans ce cas, l'intervalle n'est pas inséré dans la file $\mathcal{F}$ : il est supprimé.

Dans le cas contraire, cet intervalle est inséré dans la file $\mathcal{F}$, sauf s'il remplit les critères conduisant à l'insérer dans une des deux files résultats que nous allons décrire ci-dessous.

L'ordre d'insertion des intervalles dans la file $\mathcal{F}$ est important, puisque c'est à chaque fois le premier élément de cette file qui sera extrait et traité par l'algorithme.

Les éléments de la file $\mathcal{F}$ sont en fait des intervalles indéterminés. En effet, les intervalles non admissibles sont éliminés, et les intervalles admissibles sont réduits à un point, et donc, comme nous le verrons, soit insérés dans la liste $f_{res}^{sat}$, soit éliminés (s'ils sont de moins bonne qualité que le meilleur point rencontré par l'algorithme).

Les essais effectués sur divers cas de conflit avec différentes manières d'insérer les intervalles dans la file $\mathcal{F}$ ont mené au meilleur résultat (en terme de rapidité avec laquelle la solution est obtenue) en insérant les intervalles dans $\mathcal{F}$ de manière à ce qu'ils soient ordonnés par ordre croissant de la pire séparation à laquelle ils mènent (correspondant à la valeur de $dist_{min}(T)$), tant qu'aucun point respectant les contraintes de séparations n'a été trouvé, et ensuite de manière à ce qu'ils y soient ordonnés par ordre décroissant du meilleur retard auquel ils pourraient mener (la valeur de ce meilleur retard est donnée par $r_{min}(T)$).

Cela demande de réordonner la file $\mathcal{F}$ la première fois qu'un point admissible pour les contraintes de séparation est trouvé, c'est-à-dire la première fois qu'un estimateur est calculé en un tel point.

Les deux files d'intervalles-résultats utilisées dans le cadre de l'algorithme sont une file d'intervalles-résultats admissibles, $f_{res}^{sat}$, et une file d'intervalles-résultats indéterminés, $f_{res}^{ind}$. Ces files sont toutes les deux égales, au début du déroulement de l'algorithme, à la file vide. Ces deux files sont remplies au cours du déroulement de l'algorithme par des intervalles répondant aux critères suivants :

**Insertion dans $f_{res}^{sat}$ :** Un intervalle $X$ généré par l'algorithme sera inséré dans la file $f_{res}^{sat}$ s'il satisfait les trois critères suivants, que dans la suite nous appellerons les critères $C_{sat}$ :

1. Sa taille est inférieure à une valeur de référence, notée $\epsilon_{res}^{sat}$.
2. L'intervalle $X$ est admissible, ou bien, s'il n'est pas admissible, son estimateur a été calculé en un point respectant les contraintes de séparations.
3. il mène à un retard inférieur ou égal au meilleur retard obtenu par l'algorithme.

On a vu que si l'intervalle $X$ est admissible, il est réduit par l'algorithme à un intervalle dégénéré (voir définition 2.3) correspondant au point de l'intervalle $X$ entraînant le retard le plus faible.

Si l'intervalle $X$ est indéterminé mais que son estimateur a été calculé en un point en lequel les contraintes de séparations sont respectées, l'intervalle $X$ est réduit à un intervalle dégénéré correspondant à ce point.

Les intervalles sont insérés dans la file $f_{res}^{sat}$ de manière à être ordonnés selon l'ordre croissant des retards qu'ils entraînent. Les intervalles de la file $\mathcal{F}$ étant des intervalles dégénérés, réduits à des points, chacun d'entre eux correspond à une valeur précise pour la somme des retards des avions. Le premier intervalle de la file, à la fin de l'optimisation correspond donc au retard minimal calculé par l'algorithme, et sera présenté comme solution.

**Insertion dans $f_{res}^{ind}$ :** On n'insère des intervalles dans la file $f_{res}^{ind}$ que si la file $f_{res}^{sat}$ est vide. En effet les intervalles contenus dans la file $f_{res}^{sat}$ sont, comme nous venons de le voir, admissibles (ou bien ils sont admissibles et réduits à un point, ou bien ils sont réduits à un point qui respecte les contraintes de séparations, c'est-à-dire réduits à des intervalles dégénérés admissibles), tandis que les intervalles de $f_{res}^{ind}$ ne seront pas admissibles : si $f_{res}^{sat}$ n'est pas vide, il est inutile de conserver les intervalles de $f_{res}^{ind}$, qui fournissent une moins bonne réponse au problème de l'évitement. À la condition que $f_{res}^{sat}$ soit vide,un intervalle sera inséré dans $f_{res}^{ind}$ s'il satisfait les critères suivants, appelés dans la suite les critères $C_{ind}$ :

- sa taille est plus petite qu'une valeur de référence, notée $\epsilon_{res}^{ind}$. La valeur choisie pour $\epsilon_{res}^{ind}$ est plus petite que celle choisie pour $\epsilon_{res}^{sat}$. En effet, les intervalles susceptibles d'être insérés dans la file $\epsilon_{res}^{ind}$ ne doivent être considérés que si aucun intervalle n'a pu être intégré dans la file $f_{res}^{sat}$. Il faut noter que le fait de continuer à générer des intervalles en divisant et réinsérant dans la file $\mathcal{F}$ des intervalles de cette file peut permettre de générer un intervalle susceptible d'être inséré dans $f_{res}^{sat}$ (l'algorithme s'arrête alors).

- Il est indéterminé, et son estimateur correspond à un point qui ne satisfait pas les contraintes de séparation (s'il est non admissible il est supprimé, et s'il est admissible, ou si son estimateur satisfait les contraintes de séparation, il est inséré dans $f_{res}^{sat}$).
- Son estimateur a été calculé en un point menant à une meilleure séparation que celle qui est entraînée par le meilleur élément déjà contenu dans $f_{res}^{ind}$.

On ne sait pas, puisque cet intervalle est indéterminé et que son estimateur a été calculé en un point ne respectant pas les contraintes de séparation, si l'intervalle $X$ contient ou non un point respectant les contraintes de séparation (voir section 2.8.4). Mais on connaît, aux imprécisions dues à la discrétisation du temps près, la plus mauvaise séparation entre deux avions sur la durée de l'évitement rendue possible par l'intervalle $X$. Cette valeur est approchée par la valeur de $dist_{min}$ calculée pour l'intervalle $X$ (voire l'encadrement 2.81). Cependant, l'estimateur de $X$ fournit un point qui certes ne respecte pas les contraintes de séparation mais qui conduit à une séparation au moins aussi bonne que $d_{min}$ entre les avions. Cette séparation est en fait la meilleure séparation dont on soit sûr qu'elle est rendue possible par l'intervalle $X$.

Pour cette raison, on réduit aussi, avant de l'insérer dans $f_{res}^{ind}$, l'intervalle $X$ en un intervalle dégénéré correspondant au point en lequel a été calculé son estimateur.

Les intervalles sont insérés dans la file $f_{res}^{ind}$ de manière à être ordonnés selon l'ordre décroissant de la séparation qu'ils entraînent. Dans le cas où la file $f_{res}^{sat}$ serait demeurée vide, jusqu'à la fin de l'optimisation, c'est le premier élément de $f_{res}^{ind}$ qui serait présenté comme solution. Il faut cependant noter que cette solution n'est pas une bonne solution, puisqu'elle ne respecte pas les contraintes de séparation, elle est seulement la *moins mauvaise solution rencontrée par l'algorithme*.

On voit donc que dès que la file $f_{res}^{sat}$ n'est pas vide, les intervalles dont la taille est inférieure à $\epsilon_{res}^{sat}$ et qui ne remplissent pas les critères d'insertion dans $f_{res}^{sat}$ sont éliminés.

Cet algorithme suit alors les étapes suivantes (section 2.3) :

1. *La file $\mathcal{F}$ est initialisée. Elle contient $2^n$ éléments, qui sont les éléments de $I_{opt}^n$ de la forme $(([0,t_f],[0,t_f],\alpha^1),..,([0,t_f],[0,t_f],\alpha^n))$, correspondant chacun à une des $2^n$ combinaisons possibles pour les angles de déviation $\alpha^i$ des $n$ avions.*
2. *On extrait le premier élément $X$ de cette file.*
3. *Calcul de l'intervalle image et d'un estimateur $f_{est}(X)$ de cet intervalle.*
4. *$f_{est}^* \leftarrow f_{est}(X)$*
5. *Tant que $\mathcal{F} \neq \emptyset$ on répète les étapes suivantes :*
   (a) *On extrait $X$ le premier élément de la file $\mathcal{F}$.*
   (b) *Si $X$ satisfait le critère $C_{sat}$, on insère $X$ dans $f_{res}^{sat}$.*
   (c) *Si $f_{res}^{sat}$ est vide et si $X$ satisfait le critère $C_{ind}$, insère $X$ dans $f_{res}^{ind}$.*

(d) *Si $X$ n'a été inséré ni dans $f_{res}^{sat}$ ni dans $f_{res}^{ind}$, si la taille de $X$ est supérieure à $\epsilon_{res}^{sat}$ ou si $f_{res}^{sat}$ est vide, on coupe $X$ en deux intervalles $X_1$ et $X_2$, selon la méthode indiquée dans la section 2.8.2 en vérifiant que ces deux intervalles satisfont la condition 2.58 (à défaut de quoi ils sont supprimés), et en les normalisant, si besoin est, pour qu'ils satisfassent les conditions 2.59 et 2.60*

*Les cinq étapes suivantes seront menées pour $j = 1$ et $j = 2$.*

i. *calcul d'un estimateur $f_{est}(X_j)$ et l'intervalle image de $X_j$.*
ii. *Si $X_j$ est non admissible, il est supprimé.*
iii. *Si $X_j$ est admissible, réduction de l'intervalle $X_j$ et de son intervalle image.*
iv. *Si $f_{est}(X_j) < f_{est}^*$ $f_{est}^* \leftarrow f_{est}(X_j)$.*
v. *Si $f_{min}(X_j) \leq f_{est}^*$, on insère $X_j$ dans la file $\mathcal{F}$.*

6. *Quand $\{ = \emptyset$, si $f_{res}^{sat}$ n'est pas vide, le résultat de l'algorithme est le premier élément de $f_{res}^{sat}$ (une bonne solution). Si $f_{res}^{sat}$ est vide et $f_{res}^{ind}$ ne l'est pas le résultat de l'algorithme est le premier élément de $f_{res}^{ind}$ (une mauvaise solution). Si $f_{res}^{sat}$ et $f_{res}^{ind}$ sont vides, il n'y a pas de solution.*

L'approche globale, nous le verrons dans la section 2.9.2, mène à des temps de calcul très importants dès que le nombre d'avions dépasse 2. C'est pourquoi nous utilisons la technique de l'élagage.

L'élagage consiste à modifier l'étape (*5(d)v*) de l'algorithme présenté ci-dessus. Sans élagage, cette étape consiste à insérer dans la file $\mathcal{F}$ un intervalle $T$ qui vient d'être créé par l'algorithme, quand la borne inférieure de son intervalle image est inférieure ou égale à la valeur du meilleur estimateur calculé, c'est-à-dire quand il est possible que l'intervalle $T$ contienne un point au moins aussi bon que le meilleur point déjà rencontré.

Avec l'élagage, à cette étape de l'algorithme, on n'insère l'intervalle $T$ dans la file $\mathcal{F}$ que s'il contient le point en lequel a été calculé le meilleur estimateur ou si il a une probabilité non nulle d'apporter, par rapport au meilleur point calculé, une amélioration supérieure à une valeur fixée, appelée la coupure d'élagage, et notée $c_e$, c'est-à-dire si on a, en notant $est_0$ le meilleur estimateur déjà calculé par l'algorithme et $i_{min}(T)$ la borne inférieure de l'intervalle image de $T$ :

$$i_{min}(T) \leq est_0 - c_e \tag{2.87}$$

Dans le cas présent, la coupure d'élagage correspondra à une précision requise sur les retards obtenus. Cette précision sera telle que la différence entre la valeur de la fonction $f_{opt}^n$ en un point respectant les contraintes de séparations et un point ne les respectant pas sera toujours supérieure à $c_e$.

Les effets de l'élagage, tant sur le temps de calcul que sur la qualité des résultats obtenus, seront décrits dans la section 2.9.2.

Le même algorithme est utilisé dans le cadre de l'approche séquentielle, successivement pour les $n$ avions, à cette différence près que l'optimisation est menée sur l'ensemble $E_{opt}^1$, que les éléments des différentes files manipulées par l'algorithme sont des éléments de $I_{opt}^1$, que la file $\mathcal{F}$ contient initialement les éléments $([0, t_f], [0, t_f], \alpha)$ et $([0, t_f], [0, t_f], -\alpha)$ (voir section 2.8.2), et que les intervalles images et les estimateurs des intervalles sont calculés en fonction des contraintes de séparation imposées par les trajectoires prévues pour les avions précédant, dans l'ordre de priorité, l'avion dont on génère la trajectoire.

L'approche séquentielle procédera donc de la manière suivante :

1. *On numérote les avions selon l'ordre de priorité.*
2. *L'avion $a^1$ conserve sa trajectoire initiale.*
3. *Pour $i = 2$ à $n$, on suit les étapes suivantes :*
   (a) *On applique pour générer une trajectoire pour l'avion $a^i$ l'algorithme ci-dessus, les contraintes étant imposées par les trajectoires générées par les avions $a^0$ à $a^{i-1}$.*
   (b) *Si une trajectoire d'évitement est trouvée pour l'avion $a^i$ (l'algorithme fournit une solution, bonne ou mauvaise), les temps de manœuvres et le sens de déviation la décrivant sont mémorisés, et la trajectoire ainsi prévue pour l'avion $a^i$ imposera des contraintes sur les trajectoires des avions $a^j$, $j > i$.*
   (c) *Si aucune trajectoire d'évitement n'est trouvée, fin de l'optimisation, pas de solution.*
4. *Si une solution a été trouvée pour chaque avion, l'algorithme propose les trajectoires prévues pour les $n$ avions, en précisant les retards entraînés par ces trajectoires, et la séparation assurée dans les cas où c'est une mauvaise solution qui a été trouvée à l'étape (3b).*
5. *Si une solution n'a pas été trouvée pour chacun des $n$ avions, l'algorithme donne le numéro de l'avion à partir duquel il n'y a plus eu de solution.*

### 2.8.6 Signification des résultats obtenus

Les algorithmes présentés dans les sections précédentes peuvent donner trois résultats différents, selon l'état des files $f_{res}^{sat}$ et $f_{res}^{ind}$ à la fin du déroulement de l'algorithme (c'est-à-dire quand la file $\mathcal{F}$ est vide). L'algorithme peut fournir une *bonne solution*, correspondant au premier élément de la file $f_{res}^{sat}$ si celle ci n'est pas vide. Si elle est vide mais que $f_{res}^{ind}$ ne l'est pas, le premier élément de cette dernière file fournit une *mauvaise* solution. Si les files $f_{res}^{sat}$ et $f_{res}^{ind}$ sont toutes deux vides, alors l'algorithme ne fournit pas de solution.

Pour traduire les résultats obtenus grâce aux algorithmes décrits dans ce chapitre nous aurons besoin de définir la *largeur intérieure* d'un élément de $T_{opt}^n$.

**Définition 2.18 – Largeur intérieure d'un élément de $I_{opt}^n$** – *Pour $X \in I_{opt}^n$, avec les notations $X = (([t_{0min}^1, t_{0max}^1], [t_{1min}^1, t_{1max}^1], \alpha^1), .., ([t_{0min}^n, t_{0max}^n], [t_{1min}^n, t_{1max}^n], \alpha^n))$, on définit $l_i(X)$, la largeur intérieure de $X$ de la manière suivante :*

$$l_i(X) = \min_{\substack{i \in \{1,..,n\} \\ j \in \{0,1\}}} (t_{jmax}^i - t_{jmin}^i)$$

**Pas de solution :** Si l'algorithme ne fournit aucune solution, c'est-à-dire si, à la fin de l'optimisation, les deux files résultats $f_{res}^{sat}$ et $f_{res}^{ind}$ sont toutes deux vides, alors on sait qu'il n'existe pas de trajectoire conforme à la modélisation choisie pour l'évitement assurant, au cours de l'évitement, une séparation entre avions au moins égale à la norme séparation majorée (voir la définition 2.17 de la section 2.7.2).

En effet, considérons le cas dans lequel, quand la file $\mathcal{F}$ est vide, les deux files résultats sont toutes deux vides.

Dans ce cas, on sait qu'on n'a pas calculé d'estimateur en un point admissible pour les contraintes de séparation (c'est-à-dire, ici, assurant sur les pas de temps une séparation au moins égale à la norme de séparation majorée $n_h^*$). En effet, si c'était le cas, la file $\mathcal{F}$ n'aurait pas pu se vider sans qu'un intervalle contenant ce point et de taille inférieure à $\epsilon_{res}^{sat}$ n'ait été généré, et donc inséré dans la file $f_{res}^{sat}$ (voir sections 2.8.4 et 2.8.5).

Supposons maintenant qu'il existe un point dans l'espace de recherche tel qu'il définit pour les avions des trajectoires leur permettant d'être toujours séparés deux à deux d'une distance supérieure à la norme de séparation majorée $n_h^*$ (notons un tel point $x_{sat}$). On a vu (section 2.7.2) qu'un intervalle non admissible ne pouvait contenir un tel point. Donc, si un tel point existait, tout intervalle le contenant serait soit admissible, soit indéterminé. Or, on a vu que l'algorithme ne peut avoir calculé d'estimateur en un point admissible, donc il n'a pas rencontré d'intervalle admissible.

Nous en déduisons que tout intervalle contenant le point $x_{sat}$ est indéterminé, et ne pourrait être supprimé de la file $\mathcal{F}$ que dans un des cas suivants :

- Le meilleur estimateur obtenu est supérieur à la borne inférieure de l'image de cet intervalle. Or, si un intervalle contient $x_{sat}$, la borne inférieure de l'image de cet intervalle est plus petite que la valeur qu'aurait $f_{opt}^n$ en $x_{sat}$. Par définition de la fonction $f_{opt}^n$, et puisque le point $x_{sat}$ respecte les contraintes de séparation, cela ne serait possible que si le meilleur estimateur rencontré par l'algorithme avait été calculé en un point assurant une séparation supérieure à $n_h^*$, et on a vu que cela ne pouvait pas être le cas.
- La taille de cet intervalle est inférieure à $\epsilon_{res}^{sat}$, et la file $f_{res}^{sat}$ n'est pas vide, ce qui n'est pas possible, puisqu'on suppose que la file $f_{res}^{sat}$ est vide tout au long de l'optimisation.
- La taille de cet intervalle est inférieure à $\epsilon_{res}^{ind}$, mais dans ce cas, puisque la file $f_{res}^{sat}$ est vide, cet intervalle serait inséré dans la file $f_{res}^{ind}$, ce qui n'est pas possible, puisque cette file est supposée rester vide tout au long de l'optimisation.

L'algorithme présenté ici, dans les cas où il ne donne aucun résultat, permet donc d'assurer qu'il n'existe pas de trajectoires d'évitement conforme à la modélisation choisie permettant aux avions de ne jamais s'approcher l'un de l'autre d'une distance inférieure à la norme de séparation majorée $n_h^*$.

**Une bonne solution :** Si l'algorithme fournit une *bonne* solution, c'est-à-dire si la file $f_{res}^{sat}$ n'est pas vide à la fin de l'optimisation, la solution proposée par l'algorithme donne des trajectoires assurant que les avions restent séparés sur toute la durée de l'évitement, c'est-à-dire que la distance entre deux avions reste toujours supérieure à la norme de séparation $n_h$. Notons $r_{min}^s$ le retard correspondant à ce point. On sait alors qu'il n'existe pas d'intervalle admissible de largeur intérieure supérieure ou égale à $2\epsilon_{res}^{sat}$ dont le point milieu mènerait à un retard strictement inférieur à $r_{min}^s$.

En effet, si un tel intervalle existait (appelons le $X_0$), l'algorithme aurait nécessairement généré un intervalle admissible de taille inférieure à $\epsilon_{res}^{sat}$ et contenant ce point milieu. Cet intervalle, dont le meilleur point conduirait à un retard strictement inférieur à $r_{min}^s$ serait alors réduit à l'intervalle dégénéré correspondant à ce meilleur point, et inséré dans $f_{res}^{sat}$. Les éléments de $f_{res}^{sat}$ étant ordonnés par ordre de retard croissant, c'est cet intervalle qui aurait alors été proposé par l'algorithme comme solution du problème.

**Une mauvaise solution :** Si l'algorithme fournit une *mauvaise* solution, c'est-à-dire que la file $f_{res}^{sat}$ est vide à la fin de l'optimisation mais que la file $f_{res}^{ind}$ ne l'est pas, la solution présentée par l'algorithme est un intervalle dégénéré, noté $X_{sol}$, donnant des trajectoires n'assurant pas la séparation des avions au cours de l'évitement. On connaît alors la valeur $dist_{min}$ calculée pour cet intervalle (voir section 2.8.4), que l'on notera $dist_{min}^{s}$.

- On sait alors qu'il n'existe pas d'intervalle admissible de largeur intérieure supérieure ou égale à $2\epsilon_{res}^{ind}$. En effet, tant que $f_{res}^{sat}$ est vide, les intervalles indéterminés de tailles supérieures ou égales à $\epsilon_{res}^{ind}$ sont conservés dans la file $\mathcal{F}$ pour être ensuite divisés en deux intervalles. Si un intervalle admissible de largeur intérieure supérieure ou égale à $2\epsilon_{res}^{ind}$ existait (appelons le encore$X_0$), alors l'algorithme aurait nécessairement généré un intervalle admissible de taille inférieure à $\epsilon_{res}^{ind}$ contenu dans $X_0$ et l'aurait inséré dans $f_{res}^{sat}$.
- S'il existe un point admissible, que nous appelons $x_s$, on considère $X_s$ l'élément de $I_{opt}^{n}$ centré en $x_s$ ne contenant que des intervalles de manœuvre $T_0^i$ et $T_1^i$ de taille égale à $2\epsilon_{res}^{ind}$. On sait que s'il existe un tel point admissible, il existe un point de $X_s$ définissant des trajectoires telles qu'au cours de l'évitement deux avions s'approcheront l'un de l'autre d'une distance inférieure ou égale à $dist_{min}^{s}$. En effet, dans le cas où un tel point $x_s$ existerait, l'algorithme générerait nécessairement un intervalle indéterminé de taille inférieure à $\epsilon_{res}^{ind}$ contenu dans l'intervalle $X_s$, et l'insérerait dans la file $f_{res}^{ind}$. Si aucun point de cet intervalle n'entraînait une séparation inférieure ou égale à $dist_{min}^{s}$, la valeur de $dist_{min}$ calculée pour cet intervalle serait supérieure à $dist_{min}^{s}$ (voir section 2.7.2), et cet intervalle serait inséré dans $f_{res}^{ind}$ devant $X_{sol}$, puisque les éléments de $f_{res}^{ind}$ sont classés par ordre de $d_{min}$ croissant, et serait donc proposé comme solution à la place de $X_{sol}$.

Cette deuxième propriété est très importante dans le cadre du problème de l'évitement : en effet, supposons que les avions suivent les trajectoires proposées par l'algorithme avec une certaine imprécision sur les instants de manœuvres. Cette imprécision ne pourra qu'augmenter si on tient compte du fait que les changements de cap des avions ne sont pas instantanés, ce qui sera nécessaire dans le cadre d'une application plus réaliste de méthodes telles que celles qui sont présentées ici à des cas de conflit issus du trafic réel.

Supposons que $\epsilon_{res}^{ind}$ est choisi tel que cette imprécision est plus grande que $2\epsilon_{res}^{ind}$. Supposons de plus que l'algorithme, sur un conflit donné, ne trouve qu'une mauvaise solution. Des trajectoires d'évitement par point tournant sans conflit (assurant que les distances entre avions restent supérieures à la norme de séparation $n_h$) existent peut-être, que l'algorithme n'a pas trouvé. Cependant, même si ces trajectoires étaient connues, les avions, en voulant les suivre, pourraient s'approcher l'un de l'autre d'une distance inférieure à la séparation correspondant à la mauvaise solution donnée par l'algorithme, du fait de cette imprécision dans les instants de manœuvres.

Il faut noter que dans le cas de l'approche séquentielle, ces interprétations sont valables à chaque fois que l'on utilise l'algorithme pour générer la trajectoire d'un avion, en considérant comme fixées les trajectoires des avions le précédant dans l'ordre de priorité.

Ainsi, si l'algorithme ne trouve pas de solution pour l'avion $a^i$, cela signifie qu'il n'existe pas de trajectoire d'évitement par point tournant permettant à l'avion $a^i$ d'atteindre sa destination en restant séparé des avions le précédant dans l'ordre de priorité d'une distance supérieure à $n_h^*$, si ces avions suivent les trajectoires qui ont été générées pour eux aux itérations précédentes de l'algorithme séquentiel.

Il est tout à fait possible cependant que, au cas où l'un de ces avions suivrait une trajectoire différente (même légèrement), l'algorithme trouve une bonne solution pour l'avion $a^i$.

## 2.9 Résultats obtenus

Les résultats présentés ici ont été obtenus sur un PC équipé d'un processeur de type pentium II de 300 MHz.

Nous présentons les résultats obtenus en utilisant des trajectoires d'évitement par point tournant, l'utilisation de trajectoires par offset posant des problèmes que nous décrivons dans la section suivante.

### 2.9.1 Problèmes posés par l'évitement par offset

Nous avons constaté que les algorithmes présentés dans les sections précédentes étaient très sensiblement moins efficaces quand ils utilisaient des trajectoires d'évitement par *offset* que quand ils utilisaient des trajectoires d'évitement par point tournant. Les temps de calcul étaient en effet très grands, même pour des conflits à deux avions (approche globale) ou pour l'approche séquentielle de la résolution de conflit (de l'ordre de plusieurs dizaines minutes de temps cpu), alors que sur ce type de problème la modélisation par point tournant mène à des temps de calcul très faibles (voir sections 2.9.2 et 2.9.3).

Cela est dû au fait que les algorithmes présentés ici ne sont pas adaptés à la minimisation du retard entraîné par les trajectoires d'évitement par offset.

En effet, le retard entraîné par une trajectoire d'évitement par offset est une fonction de la différence $t_1 - t_0$ entre les deux instants de manœuvres. Toutes les trajectoires correspondant à la même différence entre les instants de manœuvres mènent au même retard.

La figure 2.7 montre trois trajectoires par offset menant au même retard. Un point $p^1(t)$ correspond à la position de l'avion $a^1$ au temps $t$. Les trois trajectoires de manœuvres montrées par la figures correspondent aux trois couples de temps de manœuvres $(t_0, t_1)$, $(t'_0, t'_1)$ et $(t''_0, t''_1)$, avec $t_1 - t_0 = t'_1 - t'_0 = t''_1 - t''_0$. Si la trajectoire de l'avion $a^1$ correspondant à $(t''_0, t''_1)$ assure que les deux avions restent séparés, les deux autres trajectoires possibles pour l'avion $a^1$ qui apparaissent sur la figure 2.7 l'assurent aussi, ainsi que toutes les trajectoires intermédiaires (la trajectoire de $a_2$ est fixée).

Il existe ainsi de nombreuses solutions équivalentes. L'algorithme conserve alors dans la file $\mathcal{F}$ un nombre très important d'intervalles indéterminés, contenant chacun un point définissant une solution équivalente à la meilleure solution obtenue. Tous ces intervalles sont successivement coupés jusqu'à ce qu'ils soient de taille inférieure à $\epsilon_{res}^{sat}$ (voir section 2.8.5), d'où l'importance des temps de calcul obtenus.

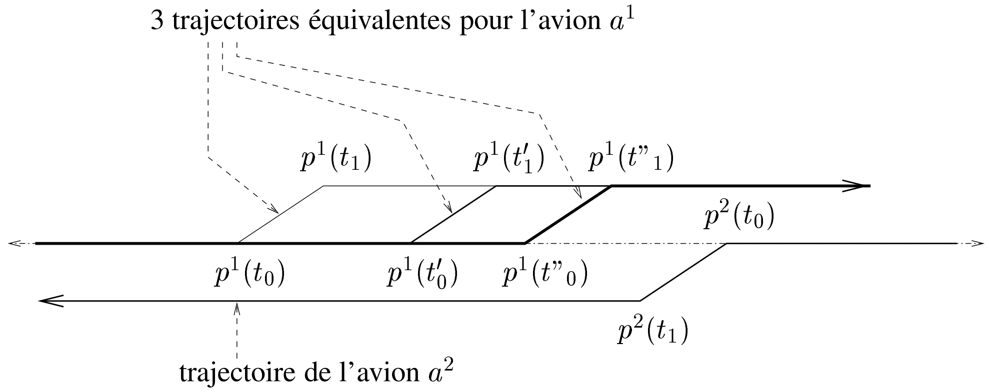



FIG. 2.7: *Trois trajectoires par offset menant au même retard*

Ce problème ne se pose pas dans le cas de l'utilisation des trajectoires par point tournant, alors que pourtant dans ce cas aussi, le retard est une fonction de la différence entre les instants de manœuvres (on a vu dans la section 2.7.3 que l'expression du retard était, à un facteur 2 près, la même pour les deux modélisations). Cette différence est due à ce que le fait de *décaler* une trajectoire d'évitement par offset n'a pas le même effet que le fait de décaler une trajectoire d'évitement par point tournant.

Par *décaler* une trajectoire par offset définie par un sens de déviation donné et un couple d'instants de manœuvres $(t_0, t_1)$, on entend remplacer cette trajectoire par celle qui est définie par le même sens de déviation et des instants de manœuvres de la forme $(t_0+c, t_1+c)$. Les trajectoires montrées sur la figure 2.7 sont décalées les unes par rapport aux autres.

Après l'instant $t''_1$, correspondant à la fin de la mise en offset la plus tardive de ces trois trajectoires, pour un instant $t$ donné, la position de l'avion est la même quelle que soit celle de ces trois trajectoires qu'il a suivie. Si, dans le cas où $a^1$ suit la trajectoire définie par $(t''_0, t''_1)$, l'instant correspondant à la distance minimale entre les deux avions arrive après $t''_1$, toute trajectoire obtenue en décalant cette trajectoire vers la gauche lui sera équivalente non seulement en terme de retard, mais aussi en terme de séparation.

Cela n'est plus le cas avec l'évitement par point tournant : décaler une trajectoire par point tournant revient à décaler non seulement le départ de l'avion de sa trajectoire d'origine, mais aussi son retour.

L'utilisation de l'élagage permettrait de résoudre ce problème : un intervalle n'est plus conservé dans la file $\mathcal{F}$ dès qu'il peut contenir une solution équivalente à la meilleure solution trouvée, mais seulement s'il peut contenir une solution apportant une diminution non nulle du retard. L'utilisation de l'élagage ne nous a pas semblé cependant une solution appropriée. Elle entraîne une perte d'optimalité de la solution qui, dans ce cas, n'est pas justifiée. Elle mènerait à une solution, parmi d'autres solutions équivalentes, seulement parce qu'elle serait la première obtenue. Dans le cas de l'évitement par offset, il nous semblerait plus intéressant de suivre une des deux voies ci-dessous :

- Plutôt que de minimiser le retard seulement, on pourrait chercher à la fois à minimiser le retard et à reculer le plus possible dans le temps les manœuvres des avions. Cela permettrait de distinguer les solutions équivalentes en terme de retard et de séparation selon un critère qui a un sens dans le cadre de l'évitement : reculer le plus possible la mise en offset de l'avion permet de limiter le plus possible le temps pendant lequel l'avion est écarté de sa route prévue. En faisant quitter trop tôt sa route à un avion, on risque de générer inutilement de nouveaux conflits. Cela serait possible en minimisant non plus le retard des avions, donné par :

$$\sum_{1}^{n} r_{min}^{i}(t_0^i, t_1^i)$$

Mais une grandeur de la forme :

$$\sum_{1}^{n} r_{min}^{i}(t_0^i, t_1^i) - \kappa t_0^i$$

où $\kappa$ serait une constante positive. Le choix de cette constante serait cependant délicat. Une petite valeur de $\kappa$ favoriserait les déviations faibles mais précoces, une grande $\kappa$ favoriserait les déviations tardives, mais importantes.

- Plutôt que de chercher à reculer la mise en offset, on pourrait chercher à optimiser aussi le moment du retour sur la trajectoire d'origine, de manière à minimiser la durée de l'offset (temps pendant lequel l'avion est sur une trajectoire parallèle à sa trajectoire d'origine). La trajectoire serait alors définie par 3 instants de manœuvres (en plus du sens de déviation). Les instants $t_0$ et $t_1$, plus un instant $t_2$ correspondant à l'instant auquel l'avion amorce son retour vers sa trajectoire d'origine. La méthode présentée ici devrait être modifiée pour tenir compte de ce troisième instant de manœuvre. Il serait toujours possible de calculer simplement la forme de l'ensemble des positions possibles d'un avion à un instant $t$ donné, comme cela est fait dans la section 2.6.4. Les formes obtenues pour ces ensembles seraient semblables à celles obtenues pour l'évitement par point tournant 2.5.8. Cependant, le *dosage*, dans la fonction à minimiser, entre le retard de l'avion et la durée de l'offset serait, là aussi, délicat.

### 2.9.2 Approche globale

L'approche globale mène à un problème devenant vite, quand le nombre d'avions augmente, trop complexe pour être traité par des méthodes d'optimisation déterministes telles que celle qui est présentée dans ce chapitre. Nous verrons que ce sera aussi le cas pour la méthode basée sur l'algorithme $A^*$ que nous présentons dans le chapitre 3.

Celles-ci seront souvent, notamment, trop coûteuses en temps de calcul, ou en mémoire (voir chapitre 3), pour servir de base à un système d'évitement automatique centralisé.

Pour un petit nombre d'avions, des méthodes de ce type peuvent néanmoins être utilisées pour valider des méthodes stochastiques, beaucoup plus rapides, mais ne présentant pas de garanties réelles quant à la qualité des résultats obtenus, par rapport à l'optimum, telles que celles présentées par Nicolas Durand dans [Dur96], ou pour évaluer la qualité de résultats obtenus par l'approche séquentielle (voir section 2.9.3).

La méthode décrite dans ce chapitre traite assez rapidement divers cas de conflits à 2 avions.

Les cas de conflit à trois avions mènent à des problèmes beaucoup plus grands.

Supposons que nous cherchions de manière exhaustive, après avoir découpé l'ensemble sur lequel est menée la recherche ($E^3_{opt} = ([0, t_f] \times [0, t_f] \times \{\alpha, -\alpha\})^3$) en intervalles de manœuvres de largeur $\epsilon^{sat}_{res}$, associés aux différents sens de déviations possibles pour chaque avion, le meilleur de ces intervalles.

Supposons que l'on ait $t_f = 800$ s. et $\epsilon^{sat}_{res} = 10\ s$, On a donc $n_{int} = 80$ intervalles de taille $\epsilon^{sat}_{res}$ dans $[0, t_f]$. Pour chaque avions, pour un sens de déviation donné, compte tenu du fait que les temps de manœuvres $t^i_0$ et $t^i_1$ doivent respecter l'inégalité $t^i_0 \leq t^i_1$, on obtient $n_{int}(n_{int} + 1)/2$ possibilités pour les intervalles de manœuvres.

Pour couvrir par de tels intervalles la totalité de l'ensemble $E^3_{opt}$, chacun des 3 avions pouvant prendre deux sens de déviation, il faudrait donc $N_{int} = (n_{int}(n_{int} + 1))^3$ éléments de $I^3_{opt}$, c'est-à-dire avec les valeurs prises pour $\epsilon^{sat}_{res}$ et $t_f$, $N_{int} = 2,7\ .\ 10^{11}$. Une recherche exhaustive nécessiterait donc dans ce cas un nombre de calculs des séparations et des retards entraînés par un élément donné de $I^n_{opt}$ de l'ordre de la dizaine de milliards.

Nous présentons ici la méthode globale sur un cas-test de conflit à 3 avions suivants : 3 avions, équitablement répartis, au temps $t = 0$, sur un cercle de rayon 28 NM, convergent vers le centre de ce cercle à une vitesse de 420 kt.

Les grandeurs caractéristiques de l'algorithme sont les suivantes :

- La durée de l'évitement est $t_f$ = 800 secondes ;
- La taille maximale des intervalles insérés dans $f^{sat}_{res}$ est $\epsilon^{sat}_{res}$ = 10 secondes ;
- La taille maximale des intervalles insérés dans $f^{ind}_{res}$ est $\epsilon^{ind}_{res}$ = 5 secondes.
- La norme de séparation est $n_h = 5$ NM.
- Le pas de temps utilisé pour le calcul des distances entre avions au cours de l'évitement est de 10 secondes, ce qui donne comme valeur pour la norme majorée de séparation majorée à $n_h = 5,52$ NM.

La solution montrée sur la figure 2.8 a été obtenue après avoir calculé les distance entre avions et les retards entraînés par les intervalles de manœuvres sur environ $500\,000$ éléments de $I^n_{opt}$ (correspondant à 25 100 secondes de temps cpu sur un ordinateur équipé d'un processeur type *pentium 2* de 300 $MHz$).

Ces trajectoires correspondent, pour chacun des 3 avions, à un retard de 9 secondes. Le cas-test de conflit présenté ici étant symétrique, il n'est pas surprenant que les 3 trajectoires proposées pour les 3 avions soient identiques, à une rotation près.

Ce résultat, très satisfaisant en terme de qualité, a cependant nécessité un temps de calcul très important : 25 100 secondes, (c'est-à-dire 7 heures) de temps cpu. Il peut être intéressant, en utilisant la technique de l'élagage (voir section 2.8.5) d'obtenir, pour 3 avions, des résultats approchés en des temps de calcul moindres. Pour un nombre d'avion supérieur à 3, il est même inenvisageable de lancer un programme traduisant l'approche globale du problème de l'évitement sans recours à l'élagage.

Les résultats présentés dans le tableau 2.1 ont été obtenus pour les mêmes grandeurs caractéristiques que celles qui sont énumérées ci-dessus, pour différentes valeurs de la coupure d'élagage, sur des cas tests semblables à celui illustré, pour 3 avions, par la figure 2.8 : au temps $t = 0$, les avions sont régulièrement répartis sur un centre de rayon 28 NM et volent à la vitesse de $420$ kt, en convergeant vers le centre.

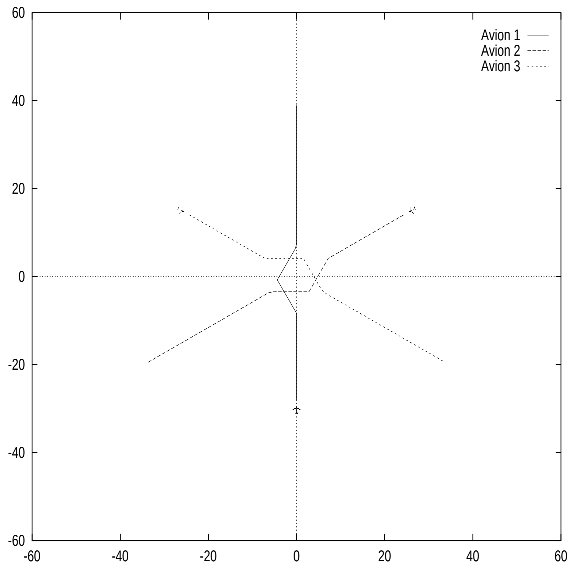



FIG. 2.8: *Conflit à 3 avions, évitement par point tournant*

TAB. 2.1: *Différentes coupures d'élagage pour différents nombres d'avions*

| Nombre d'avions | Coupure d'élagage | Retard moyen | Nombre d'éléments de $I_{opt}^n$ traités | Temps de calcul |
|---|---|---|---|---|
| 3 | 30 s | 22 s | 200 | 10 s |
| 3 | 15 s | 20 s | 13 000 | 670 s |
| 3 | 0 s | 16 s | 500 000 | 25 100 s |
| 4 | 45 s | 42 s | 350 | 27 s |
| 4 | 30 s | 31 s | 45 000 | 4 300 s |
| 5 | 75 s | 66 s | 160 | 18 s |
| 5 | 45 s | 39 s | 116 000 | 14 500 s |
| 6 | 120 s | 76 s | 90 | 23 s |
| 6 | 90 s | 58 s | 500 | 140 s |
| 6 | 60 s | 49 s | 160 000 | 36 500 s |
| 7 | 120 s | 102 s | 1 600 | 425 s |
| 7 | 90 s | 88 s | 3 900 | 1 100 s |

On voit que sur ces cas tests, le recours à l'élagage permet de réduire très fortement le nombre d'intervalles de $I^n_{opt}$ et donc le temps de calcul. Les détériorations observées en termes de qualité sont moins importantes que la détérioration maximale possible avec l'élagage : rappelons que l'utilisation d'une coupure d'élagage $c_e$ assure que la différence entre le résultat obtenu et le résultat qu'on obtiendrait sans élagage est inférieure ou égale à $c_e$.

### 2.9.3 Approche séquentielle

Dans le cas d'un conflit impliquant $n$ avions, l'approche séquentielle suit les étapes suivantes : on fixe un ordre de priorité entre les avions, puis on génère, pour chaque avion tour à tour, en suivant cet ordre de priorité, au moyen de l'algorithme présenté dans les sections précédentes, une trajectoire d'évitement permettant à l'avion de rester séparé des avions qui le précèdent dans l'ordre de priorité (les trajectoires de ces avions, générées précédemment, sont fixées). Le premier avion selon cet ordre de priorité conserve sa trajectoire initiale, il n'est soumis à aucune contrainte. Pour $i$ supérieur à 1, le $i$-ème avion doit satisfaire les contraintes traduisant la séparation, sur la durée de l'évitement, entre lui et chacun des $i$ avions dont les trajectoires sont déjà fixées au moment où la sienne est générée.

Un des problèmes que pose l'approche séquentielle est que les résultats obtenus dépendent fortement, pour un même cas de conflit, de l'ordre dans lequel les avions génèrent leur trajectoire, c'est-à-dire de l'ordre de priorité existant entre les avions impliqués dans un conflit.

Nous partons du type de cas test présenté dans la section précédente, correspondant à un conflit à 5 avions, répartis sur un cercle de rayon 28 NM, convergeant vers le centre de ce cercle à la vitesse de $420$ kt.

les grandeurs caractéristiques de l'algorithme sont les suivantes :

- La durée de l'évitement est $t_f$ = 800 secondes.
- La taille maximale des intervalles insérés dans $f^{sat}_{res}$ est $\epsilon^{sat}_{res}$ = 10 secondes.
- La taille maximale des intervalles insérés dans $f^{ind}_{res}$ est $\epsilon^{ind}_{res}$ = 5 secondes.
- La norme de séparation est $n_h = 5$ NM.
- Le pas de temps utilisé pour le calcul des distances entre avions au cours de l'évitement est de 10 secondes, ce qui porte la norme de séparation majorée à $n_h = 5,52$ NM.

Nous avons traité ce cas de conflit test avec la méthode séquentielle, pour les 120 combinaisons possibles d'ordre de priorité entre les avions.

Ce cas de conflit est invariant par rotation de $2\pi/5$, et par rotation de $-2\pi/5$ autour du centre du cercle vers lequel convergent les avions. Ainsi l'ordre de priorité $(1,2,3,4,5)$, dans lequel l'avion $i$ passe en $i$-ème position, devrait mener aux mêmes solutions (à une rotation près) que les ordres de priorité $(2,3,4,5,1)$, $(3,4,5,1,2)$, $(4,5,1,2,3)$, $(5,1,2,3,4)$.

On devrait obtenir aussi des trajectoires symétriques (c'est-à-dire correspondant aux mêmes temps de manœuvres, et à des sens de déviation opposés) pour les ordres de priorité $(5,4,3,2,1)$, $(4,3,2,1,5)$, $(3,2,1,5,4)$, $(2,1,5,4,3)$ et $(1,5,4,3,2)$.

On s'attend donc à obtenir en fait sur les 120 combinaisons, 12 solutions différentes, les autres se déduisant de celles-ci par rotations et symétries.

Le résultat que nous avons obtenu est conforme à ces attentes. Sur les 120 ordres de priorité possible, 90 ont mené à une bonne solution : une trajectoire possible pour les 5 avions, respectant la norme de séparation, c'est-à-dire que pour les 4 avions pour lesquels l'algorithme a cherché une trajectoire d'évitement (il n'en cherche pas pour le premier avion dans l'ordre de priorité, qui garde sa trajectoire d'origine), l'algorithme a trouvé un intervalle répondant au critère conditionnant l'insertion dans la file $f_{res}^{sat}$.

Pour 20 ordres de priorité, l'algorithme a trouvé une solution pour les deuxième, troisième, et quatrième avions, mais pas pour le cinquième : lors de la recherche d'une trajectoire pour le cinquième avion, les contraintes imposées par les quatre avions précédents étaient telles qu'aucune trajectoire correspondant à l'évitement par point tournant n'était possible. À la fin de la recherche d'une trajectoire pour le cinquième avion (quand la file $\mathcal{F}$ était vide), les deux files résultats $f_{res}^{sat}$ et $f_{res}^{ind}$ étaient vides.

Ces 20 cas correspondent aux ordres de priorité $(3, 1, 4, 5, 2)$ et $(3, 5, 4, 1, 2)$ et aux ordres de priorité qui se déduisent de ces deux ordres de priorité par les permutations circulaires et les symétries décrites ci-dessus.

Pour 10 ordres de priorité, c'est dès le quatrième avion qu'il n'existe plus de solution : ce sont les ordres de priorité se déduisant de $(3, 1, 4, 2, 5)$.

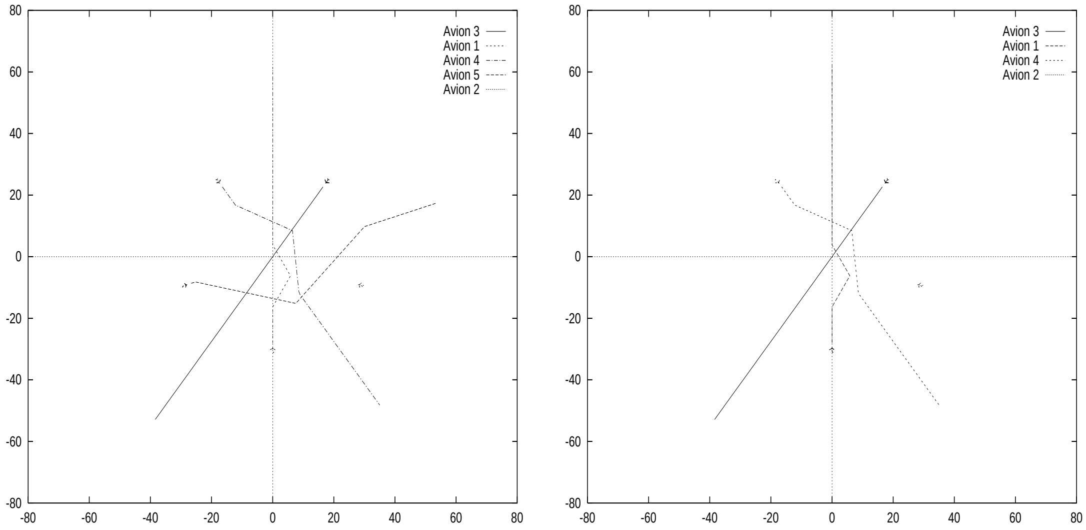
La figure 2.9 montre, à gauche, les trajectoires obtenues avec l'ordre de priorité $(3, 1, 4, 5, 2)$, pour lequel le dernier avion ne peut pas passer, et à droite celles qui ont été obtenues avec l'ordre de priorité $(3, 1, 4, 2, 5)$, pour lequel c'est le quatrième avion qui ne peut pas passer. Sur cette figure, comme sur celles qui suivent dans cette section, l'ordre des avions sur la légende correspond à leur ordre de priorité. L'avion ne correspondant à aucune trajectoire sur le schéma est le premier avion pour lequel il n'existe pas de trajectoire assurant une séparation au moins égale à la norme de séparation majorée. On remarque que les trois premiers avions dans l'ordre de priorité sont les mêmes dans les deux cas. Leurs trajectoires sont identiques dans les deux cas. Une fois que les trois premiers avions sont passés (3,1,4), l'avion 5 peut passer, mais pas l'avion 2.

Les ordres de priorité permettant à chacun des 5 avions d'adopter une trajectoire sans conflit mènent cependant à des solutions très différentes en terme de qualité : le retard moyen sur les 5 avions impliqués dans le conflit varie ainsi, selon les ordres de priorité des avions, de 20,8 secondes à 26,1 secondes (dans tous les cas le retard du premier avion à fixer sa trajectoire est nul, le retard moyen des 4 autres avions, qui eux, sont déviés, variant de 26 secondes dans le cas le meilleur à 32 secondes dans le pire).

Ces différences dans les retards moyens correspondent à des inégalités plus ou moins grandes entre les retards subis par les différents avions. Ces inégalités apparaissent clairement si l'on considère les trajectoires des avions dans les différents cas.

La figure 2.10 montre deux cas, symétriques l'un de l'autre, menant au retard minimal : les ordres de priorité $(1, 2, 3, 4, 5)$ à gauche, et $(1, 5, 4, 3, 2)$ à droite.

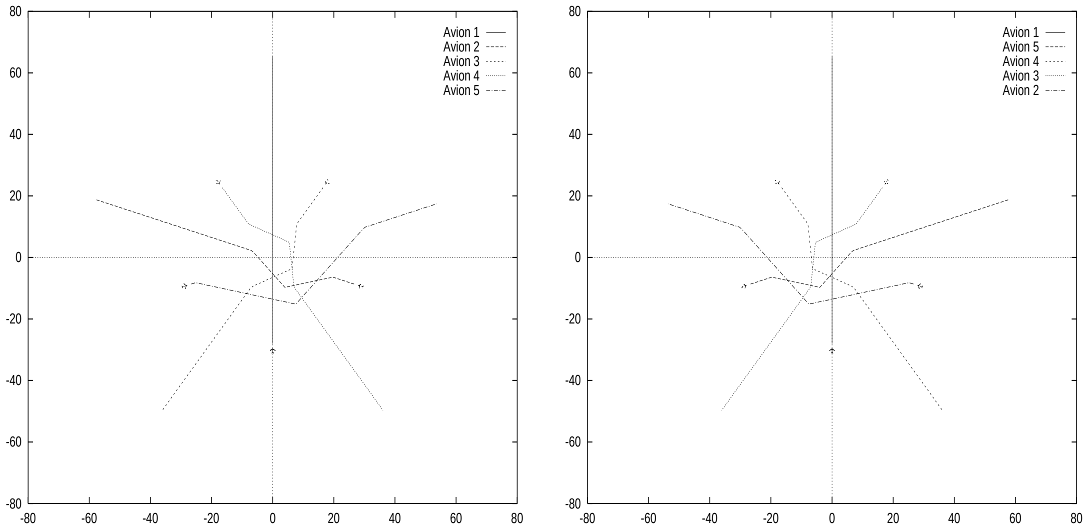
Il n'est pas surprenant que ces ordres de priorité mènent, sur ce cas test de conflit, au retard minimal. En effet, ils conduisent les premiers avions à adopter des trajectoires s'approchant des trajectoires *en rond point* telles que celles qu'on a obtenu pour trois avions dans le cas global. Le cinquième avion, plus contraint que les autres par les trajectoires des autres avions, est obligé de suivre une route beaucoup plus longue.




(a) Ordre de priorité $(3, 1, 4, 5, 2)$

(b) Ordre de priorité $(3, 1, 4, 2, 5)$

FIG. 2.9: *Deux ordres de priorité sans solution.*

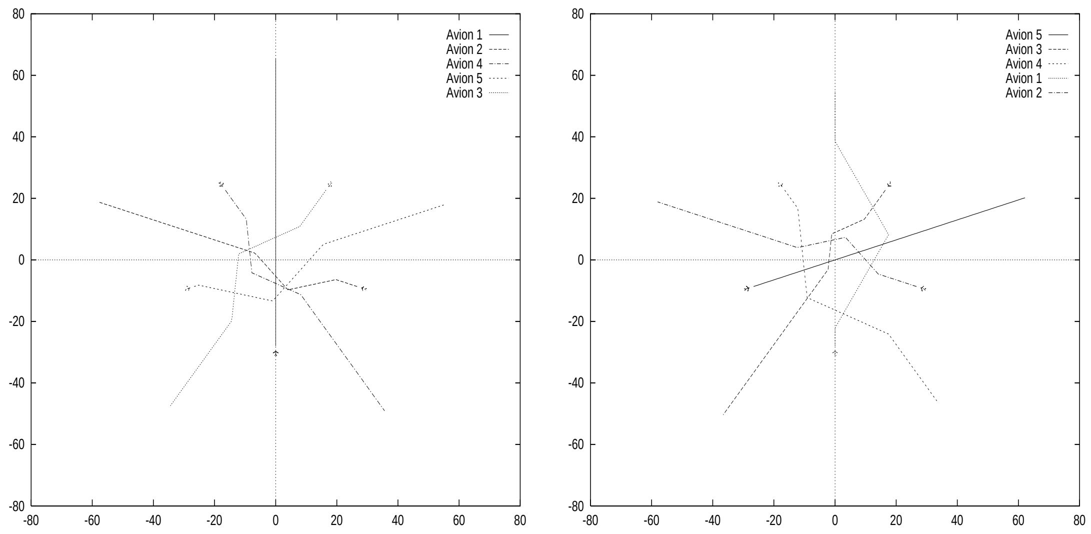



(a) Ordre de priorité $(1, 2, 3, 4, 5)$

(b) Ordre de priorité $(1, 5, 4, 3, 2)$

FIG. 2.10: *Deux des ordres de priorité les plus efficaces.*

La figure 2.11 montre les trajectoires obtenues avec deux ordres de priorité moins efficaces, mais permettant tout de même aux 5 avions de générer une trajectoire sans conflit.




(a) Ordre de priorité $(1, 2, 4, 5, 3)$ (b) Ordre de priorité $(5, 3, 4, 1, 2)$

FIG. 2.11: *Deux ordres de priorité moins efficaces.*

L'ordre de priorité $(1, 2, 4, 5, 3)$ (à gauche) mène à un retard moyen de 26,1 secondes. L'ordre de priorité $(5, 3, 4, 1, 2)$ (à droite) donne, lui, un retard moyen de 26,1 secondes. On voit que dans ce cas ce sont les deux derniers avions à fixer leurs trajectoires qui subissent une déviation très importante, et non plus seulement le dernier.

Les résultats présentés dans cette section appellent les commentaires suivants :

- Dans certains cas, selon les ordres de priorité, l'algorithme ne donne aucune solution. Cela signifie que pour l'avion en question (celui pour lequel aucune solution n'est trouvée), compte tenu des trajectoires déjà fixées pour les avions qui sont prioritaires par rapport à lui, aucune trajectoire d'évitement correspondant à la modélisation par point tournant, avec l'angle $\alpha$ fixé ne lui permet d'assurer une séparation supérieure à la norme de séparation majorée $n_h^*$.

  Une autre modélisation, ou même seulement un autre angle $\alpha$ pourrait lui permettre d'adopter une trajectoire d'évitement assurant sa séparation des autres avions. De même, nous n'avons pas prévu de possibilité de changement de niveau de vol, qui pourrait fournir une échappatoire.

  Il serait donc intéressant, dans le cadre de l'utilisation d'une méthode d'évitement basée sur de tels ordres de priorité, de proposer aux avions différentes modélisations des trajectoires d'évitement.

- Il est possible, dans le cas de l'ordre de priorité $(3, 1, 4, 5, 2)$ par exemple, que l'optimisation de la trajectoire de l'avion $a^2$, quatrième dans l'ordre de priorité, ait mené à une autre trajectoire pour cet avion (plusieurs solutions différentes peuvent être obtenues, à cause des différentes incertitudes liées à la largeur des pas de temps et à la valeur de $\epsilon_{res}^{sat}$). Il est possible que l'une de ces trajectoires pour l'avion $a^2$ aurait permis de trouver une trajectoire pour l'avion $a^4$. C'est là une illustration supplémentaire des limites de l'approche séquentielle : on ne peut pas choisir *a priori* parmi les trajectoires possibles pour un avion celle qui permettra à l'avion suivant dans l'ordre de priorité de trouver une trajectoire assurant sa séparation.

- Dans les cas que nous présentons ici, il n'est pas surprenant que les avions génèrent des contraintes très fortes : le temps avant le conflit est très court, les avions sont convergents. Des situations aussi défavorables sont peu probables sur des cas de conflits apparaissant sur un trafic réel. L'évaluation à proprement parler de ses méthodes ne pourra se faire que sur une simulation de trafic réel. Mais dans ce cas, il faudra tenir compte des incertitudes portant sur les positions des avions (notamment dues aux incertitudes portant sur leur vitesse-sol). Les avions ne peuvent garantir qu'ils suivront effectivement les trajectoires qu'ils prévoient de suivre.

  Un avion générant sa trajectoire en fonction des trajectoires prévues par les avions plus prioritaires que lui devra prendre en compte ces incertitudes. Ainsi, dans le cadre d'une simulation du trafic réel, des situations *a priori* moins défavorables peuvent aussi mener à des contraintes très lourdes.

La figure 2.12 montre qu'il est possible qu'aucune trajectoire ne soit obtenue, dès le troisième avion à générer sa trajectoire. Pour le même conflit à 3 avions, semblable aux conflits test présentés précédemment, mais avec des angles faibles entre les trajectoires de avions, on peut obtenir, selon l'ordre de priorité choisi, des résultats très différents : pas de solution pour le troisième avion (figure 2.12 (c)), une solution, mais avec un détour très important pour le dernier avion à générer sa trajectoire (figure 2.12 (b)), ou une solution permettant aux deux avions déviés d'obtenir un retard très faible (figure 2.12 (b)).

Ces résultats ont été obtenus avec les mêmes paramètres que les résultats précédent. Il faut noter que, toujours avec ces mêmes paramètres, la méthode globale, sans élagage ($c_e = 0$), donne les mêmes trajectoires que la méthode séquentielle avec l'ordre de priorité $(1, 2, 3)$.

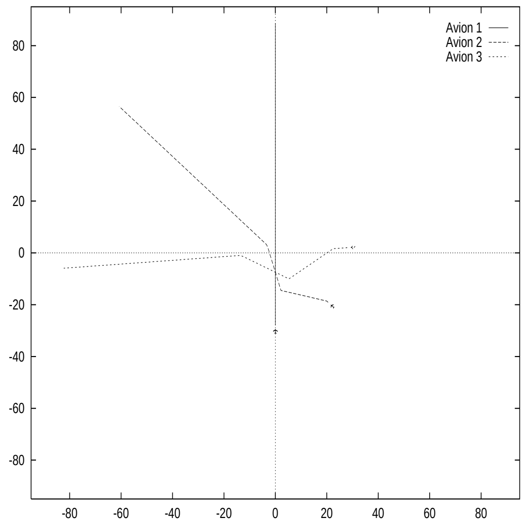
Ceci montre encore une fois la très grande importance de l'ordre de priorité utilisé, puisque, même pour un conflit à 3 avions, la méthode séquentielle, qui peut, pour un ordre de priorité donné, mener à d'aussi bons résultat que l'approche globale, peut, avec un autre ordre de priorité, n'en trouver aucun.

## 2.10 Conclusion

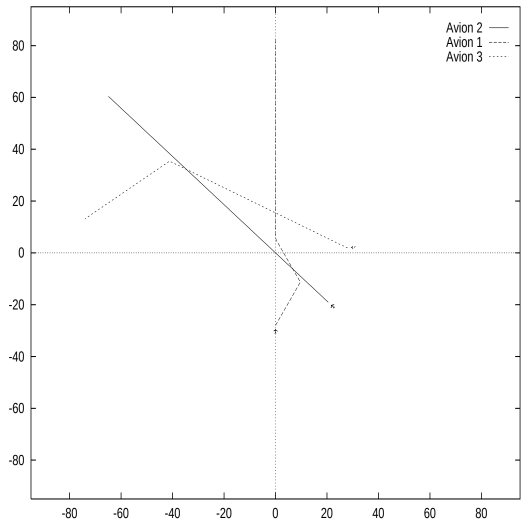
Les méthodes présentées ici ont des défauts et des qualités que l'on pouvait attendre en effet de méthodes déterministes d'optimisation globale.

Leur principale qualité, par rapport à des méthodes stochastiques, est de permettre, du moins dans le cadre de l'approche globale, d'obtenir, au vu de la solution qu'on obtient, une idée de la meilleure solution possible (voir section 2.8.6).

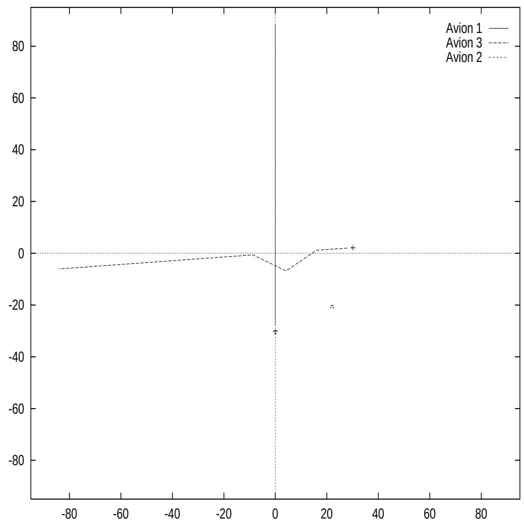
Cela s'applique certes aussi dans le cas de l'approche séquentielle, mais est moins significatif : ainsi, si, comme dans les exemples de la section 2.9.3, on ne trouve pas de solution pour le cinquième avion, lors de la résolution d'un conflit à 5 avions, cela signifie bien qu'il n'existe pas




(a) Ordre de priorité $(1, 2, 3)$




(b) Ordre de priorité $(2, 1, 3)$




(c) Ordre de priorité $(1, 3, 2)$, pas de solution

FIG. 2.12: *Trois ordres de priorité menant à des résultats très différents.*

de trajectoire normalisée permettant d'assurer que cet avion restera toujours distant des 4 autres d'au moins $n_h^*$ (norme de séparation majorée). Mais cela n'est valable qu'en considérant comme fixées les trajectoires déjà générées pour ces 4 autres avions. Cela ne signifie pas qu'il n'existe pas de solution à ce conflit à 5 avions. Cette dépendance des résultats obtenus par rapport à l'ordre de priorité est le défaut majeur de l'approche séquentielle. Non seulement les solutions proposées ne s'approchent pas forcément de l'optimum global, mais on peut même ne pas trouver de solution quand il en existe.

Le principal défaut de l'approche globale réside dans un temps de calcul qui en interdit l'usage en pratique (on ne peut pas passer 7 heures à calculer la solution à un conflit qui aura lieu dans moins de 20 minutes). Certes l'utilisation de l'élagage, sur les exemples donnés dans la section 2.9.2, entraîne des diminutions très importantes des temps de calcul. Mais ces diminutions ne peuvent pas être déterminées *a priori*. L'utilisation de l'élagage ne suffit donc pas à permettre l'utilisation de ces méthodes *en temps réel*, c'est-à-dire en garantissant qu'on obtiendra le résultat en un laps de temps donné.

À côté de ces défauts et qualités qui sont ceux de beaucoup de méthodes déterministes d'optimisation globale (fiabilité des résultats mais temps de calcul important), les méthodes présentées ici ont une qualité due à l'utilisation de méthodes d'intervalles, elles permettent de prendre en compte l'incertitude sur les instants de manœuvres. Les incertitudes sur les avions pourront être prises en compte en adaptant la forme des ensembles des positions possibles des avions à un instant $t$, pour des intervalles de manœuvres donnés.

Nous avons vu que les méthodes présentés ici reposaient sur le fait qu'on pouvait calculer facilement ces ensembles des positions possibles des avions, pour des intervalles de manœuvres donnés, à un instant $t$ donné. C'est justement dans le calcul et l'utilisation de ces ensembles que résident les défauts majeurs de ces méthodes :

- La manière dont ces ensembles sont calculés (voir section 2.5.8 et 2.6.4) suppose que les vitesses des avions soient constantes au cours de l'évitement. Si l'on souhaite généraliser la méthode à la séparation d'avions évolutifs (c'est-à-dire en montée ou en descente), ou permettre l'utilisation de manœuvres d'évitement dans le plan vertical, cela constituera un obstacle certain : les vitesses des avions ne pourront plus alors être considérées comme constantes.
- Toutes les modélisations de trajectoires d'évitement ne peuvent pas être utilisées dans le cadre de ces méthodes (on a vu par exemple dans la section 2.5 que dans le cas de l'évitement par point tournant, il était nécessaire que l'angle de retour sur la trajectoire soit fixe, ce qui interdit l'utilisation de trajectoire par point tournant avec retour direct, qui sont présentées dans le chapitre suivant).

La méthode présentée dans le chapitre suivant, basée sur l'utilisation d'un algorithme $A^*$, ne présente pas ces deux derniers inconvénients.

# Chapitre 3

# Application de l'algorithme $A^*$ à la génération de trajectoires d'évitement

## 3.1 Introduction

Pour résoudre le problème de l'évitement aérien, on cherche en général à construire des trajectoires sans conflit ayant une forme assez simple pour être facilement transmises et suivies par les pilotes. Ces trajectoires d'évitement sont souvent définies par des instants auxquels ont lieu des manœuvres pré-déterminées. C'est par exemple le cas des deux modélisations de trajectoires d'évitement décrites dans le chapitre précédent.

Les trajectoires d'évitement par point tournant avec retour direct décrites dans la section 3.2 de ce chapitre sont des trajectoires répondant naturellement à cette exigence de simplicité. Elles aussi sont définies par deux instants de manœuvres, auxquels prennent place des changements de caps. Les segments de droites successifs dont sont constituées ces trajectoires sont appelés les *étapes* de la trajectoire. Pour un avion se trouvant à une étape donnée de la trajectoire, effectuer un virage (éventuellement parmi plusieurs virages possibles) et passer ainsi à l'étape suivante peut être vu comme effectuer un choix entre plusieurs actions possibles (un des virages possibles, ou pas de virage). Cette manière de considérer de telles trajectoires justifie le fait de chercher à les traduire en terme de choix successifs, ce qui permet de représenter l'ensemble des trajectoires d'évitement possibles pour un avion comme un arbre, et chacune de ces trajectoires comme un chemin dans cet arbre. On peut alors, après avoir défini le *coût* d'un tel chemin, utiliser pour générer des trajectoires d'évitement optimales un algorithme tel que l'algorithme $A^*$, algorithme de recherche de chemin de coût minimal dans un arbre à coût additif, décrit dans la section 3.3. Cet algorithme souvent utilisé pour la génération de trajectoires en robotique nous a semblé pouvoir être utilisé pour générer des trajectoires sans conflit.

L'utilisation de cet algorithme obligera à restreindre l'ensemble des trajectoires envisageables, en discrétisant le temps, et en imposant que les instants de manœuvres prennent place sur des pas de temps (section 3.4). En effet, nous souhaitons traduire ces trajectoires en terme de choix *successifs*. En l'absence de discrétisation du temps, les choix peuvent s'opérer à tout instant du temps continu, l'ensemble des trajectoires possibles (issues de tels choix) ne peut pas se représenter comme un arbre.

Comme nous l'avions fait avec l'algorithme utilisant les méthodes d'intervalles présenté dans le chapitre précédent, nous décrivons deux approches différentes de la génération de trajectoires d'évitement au moyen d'un algorithme $A^*$. L'approche globale (section 3.6.1) répond aux besoins d'un système d'évitement *centralisé* (définition 1.9), et l'approche séquentielle (section 3.6.2) à ceux d'un système d'évitement *embarqué* (définition 1.10).

Les résultats obtenus avec chacune de ces approches sont présentés dans les sections 3.7.1 (approche globale) et 3.7.2 (approche séquentielle). Des éléments de comparaison entre les méthodes de ce chapitre et celles du chapitre 2, utilisant un algorithme *Branch and Bound* basé sur des méthodes d'intervalles, sont donnés dans la section 3.8.

La génération de trajectoires sans conflit, selon un mode séquentiel, par un algorithme $A^*$ a été reprise par Géraud Granger, dans le cadre de son stage de DEA effectué au LOG[1] pour développer une première approche d'un système d'évitement embarqué. Ce système a été testé sur une journée de trafic réel grâce au simulateur de trafic CATS [ABDM96]. Ce travail est résumé dans la section 3.9. Pour plus de détails, on pourra se reporter à [Gra98].

## 3.2 Modélisation des trajectoires d'évitement

Nous utiliserons des trajectoires d'évitement du type *point tournant avec retour direct vers la destination*, c'est-à-dire que les avions suivront des trajectoires normalisées, constituées de 4 étapes définies de la manière suivante :

1. Première étape ($E_0$) : jusqu'à un temps $t_0 \geq 0$, l'avion suit sa trajectoire d'origine.
2. Deuxième étape ($E_1$) : au temps $t_0$, l'avion est dévié de sa trajectoire d'origine, vers la droite ou vers la gauche, mais avec un changement de cap dont la valeur absolue est déterminée, et garde son nouveau cap jusqu'au temps $t_1$.
3. Troisième étape ($E_2$) : au temps $t_1$, l'avion prend le cap qui lui permet de rejoindre directement sa position finale.
4. Quatrième étape ($E_{fin}$) : l'avion a atteint sa position finale, et poursuit sa route sur sa trajectoire initialement prévue.

La *position finale* d'un avion, que nous appellerons aussi sa *destination*, est la position qu'occuperait l'avion, en l'absence de déviation, à la fin de la période de temps sur laquelle est assurée l'évitement. Cela correspond à la *durée de l'évitement* $t_f$ définie et utilisée dans le chapitre précédent.

La figure 3.1 montre une trajectoire d'évitement par point tournant avec retour direct vers la destination, répondant à la définition d'une trajectoire d'évitement normalisée donnée ci-dessus : l'avion est en sa position d'origine, notée $O$, au temps $t = 0$. Sa position finale est notée $D$. L'avion conserve son cap jusqu'au temps $t = t_0$. Sur l'exemple de la figure 3.1, au temps $t = t_0$, il opère un virage à gauche. Il prend au temps $t_1$ un cap de retour direct vers sa position finale. La trajectoire originellement prévue pour l'avion est indiquée en pointillés.

Notons que cette modélisation des trajectoires d'évitement est différente de la modélisation par point tournant utilisée dans le chapitre 2 et décrite dans la section 2.5. La modélisation utilisée ici

[1] Laboratoire d'Optimisation Globale ENAC/CENA

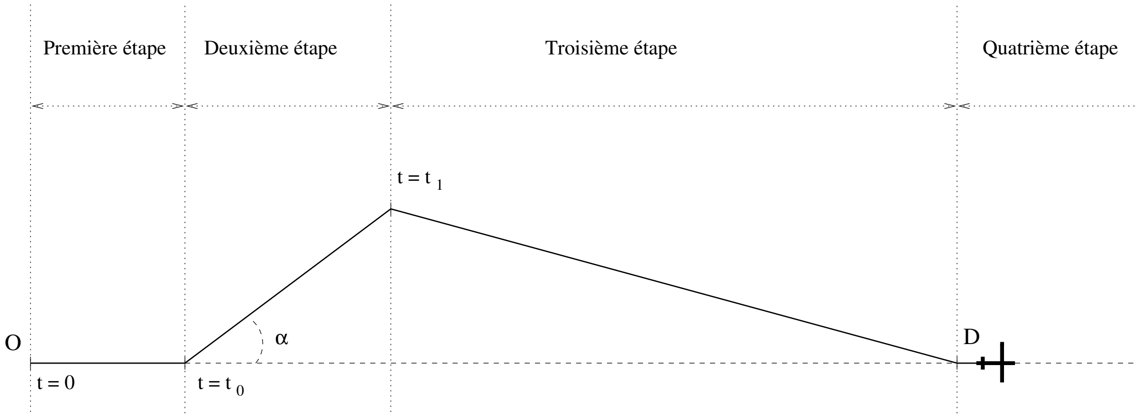



FIG. 3.1: *Trajectoire d'évitement avec un point tournant*

se distingue par le fait que les avions, lorsqu'ils prennent un cap de retour vers leurs trajectoires d'origine, prennent un cap direct vers leurs destinations, alors que dans le cadre modélisation présentée dans le chapitre 2, les avions prenaient un cap leur permettant de revenir sur leurs trajectoires initiales avec un *angle de retour* fixé. C'est pour distinguer ces deux modélisations que nous avons appelée celle qui sera utilisée dans ce chapitre la modélisation par *point tournant avec retour direct vers la destination*. Cependant, pour plus de simplicité, dans la suite de ce chapitre, elle sera appelée modélisation par *point tournant*. Cette modélisation étant la seule utilisée dans ce chapitre, il n'y aura pas risque d'ambiguité.

**Définition 3.1 – Trajectoire d'évitement normalisée –** *On appellera "trajectoire d'évitement normalisée", ou plus simplement "trajectoire normalisée" toute trajectoire correspondant à la description que nous avons donnée ci-dessus.*

Dans la suite, les étapes des trajectoires d'évitement normalisées seront aussi appelées *étapes de l'évitement*.

Une telle trajectoire d'évitement normalisée est donc entièrement déterminée par les valeurs de $t_0$ et de $t_1$ et par la direction de la déviation subie par l'avion au temps $t_0$ (à droite ou à gauche).

À chaque étape de cette trajectoire correspondent certains choix possibles : opérer un virage (et passer à l'étape suivante) ou non, et éventuellement dans quelle direction opérer le virage. Le fait de voir une trajectoire comme une suite de choix mène à envisager de représenter l'ensemble des trajectoires possibles par un arbre, et d'utiliser un algorithme de recherche de chemin optimal dans un arbre pour générer une trajectoire d'évitement optimale.

Nous présentons dans la section suivante (section 3.3) l'algorithme $A^*$. Dans la section 3.4, nous apportons à la modélisation par point tournant présentée ici les restrictions permettant de représenter une telle trajectoire comme un chemin d'un arbre.

## 3.3 L'algorithme $A^*$

La méthode présentée ici utilise l'algorithme $A^*$, qui est un algorithme de recherche de chemin de coût minimal dans un arbre à coût additif. Cet algorithme a été décrit pour la première fois par Hart, Nilsson et Raphael dans [HNR68] Nous donnons dans cette section une description rapide de l'algorithme $A^*$. Pour plus de détails, on pourra se reporter, par exemple, à [AS92].

L'algorithme $A^*$ est un algorithme permettant de rechercher dans un arbre le plus court chemin menant d'un état initial à un état terminal. Il s'agit d'un algorithme de recherche de type *meilleur en premier*, c'est-à-dire d'un algorithme dans lequel on développe à chaque étape le meilleur nœud généré non encore développé.

Par "meilleur nœud" on entend le nœud de coût total estimé minimal (cette notion de coût total estimé sera précisée ci-dessous).

**Définition 3.2 – Nœud développé –** *On dit qu'un nœud est développé quand tous ses successeurs ont été générés.*

L'utilisation de l'algorithme $A^*$ nécessite la connaissance des données suivantes :

- $u_0$ : l'état initial.
- $T$ : l'ensemble des états terminaux.
- $p_1, p_2, .., p_n$, l'ensemble des règles de production, permettant de générer les fils d'un nœud en fonction de l'état correspondant à ce nœud. Si $v$ et $u$ sont deux nœuds, et si $v$ est généré à partir de $u$ en utilisant la règle $p_i$, on note : $v \overset{p_i}{\leftarrow} u$.
- $k(u, v)$ : si $u$ et $v$ sont des nœuds de l'arbre, $k$ est une fonction donnant le coût du passage par l'arc $(u, v)$. Le coût total d'obtention d'un état donné est égal à la somme des coûts des arcs reliant l'état initial à cet état, le coût de chacun de ces arcs étant calculé au moyen de la fonction $k$. Dans le cas qui nous intéresse ici, $k(u, v)$ devra traduire la fonction que nous cherchons à minimiser, c'est-à-dire l'allongement des trajectoires.
- $h(u)$ : $u$ est un nœud de l'arbre, $h$ est une fonction heuristique qui estime le coût de passage de $u$ à un état terminal, c'est-à-dire le minimum sur tous les chemins $(u, v_1, .., v_n, u_t)$, où $u_t$ est un état terminal, de la somme :

$$k(u, v_1) + k(v_1, v_2) + .. + k(v_n, u_t)$$

  Ce minimum sera noté $h^*(u)$.

Pour décrire le fonctionnement d'un algorithme $A^*$, nous aurons besoins des variables suivantes :

- $G$ : la liste des états générés mais non encore développés.
- $premier(G)$ : une fonction qui renvoie le premier élément de la liste $G$.
- $D$ : la liste des états générés et développés.
- $g(v)$ : une fonction de l'état $(v)$ donnant le coût du trajet pour atteindre l'état $(v)$ à partir de l'état initial.
- $f(v)$ : une fonction de l'état $(v)$ donnant le coût total estimé de l'état $(v)$, c'est-à-dire que $f(v) = g(v) + h(v)$, où $h(v)$ est l'heuristique définie ci-dessus.

- $insérer\_selon\_f(G,v)$ : une fonction qui insère $v$ dans $G$ dans l'ordre $f$ croissant puis $g$ décroissant.
- $père(v)$ : un tableau indexé par les états, donnant pour un état $(v)$, l'état qui l'a généré.

Une fois définies ces variables, l'algorithme $A^*$, dans sa forme la plus générale, peut se décrire de la manière suivante :

1. $G \leftarrow (u_0)$ ; $D \leftarrow \emptyset$ ; $g(u_0) \leftarrow 0$ ; $f(u_0) \leftarrow 0$
2. **Tant que** $G \neq \emptyset$ **faire**
   (a) $u \leftarrow premier(G)$ ; $G \leftarrow G \setminus (u)$ ; $D \leftarrow D \cup (u)$
   (b) **Si** $u \in T$ **alors** donner_solution ; **fin_programme** ; **fin_si**
   (c) **Pour** $i$ **allant de** $1$ **jusqu'à** $n$ **faire**
      i. $v \stackrel{p_i}{\leftarrow} u$
      ii. **Si** $v \notin D \cup G$ **ou** $[g(v) > g(u) + k(u,v)]$ **alors**
         A. $g(v) \leftarrow g(u) + k(u,v)$
         B. $f(v) = g(v) + h(v)$
         C. $père(v) \leftarrow u$
         D. $insérer\_selon\_f(G,v)$
      iii. **fin_si**
   (d) **fin_faire**
3. **fin_faire**

Les performances de l'algorithme $A^*$ dépendent principalement de l'heuristique $h$. Rappelons que, pour un état $u$, $h(u)$ tente d'approcher $h^*(u)$, qui est le minimum de la somme :

$$k(u,v_1) + k(v_1,v_2) + .. + k(v_n,u_t)$$

sur tous les chemins $(u,v_1,..,v_n,u_t)$, $u_t$ étant un état terminal de l'arbre. Concernant cette heuristique $h$, on a les définitions suivantes :

**Définition 3.3 – Heuristique parfaite –** *Une heuristique h est dite parfaite si et seulement si :*

$$\forall u,v, \;\; h(u) = h(v) \Leftrightarrow h^*(u) = h^*(v)$$

**Définition 3.4 – Heuristique presque parfaite –** *Une heuristique h est dite presque parfaite si et seulement si :*

$$\forall u,v, \;\; h(u) < h(v) \Leftrightarrow h^*(u) < h^*(v)$$

**Définition 3.5 – Heuristique monotone –** *Une heuristique h est dite monotone si et seulement si :*

$$\forall u, \forall v \; descendant \; de \; u, \; h(u) - h(v) \leq k(u, v)$$

**Définition 3.6 – Heuristique minorante –** *Une heuristique h est dite minorante si et seulement si :*

$$\forall u, \;\; h(u) \leq h^*(u)$$

En fonction de la qualité de l'heuristique, les performances et la complexité de l'algorithme varient :

- Si l'heuristique $h$ est parfaite (voir définition 3.3), alors le premier chemin développé est un chemin de coût minimal entre l'état initial et un état terminal : l'algorithme converge alors immédiatement vers son but ;
- si l'heuristique $h$ est minorante (voir définition 3.6), alors l'algorithme $A^*$ trouve toujours un chemin optimal ;
- une heuristique monotone est minorante (voir définitions 3.5 et 3.6) et de plus, si $h$ est monotone, l'algorithme garantit que pour tout nœud $u$ développé, le chemin calculé jusqu'à $u$ est optimal, c'est-à-dire que $g(u)$ est égal au coût du plus court chemin menant du nœud initial de l'arbre au nœud $u$.

Si on s'intéresse à la complexité de l'algorithme $A^*$, on peut citer les propriétés suivantes, qui précisent l'ordre de grandeur du nombre de nœuds de l'arbre qui seront considérés par l'algorithme avant d'obtenir la solution, en fonction du nombre total de nœuds de l'arbre, noté $N$ :

- dans le cas le plus général, la complexité est en $2^N$ ;
- si l'heuristique $h$ est minorante (voir définition 3.6), on peut, en modifiant très légèrement l'algorithme, obtenir une complexité en $N^2$ ;
- si l'heuristique $h$ est monotone (voir définition 3.5), la complexité est linéaire en $N$.

Dans l'application que nous allons présenter de l'algorithme $A^*$ à la résolution de conflit, cette mesure de la complexité ne sera pas facilement utilisable, car le nombre total de nœuds de l'arbre de recherche que nous utilisons sera difficile à estimer. Il existe une autre mesure de la complexité de l'algorithme $A^*$, donnant un ordre de grandeur du nombre de nœuds de l'arbre qui seront considérés par l'algorithme avant d'obtenir la solution, en fonction du nombre d'arcs entre l'état initial l'état de l'arbre et un état final. Ce nombre d'arcs étant noté $M$ et le facteur de branchement de l'arbre étant noté $K$, on a les propriétés suivantes :

- si $h$ est presque parfaite, la complexité de l'algorithme est linéaire.
- si $h$ est monotone, la complexité de l'algorithme est exponentielle, en $K^M$.
- on a, pour une heuristique minorante, des raffinements du calcul de la complexité de l'algorithme en fonction de l'écart entre $h$ et $h^*$. Dans l'application de l'algorithme $A^*$ à l'évitement que nous allons présenter ici, nous n'aurons pas de moyen d'évaluer cette différence suffisamment précisément pour l'utiliser dans le calcul de la complexité.

## 3.4 Trajectoires d'évitement discrétisables

L'utilisation d'un algorithme de parcours d'arbre, comme l'algorithme $A^*$, pour la recherche d'une trajectoire d'évitement, nécessite de pouvoir construire un arbre possédant les caractéristiques suivantes :

- La racine de l'arbre correspond à *l'état initial de l'évitement* : au temps 0, les avions ont leurs positions, vitesses et caps initiaux.
- Les feuilles terminales de l'arbre correspondent aux *états finaux de l'évitement* : tous les avions ont atteint leurs positions finales.
- Toutes les trajectoires possibles dans le cadre de la modélisation adoptée peuvent être représentées par un chemin entre la racine de l'arbre et une des feuilles.
- Le coût d'un chemin sur l'arbre doit correspondre à la fonction que l'on cherche à minimiser (dans ce chapitre nous nous attacherons à minimiser l'allongement des trajectoires, et c'est donc cet allongement que devra traduire la fonction donnant le coût d'un état correspondant à un nœud de l'arbre).

La modélisation que nous utiliserons pour les trajectoires d'évitement devra permettre la construction d'un tel arbre.

Décrire ne trajectoire normalisée, telle qu'elle est définie dans la section 3.2 (définition 3.1), par un chemin d'un arbre implique de la traduire en terme d'embranchements, c'est-à-dire en terme de choix successifs.

Considérons les possibilités de choix offertes à un avion au cours du déroulement de sa trajectoire d'évitement :

Lorsqu'un avion est encore à la première étape de l'évitement ($E_0$), il a à tout moment le choix entre trois actions :

1. Poursuivre sa route en conservant son cap d'origine, en restant dans la première étape de l'évitement.
2. Modifier sa trajectoire en opérant un virage normalisé à gauche et passer ainsi à la deuxième étape de l'évitement ($E_1$).
3. Opérer un virage normalisé à droite et passer à la deuxième étape de l'évitement ($E_1$).

Si l'avion opère un virage, à droite ou à gauche, et passe ainsi à la seconde étape de l'évitement, il n'a plus le choix, alors, qu'entre deux actions possibles :

1. Poursuivre sa route en conservant son cap modifié, en restant dans la deuxième étape de l'évitement ($E_1$).
2. Opérer un virage pour suivre un cap direct vers sa position finale, il passe alors à la troisième étape de l'évitement ($E_2$).

S'il opère ce virage et passe à la troisième étape de sa trajectoire d'évitement l'avion n'a plus alors aucun choix : il garde le même cap jusqu'à ce qu'il atteigne sa position finale et reprend à ce moment sa route originelle.

Cette traduction en terme de choix, pour chaque avion, de la trajectoire d'évitement normalisée ne permet toujours pas, cependant, de définir l'arbre sur lequel opérera l'algorithme $A^*$. En effet, si $t_0$ et $t_1$ peuvent prendre n'importe quelle valeur réelle pourvu que l'on ait $0 \leq t_0 \leq t_1$, les choix décrits ci-dessus peuvent avoir lieu à tout moment.

Nous souhaitions traduire la trajectoire d'évitement normalisée en terme de choix successifs. Il est nécessaire pour cela de discrétiser le temps.

Le temps est donc divisé en *pas de temps* d'une longueur constante, notée $l_t$. À chaque pas de temps, chaque avion pourra modifier ou non sa trajectoire, selon l'étape de l'évitement dans laquelle il se trouvera. On a donc ainsi une suite discrète d'instants auxquels des choix sont opérés. Les temps $t_0$ et $t_1$ prennent des valeurs discrètes et les trajectoires d'évitement ainsi définies peuvent être représentées par les chemins d'un arbre.

Nous pouvons maintenant, à partir de définitions 3.1 donner les définitions des *trajectoires d'évitement normalisées discrétisables* et des *jeux de trajectoires admissibles* que nous utiliserons dans la suite de ce chapitre :

**Définition 3.7 – Trajectoire d'évitement normalisée discrétisable –** *Dans la suite de ce chapitre, l'expression* trajectoires d'évitement normalisées discrétisables *désignera des trajectoires telles qu'elles sont décrites dans la définition 3.1, c'est-à-dire des trajectoires comportant les différentes étapes de l'évitement décrites dans la section 3.2 et qui sont de plus compatibles avec la discrétisation du temps, c'est-à-dire pour lesquelles les temps $t_0$ et $t_1$ correspondent à des pas du temps discrétisé.*

Dans la suite de ce chapitre, on pourra, pour plus de simplicité employer l'expression *trajectoire d'évitement normalisée* au lieu de *trajectoire d'évitement normalisée discrétisable*. Toutes les trajectoires normalisées considérées seront discrétisables, il n'y aura donc pas risque d'ambiguïté.

**Définition 3.8 – Jeu de trajectoires admissible –** *Dans le cadre de la résolution d'un conflit donné, impliquant $n$ avions, on appellera "jeu de trajectoires admissible", l'ensemble de $n$ trajectoires d'évitement normalisées discrétisables (définition 3.7) et telles que si les avions suivent ces trajectoires, alors :*

- *tous les avions atteignent leurs positions finales ;*
- *entre le temps 0 et le temps auquel tous les avions ont atteint leurs positions finales, il n'y a aucun conflit entre des avions initialement impliqués dans le conflit que l'on s'attachait à résoudre.*

Nous définissons aussi les *longueurs* des trajectoires d'évitement normalisées et des jeux de trajectoires admissibles :

**Définition 3.9 – Longueur d'une trajectoire d'évitement normalisée –** *On appelle "longueur" d'une trajectoire d'évitement normalisée d'un avion la longueur de la partie de cette trajectoire située entre la position initiale et la position finale de cet avion. L'unité de longueur est le mille nautique ($NM$).*

**Définition 3.10 – Longueur d'un jeu de trajectoires admissible –** *On appelle "longueur" d'un jeu de trajectoires admissible la somme des longueurs, au sens de la définition précédente, des trajectoires d'évitement normalisées constituant ce jeu de trajectoires admissible. Dans la suite de ce chapitre, pour un jeu de trajectoires admissible $J$, on notera $L(J)$ sa longueur.*

## 3.5 L'arbre de recherche

Pour les raisons évoquées dans la section précédente, le temps est discrétisé. À chaque pas de temps on peut considérer l'état de l'évitement, défini de la manière suivante :

**Définition 3.11 – État de l'évitement –** *Pour un évitement impliquant $n$ avions, notés $a_i$ pour $i = 1\ ..\ n$, un état de l'évitement est donné par la position et le cap de chaque avion $a_i$, et l'étape de la trajectoire d'évitement dans laquelle se trouve chaque avion $a_i$.*

Chaque nœud de l'arbre sur lequel opérera l'algorithme $A^*$ correspondra à un état de l'évitement et à un pas de temps donnés.

Ainsi la racine de l'arbre correspond au temps 0 et à l'état dans lequel tous les avions sont en leurs positions initiales, ont leurs caps initiaux et sont tous à l'étape $E_0$.

À partir de la définition des trajectoires d'évitement normalisées (définition 3.1), on peut définir les règles de génération permettant de créer les fils d'un nœud courant de l'arbre.

On considère un nœud de l'arbre, noté $u$, on note $j_u$ le pas de temps correspondant, et pour chaque avion $a_i$ on note $(x_i(u), y_i(u))$ sa position, $c_i(u)$ son cap, et $E_{a_i}(u)$ l'étape de la trajectoire normalisée dans laquelle il se trouve.

Le pas de temps correspondant à tous les fils du nœud $u$ sera $j_u + 1$.

Nous avons vu dans la section précédente quels sont à chaque pas de temps les choix possibles pour un avion. Selon sa position, son cap et l'étape de l'évitement dans laquelle il se trouvait dans l'état de l'évitement correspondant au nœud $u$, chaque avion $a_i$ pourra donc occuper différentes positions, suivre différents caps, être dans différentes étapes de l'évitement. Les différents états correspondant aux différents fils du nœud $u$ seront donc obtenus par la combinaison des différents choix possibles pour les avions à partir de l'état correspondant au nœud $u$.

L'exemple ci-dessous illustre ce qui précède.

On génère les trajectoires de 3 avions, et on a au nœud $u$ l'état suivant :

- Avion $a_1$ en $(x_1(u), y_1(u))$ avec le cap $c_1(u)$, en première étape de l'évitement (étape $E_0$).
- Avion $a_2$ en $(x_2(u), y_2(u))$ avec le cap $c_2(u)$, en deuxième étape de l'évitement (étape $E_1$).
- Avion $a_3$ en $(x_3(u), y_3(u))$ avec le cap $c_3(u)$, en troisième étape de l'évitement (étape $E_2$).

Au pas de temps suivant, on peut choisir pour l'avion $a_1$ parmi 3 possibilités, pour l'avion $a_2$ entre 2 seulement, et l'avion $a_3$, en étape $E_2$, n'a aucun choix. L'avion $a_3$, en effet, soit a atteint sa position finale et passe à l'étape $E_{fin}$, soit ne l'a pas atteinte et dans ce dernier cas poursuit sa route vers elle sans changement de cap. Ces deux possibilités ne dépendent d'aucun choix, mais découlent simplement de la position de l'avion $a_3$. Supposons que cette position est telle que l'avion $a_3$ poursuit sa route.

Le nœud $u$ engendre donc 6 fils (voir figure 3.2) , correspondant aux 6 cas suivants :

1. Les trois avions poursuivent leurs routes en conservant leurs caps et en restant aux mêmes étapes de l'évitement.

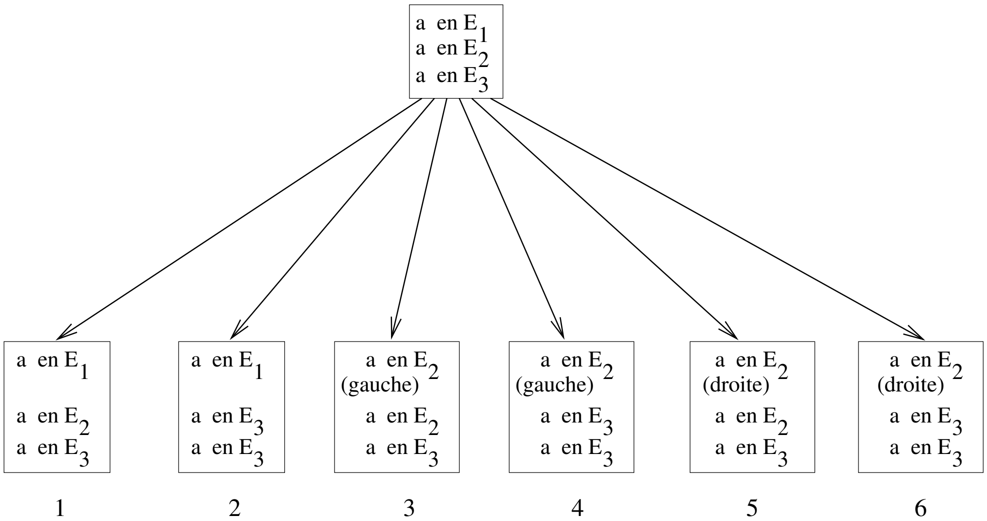



FIG. 3.2: *Les fils du nœud u*

2. Les avions $a_1$ et $a_3$ poursuivent leurs routes en conservant leurs caps et en restant aux mêmes étapes de l'évitement, tandis que l'avion $a_2$ prend un cap direct vers sa position finale, passant ainsi à l'étape $E_2$.
3. L'avion $a_1$ opère un virage à gauche, passant ainsi à l'étape $E_1$, les deux autres avions conservant leurs caps courants et restant aux mêmes étapes de l'évitement.
4. L'avion $a_1$ opère un virage à gauche et passe à l'étape $E_1$ tandis que l'avion $a_2$ prend un cap direct vers sa position finale et passe à l'étape $E_2$ et que l'avion $a_3$ conserve son cap courant et reste à l'étape $a_3$.
5. L'avion $a_1$ opère un virage à droite et passe à l'étape $E_1$, les deux autres avions conservant leurs caps courants et restant aux mêmes étapes.
6. L'avion $a_1$ opère un virage à droite et passe à l'étape $E_1$ tandis que l'avion $a_2$ prend un cap direct vers sa position finale et passe à l'étape $E_2$ et que l'avion $a_3$ conserve son cap courant et reste à l'étape $E_2$.

Au pas de temps suivant, une fois fait le choix d'opérer ou non un virage et, dans le cas du passage de l'étape $E_0$ à l'étape $E_1$, une fois le sens du le sens du virage choisi, on est dans un des 6 cas décrits ci-dessus. Les positions et les caps des avions sont alors totalement déterminés. On a donc pour les fils du nœud $u$ 6 états possibles parfaitement déterminés.

Ainsi se déduisent, à partir de l'état de l'évitement correspondant au nœud $u$, les différents états de l'évitement correspondant aux différents fils du nœud $u$. En appliquant à un nœud de l'arbre de recherche les règles définies ci-dessus, on a donc pour un nœud donné de l'arbre un nombre de fils dépendant de l'état correspondant à ce nœud, qu'on peut calculer de la manière suivante :

> *Pour un évitement impliquant $n$ avions, si $u$ est un nœud de l'arbre de recherche correspondant à un état dans lequel il y a $n_0$ avions à l'étape $E_0$ de l'évitement, $n_1$ avions à l'étape $E_1$, $n_2$ avions à l'étape $E_2$ et $n_{fin}$ avions à l'étape $E_{fin}$, le nœud $u$ génère un nombre de fils égal à $3^{n_0} * 2^{n_1}$.*

Ceci découle directement du fait qu'il y a 3 choix possibles pour chaque avion en première étape, et deux pour chaque avion en deuxième étape. On voit que les nombres d'avions en troisième étape ($n_3$) et en dernière étapes ($n_{fin}$) dans l'état de l'évitement correspondant au nœud $u$ n'influent pas sur le nombre de fils du nœud $u$, les seuls choix possibles correspondant aux avions en première ou en deuxième étape de l'évitement.

## 3.6 Algorithmes utilisés

Comme pour l'application de méthodes d'intervalles au problème de l'évitement (chapitre 2), nous avons utilisé un algorithme de type $A^*$ pour traduire deux approches du problème de l'évitement : l'approche *globale* d'une part, qui correspond à un système d'évitement centralisée, et d'autre part l'approche *séquentielle*, qui correspond à un système d'évitement *embarqué*.

Ces deux approches sont semblables, dans leurs principes, aux approches correspondantes décrites et utilisées dans le chapitre 2 : l'approche globale génère simultanément les trajectoires des $n$ avions impliquées dans le conflit et vise à minimiser la somme des retards de avions (c'est-à-dire la somme des longueurs de leurs trajectoires d'évitement), tandis que l'approche séquentielle génère ces trajectoires successivement, après qu'un ordre de priorité a été fixé entre les avions, en minimisant le retard de chaque avion, compte tenu des contraintes imposées par les trajectoires déjà générées.

### 3.6.1 Approche globale

Nous présentons dans cette section un algorithme basé sur l'algorithme $A^*$ dont le but est de trouver, pour les avions impliqués dans un conflit, des trajectoires d'évitement normalisées optimales, c'est-à-dire répondant à la définition suivante :

**Définition 3.12 – Trajectoires d'évitement normalisées optimales –** *Dans le cadre de la résolution d'un conflit impliquant $n$ avions, on appelle trajectoires d'évitement normalisées optimales $n$ trajectoires normalisées discrétisables (définition 3.7) telles que :*

1. *elles forment un jeu de trajectoires admissible (voir définition 3.8) ;*
2. *ces trajectoires sont telles qu'elles correspondent à un allongement minimal des trajectoires des avions, en ce sens que si on note $J_{opt}$ le jeu de trajectoires admissible donné par l'algorithme, la longueur de $J_{opt}$ (voir définition 3.10) est minimale sur l'ensemble des jeux de trajectoires admissible pour la résolution du conflit, c'est-à-dire que l'on a pour tout jeu de trajectoires admissible $J$ :*

$$L(J) \geq L(J_{opt}) \tag{3.1}$$

Dans la suite de ce chapitre, on emploiera, pour plus de simplicité, le terme de *trajectoires optimales* pour désigner des trajectoires d'évitement normalisées optimales.

L'algorithme d'optimisation globale que nous utilisons est l'algorithme $A^*$ générique tel qu'il a été décrit dans la section 3.3, pour lequel il nous reste à définir les données suivantes :

**L'état initial :** comme nous l'avons vu (section 3.5), l'état initial est l'état de l'évitement dans lequel les avions occupent leurs positions initiales, ont leurs caps initiaux et sont tous en étape ($E_0$). Cet état correspond au temps 0.

**Les états terminaux :** les états terminaux sont les états de l'évitement dans lesquels tous les avions sont en dernière étape de l'évitement ($E_{fin}$), c'est-à-dire quand tous les avions ont atteint leurs positions finales.

**Les règles de production :** Nous avons vu comment sont générés les fils d'un nœud en fonction de l'état correspondant à ce nœud dans la section 3.5. On a ainsi un ensemble de règles de production : on passe d'une règle à l'autre en modifiant le choix opéré pour l'un des avions.

L'ordre dans lequel ces règles sont générées n'a pas d'importance : l'ordre dans lequel les différents fils d'un même nœud sont traités par l'algorithme ne dépendra pas de l'ordre dans lequel ils ont été générés, mais uniquement des différentes valeurs que prennent sur eux la fonction coût et la fonction heuristique.

**La fonction coût :** On cherche à déterminer le coût du passage d'un nœud $u$ de l'arbre à un nœud $v$ généré à partir de $u$. Le coût de passage de $u$ à $v$ devra traduire l'allongement des trajectoires entraîné par les actions menées par les avions pour passer de l'état de l'évitement correspondant à $u$ à l'état de l'évitement correspondant à $v$.

On cherche à minimiser la somme des longueurs des trajectoires d'évitement entre les positions initiales et les positions finales des avions. La distance parcourue par un avion doit donc être systématiquement prise en compte tant qu'il n'a pas atteint sa position finale, et ne doit plus être prise en compte une fois qu'il a atteint cette position finale, c'est-à-dire une fois qu'il est à l'étape $E_{fin}$ d'une trajectoire d'évitement normalisée.

On veut de plus ne considérer que des jeux de trajectoires admissibles (voir définition 3.8), c'est-à-dire qu'on ne considérera que des trajectoires telles que si les avions suivent ces trajectoires, il n'y a à aucun moment conflit entre eux.

Or, d'après la définition de trajectoires d'évitement normalisées (voir définition 3.7), les changements de cap (et éventuellement de vitesse) se font de manière discrète, à des instants correspondant à des pas de temps, et donc entre deux pas de temps consécutifs les trajectoires des avions sont rectilignes uniformes.

Puisque $u$ est le père de $v$, il y a un pas de temps et un seul entre le nœud $u$ et le nœud $v$. Les trajectoires des avions entre leurs positions dans l'état de l'évitement correspondant au nœud $u$ et leurs positions dans l'état de l'évitement correspondant au nœud $v$ seront rectilignes et uniformes. Or, vérifier que les conditions de séparation sont respectées entre deux avions dont les trajectoires sont rectilignes et uniforme revient à résoudre une simple équation du second degré.

Il est donc facile de vérifier si le passage du nœud $u$ au nœud $v$ entraîne un conflit entre les avions. Si c'est le cas, il est inutile que le nœud $v$ soit développé par l'algorithme.

Selon que le passage du nœud $u$ au nœud $v$, où $u$ est le père du $v$, entraîne ou non un conflit entre les avions, on définira donc le coût $k(u, v)$ de l'arc $(u, v)$ de deux manières différentes :

- Si les avions restent séparés lors du passage du nœud $u$ au nœud $v$ : les $n$ avions qui sont impliqués dans le conflits sont notés $a_i$ pour $i = 1 .. n$ ; on note $d_{a_i}(u, v)$ la distance entre la position de l'avion $a_i$ dans l'état de l'évitement correspondant au nœud $u$ et sa position dans l'état de l'évitement correspondant au nœud $v$, et $A_u$ l'ensembles des avions qui, dans l'état de l'évitement correspondant au nœud $u$, n'ont pas atteint leurs positions finales, c'est-à-dire ne sont pas encore à l'étape de l'évitement $E_{fin}$ ; $k(u,v)$, le coût de l'arc menant du nœud $u$ au nœud $v$ est alors donné par la somme suivante :

$$k(u,v) = \sum_{a_i \in A_u} d_{a_i}(u,v) \tag{3.2}$$

- Si au contraire les avions ne restent pas séparés lors du passage du nœud $u$ au nœud $v$. On ne souhaite pas développer le nœud $v$ : tout chemin contenant l'arc $(u,v)$ correspond à un jeu de trajectoires ne maintenant pas les avions séparés. On affecte donc à l'arc $(u,v)$ un coût considéré comme étant infini[2], l'algorithme que nous utiliserons ne développera pas les nœuds de coût infini.

  L'algorithme $A^*$ développant d'abord les nœuds de coût total estimé minimal, de tels nœuds ne seraient de toute manière développés que s'il ne restait dans l'arbre aucun nœud non encore développé de coût non infini. Dans ce cas, l'algorithme que nous utilisons s'arrête, et signale qu'il n'a pas trouvé de solution admissible. Nous verrons dans la section 3.6.3 quelle information cela donne quant à l'existence d'une solution au problème de l'évitement.

**L'heuristique :** rappelons que le but de l'heuristique est d'approcher, pour un nœud $v$ généré par l'algorithme, le minimum des coûts des chemins menant du nœud $v$ à un état terminal.

On note, pour un état $v$ et des avions $a_i$ avec $i = 1 .. n$, $d_{a_i}(v)$ la distance entre la position de l'avion $a_i$ dans l'état correspondant au nœud $v$ et sa position finale. La valeur de l'heuristique $h$ en $v$ est la somme :

$$h(v) = \sum_{i=1}^{n} d_{a_i}(v) \tag{3.3}$$

Cette somme est égale au coût du chemin de l'arbre correspondant au cas où, à partir de l'état de l'évitement correspondant au nœud $v$, tous les avions prendraient des caps directs vers leurs positions finales et les rejoindraient sans qu'il y ait conflit entre eux. Cela n'est bien entendu pas toujours possible. Cependant, le nœud $v$ étant donné, trois cas de figure peuvent se présenter :

- Soit il existe un tel chemin, c'est-à-dire que les avions peuvent rejoindre directement leurs positions finales sans conflit ; dans ce cas ce chemin est le chemin de coût minimal (les avions allant en ligne droite à leurs positions finales il n'existe pas de trajectoires plus courtes les y menant), et son coût est égal, par définition des distances $d_{a_i}(v)$ et de la fonction $k$, à $h(v)$ telle qu'elle a été définie ci-dessus. On a donc $h(v) = h^*(v)$, où $h^*(v)$ est le coût minimal pour passer de $v$ à un état terminal (voir section3.3).
- Soit un tel chemin n'existe pas, ce qui signifie qu'il y aurait conflit si tous les avions suivaient des routes directes vers leurs positions finales, mais les avions pourront atteindre leurs positions finales sans conflit en suivant, à partir de leurs positions, de leurs caps et des étapes de l'évitement auxquelles ils sont dans l'état de l'évitement

[2]En pratique un coût suffisamment grand pour être supérieur au coût total de n'importe quelle trajectoire sans conflit.

correspondant au nœud $v$, des trajectoires d'évitement normalisées. Ces trajectoires ne seront pas toutes directes. Celles d'entre elles qui ne sont pas directes sont strictement plus longues que les routes directes menant les avions correspondant à leurs positions finales. Les coûts des chemins correspondant à de telles trajectoires ne pourront alors qu'être strictement supérieurs à $h(v)$ puisque ces trajectoires ne sont pas toutes directes, on a donc dans ce cas $h(v) < h^*(v)$.

- Soit les avions ne pourront pas atteindre leurs positions finales sans conflit en suivant, à partir de l'état de l'évitement correspondant au nœud $v$, des trajectoires normalisées ; dans ce cas, tous les chemins menant de $v$ à un état terminal passeraient par un nœud correspondant à un état de l'évitement que les avions ne peuvent pas atteindre en demeurant séparés deux à deux, et ces chemins auraient donc tous, par définition de la fonction coût, un coût infini, et on a donc aussi dans ce cas, comme dans le cas précédent, $h(v) < h^*(v)$.

On voit que dans tous les cas on a pour tout nœud $v$ :

$$h(v) \leq h^*(v) \tag{3.4}$$

ce qui signifie que l'heuristique $h$ ainsi définie est minorante (voir définition3.6).

On a, pour deux nœud $u$ et $v$, si $v$ est un descendant de $u$ :

$$h(u) \leq k(u, v) + h(v) \tag{3.5}$$

En effet, soit $k(u, v)$ est infini, soit $k(u, v)$ est égal à la somme des déplacements nécessaires pour amener les avions de leurs positions correspondant au nœud $u$ à leurs positions correspondant au nœud $v$. $k(u, v) + h(v)$ est alors la somme des longueurs des trajectoires des avions pour aller de leurs positions correspondant à $u$ à leurs destinations, en suivant, entre leurs positions correspondant à $u$ et leurs positions correspondant à $v$, un chemin de coût $k(u, v)$, tandis que $h(u)$ est la somme des longueurs des trajectoires directes des avions de leurs positions correspondant à $u$ à leurs destinations. Les trajectoires directes étant les plus courtes pour rejoindre les destinations, on a bien toujours l'inégalité 3.5.

L'heuristique $h$ est donc non seulement minorante, mais aussi monotone (définition 3.5).

Ces données étant précisées, l'algorithme d'optimisation globale que nous utilisons est semblable à l'algorithme $A^*$ générique tel qu'il a été défini dans la section 3.3.

Cependant, comme nous le verrons dans la section 3.7.1, l'approche globale est très coûteuse en mémoire, l'ensemble des nœuds générés devant être conservé en mémoire. Pour limiter le nombre de nœuds de l'arbre de recherche générés, nous avons utilisé, comme dans le cadre de l'utilisation des méthodes d'intervalles, une technique d'élagage. Cette technique d'élagage consiste à modifier la fonction heuristique.

**L'heuristique d'élagage :** Le principe de la technique d'élagage utilisée avec l'algorithme $A^*$ consiste à ne pas développer de nœuds dont on sait qu'il n'apporteront pas une amélioration plus grande qu'un seuil d'élagage donné par rapport à une solution déjà trouvée.

La fonction heuristique utilisée en l'absence d'élagage, décrite ci-dessus, est une heuristique minorante, c'est-à-dire que l'on sait que le coût d'un chemin allant de l'état initial à un état final et passant par un nœud $v$ est forcément supérieur à $g(v) + h(v)$, où $g(v)$ est le coût du chemin reliant l'état initial au nœud $v$ et $h(v)$ est la valeur de l'heuristique en $v$.

Supposons que l'algorithme a trouvé une solution, c'est-à-dire un chemin reliant l'état initial à un état final, de coût $g_s$. Si l'algorithme a généré un nœud $u$, il ne cherchera à le développer en générant des fils de ce nœud, que si on a :

$$g(v) + h(v) < g_s \tag{3.6}$$

Cette relation signifie qu'il est possible qu'un chemin passe par $u$ en reliant l'état initial à un état terminal et en ayant un coût global inférieur au coût de la solution trouvée.

La technique d'élagage que nous utilisons consiste à utiliser une heuristique d'élagage, dépendant d'un taux d'élagage, $t_e$. cette nouvelle fonction heuristique est notée $h_{t_e}$, sa valeur au nœud $v$ est donnée par :

$$h_{t_e}(v) = t_e g(v) + (1 + t_e)h(v) \tag{3.7}$$

Supposons cette fois encore que l'algorithme a trouvé une solution de coût $g_s$. Le nœud $v$ sera développé si et seulement s'il est possible qu'un chemin passe par $u$ en reliant l'état initial et tel que son coût global soit égal à $g_s(u)$ avec :

$$g_s > (1 + t_e)g_s(u) \tag{3.8}$$

Cette heuristique n'est plus minorante. Il est d'ailleurs clair, par sa définition même, qu'elle ne garantit pas de trouver le chemin de coût minimal entre état initial et un état final, c'est-à-dire, ici, qu'elle ne garantit pas de trouver un jeu de trajectoires normalisées optimales (définition 3.12). Cependant, supposons qu'un tel jeu de trajectoire existe, et ait un coût $g_{opt}$. La définition de l'heuristique d'élagage $h_{t_e}$ entraîne que :

- Si le coût de cette solution (c'est-à-dire la somme des longueurs des trajectoires composant ce jeu de trajectoires admissible) est noté $g_{sol}$, on a :

$$g_{sol} \leq (1 + t_e)g_{opt} \tag{3.9}$$

- L'algorithme décrit ici, s'il utilise cette heuristique d'élagage $h_{t_e}$, trouvera une solution, c'est-à-dire un jeu de trajectoires admissible. En effet, si un jeu de trajectoires admissible existe, de coût $g_{ad}$, la différence entre son coût et celui d'un jeu de trajectoire entraînant un conflit sera supérieur à $(1 + t_e)g_{ad}$.

On remarque que la perte possible en terme de qualité est un pourcentage de la longueur totale des trajectoires sur la durée de l'évitement. Dans le cas de l'élagage utilisé pour les méthodes d'intervalles, on pouvait fournir une borne de la perte de qualité en terme de majoration du retard. Nous verrons dans la section 3.7.1 que, si elle permet en effet de réduire fortement le nombre de nœuds générés, l'utilisation de l'élagage nécessite des taux d'élagage relativement importants, et que son efficacité n'est pas suffisante pour obtenir une solution au-delà de 4 avions.

### 3.6.2 Approche séquentielle

Dans le cadre de l'approche séquentielle, on cherche toujours pour les $n$ avions des trajectoires d'évitement normalisées répondant à la définition 3.7, mais on ne cherche plus à générer simultanément $n$ trajectoires d'évitement normalisées optimales pour les $n$ avions. On procède séquentiellement, selon les étapes suivantes :

1. On établit entre les avions un ordre de priorité. On supposera ensuite que les avions sont numérotés de 1 à $n$ selon cet ordre.
2. La trajectoire de l'avion $a_1$ est inchangée : l'avion $a_1$ conserve sa trajectoire initialement prévue, c'est-à-dire, ici, la trajectoire qui va de sa position initiale à sa position finale en suivant une ligne droite.
3. On génère ensuite successivement les trajectoires des avions qui sont numérotés de 2 à $n$. Pour $1 < i \leq n$, on considère que les trajectoires des avions $j$, pour $j < i$ sont fixées et on génère la trajectoire de l'avion $i$ de manière à ce que :
   - l'avion $a_i$ soit à tout moment séparé de chaque avion $a_j$, pour $j < i$ ;
   - l'allongement de la trajectoire de l'avion $a_i$ soit minimal, compte tenu de ces contraintes de séparation.

   La trajectoire de chaque avion $a_i$ est générée au moyen d'un algorithme $A^*$ dont nous allons maintenant préciser les caractéristiques principales.

Comme dans la section 3.6.1, le temps et discrétisé et la recherche de trajectoire se fera sur l'ensemble des trajectoires d'évitement normalisées discrétisables.

Les gains très importants en terme de temps de calcul constatés avec une première version de cet algorithme séquentiel nous ont permis d'envisager de modifier les trajectoires d'évitement normalisées que nous utiliserons. Selon la définition d'une trajectoire d'évitement normalisée donnée dans la section 3.2, chaque avion avait, au temps $t_1$ la possibilité d'opérer soit un virage à gauche, soit un virage à droite, mais toujours d'un angle fixé, par exemple de 30 degrés. Dans l'algorithme séquentiel tel que nous le présentons ici, chaque avion au temps $t_1$ a pareillement le choix entre un virage à droite ou un virage à gauche, mais le changement de cap entraîné par ce virage pourra prendre trois valeurs différentes. Dans l'application que nous présentons ici, les avions pourront opérer un virage de 10, 20 ou 30 degrés.

Considérons, pour $i_0 > 1$, l'arbre de recherche utilisé par l'algorithme $A^*$. Ses caractéristiques sont différentes de celle de l'arbre de recherche qu'utilise l'algorithme d'optimisation globale :

**Les états de l'évitement :** les états de l'évitement utilisés par l'algorithme d'optimisation séquentielle sont définis de la même manière que ceux qui sont décrits dans la section précédente (voir définition 3.11).

L'état initial est l'état correspondant au temps 0, dans lequel les avions occupent leurs positions initiales, ont leurs caps initiaux, et sont à la première étape de l'évitement. Les états terminaux sont les états dans lesquels tous les avions ont atteint leurs positions finales.

Tous les états de l'évitement qui étaient possibles dans le cadre de l'algorithme d'optimisation globale ne le sont plus forcément dans le cadre de l'algorithme d'optimisation séquentielle. En effet, pour $i < i_0$, la trajectoire d'un avion $a_i$ est fixée : pour chaque pas de temps, la position et le cap de l'avion $a_i$, ainsi que l'étape de la trajectoire dans laquelle il se trouve, ont été déterminés lors des itérations précédentes de l'algorithme.

**Les règles de production :** le fait que les trajectoires des avions $a_i$ sont fixées, pour $i < i_0$, modifie les règles de production. En effet, à un pas de temps donné, on ne peut opérer de choix qu'en ce qui concerne l'avion $a_{i_0}$ : les étapes des trajectoires auxquelles sont les autres avions, de même que leurs positions et leurs caps, sont déjà déterminées.

De plus, si l'avion $a_{i_0}$ est à le première étape de sa trajectoire, il n'a plus le choix entre 3 possibilités (poursuivre sa route, tourner à gauche ou tourner à droite), mais entre 7 possibilités, puisqu'il peut opérer trois virages différents de chaque côté.

Si on considère un nœud $u$ de l'arbre de recherche, l'ensemble des états possibles pour les nœuds qui seront générés à partir de $u$ sera donc l'ensemble des états dans lesquels :

- L'avion $a_{i_0}$ occupe une position, a un cap et est à une étape de sa trajectoire compatible avec les position, cap et étape de trajectoire qu'il avait l'état correspondant au nœud $u$, selon les règles décrites ci-dessus.
- Pour $i < i_0$, les positions, caps et étapes des avions $a_i$, sont totalement déterminés par le pas de temps correspondant au fils du nœud $u$, conformément aux trajectoires qui ont été précédemment générées pour eux par l'algorithme.

Le nombre de nœuds générés à partir du nœud $u$ dépend donc uniquement de l'étape dans laquelle se trouvait l'avion $a_{i_0}$ dans l'état de l'évitement correspondant au nœud $u$ :

- Si l'avion $a_{i_0}$ était à la première étape de sa trajectoire d'évitement, le nœud $u$ a 7 fils (soit l'avion $a_{i_0}$ garde le même cap, soit il est dévié vers la droite, soit il est dévié vers la gauche, avec à chaque fois 3 angles de virages possibles).
- Si l'avion $a_{i_0}$ se trouvait à la seconde étape de sa trajectoire d'évitement, le nœud $u$ a 2 fils (soit l'avion $a_{i_0}$ conserve son cap, soit il change de cap pour prendre une route directe vers sa position finale).
- Dans les autres cas le nœud $u$ n'a qu'un fils.

**la fonction coût :** la fonction coût est semblable à celle qui est utilisée dans le cas de l'optimisation globale et qui est décrite dans la section 3.6.1, mais on ne calcule que l'allongement de la trajectoire de l'avion $a_{i_0}$ : les allongements des trajectoires des autres avions sont fixés puisque ces trajectoires ont déjà été déterminées.

**L'heuristique :** de même que la fonction coût, la fonction heuristique est similaire à celle que nous avons décrite dans la section 3.6.1, mais en ne prenant en compte que l'avion $a_{i_0}$. Il ne sera pas nécessaire, du fait du faible facteur de branchement de l'arbre parcouru, d'avoir recours à l'heuristique d'élagage. L'heuristique reste ainsi minorante.

Ainsi, pour un conflit impliquant $n$ avions, au lieu de mener une seule optimisation, au moyen d'un algorithme $A^*$ utilisant un arbre de recherche dont le facteur de branchement[3] est $3^n$, on mène $n-1$ optimisations au moyen, à chaque fois d'un algorithme $A^*$ utilisant un arbre de recherche dont le facteur de branchement est de 7.

### 3.6.3 Signification des résultats obtenus

L'heuristique que nous utilisons est minorante (en l'absence d'élagage), l'algorithme trouve donc le chemin de coût minimal dans l'arbre qu'il parcourt (voir section 3.3). Cependant, les chemins de l'arbre parcouru par l'algorithme, tant dans la cadre de l'approche globale que de l'approche séquentielle, ne représentent pas la totalité des trajectoires par point tournant possibles,

[3]il s'agit nombre de descendant maximal d'un nœud, correspondant aux états de l'évitement dans lesquels tous les avions sont à l'étape $E_0$ (voir section 3.5).

mais seulement les trajectoires normalisées discrétisables, c'est-à-dire telles que les instants de virages $t_0$ et $t_1$ prennent place sur des pas du temps discrétisé.

Quand l'algorithme propose une solution, elles correspond donc aux meilleures trajectoires normalisées discrétisables possibles. Il peut exister des trajectoires normalisées non discrétisables, respectant les contraintes de séparation, et menant à un retard plus faible.

La forme des trajectoires normalisées ne permet pas d'obtenir une expression directe d'un minorant du meilleur retard possible, connaissant le retard obtenu par l'algorithme.

Dans le cas où l'algorithme trouve une solution, en notant $l_t$ la largeur des pas de temps et $r_{min}$ la valeur de la somme des retards correspondant à la solution trouvée par l'algorithme, on sait, en reprenant les notations du chapitre 2, qu'il n'existe pas d'intervalles de manœuvres (définition 2.12) de largeur intérieure (définition 2.18) supérieure à $l_t$ tel que tous les instants de manœuvres définis par cet intervalle correspondent à des trajectoires entraînant un retard inférieur à $r_{min}$.

Comme dans le cas de l'utilisation de méthodes d'intervalles (chapitre 2), l'interprétation du fait que l'algorithme ne trouve pas de solution nécessite l'introduction d'une norme de séparation majorée.

Supposons que les $n$ avions suivent des trajectoires d'évitement par point tournant quelconques, c'est-à-dire des trajectoires telles que les instants $t_0$ et $t_1$ correspondants ne coïncident pas avec des pas du temps discrétisé.

Il est immédiat de vérifier que, quelle que soit la trajectoire normalisée suivie par l'avion $a_i$, il existe une trajectoire normalisée discrétisable pour l'avion $a_i$ telle que, à tout instant $t$, la distance entre la position effective de l'avion $a_i$ et la position qu'il occuperait s'il suivait cette trajectoire normalisée discrétisable reste inférieure à une distance $\delta_i$ :

$$\delta_i = e_t\, v_i \tag{3.10}$$

où $v_i$ est la vitesse de l'avion $a_i$ et $l_t$ la largeur des pas de temps.

Cette trajectoire est la trajectoire normalisée discrétisable dont les instants de manœuvres $t_0$ et $t_1$ sont les plus proches de ceux de la trajectoire de $a_i$.

On note $\delta_{max}$ le maximum de ces distances $\delta_i$ :

$$\delta_{max} = l_t \max_{i \in \{1,..,n\}} v_i \tag{3.11}$$

Il est clair alors que quelles que soient les trajectoires normalisées des $n$ avions, il existe un jeu de trajectoires admissible, tel que si la distance minimale entre les avions au cours de l'évitement est notée $d_{min}^{ef}$, et la valeur qu'aurait cette distance minimale, s'ils suivaient les trajectoires de ce jeu de trajectoires admissible est notée $d_{min}^{ad}$, on a :

$$|d_{min}^{ef} - d_{min}^{ad}| \leq 2\,\delta_{max} \tag{3.12}$$

Nous pouvons maintenant définir une norme de séparation majorée.

**Définition 3.13 – Norme de séparation majorée –** *À partir de la norme de séparation horizontale, notée $n_h$, on définit une norme de séparation majorée, notée $n_h^*$, par :*

$$n_h^* = n_h + 2\,\delta_{max} \tag{3.13}$$

*où $\delta_{max}$ est donnée par l'équation 3.11*

Ainsi, si l'algorithme ne trouve pas de solution, on sait qu'il n'existe pas de trajectoires normalisées, discrétisables ou non, assurant que la distance entre les avions reste, tout au long de l'évitement, supérieure à la norme de séparation majorée.

Ce résultat ressemble à celui que nous avions dans le chapitre 2 dans le cas où les algorithmes utilisant des méthodes d'intervalles ne trouvaient pas de solution (section 2.8.6). Cependant, une différence notable existe entre ces deux cas. Dans les deux cas, la différence entre la norme de séparation $n_h$ et la norme de séparation majorée $n_h^*$ est proportionnelle à la largeur d'un pas de temps. Dans le cas des méthodes d'intervalles, ce pas de temps est utilisé pour vérifier la séparation des avions, le temps de calcul est linéaire par rapport au nombre de ces pas de temps. Dans le cas de l'utilisation des méthodes $A^*$, les pas de temps correspondant aux instants où les avions peuvent effectuer ou non un virage. C'est donc la longueur des branches de l'arbre parcourues par l'algorithme qui est proportionnelle au nombre des pas de temps.

## 3.7 Résultats

Les résultats présentés ici ont été obtenus sur un ordinateur (PC) équipé d'un processeur *pentium 2* de 400 MHz. La taille de la mémoire utilisable par le programme a été fixée à 96 méga-octets.

### 3.7.1 Approche globale

Le principal problème rencontré par l'algorithme $A^*$, quand il est utilisé dans le cadre de l'approche globale du problème de l'évitement (section 3.6.1) est que l'algorithme génère un nombre très grand de nœuds de l'arbre qu'il explore. Ces nœuds, gardés en mémoire, entraînent vite un dépassement des capacités mémoires allouées.

Nous avons testé cet algorithme sur des cas-tests de conflits semblables à ceux utilisés pour tester l'algorithme utilisant les méthodes d'intervalles (chapitre 2). Ces cas tests, ici, sont de la forme suivante :

- Au temps 0 les avions se trouvent tous dans un même plan horizontal et sont régulièrement répartis sur un cercle de 28 NM de rayon.
- Les avions convergent tous vers le centre de ce cercle.
- La position finale de chaque avion est le point de la trajectoire de l'avion situé à 80 NM au-delà du centre de ce cercle.
- Les avions volent tous dans un plan horizontal, à la même vitesse constante de 420 nœuds.

| Pas de temps (en secondes) | Taux d'élagage | | | |
|---|---|---|---|---|
| | 0 | 0,05 | 0,10 | 0,15 |
| 45 | $\Delta_l = 6,75\ NM$ $N_n = 130\,389$ | $\Delta_l = 6,75\ NM$ $N_n = 3\,199$ | $\Delta_l = 6.75\ NM$ $N_n = 2\,209$ | $\Delta_l = 6,75\ NM$ $N_n = 2\,083$ |
| 30 | dépassement mémoire | $\Delta_l = 9,0\ NM$ $N_n = 14\,805$ | $\Delta_l = 9,0\ NM$ $N_n = 8\,855$ | $\Delta_l = 9,0\ NM$ $N_n = 7\,256$ |
| 20 | dépassement mémoire | $\Delta_l = 5\ NM$ $N_n = 33\,547$ | $\Delta_l = 9,3\ NM$ $N_n = 24\,893$ | $\Delta_l = 9,3\ NM$ $N_n = 22\,214$ |
| 15 | dépassement mémoire | $\Delta_l = 5,5\ NM$ $N_n = 123\,020$ | $\Delta_l = 5,5\ NM$ $N_n = 89\,306$ | $\Delta_l = 5,5\ NM$ $N_n = 79\,470$ |
| 10 | dépassement mémoire | dépassement mémoire | $\Delta_l = 4,4\ NM$ $N_n = 337\,280$ | $\Delta_l = 6,0\ NM$ $N_n = 287\,124$ |

TAB. 3.1: *Influence du pas de temps et du taux d'élagage sur le cas-test à trois avions*

En l'absence d'évitement les avions se rejoindraient tous au centre du cercle en 4 minutes.

La norme de séparation, $n_h$, est fixée à 4 NM.

La largeur du pas de temps sera notée $l_t$. Si l'élagage est utilisé, le taux d'élagage sera noté $t_e$.

Le cas-test à deux avions est traité sans difficulté par l'algorithme pour des pas de temps de 15, et même de 10 secondes : pour 15 secondes, 75 000 nœuds de l'arbre ont dû être générés, en 15 secondes de temps cpu. Pour $l_t = 10\ s$, 202 000 nœuds ont été généré pour un temps de calcul de 46 secondes de temps cpu.

Le conflit à trois avions pose beaucoup plus de problème. Le temps de calcul en lui même n'est pas une donnée traduisant la difficulté qu'a rencontrée l'algorithme. Le nombre de nœuds générés (et l'encombrement de la mémoire qu'il entraîne) est le facteur limitant. Dans les cas à trois avions, 400 000 nœuds environ suffisaient à dépasser la mémoire allouée (96 méga-octets), en 2 minutes de temps cpu.

Pour pallier ce problème de consommation de mémoire, on pourrait utiliser, au lieu de l'algorithme $A^*$, un algorithme de recherche en approfondissement itératif, tel que celui qui est décrit dans [Kor85].

Nous avons testé l'approche globale sur le cas-test à trois avions, avec différentes valeurs pour le pas de temps $t_e$ et pour le taux d'élagage $l_t$. Les résultats obtenus sont présentés dans le tableau 3.1.

Le résultat le plus important donné par ce tableau est que, à moins d'augmenter le pas de temps jusqu'à 45 secondes, on ne peut pas, avec la mémoire dont on dispose, obtenir de résultat sans élagage. Rappelons que la recherche d'une solution, c'est-à-dire de trajectoires sans conflit, est limitée à l'ensemble des trajectoires normalisées discrétisables. Si le pas de temps augmente, le nombre de ces trajectoires discrétisables diminue, ce qui explique que le nombre de nœuds générés par l'arbre diminue. Mais le risque de ne pas trouver de solution au problème augmente. Il est tout à fait possible que, dans certains cas de conflit, les seules solutions possibles consistent à placer un des virages d'un avion dans un intervalle de temps restreint. Le fait d'augmenter le pas de temps risque d'amener à ne pas trouver de telle solution.

Un autre résultat visible dans le tableau 3.1 est que l'utilisation de l'élagage diminue fortement le nombre de nœuds générés par le réseau sans entraîner de détérioration trop importante de la performance (en terme d'allongement de la trajectoire).

Ces résultats, obtenus sur des cas-tests, ne sont pas forcément généralisables. Rappelons que l'utilisation d'un taux d'élagage de 0,1, par exemple, peut théoriquement entraîner un allongement allant jusqu'à 10 % de la longueur des trajectoires obtenues par rapport aux trajectoires (discrétisables) optimales. Avec les valeurs numériques utilisées ici, une détérioration de 10 % correspond à un allongement de $10,8\ NM$ de la trajectoire de chacun des avions, c'est-à-dire à une augmentation du retard de l'ordre de 90 secondes par avion. L'élagage utilisé avec les méthodes d'intervalles (voir chapitre 2) menait à des détériorations maximales des performances souvent plus faibles (voir sections 2.8.5 et 2.9.2).

Cependant, même si la limite supérieure de la détérioration des résultats introduite par l'utilisation de l'élagage est élevée, ce n'est pas uniquement *par chance* que les résultats obtenus sur les cas tests, et présentés dans le tableau 3.1 montrent une détérioration très faible de ces résultats. En effet, l'algorithme

$$A^*$$

développe d'abord les nœuds dont le coût estimé est plus faible, c'est-à-dire ici les nœuds correspondant aux déviations les moins grandes pour les avions. Il n'est pas illogique que les premières solutions trouvées puissent souvent compter parmi les plus courtes. Cependant, rien n'empêche, théoriquement, que dans certains cas de conflit, l'élagage n'entraîne une augmentation importante du retard. Seule une étude statistique sur des cas de conflits issus du trafic réel pourrait donner une mesure véritablement significative des conséquences de l'utilisation de l'élagage.

L'avantage de l'utilisation de l'élagage, par rapport à l'augmentation du pas de temps, demeure qu'il est possible qu'une solution existe pour un pas de temps de 10 secondes, et pas pour un pas de temps de 30 secondes. Dans ce cas, pour un pas de temps de 10 secondes, tout taux d'élagage mènera à une solution (pas forcément optimale), tandis que, même sans élagage, on ne trouvera aucune solution pour un pas de temps de 30 secondes.

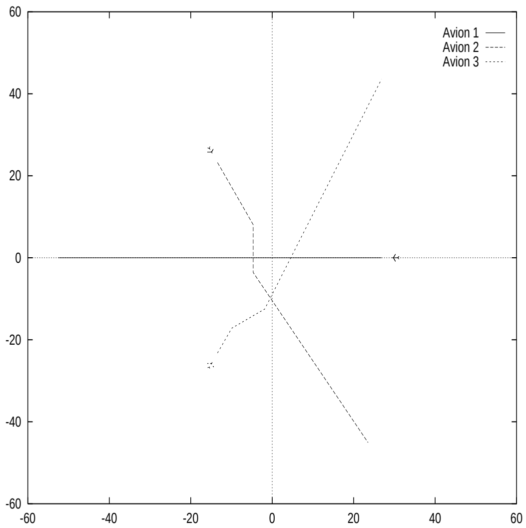



FIG. 3.3: *Approche globale, 3 avions,* $l_t = 10\ s$, $t_e = 0,1$.

La figure 3.3 montre la meilleure solution correspondant aux tests dont les résultats sont présentés dans le tableau 3.1, c'est-à-dire la solution obtenue avec un pas de temps de 10 secondes et un taux d'élagage de 0,1, menant à une somme des allongements de $4,4$ NM, ce qui correspond à un retard moyen de 12 secondes par avion (en fait seuls les avions 2 et 3 sont déviés, subissant chacun un retard de 18 secondes. La solution qui consisterait à dévier les 3 avions dans le même sens, *en rond point*, comme par exemple la solution obtenue par les méthodes d'intervalles (voir section 2.9.2) mènerait peut-être à des retards plus faibles. Ce n'est pas sûr, compte tenu des contraintes imposées sur les trajectoires par la discrétisation. De plus, cette solution a été obtenue avec élagage, et ne donne pas forcément les meilleures trajectoires d'évitement, même parmi les trajectoires discrétisables. Cependant, les retards obtenus restent faibles.

Sur le cas test à 4 avions, l'algorithme n'a trouvé aucune solution, sans élagage, pour des pas de temps allant jusqu'à 90 secondes. Les seuls résultats obtenus l'ont été pour des valeurs encore trop importantes du pas de temps et du taux d'élagage, comme $l_t = 45\ s$ et $t_e = 0,2$. On peut donc considérer que l'approche globale telle qu'elle est décrite ici n'est pas applicable à des conflits impliquant plus de 3 avions.

### 3.7.2 Approche séquentielle

Dans un premier temps, nous avons appliqué l'algorithme d'optimisation séquentielle décrit dans la section précédente à des cas test de conflits similaires à ceux qui ont été utilisés dans la section précédente pour tester l'approche globale (les positions initiales sont réparties sur un cercle de rayon 28 NM, les trajectoires originellement prévues pour les avions convergent vers le centre de ce cercle, les positions finales des avions sont à 80 NM audelà du centre du cercle, et les avions volent dans un plan horizontal, à la vitesse de 420 kt).

Comme c'était le cas avec les méthodes d'intervalles (section 2.9.3), les résultats obtenus dépendent très fortement de l'ordre de priorité entre les avions.

Considérons le conflit présenté sur les figure 3.4 et 3.5. La différence par rapport aux cas tests décrits précédemment est que maintenant les avions ne sont plus équirépartis sur ce cercle de 28 NM de rayon, mais ont des trajectoires originellement prévues faisant entre elles des angles plus faibles.

Ces trois schémas représentent les trajectoires fournies par l'algorithme d'optimisation séquentiel pour exactement la même configuration de conflit à 3 avions, mais pour 3 ordres de priorité différents.

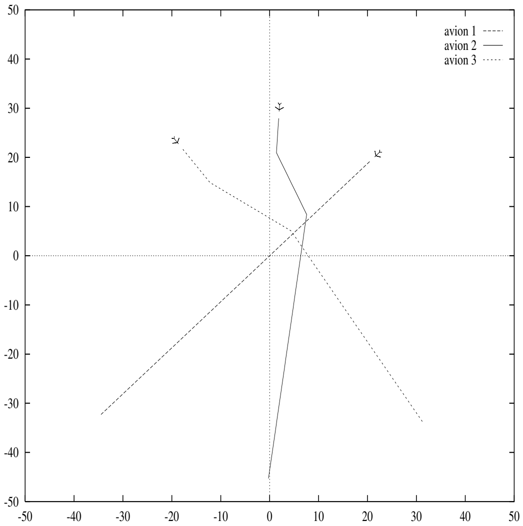
Les ordres selon lesquels les avions apparaissent sur les légendes des figures 3.4 et 3.5 donnent les ordres de priorités qui ont été utilisés.

Dans le cas illustré à droite de la figure 3.4, le deuxième avion, en passant derrière le premier avion, libère de l'espace au troisième avion qui peut alors passer derrière lui sans être trop dévié.

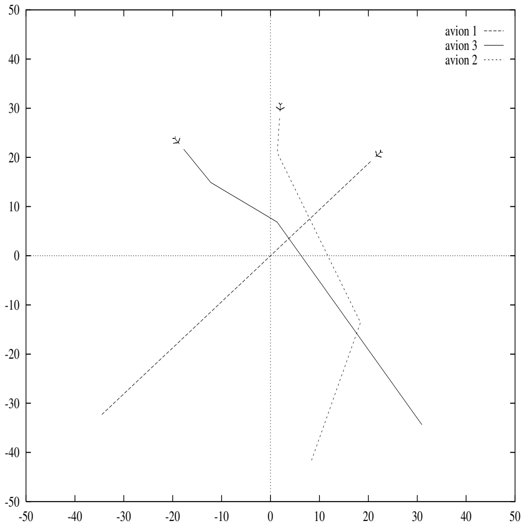
Dans le cas illustré à gauche, au contraire, le troisième avion dans l'ordre de priorité est l'avion dont la position initiale est située entre les positions initiales des deux premiers avions. Le deuxième avion, cette fois, en passant derrière le premier, gêne le troisième, qui, pour passer à son tour derrière lui doit subir une déviation beaucoup plus importante.

La figure 3.5 montre ce qui se passe, toujours pour le même conflit que sur la figure 3.4, si les avions sont classés selon un ordre encore différent : cette fois c'est l'avion dont la position initiale est entre les positions initiales des deux autres qui est prioritaire.

Dans ce cas, l'algorithme ne trouve aucun jeu de trajectoires admissible : les trajectoires des deux premiers avions sont telles qu'elles empêchent totalement le troisième avion (à droite sur la figure) de rejoindre sa position finale sans entrer en conflit avec aucun des deux autres et en utilisant une trajectoire d'évitement normalisée.




(a) Ordre de priorité $(1, 2, 3)$




(b) Ordre de priorité $(1, 3, 2)$

FIG. 3.4: *Deux résolutions de différentes qualités d'un même conflit*

La modélisation utilisée ici peut être utilisée dans le cadre de génération de trajectoires d'évitement, de manière globale, au moyen d'algorithmes génétiques[4], décrites par Nicolas Durand.

Ceci nous a permis de comparer le meilleur résultat obtenu pour un cas test de conflit à 8 avions les résultats obtenus avec les algorithmes génétiques.

Il faut préciser que la méthode d'optimisation que nous utilisons ici comme référence, qui utilise un algorithme génétique, recherche un optimum global, mais n'en garantit pas l'obtention systématique (en terme d'allongement de trajectoire). Cependant, cette méthode a montré son efficacité, même appliquée à la résolution de cas de conflits issus du trafic aérien réel [Dur96].

Les deux algorithmes que nous comparons utilisent la même modélisation des trajectoires d'évitement, c'est-à-dire que :

- le temps est discrétisé et le pas de temps est de 5 secondes ;
- à chaque avion $a_i$ correspond un temps $t_0^i$ auquel l'avion $a_i$ s'écarte de sa trajectoire originelle, en opérant un changement de cap de 10, 20 ou 30 degrés, à gauche ou à droite ;
- à chaque avion $a_i$ correspond un temps $t_1^i \geq t_0^i$, auquel l'avion $a_i$ prend un cap de retour direct vers sa position finale.

[4]Les algorithmes génétiques seront présentés dans chapitre 4 et utilisés dans les chapitres 5 et 6.

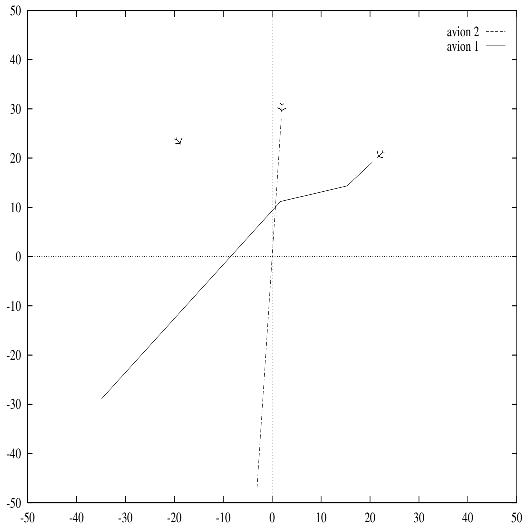



FIG. 3.5: *Le même conflit à 3 avions, non résolu, ordre de priorité* $(2,1,3)$

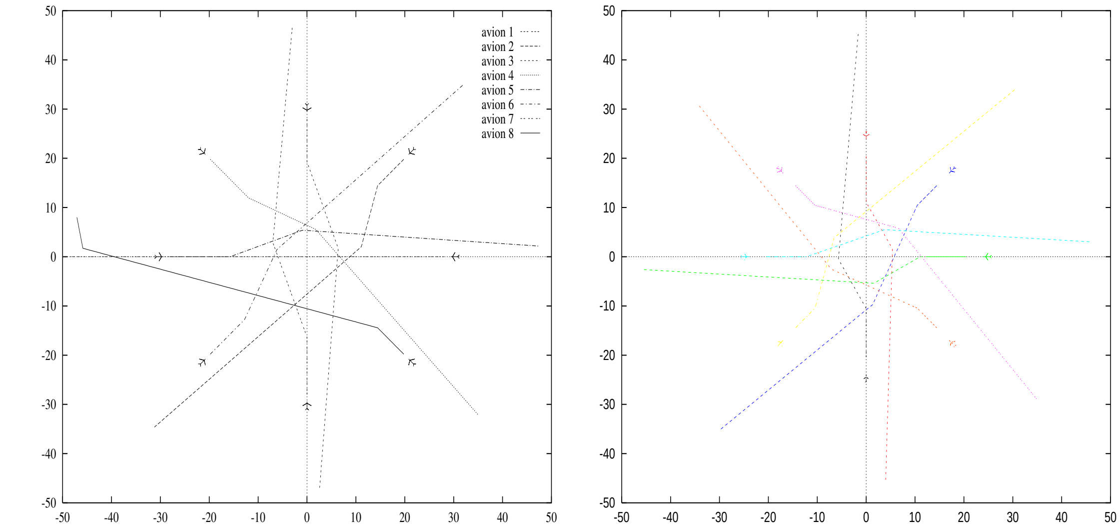
La figure 3.6 montre les trajectoires proposées pour les 8 avions par chacun des deux algorithmes : l'algorithme séquentiel à gauche, et à droite l'algorithme génétique globale. On voit, à gauche, que l'algorithme séquentiel impose à l'avion numéro 8 une déviation beaucoup plus importante que celles qui sont subies par les autres avions (les avions sont numérotés selon l'ordre de priorité, c'est-à-dire selon l'ordre dans lequel sont générées leurs trajectoires). Sur la figure de droite, on remarque qu'au contraire tous les avions subissent des déviations à peu près équivalentes.

Ces résultats sont présentés plus précisément sur le tableau 3.2. Comme on le voit, en terme de retard moyen, la méthode séquentielle est moins bonne que la méthode globale, ce qui n'a rien de surprenant, mais elle mène, sur cet exemple précis, à un retard moyen d'un ordre de grandeur comparable.

| Algorithme utilisé | $A^*$ séquentiel | GA global |
|---|---|---|
| Avion 1 | 0 | 21 |
| Avion 2 | 17 | 19 |
| Avion 3 | 12 | 22 |
| Avion 4 | 9 | 26 |
| Avion 5 | 10 | 20 |
| Avion 6 | 10 | 21 |
| Avion 7 | 13 | 21 |
| Avion 8 | 153 | 17 |
| Moyenne | 28 | 20,9 |

TAB. 3.2: *Retard en secondes de chaque avion selon l'algorithme utilisé*

Ce résultat montre encore une fois que l'approche séquentielle peut mener à de bons résultats en terme de retards, *pourvu que l'ordre de priorité entre les avions soit bien choisi*. Avec d'autres




(a) $A^*$, approche séquentielle

(b) Algorithme génétique, optimisation globale

FIG. 3.6: *Trajectoires d'évitement pour un conflit impliquant 8 avions.*

ordres de priorité, l'algorithme séquentiel n'aurait pas trouvé de solution. Ce problème est en fait identique au problème rencontré avec les méthodes d'intervalles, et étudiées dans la section 2.9.3. Le choix de cet ordre de priorité demeure donc un point crucial dans l'emploi de ce type de méthode séquentielle.

## 3.8 Comparaison avec les méthodes d'intervalles

Comme la méthode utilisant l'analyse d'intervalles présentée dans le chapitre précédent, l'algorithme $A^*$ n'est pas adapté à la recherche de l'optimum global dans le cas d'un conflit à plus de 2 avions. Le facteur limitant, dans le cas de l'algorithme $A^*$, n'est plus le temps de calcul, mais la taille de la mémoire nécessitée par le programme.

Dans ce cadre, l'élagage utilisé avec l'algorithme $A^*$ s'est montré bien moins efficace que dans le cas de l'utilisation d'un algorithme basé sur les méthodes d'intervalles.

Pour les méthodes d'intervalles, sur les cas tests considérés, l'utilisation de l'élagage a permis de traiter des cas de conflits allant jusqu'à 7 avions (section 2.9.2). Avec l'algorithme $A^*$, il ne permet pas de résoudre un conflit à 4 avions sans atteindre des valeurs d'élagage prohibitives (section 3.7.1).

De plus, le coût maximal théorique de l'élagage est beaucoup plus important dans le cas de l'algorithme $A^*$ : un taux d'élagage de 0,1 à une augmentation du retard de 90 secondes par avion (section 3.7.1). De telles coupures d'élagage n'étaient pas nécessaires pour obtenir de bons résultats rapidement sur les conflits à 3 et 4 avions, en utilisant les méthodes d'intervalles.

Dans les deux cas ($A^*$ et intervalles), le fait de ne pas obtenir de solution signifie qu'il n'existe pas de jeux de trajectoires tels que les distances entre avions restent toujours supérieures à une norme de séparation majorée (voir section 2.8.6 pour les intervalles, section 3.6.3 pour $A^*$). Pour que cette propriété soit intéressante, il faut que cette majoration de la norme soit faible. Dans les deux cas, cette majoration est proportionnelle à la largeur des pas de temps utilisés.

Cependant, dans le cas des intervalles la discrétisation du temps est seulement utilisée pour calculer les positions des avions et donc les séparations. Le temps de calcul est proportionnel au nombre de ces pas de temps (voir section 2.8.5). Dans le cas de l'algorithme $A^*$, c'est le nombre de nœuds des chemins de l'arbre de recherche qui est proportionnel au nombre des pas de temps, puisque si un nœud $u$ correspond à un pas de temps $j_u$, un fils $v$ de ce nœud $u$ correspond à des positions possibles pour les avions au pas de temps $j_u + 1$. Or, dans le cas de l'approche globale, l'heuristique utilisée étant monotone, la complexité de l'algorithme est exponentielle, en $K^M$, où $K$ est le facteur de branchement de l'arbre et $M$ le nombre d'arcs entre l'état initial et un état final. On a vu d'ailleurs dans la section 3.7.1 qu'en effet le fait de réduire les pas de temps obligeait à augmenter fortement le taux d'élagage pour obtenir un résultat.

Les résultats obtenus avec les deux méthodes ($A^*$ et intervalles) dans le cadre de l'approche séquentielle sont comparables. En fait, les principaux inconvénients de l'approche séquentielle (pas de recherche d'optimalité globale, très forte dépendance des solutions obtenues, en terme de qualité voire d'existence, par rapport à l'ordre de priorité choisi) découlent du fait de générer les trajectoires des avions séquentiellement, et pas de l'algorithme utilisé pour les générer.

Au vu de ce qui précède, on pourrait conclure à la supériorité des méthodes d'intervalles, équivalentes dans le cadre de l'approche séquentielle, plus performantes dans le cadre de l'approche globale.

Cependant, si l'on se place dans la perspective d'utiliser ces méthodes de manière plus réaliste, dans le cadre du trafic réel, l'algorithme $A^*$ nous semble sans conteste plus indiqué que les méthodes d'intervalles.

En effet, les méthodes présentées dans le chapitre 2 reposaient sur le fait de pouvoir calculer, pour des intervalles de manœuvres donnés, l'ensemble des positions possibles pour les avions à un instant $t$ donné. Ce calcul n'est possible que pour certaines modélisations des trajectoires d'évitement et, surtout, nécessite de supposer que les vitesses des avions sont constantes, ce qui oblige à ne considérer que des avions volant dans un plan horizontal, et n'utiliser que des manœuvres d'évitement maintenant les avions dans ce plan horizontal[5].

La méthode présentée dans ce chapitre ici, utilisant l'algorithme $A^*$ s'applique, elle, directement, dès que les trajectoires d'évitement choisies peuvent être traduites en terme de choix successifs (toujours en supposant que ces choix ne peuvent avoir lieu qu'à des pas de temps successifs). La nature de ces choix (changement de cap ou de vitesse, mise en palier, en montée ou en descente) n'importe pas, tant que le nombre de ces choix reste raisonnable (le facteur de branchement de l'arbre obtenu dépend de ce nombre de choix).

Ainsi, dans le cadre d'un système d'évitement embarqué basé sur la génération des trajec-

[5]Quand nous disons que ce calcul des positions possibles des avions n'est réalisable qu'à ces conditions, cela s'applique, bien entendu, à ce calcul tel qu'il est effectué dans le chapitre 2. Nous n'affirmons pas que l'utilisation de méthodes d'intervalles empêche de faire varier les vitesses des avions. Nous disons que les méthodes du chapitre 2 ne peuvent pas s'appliquer *directement* si les vitesses varient. Leur adaptation pour qu'elles puissent s'appliquer à des avions évolutifs serait sans doute possible, mais en tout cas n'est pas immédiate.

toires des avions de manière séquentielle, l'utilisation de l'algorithme $A^*$ nous semblerait plus appropriée que celle de méthodes d'intervalles telles que celles présentées dans le chapitre 2.

## 3.9 Application à des cas de conflit issus du trafic réel

Dans le cadre de son stage de DEA au sein de l'équipe du Laboratoire d'Optimisation Globale de l'ENAC et du CENA, Géraud Granger a adapté l'approche séquentielle de la méthode présentée dans ce chapitre à un système d'évitement embarqué utilisable sur des cas de conflits issus du trafic réel, et a testé la méthode obtenue à l'aide du simulateur CATS [ABDM96]. Un très bref résumé de ce travail est donné dans cette section. Pour plus de détails, on se reportera à [Gra98]

La première version de ce système, telle qu'elle est exposée dans [Gra98] ne prévoit que des manœuvres d'évitement dans le plan horizontal, et ne prend en compte que la norme de séparation horizontale.

Le système est basé sur le fait que chaque avion a connaissance des avions se trouvant à l'intérieur d'un cercle dont il est le centre et dont le rayon est égal à un *rayon de détection*. Ce rayon de détection est fixé à 90 NM. La génération des trajectoires d'évitement se fait selon un ordre de priorité, obtenu par la méthode d'échange de *jetons* décrite dans la section 2.8.1.

La méthode présentée dans [Gra98] s'attache à résoudre les conflits susceptibles d'avoir lieu à l'intérieur d'un horizon temporel noté $F_a$. Les trajectoires d'évitement des avions sont calculées périodiquement, avec une période $\Delta$. Les valeurs numériques choisies par Géraud Granger sont $F_a = 5\ mn$ et $\Delta = 1\ mn$.

Les trajectoires d'évitement retenues sont les trajectoires par point tournant avec retour direct vers la destination décrites dans la section 3.2. Le changement de cap correspondant au premier virage de la trajectoire d'évitement peut prendre, comme dans la section 3.7.2, 3 valeurs différentes : 10, 20 ou 30 degrés.

Le critère retenu pour générer un ordre de priorité, au moyen de la méthode exposée dans la section 2.8.1, tient compte de l'étape de la trajectoire d'évitement par point tournant à laquelle se trouve chacun des avions. En effet, si un avion a déjà effectué son premier virage, et est en deuxième étape de l'évitement, il est moins *manœuvrable* qu'un avion qui est encore en première étape, en ce sens qu'il n'a plus le choix qu'entre deux actions (continuer de s'éloigner de sa trajectoire d'origine ou revenir vers elle), alors qu'un avion qui est en première étape de l'évitement à le choix entre 7 actions (rester sur sa trajectoire, ou tourner à droite ou a gauche, avec pour chaque direction de virage 3 valeurs différentes de changement de cap).

Les avions qui ont déjà effectué un premier virage sont prioritaires par rapport aux avions qui ne l'ont pas encore effectué, à l'intérieur de chacune de ces deux catégories, l'ordre de priorité est donné par les numéros CAUTRA des avions.

Une nouvelle trajectoire d'évitement est calculée pour chaque avion avec une périodicité $\Delta$, à partir de nouvelles données (nouvelles positions des autres avions, prise en compte de nouveaux avions, modifications de l'ordre de priorité). La génération successive des trajectoires des avions doit se faire à l'intérieur d'une de ces périodes. Durant l'intervalle de temps $\Delta$, les avions volent toujours. Le système ne peut donc changer la trajectoire prévue de chacun des avions pendant la première période de durée $\Delta$. Ces trajectoires correspondent donc à la prévision effectuée au tour précédent. Par contre le système reste libre de modifier toutes les trajectoires entre $\Delta$ et $F_a$, sauf, bien entendu, celles des avions qui sont déjà en train de revenir sur leurs trajectoires d'origines. En effet, cette partie de la manœuvre est complètement déterminée par le début de la manœuvre.

Au début de chacune de ces période de durée $\Delta$, l'avion opère une détection de conflits, il prévoit une nouvelle trajectoire pour les $F_a$ prochaines minutes (voir figure 3.7). Lors de la recherche d'une nouvelle trajectoire, celle-ci ne peut pas être modifiée pendant une première période de durée $\Delta$. Si la modification intervient entre $\Delta$ et $2\,\Delta$, elle est nécessairement prise en compte, car à la prochaine itération, elle sera comprise dans la partie non modifiable de la trajectoire. Si la modification intervient après $2\,\Delta$, elle pourra être remise en cause à l'étape suivante.

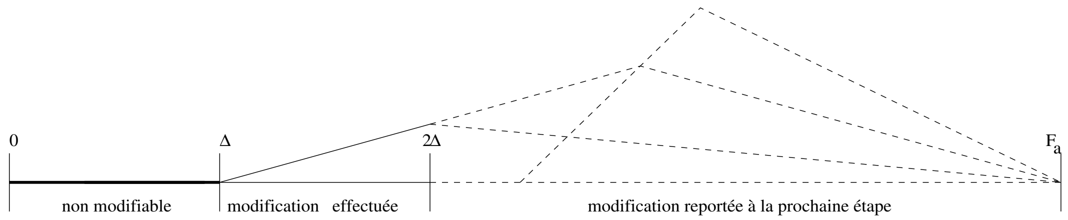



FIG. 3.7: *Prévision pour les* $5$ *prochaines minutes*

Ce système a été testé grâce au simulateur de trafic CATS [ABDM96], sur une journée[6] de trafic dans l'espace aérien français.

Un tel système est destiné à être utilisé dans l'espace aérien supérieur, où le trafic est moins dense, c'est-à-dire aux zones qui seront réservées au free flight (voir chapitre 1.3). Dans le cadre de la simulation effectuée par Géraud Granger, les avions ne sont pris en compte qu'au-dessus du niveau de vol 320[7]. Les plans de vol de certains avions sont légèrement modifiés par le simulateur de manière à ce que les avions arrivant dans cette zone soient séparés pendant une durée égale à $\Delta$ (cela est rendu nécessaire par le fait que leurs trajectoires ne peuvent être modifiées qu'au bout de cette durée). Dans une application d'un tel système, il serait nécessaire qu'une séparation de cet ordre soit assurée à l'interface entre les zones contrôlées et la zone *free flight*.

Il y a eu 6 381 vols dans l'espace aérien français sur la journée considérée, dont 2 763 au-dessus du niveau 320. Ces 2 763 vols, en l'absence de manœuvres d'évitement, génèrent 641 conflits.

Il ne reste plus, après application de la méthode décrite ici, que 4 conflits, impliquant en tout 9 avions. 367 avions ont été manœuvrés, le retard moyen sur les avions manœuvrés est de $27,4\ s$. Le retard maximum est de $402\ s$.

On constate qu'une simple manœuvre dans le plan vertical (changement temporaire de niveau de vol) permettrait de résoudre les 4 conflits restants.

On constate, de même que dans la section3.7.2, l'influence de l'ordre de priorité choisi. En prenant, à l'intérieur de chacune des deux catégories d'avions distinguées (ceux qui ont déjà effectué un premier virage et ceux qui n'en ont pas effectué) l'ordre inverse de celui choisi précédemment (les avions ayant déjà manœuvré restant prioritaires), on obtient des résultats comparables, mais les conflits qui n'ont pas été résolus ne sont pas les mêmes.

[6]Le 21 juin 1996, un vendredi, jour d'intense trafic.
[7]32 000 pieds.

## 3.10 Conclusion

L'algorithme $A^*$ est bien adapté à la génération de trajectoires. Étant bien moins dépendant de la modélisation utilisée que les méthodes d'intervalles, il est plus adapté à une utilisation effective dans le cadre d'un système d'évitement automatisé.

Il ne peut cependant pas être utilisé pour résoudre, avec recherche de l'optimum global, des conflits impliquant plus de deux avions.

Il pourrait être utilisé dans le cadre d'un système d'évitement distribué. Un tel système devra obligatoirement fonctionner selon une approche *séquentielle*, dans le cadre de laquelle les trajectoires des avions sont générées successivement, selon un *ordre de priorité*, les trajectoires déjà générées imposant des contraintes aux trajectoires qui restent à générer. En effet, la méthode présentée dans ce chapitre nécessite que la trajectoire d'évitement d'un avion ait été générée jusqu'à sa destination avant que celui-ci ne s'y engage, tout comme celle présentée dans le chapitre 2.

On trouvera dans [Gra98] une adaptation d'une telle approche séquentielle à une utilisation plus réaliste de l'algorithme $A^*$ (voir section 3.9). Les résultats obtenus sur une journée de trafic, grâce au simulateur CATS, sont suffisamment encourageant pour que cet axe de recherche soit poursuivi. Cependant, le principal problème que nous avons rencontré ici avec l'approche séquentielle se pose toujours (bien que sur un nombre très limité de cas) : pour certains conflits, et pour un ordre de priorité donné, il est possible qu'on ne puisse pas générer de trajectoire d'évitement pour tous les avions.

Ce problème est la manifestation la plus grave du défaut majeur de l'approche séquentielle par rapport à une approche globale : il n'y a pas de recherche de l'optimum global, pour ce qui est de l'ensemble des trajectoires d'évitement des $n$ avions impliquées dans le conflit.

La non optimalité des solutions peut n'être pas vue comme un problème majeur, mais il est nécessaire que des solutions acceptables puissent être proposées. Un système d'évitement utilisant une approche séquentielle devra obligatoirement proposer une solution à tous les conflits pour qu'on puisse envisager d'en équiper les avions.

Ce type d'approche séquentielle pose, si on veut l'utiliser en pratique, d'autres problèmes qu'il faudra résoudre si l'on veut rendre robustes des systèmes l'utilisant. Il nécessite la diffusion par les avions d'informations les concernant (notamment les informations permettant d'établir l'ordre de priorité, et la description trajectoires d'évitement, une fois qu'elles sont fixées) ; si un problème survient dans ces communication, à l'émission et à la réception, le fonctionnement du système risque d'être remis en cause.

La méthode présentée dans ce chapitre peut ainsi s'intégrer dans un système d'évitement embarqué. La conception d'un tel système pose cependant de très nombreux autres problèmes.

# Chapitre 4

# Les Algorithmes Génétiques

## 4.1 Présentation générale

### 4.1.1 Principe

Les Algorithmes Génétiques forment une branche d'une famille d'algorithmes d'optimisation que l'on peut appeler les *Algorithmes Évolutionnaires*. Cette famille comprend principalement, en plus des Algorithmes Génétiques, les *Stratégies d'évolution* (en anglais *Evolution Strategies*, souvent désignées par ES), la *Programmation Evolutionnaire* (en anglais *Evolutionnary Programming*, EP) et la *Programmation Génétique* (en anglais *Genetic Programming*, EP). Ces trois derniers types d'algorithmes seront très brièvement présentés dans la section 4.7. Depuis quelques années, les frontières entre ces différentes branches des Algorithmes Évolutionnaires se sont faites moins étanches, des opérateurs développés dans le cadre des uns sont utilisés avec les autres. Ainsi la mutation évolutive (section 4.2.2), d'abord introduite pour les stratégies d'évolution, est parfois utilisée dans le cadre des algorithmes génétiques. Nous allons présenter dans ce chapitre les Algorithmes Génétiques plus en détail, car nous les utiliserons dans le cadre des méthodes présentées dans les chapitres suivants (chapitres 5 et 6).

Le principe de base des algorithmes génétiques s'inspire, très schématiquement, de l'évolution naturelle. Le vocabulaire employé révèle clairement cette source d'inspiration : on parle de *population*, de *générations*, de *sélection*, de *reproduction*, de *croisement*, de *mutation*, de *chromosomes*.

Les origines de ces algorithmes remontent à 1962, aux travaux de Holland [Hol62]. Mais ce n'est qu'à partir de 1989 qu'ils ont connu une véritable popularité avec le livre de David E. Goldberg [Gol89].

Nous décrivons dans cette section le principe général d'un algorithme génétique employé dans le cadre de l'optimisation (minimisation ou maximisation) d'une fonction, notée $f_{opt}$, d'un espace d'états noté $E_{opt}$ dans $\mathbb{R}$. Les différents points de cette description rapide seront détaillés dans les sections suivantes.

Bien entendu, les caractéristiques d'un algorithme génétique peuvent être différentes de celles qui sont présentées ici. Nous présentons les caractéristiques communes aux algorithmes génétiques les plus fréquemment utilisés. Un panorama plus riche sera trouvé, par exemple, dans [Mic92].

L'algorithme génétique utilise une population. Les éléments de cette population, appelés *individus*, codent des points de l'ensemble $E_{opt}$. Cette population est générée aléatoirement au début de l'algorithme. On définit une fonction d'adaptation, notée $f_a$, souvent appelée *fitness*[1].

À partir de cette population initiale, et en utilisant cette fonction d'adaptation, L'algorithme génétique suit les étapes suivantes, illustrées par la figure 4.1 :

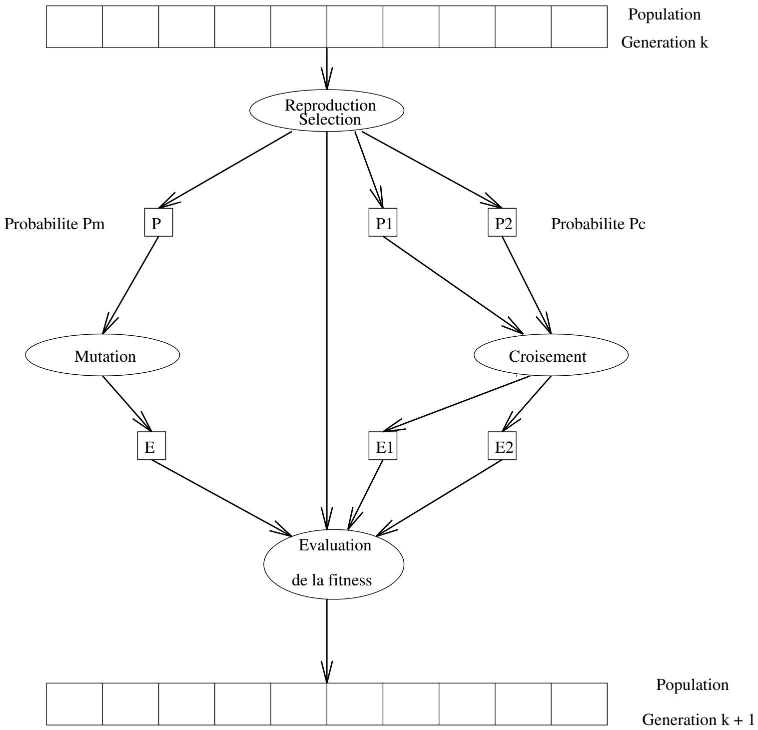



FIG. 4.1: *Principe général des algorithmes génétiques*

1. On génère d'abord les *éléments* de la *population* initiale. Ces éléments sont appelés *individus*. Ils codent des points de $E_{opt}$. La partie d'un individu qui porte le codage d'un point de l'espace d'états $E_{opt}$ est souvent appelée *chromosome*. Ce codage est aussi désigné par le terme de *génotype*, tandis que le point de $E_{opt}$ qu'il code est désigné par le terme de *phénotype*.
2. On calcule la fonction d'adaptation $f_a(e)$ pour chacun des éléments $e$ de la population. La manière dont est calculée cette fonction dépend bien entendu du problème.
3. On opère la reproduction des éléments de la population, les éléments les plus adaptés se reproduisant mieux que les éléments moins adaptés. Le produit de cette étape de reproduction est une nouvelle population, contenant le même nombre d'éléments que la population de départ.
4. On applique deux opérateurs dont l'effet est de diversifier la population : l'opérateur de *croisement* et l'opérateur de *mutation*.

[1]Les algorithmes génétiques ayant été surtout développés aux États Unis, on emploie souvent, même en français, des mots anglais pour désigner les notions qu'ils mettent en œuvre. Nous emploierons de préférence les termes français, mais conserverons leurs équivalents anglais quand ceux-ci sont imposés par l'usage

**Croisement :** on choisit aléatoirement, avec une probabilité $p_c$ fixée, des couples d'éléments de la population. L'opérateur de croisement, à partir de chaque couple de chromosomes, génère deux nouveaux chromosomes qui en sont issus et qui les remplacent dans la population. On appellera les éléments choisis dans la population, et à partir desquels on opère le croisement, les *parents*, tandis que les produits de ce croisement seront appelés les *enfants*. Les opérateurs de croisement les plus courants sont présentés dans la section la suivante.

**Mutation :** On choisit aléatoirement des éléments de la population, avec la probabilité $p_m$. Chacun de ces éléments est remplacé dans la population par un nouvel élément, obtenu à partir de lui par une perturbation aléatoire.

5. On revient à la deuxième étape, et on parcourt, à chaque itération, les 5 dernières étapes. On parle, par analogie avec les mécanismes de l'évolution naturelle dont s'inspirent les algorithmes génétiques, de *génération*. Une nouvelle population apparaît à chaque itération de ces étapes, c'est-à-dire à chaque génération. Cette population est issue, à travers la reproduction, le croisement et la mutation, de la population présente à la génération précédente. Certains éléments de la population peuvent demeurer inchangés d'une génération à l'autre s'ils n'ont subi ni croisement ni mutation. Le nombre d'éléments de la population reste constant à travers les générations.

Le plus souvent, il n'est pas possible de comparer le meilleur élément obtenu à l'optimum cherché. Il faut alors choisir un critère d'arrêt *arbitraire*. Les critères d'arrêt les plus communément utilisés sont les suivants :

- Un nombre maximal de générations peut être fixé *a priori*.
- Le processus peut être arrêté quand la population s'est stabilisée, c'est-à-dire n'évolue plus, ou quand la valeur de la fonction d'adaptation du meilleur élément a cessé d'augmenter ou n'augmente plus suffisamment rapidement.

### 4.1.2 Codage des données

Longtemps, les algorithmes génétiques ont utilisé un codage binaire. Dans ce cas, quelle que soit leur nature, les éléments de l'ensemble $E_{opt}$, sur lequel est menée l'optimisation, sont codés par des chaînes de bits. L'intérêt principal du codage binaire est de permettre l'utilisation d'opérateurs de croisement et de mutation simples. Les premiers résultats de convergence obtenus pour les algorithmes génétiques étaient basés sur l'utilisation du codage binaire (voir section 4.6).

Ce type de codage présente cependant des inconvénients :

- deux éléments voisins en terme de distance de Hamming (nombre de bits différents) ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un codage de Gray (un codage tel que deux entiers consécutifs ne diffèrent que d'un bit), ou d'autres codages adaptés au problème. Cependant, un autre problème se pose alors, qui est que le choix du codage influe sur le comportement des opérateurs.
- Pour des problèmes d'optimisation dans des espaces de grande dimension, le codage binaire peut rapidement devenir mauvais. Généralement, chaque variable est représentée par une partie de la chaîne de bits et la structure du problème n'est pas bien reflétée. L'ordre des

variables ayant une importance dans la structure du chromosome alors qu'il n'en a pas forcément dans la structure du problème.

L'utilisation du codage réel (les chromosomes sont des vecteurs de réels et non plus des chaînes de bits), pour des problèmes dans lesquels les données prennent des valeurs réelles s'est donc répandue [Gol91, Wri91]. L'avantage est ainsi de conserver les variables du problème dans le codage de l'élément de population sans passer par le codage binaire intermédiaire. La structure du problème est alors plus facilement conservée dans le codage.

Le codage binaire reste très utilisé, surtout quand il est adapté à la structure des données. Nous l'avions utilisé, par exemple, pour coder des ordres de passages entre avions, dans le cadre de l'utilisation d'algorithmes génétiques pour traiter les problèmes combinatoires générés par les différents ordres de passages possibles entre avions lors de la résolution de conflit [MDA95, Méd94]. L'algorithme génétique permet une recherche dans l'espace des combinaisons possibles de direction de déviation des avions et d'ordre de passage aux points de croisement, un algorithme d'optimisation locale étant utilisé sur la combinaison correspondant à chaque individu de la population.

### 4.1.3 Initialisation de la population et gestion des contraintes

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état. Si par contre des informations *a priori* sur le problème sont disponibles, il paraît bien évidemment naturel de générer les individus dans un sous-domaine particulier afin d'accélérer la convergence. Si le problème d'optimisation à résoudre comporte des contraintes, on veille en général à ce que les individus ainsi générés respectent ces contraintes [MJ91].

Si l'expression des contraintes du problème d'optimisation le permet, on peut ne générer, à travers les opérateurs de croisement et de mutation, que des individus respectant ces contraintes.

Cependant, une telle gestion directe des contraintes est souvent impossible, et le fait qu'un individu les respecte, ou non, est inclus dans le critère à optimiser, c'est-à-dire dans la fonction d'adaptation, sous forme de pénalités.

Un élément de population qui viole une contrainte se verra attribuer une mauvaise fonction d'adaptation et aura une probabilité forte d'être éliminé par le processus de sélection.

Il peut d'ailleurs être intéressant de conserver, tout en les pénalisant, les éléments non admissibles car ils peuvent permettre de générer des éléments admissibles de bonne qualité. Pour de nombreux problèmes, l'optimum est atteint lorsque l'une au moins des contraintes de séparation est saturée, c'est-à-dire sur la frontière de l'espace admissible.

Gérer les contraintes en pénalisant la fonction d'adaptation est difficile, un *dosage* s'impose pour ne pas favoriser la recherche de solutions admissibles au détriment de la recherche de l'optimum ou inversement. Les pénalités traduisant le non respect des contraintes dépendront évidemment du problème.

Disposant d'une population d'individus non homogène, la diversité de la population doit être entretenue au cours des générations afin de parcourir le plus largement possible l'espace d'état.

C'est le rôle des opérateurs de croisement et de mutation. En effet, sans ces opérateurs, l'effet de la sélection serait d'uniformiser la population.

## 4.2 Renouvellement de la population

### 4.2.1 Opérateur de croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. L'analogie avec la génétique est claire.

Initialement, le croisement associé au codage par chaînes de bits est le croisement à découpage de chromosomes (slicing crossover). Pour effectuer ce type de croisement sur des chromosomes constitués de $M$ gènes, on tire aléatoirement une position dans chacun des parents. On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants $E_1$ et $E_2$. Ce type de croisement est représenté sur la figure 4.2.

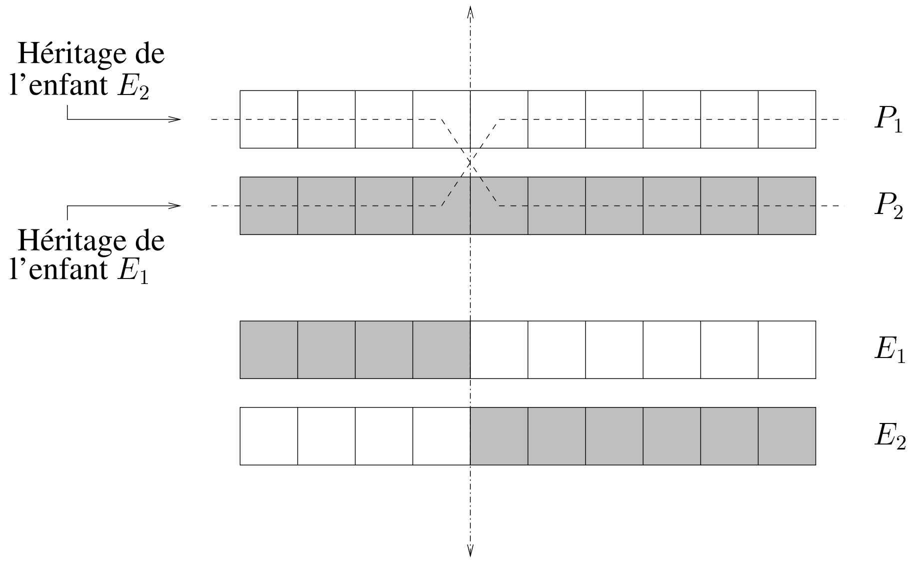



FIG. 4.2: *Un exemple de* slicing crossover

On peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaînes mais en 3, 4, etc. [BG91]. Un exemple d'un tel croisement est donné par la figure 4.3.

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets, avec un codage binaire, ou si différentes zones du chromosome ont des significations et des rôles différents.

Pour les problèmes continus avec codage réel, un croisement *barycentrique* est souvent utilisé : une position $i$ est choisie aléatoirement. Les gènes de chacun des deux parents correspondant à cette position sont respectivement notés $P_1(i)$ et $P_2(i)$. Ils définissent deux nouveaux gènes $C_1(i)$

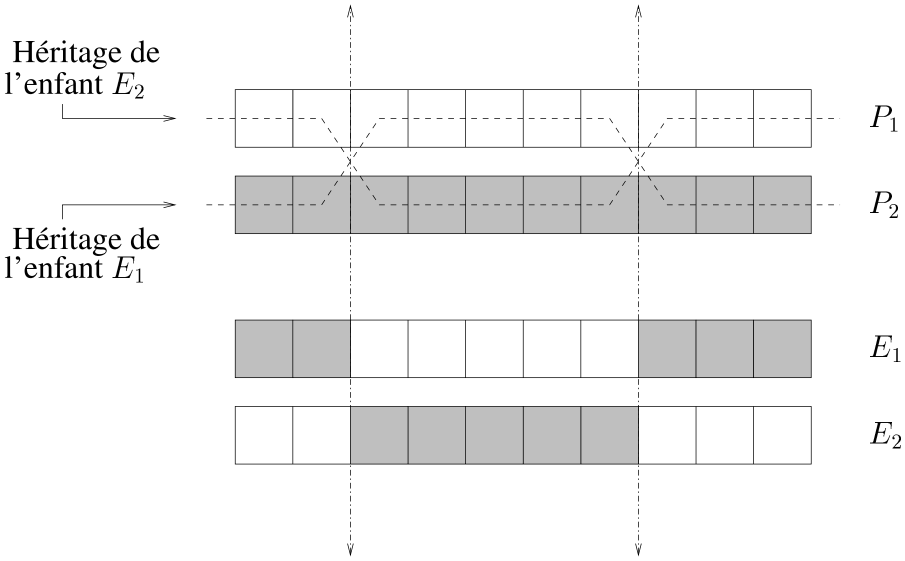



FIG. 4.3: *Un exemple de* slicing crossover *à deux points*

et $C_2(i)$ par combinaison linéaire :

$$\begin{cases} C_1(i) = \alpha P_1(i) + (1-\alpha)P_2(i) \\ C_2(i) = (1-\alpha)P_1(i) + \alpha P_2(i) \end{cases}$$

où $\alpha$ est un coefficient de pondération aléatoire adapté au domaine d'extension des gènes (il n'est pas nécessairement compris entre $0$ et $1$, il peut par exemple prendre des valeurs dans l'intervalle $[-0,5, 1,5]$ ce qui permet de générer des points entre, ou à l'extérieur des deux gènes considérés).

C'est ce type de croisement, avec coefficient compris entre $-0,5$ et $1,5$ que nous utilisons dans l'algorithme génétique au moyen duquel est réalisé l'apprentissage des réseaux de neurones, dans le cadre des techniques d'évitement automatiques décrites dans les chapitres 5 et 6.

D'autres types de croisement, plus spécifiques au problème traité peuvent bien entendu être utilisés dans le cadre d'un algorithme génétique. Ainsi, dans [Dur96], Nicolas Durand propose un croisement adapté à l'optimisation de fonctions partiellement séparables. L'efficacité du croisement dépend souvent de son adaptation au problème.

### 4.2.2 Opérateur de mutation

Le but de l'opérateur de mutation est d'apporter aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété signifie que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implantations fonctionnent de cette manière [FOW66]. Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur sur le plan théorique.

Pour les problèmes discrets, l'opérateur de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire. Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale.

Dans le cas d'un codage réel, utilisé pour un problème continu, on adapte ce principe en tirant aléatoirement un réel dans le chromosome (qui est en général un vecteur de réels), auquel on ajoute un bruit (généralement gaussien).

L'écart-type de ce bruit est difficile à choisir *a priori*. S'il est trop petit, les déplacements dans l'espace sont insuffisants en début de convergence, et l'algorithme peut rester bloqué dans un optimum local. Si le bruit est trop fort, l'algorithme pourra trouver une zone contenant l'optimum, mais sera incapable de converger localement.

Il est possible pour pallier ce problème d'avoir recours à la mutation adaptative. Le bruit est modifié au fil des générations. Pour ce faire, on calcule l'écart moyen, pour chacune des coordonnées, entre le meilleur élément à la génération $n$ et tous les autres éléments : ce nombre est alors utilisé comme écart-type du bruit gaussien sur la coordonnée en question à la génération $n+1$.

Ainsi, lorsque les éléments sont dispersés, ce qui est souvent le cas au début de l'évolution, l'amplitude du bruit gaussien sera plus grande. Quand les éléments sont moins dispersés, ce qui arrive souvent en fin d'évolution, cette amplitude sera plus faible. L'exploration de l'espace sera alors plus fine. Des exemples d'utilisation de cette technique seront trouvés dans [All96].

Il existe une autre variante de la mutation, que l'on peut appeler *mutation évolutive*. Les caractéristiques de la mutation, c'est-à-dire, par exemple, l'écart-type du bruit gaussien, sont codées par une partie du chromosome, et évoluent de génération en génération, à travers les étapes de l'algorithme génétique lui même. Cette forme de mutation reprend le principe de la *self adaptation*, introduite par Schwefel [Sch87] dans le cadre des stratégies d'évolution, principe selon lequel différentes caractéristique de l'algorithme évoluent de génération en génération. Des utilisations de cette méthode sont présentée dans [BS93]. Cette méthode s'est répandue depuis dans d'autres branches des algorithmes évolutionnaires dont les algorithmes génétiques.

### 4.2.3 Principaux types de sélection

À l'inverse d'autres techniques d'optimisation, les algorithmes génétiques ne requièrent pas d'hypothèse particulière sur la régularité de la fonction objectif. L'algorithme génétique n'utilise notamment pas ses dérivées successives, ce qui rend très vaste son domaine d'application. Aucune hypothèse sur la continuité n'est non plus requise. Néanmoins, dans la pratique, les algorithmes génétiques sont sensibles à la régularité des fonctions qu'ils optimisent.

Le fait de ne pas requérir d'hypothèse de dérivabilité ni même de continuité, permet de traiter des problèmes très complexes. La fonction à optimiser peut n'être pas calculable directement mais être le résultat d'une simulation. Cela sera justement le cas dans l'application présentée dans les chapitres 5 et 6.

La sélection permet d'identifier statistiquement les meilleurs individus d'une population, de leur assurer une *descendance*, c'est-à-dire de favoriser leur présence dans la population à la génération suivante, et d'éliminer les mauvais.

Parmi les plus fréquemment utilisées, on peut citer les deux types de sélection suivants :

- La sélection par *Roulette*, ou *Monte-Carlo* (en anglais *Roulette wheel selection*).
- La sélection par reste stochastique sans remplacement (*Stochastic remainder without replacement selection*).

Tous deux sont décrits dans [Gol89]. Le principe de *Roulette wheel selection* consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa fitness. Le principe de tirage aléatoire utilisé dans les roulettes de casinos est reproduit, mais avec une structure linéaire : à chaque individu est associé un segment de longueur proportionnelle à sa fitness. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre $0$ et $1$. On tire alors un nombre aléatoire (avec une distribution uniforme de probabilité) entre $0$ et $1$. L'élément sélectionné est celui qui correspond au segment dans lequel se trouve ce nombre. On reproduit ce tirage autant de fois qu'il y a d'éléments dans la population.

Ainsi les meilleurs éléments de la population, qui se voient attribuer des segments plus larges, plus souvent sélectionnés, seront présents en plus grand nombre dans la population.

Lorsque la taille de la population (nombre d'individus) est réduite, il est difficile d'obtenir en pratique l'espérance mathématique de sélection en raison du petit nombre de tirages effectués. Un biais de sélection plus ou moins fort existe suivant la taille de la population.

La technique du *Reste stochastique* vise à supprimer ce biais. Le principe en est le suivant :

- Pour chaque élément $i$, on calcule le rapport $r_i$ de sa fitness sur la moyenne des fitness.
- Soit $e(r_i)$ la partie entière de $r_i$, chaque élément est reproduit exactement $e(r_i)$ fois.
- La *roulette wheel selection* précédemment décrite est appliquée sur les individus affectés des fitness $r_i - e(r_i)$.

Ces deux processus de sélection sont très sensibles aux écarts de fitness et, dans certains cas, un très bon individu risque d'être reproduit trop souvent et peut même provoquer l'élimination complète de ses congénères. On obtient alors une population homogène contenant un seul type d'individu. La figure 4.4 montre un cas dans lequel cela peut arriver. On voit qu'il y a un élément proche de l'optimum local $M_2$, et dont la fonction d'adaptation est bien plus élevée que celle de tous les autres éléments. Cet élément risque d'être le seul représentant pour la génération suivante, alors que d'autres éléments, éventuellement plus proches de l'optimum global $M_1$ disparaîtront. Cet optimum ne pourra être ensuite atteint que très difficilement, au prix de nombreux essais successifs.

Ce problème peut être évité en ayant recours à la mise à l'échelle (*scaling*), présentée dans la section 4.3. En plus du *scaling*, on utilise souvent le partage (*sharing*), dont nous décrirons les variantes les principales variantes dans la section 4.4, pour conserver à la population une relative diversité, l'empêcher de se concentrer trop vite autour de ses meilleurs éléments, proches le plus souvent d'un optimum local. Une telle concentration prématurée de la population rendant très difficile d'atteindre ensuite un meilleur optimum local (rappelons que le but est d'atteindre, ou du moins d'approcher, l'optimum global).

Une autre manière de rendre la sélection moins radicale au début du déroulement de l'algorithme (on parle de *pression sélective*), et de permettre à la population de se diversifier d'abord, on pourra introduire des techniques issues du *recuit simulé* (voir section 4.5).

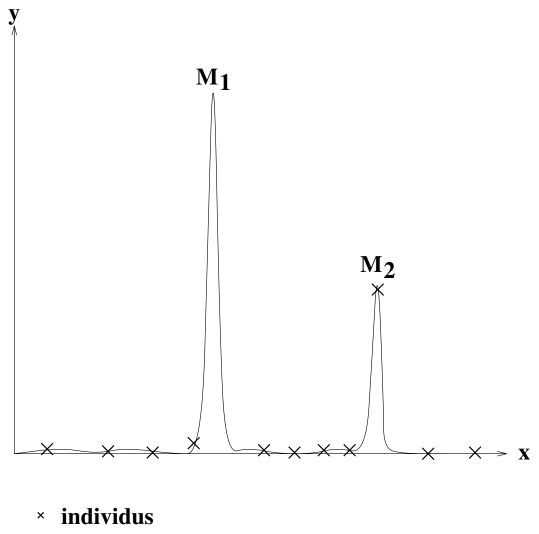



FIG. 4.4: *Exemple où les sélections classiques risquent de ne reproduire qu'un individu*

Enfin, on peut également introduire des recherches multi-objectifs, en utilisant la notion de dominance lors de la sélection. De telles sélections sont présentées dans [All96].

## 4.3 Scaling

Le *scaling* ou mise à l'échelle, modifie les fitness afin de réduire ou d'amplifier artificiellement les écarts entre les individus. Le processus de sélection n'opère plus sur la fitness réelle mais sur son image après scaling. Parmi les fonctions de scaling les plus répandues se trouvent le *ranking*, le scaling linéaire et le scaling exponentiel. Dans la suite de cette section on notera $f_r$ la fitness avant scaling[2] et $f_s$ la fitness modifiée par le scaling[3].

### 4.3.1 Ranking

Dans le cadre du ranking, l'évaluation des éléments n'est utilisée que pour les ordonner. La sélection, ensuite, ne dépendra que du rang des éléments dans cet ordre. C'est ce mode de mise à l'échelle que nous utilisons dans les méthodes présentées dans les chapitres 5 et 6. Après avoir ordonné les individus, l'individu de rang $i$ se voit attribuer une *fitness* égale à $(N_e - i)/N_e$, où $N_e$ est le nombre d'individus dans la population. On procède ensuite à une sélection par reste stochastique.

Ce type de ranking est aussi appelé *linear ranking*. Dans la suite de ce rapport (chapitres 5 et 6), pour plus de simplicité, on appellera *sélection par le rang* cette combinaison de *ranking* et de sélection par reste stochastique.

Le ranking est utile quand il est possible de savoir que tel individu répond mieux qu'un autre

[2] de l'anglais *raw fitness*

[3] scaled fitness

au problème, mais qu'on ne sait pas quantifier cette différence de qualité. Ce sera le cas dans nos applications, et c'est pourquoi nous utilisons cette technique (après avoir comparé, pour certaines des méthodes que nous présentons, avec l'utilisation directe de la sélection par reste stochastique, qui a donné de moins bons résultats).

Si l'on veut conserver l'information supplémentaire contenue dans les valeurs des fitnesses, par rapport à celle qui est donnée par les rangs (à savoir, par exemple, que la qualité d'un individu le distingue très nettement des autres), mais en en atténuant les effets pervers illustrés par la figure 4.4, on pourra utiliser un des deux types de scaling que nous allons décrire maintenant : le sacling linéaire ou le scaling exponentiel.

### 4.3.2 Scaling linéaire

Dans ce cas la fonction de scaling est définie de la façon suivante [Mic92] :

$$f_s = af_r + b$$

$$a = \frac{max' - min'}{max - min};\ b = \frac{min'.max - min.max'}{max - min}.$$

En règle générale, le coefficient $a$ est inférieur à 1, ce qui permet de réduire les écarts de fitness et donc de favoriser l'exploration de l'espace. Ce scaling est statique par rapport au numéro de génération et pénalise la fin de convergence lorsque l'on désire favoriser les modes dominants.

### 4.3.3 Scaling exponentiel

Il est défini de la façon suivante [Mic92] (voir figure 4.5) :

$$f_s = (f_r)^{k(n)}$$

où $n$ est la génération courante.

- Pour $k$ proche de zéro, on réduit fortement les écarts de fitness ; aucun individu n'est vraiment favorisé et l'algorithme génétique se comporte comme un algorithme de recherche aléatoire et permet d'explorer l'espace.
- Pour $k$ proche de $1$ : le scaling est inopérant.
- Pour $k > 1$ les écarts sont exagérés et seuls les bons individus sont sélectionnés ce qui produit l'émergence des modes.

Dans la pratique, on fait généralement varier $k$ des faibles valeurs vers les fortes valeurs au cours des générations. Pour cela on peut utiliser la formule suivante :

$$k = \left(\tan\left[\left(\frac{n}{N+1}\right)\frac{\pi}{2}\right]\right)^p$$

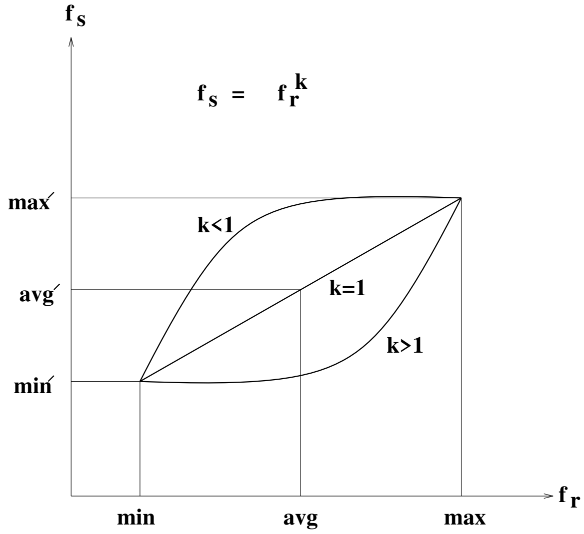



FIG. 4.5: *Fonction de scaling exponentielle*

$n$ étant la génération courante, $N$ le nombre total de générations prévues, $p$ un paramètre à choisir. L'évolution de $k$ en fonction de la génération $n$, pou $p = 0,1$ est donnée par la figure 4.6.

En début d'optimisation, on favorise l'exploration, presque aléatoire, de l'espace d'état. En fin d'optimisation, en recherche une sélection plus sévère.

## 4.4 Sharing

### 4.4.1 Principe

L'objectif du sharing (ou partage) est de répartir sur chaque sommet de la fonction à optimiser un nombre d'individus proportionnel à la fitness associée à ce sommet. La figure 4.7 présente deux exemples de répartitions de populations dans le cas d'une fonction à 5 sommets : le premier sans sharing, le second avec sharing.

De la même façon que le scaling, le sharing consiste à modifier la fitness utilisée par le processus de sélection. Pour éviter le rassemblement des individus autour d'un sommet dominant, le sharing pénalise les fitness en fonction du taux d'agrégation de la population dans le voisinage d'un individu. Il requiert l'introduction d'une notion de distance. Dans la pratique, il faut définir une distance indiquant la dissimilarité entre deux individus. Cette distance est alors utilisée pour calculer la nouvelle fitness de la façon suivante :

$$f'_i = \frac{f_i}{m'_i};\ m'_i = \sum_{j=1}^{N} S(d(x_i, x_j))$$

avec

$$S(d) = 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} \text{ si } d < \sigma_{share}$$

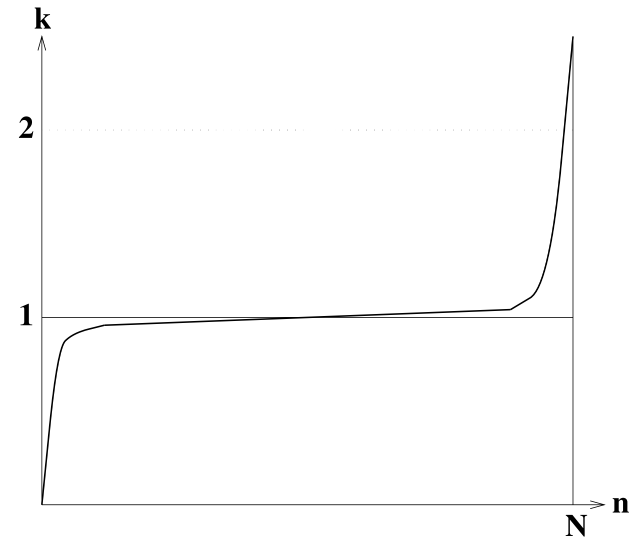



FIG. 4.6: *Allure de l'évolution de $k$ en fonction des générations*

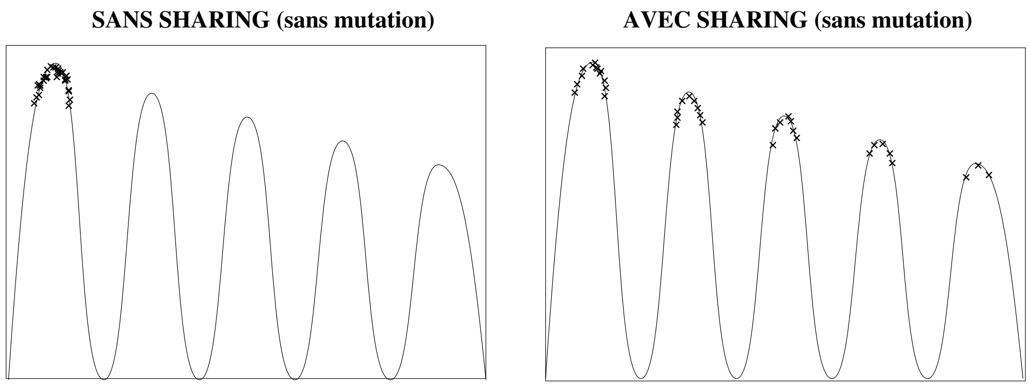



FIG. 4.7: *Objectif du sharing*

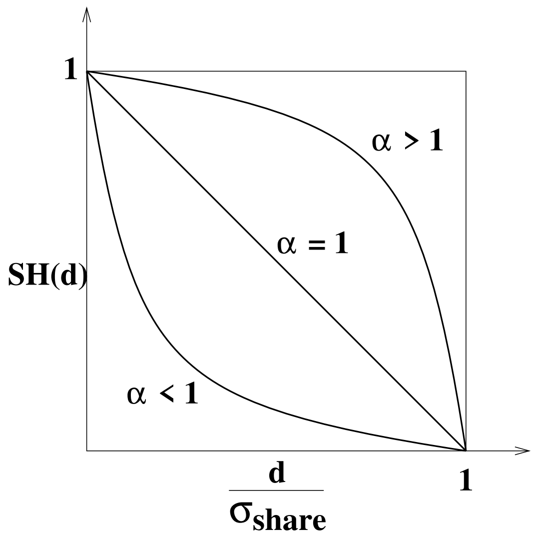



FIG. 4.8: *Allure de* $S(\frac{d}{\sigma_{share}})$

$$S(d) = 0 \text{ si } d > \sigma_{share}$$

Le paramètre $\sigma_{share}$ permet de délimiter le voisinage d'un point et dépend du problème traité. La figure 4.8 donne l'allure de $S(d)$ pour différentes valeurs de $\alpha$. Suivant la valeur donnée à $\alpha$ le sharing sera plus ou moins efficace. Ainsi pour $\alpha < 1$, on pénalise les groupes très agglomérés.

Dans la pratique ce type de sharing donne effectivement de bons résultats mais au prix de $N^2$ calculs de distances entre chromosomes à chaque génération pour une population de taille $N$. Or les algorithmes génétiques induisent une complexité en $N$ sans sharing et le fait de passer en $N^2$ peut être pénalisant, notamment pour $N$ grand.

Pour réduire ce nombre, on utilise un sharing *clusterisé*.

### 4.4.2 Sharing clusterisé

Pour effectuer ce type de sharing [YG93], on commence par identifier les différents "clusters" d'individus dans la population. Ce dernier utilise deux paramètres $d_{min}$ et $d_{max}$ respectivement pour fusionner des clusters ou en créer de nouveaux. Initialement, chaque individu de la population est considéré comme le centre d'un cluster. On applique alors successivement les deux principes suivants :

- si deux centres sont à une distance inférieure à $d_{min}$, on fusionne ces derniers dans un cluster unique dont le centre résultant est le barycentre des deux centres initiaux.
- un nouvel individu est agrégé à un cluster si sa distance au centre le plus proche est inférieure à $d_{max}$. Dans ce cas, on recalcule le centre du cluster global. Sinon, on crée un nouveau cluster contenant ce seul individu.

Ce principe de fusion-agrégation permet d'obtenir un nombre de clusters fluctuant avec la répartition des individus dans l'espace d'état. On applique ensuite le principe de sharing en modifiant les fitness de la façon suivante :

$$f_i' = \frac{f_i}{m_i'};\ m_i' = n_c\left(1 - \left(\frac{d_{ic}}{2d_{max}}\right)^{\alpha}\right)$$

avec

- $n_c$ : nombre d'individus contenus dans le cluster auquel appartient l'individu $i$.
- $\alpha$ : coefficient de sensibilité.
- $d_{ic}$ : distance entre l'individu $i$ et le centre du cluster $c$.

On montre que ce type de sharing induit une complexité en $O(N \log N)$[YG93] pour des résultats tout à fait comparables à ceux fournis par le sharing classique. Dans la pratique, on remarque que le réglage des coefficients $d_{min}$ et $d_{max}$ est assez délicat car l'efficacité de ces derniers dépend essentiellement de la connaissance *a priori* des distances inter-modes dans l'espace d'état, distances qu'il est très difficile d'estimer. La technique du *clustering adaptatif* (section 4.4.3) permet de calculer automatiquement ces quantités.

### 4.4.3 Clustering adaptatif

Nous avons vu dans la section 4.4.2 une technique de sharing clusterisé permettant de diminuer la complexité du sharing traditionnel. Cependant, comme nous l'avons souligné, les quantités $d_{min}$ et $d_{max}$ sont difficiles à calculer *a priori*. Nous présentons ici un algorithme permettant de calculer automatiquement ces quantités de façon adaptative pendant l'exécution de l'algorithme génétique.

Au départ on initialise le processus avec $d_{moy} = 0$ et $\Delta = 2.0$. Ensuite, à chaque génération, on utilise ce qui suit :

1. Calcul de $d_{min}$ et $d_{max}$ :
$$\begin{cases} d_{max} = \frac{d_{moy}}{\Delta} \\ d_{min} = \frac{d_{max}}{3} \end{cases}$$
2. Évaluation de la distance moyenne des individus par rapport aux centres des clusters :
$$d_{moy} = \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{N_c} d(E_i, C_j)\right)}{nN_c}$$
si l'on note $C_j$ le centre du groupe $j$ et $N_c$ le nombre total de clusters.
3. On compte ensuite le nombre de clusters dont le meilleur individu a une fitness supérieure à un certain pourcentage[4] de celle du meilleur élément de la population, on le note $N_{opt}$
4. On met à jour le paramètre $\Delta$ en utilisant la règle :
$$\begin{cases} \Delta = \Delta.1,05 \text{ si } \frac{N_{opt}}{N_c} > S_1 \text{ et } \Delta < 100 \\ \Delta = \Delta.0,95 \text{ si } \frac{N_{opt}}{N_c} < S_2 \text{ et } \Delta > 1 \\ \Delta \text{ reste inchangé dans les autres cas} \end{cases}$$

[4]Égal au taux de sharing.

Les valeurs $S_1$ et $S_2$ désignent deux seuils : si beaucoup de clusters sont suffisamment bons (c'est-à-dire quand leur proportion dépasse $S_1$) alors on diminue $d_{min}$ et $d_{max}$ pour accroître le nombre total de groupes et ainsi chercher d'autres optima. Au contraire, s'ils sont peu nombreux alors on va diminuer la quantité de clusters en augmentant les deux distances afin de rechercher des zones intéressantes de l'espace d'état.
Les valeurs utilisées habituellement pour les seuils sont $S_1 = 0,85$ et $S_2 = 0,75$.

Nous avons testé cette technique de sharing dans le cadre des premières applications des méthodes présentées dans les chapitres 5 et 6. Comme nous avons obtenu sur ces premiers essais de bons résultats avec elle, c'est elle que nous avons utilisé quand nous avons eu recours au sharing.

## 4.5 Algorithmes génétiques et recuit simulé

### 4.5.1 Introduction

Les algorithmes génétiques et le recuit simulé étant deux techniques d'optimisation stochastique travaillant sur les mêmes types de problèmes, il est logique de chercher à les associer afin de tirer parti de leurs avantages respectifs. Après plusieurs évaluations de ces deux techniques sur les mêmes problèmes test, on remarque que le recuit simulé converge généralement plus vite vers la solution optimale lorsque le problème est de taille raisonnable. Toutefois, il ne donne qu'une solution dans le cas des problèmes multi-modes, ceci confirme les résultats donnés dans [IR92]. A l'inverse, les algorithmes génétiques fournissent plusieurs solutions quasi-optimales mais au prix d'un temps de convergence généralement plus long. Il semble alors naturel d'associer ces deux techniques afin d'améliorer la convergence des algorithmes génétiques.

Il y a eu de nombreuses tentatives de fusion entre les algorithmes génétiques et le recuit simulé, les travaux les plus intéressants étant ceux de Mahfoud et de Goldberg [MG92].

### 4.5.2 Principe du croisement avec recuit simulé

Pour appliquer ce principe de croisement, on commence par sélectionner deux parents $P_1$ et $P_2$ dans la population (voir figure 4.9). On applique ensuite l'opérateur de croisement classique qui engendre deux enfants $C_1$ et $C_2$. Un tournoi est alors effectué entre les parents et les enfants dont les deux vainqueurs sont sélectionnés par le schéma de recuit suivant. On considère l'individu $C_1$. On tire ensuite aléatoirement un des deux parents, soit $P_i$ ce parent :

- si $C_1$ est meilleur que $P_i \Rightarrow C_1$ est sélectionné.
- sinon $C_1$ est sélectionné avec la probabilité :

$$P = e^{-\left(\frac{|f_{C_1} - f_{P_i}|}{c(n)}\right)}$$

où $c(n)$ est un coefficient décroissant en fonction de la génération courante ($n$).

On fait de même pour $C_2$ avec le parent restant et l'on détermine ainsi deux individus $C_1'$ et $C_2'$.

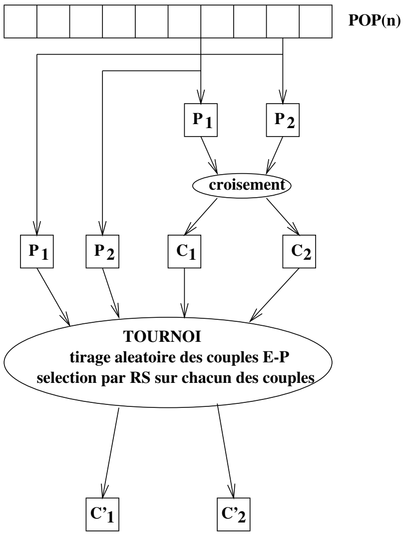



FIG. 4.9: *Principe du croisement avec recuit simulé*

L'évolution de la variable $c(n)$ se fait de la façon suivante. On utilise un schéma de recuit standard géométrique à un palier de basculement. Pratiquement, on calcule trois "températures" dont les valeurs dépendent de la connaissance des écarts $min$ et $max$ des fitness de la population initiale.

$$\begin{cases} C_s = -\frac{\Delta f_{max}}{ln\left(\frac{1}{1-k}\right)} & k = 0.75 \quad \text{"Température initiale"} \\ C_x = -\frac{\Delta f_{max}}{ln\left(\frac{1}{1-k}\right)} & k = 0.99 \quad \text{"Température de basculement"} \\ C_f = -\frac{\Delta f_{min}}{ln\left(\frac{1}{1-k}\right)} & k = 0.99 \quad \text{"Température finale"} \end{cases}$$

où $\Delta f_{min}, \Delta f_{max}$ représentent les écarts minimum et maximum des fitness de la population initiale. Le schéma géométrique fait évoluer la température courante de la façon suivante :

$$\begin{cases} C_0 = C_s \\ C_{n+1} = \alpha_1 C_n \text{ pour } C_s > C_n > C_x; \\ C_{n+1} = \alpha_2 C_n \text{ pour } C_x > C_n > C_f; \end{cases}$$

avec $0 < \alpha_1 < \alpha_2 < 1$.

Pour chaque palier, on calcule le nombre d'itérations de stabilisation à l'aide des formules :

$$N_1 = \frac{ln\left(\frac{C_x}{C_s}\right)}{ln\,\alpha_1} \quad N_2 = \frac{ln\left(\frac{C_f}{C_x}\right)}{ln\,\alpha_2}$$

Ces deux formules permettent de calculer le nombre total de générations pour un problème donné.

## 4.6 Résultats théoriques existants

Les premiers résultats théoriques concernant le fonctionnement des algorithmes génétiques ont été obtenus dans le cadre de l'utilisation du codage binaire, et des opérateurs de croisement et de mutation qui y sont associés et que nous avons décrits dans les sections 4.2.1 et 4.2.2.

Ces résultats sont connus sous le nom de théorie des schémas, ils sont largement développés dans [Gol89]. On en trouvera un exposé plus concis dans [AS92].

Cette théorie utilise les notions de séquences et de schémas :

**Définition 4.1 – Séquence –** *On appelle séquence $A$ de longueur $l(A)$ une suite $A = a_1 a_2 \ldots a_l$ avec* $\forall i \in [1, l], a_i \in V = \{0, 1\}$.

En codage binaire, les chromosomes sont des séquences.

**Définition 4.2 – Schéma –** *On appelle schéma $H$ de longueur $l$ une suite $H = a_1 a_2 \ldots a_l$ avec* $\forall i \in [1, l], a_i \in V^+ = \{0, 1, *\}$.

Une $*$ en position $i$ signifie que $a_i$ peut être indifféremment $0$ ou $1$.

Elle se base sur une étude de l'effet, sur les schémas présents dans la population, des opérateurs de l'algorithme génétique. Nous ne la développerons pas plus avant ici, puisqu'elle concerne exclusivement le codage binaire, que nous n'utilisons pas. Elle ne manque cependant pas d'intérêt pour qui veut comprendre le fonctionnement des algorithmes génétiques et peut offrir une base de réflexion quant à la manière la plus adéquate de coder les données du problème à résoudre à travers des chaînes de bits.

L'existence de résultats théoriques portant sur la convergence des algorithmes génétiques compatibles avec l'utilisation d'un codage réel est plus récente. On trouvera dans la thèse de Raphaël Cerf [Cer94] une étude des propriétés de convergence des algorithmes génétiques, et de leurs états stables. Cette étude reprend des idées utilisées par Olivier Catoni [Cat90] dans l'étude du recuit simulé. L'algorithme génétique y est considéré comme un processus markovien, *perturbé* par les différents opérateurs génétiques. L'étude menée par Raphaël Cerf se base sur la théorie de Freidlin et Wentzell [FW83].

Ces résultats théoriques, très intéressants, ne rendent cependant pas compte du fonctionnement des algorithmes génétiques tels qu'ils sont utilisés en pratiques, qui doivent leurs efficacité à des raffinements (tels que le scaling ou le sharing) dont l'effet n'a pas encore été étudié d'un point de vue théorique. L'utilisation, sur des problèmes pratiques, des algorithmes génétiques reste encore malheureusement assez déconnectée de leur étude théorique.

## 4.7 Autres Algorithmes évolutionnaires

### 4.7.1 Stratégies d'évolution

Les stratégies d'évolution sont en fait généralement désignées, même dans les milieux francophones, par leur nom anglais de *Evolution Strategies*, ou par le sigle *ES*. Elles ont été introduites par Schwefel, à Berlin en 1965. À l'origine, un individu unique produisait, au moyen d'une mutation, un descendant unique, une sélection s'opérait entre eux deux.

Dès leurs origines, une grande originalité des stratégies d'évolution par rapport aux algorithmes génétiques a résidé dans leur utilisation du codage réel (voir section 4.1.2) et de la mutation gaussienne (section 4.2.2).

Elles ont été étudiées et développées par Rechenberg, à qui on doit la règle dite du un cinquième : le taux de mutation est modifié au cours du déroulement de l'algorithme de manière à atteindre un taux de succès des mutation de un cinquième, c'est-à-dire que dans un cas sur cinq en moyenne, le produit d'une mutation (le *fils*) a une meilleure *fitness* que l'élément à partir duquel il a été obtenu (le *père*). Des techniques similaires ont pu ensuite être reprises dans le cadre d'autres algorithmes évolutionnaires, de même que la *self adaptation*, introduite par Schwefel (voir section 4.2.2).

Dans les années 70 ont été développées 2 versions des stratégies d'évolution, les $(\mu + \lambda)$-ES et les $(\mu, \lambda)$-ES, dues à Schwefel, dont on trouvera une description dans [BS93]. Dans les deux cas, $\mu$ individus donnent, au moyen de mutations (et éventuellement d'opérateurs de recombinaisons, tels que le croisement décrit dans la section 4.2.1), $\lambda$ nouveaux individus. Dans le cadre des $(\mu, \lambda)$-ES, $\lambda > \mu$, et la sélection retient $\mu$ individus sur les $\lambda$ qui ont été créés. Elle retient, dans le cadre des $(\mu + \lambda)$-ES, $\mu$ individus sur l'ensemble des $\mu$ *parents* et des $\lambda$ *enfants*. La sélection est déterministe : les meilleurs $\lambda$ individus (en termes de *fitness*) sont conservés, les $\lambda$ moins bons sont éliminés.

Les stratégies d'évolution, en plus de former un groupe d'algorithmes très intéressant en eux mêmes, ont eu une influence positive sur développement des algorithmes génétiques, notamment par la mise au point de nouveaux opérateurs de mutation, et par l'expérience du codage réel qu'elles ont permis d'acquérir.

### 4.7.2 Programmation Évolutionnaire

Le principe de la programmation évolutionnaire, ou Evolutionnary Programming a d'abord été présenté par L. J. Fogel [FOW66] en 1966.

Ces algorithmes ont été grandement développés (notamment par le passage au codage réel) et popularisés par D. B. Fogel [FFAF92].

Leurs principales caractéristiques, par rapport aux autres algorithmes évolutionnaires, étaient, à l'origine, le fait qu'ils n'utilisaient pas de croisement, et surtout l'utilisation d'un mode de sélection originale, la sélection par tournoi.

Nous donnons ici une description simple d'un algorithme de programmation évolutionnaire (cette famille d'algorithmes s'est rapidement enrichie et diversifiée). $\mu$ individus en donnent $\mu$ par mutation. Si le codage utilisé est un codage réel, cette mutation est gaussienne, et son écart-type peut, au moment de la mutation d'un individu, être calculé à partir de la *fitness* de cet individu.

Notons $\mathcal{P}(n)$ la population des $\mu$ individus présent à la génération $n$, et $\mathcal{P}'(n)$ les $\mu$ individus obtenus à partir d'eux par mutation. On va d'abord attribuer une note à chacun des 2 $\mu$ individus de $\mathcal{P}(n) \cup \mathcal{P}'(n)$. Pour chaque individu $e$ de $\mathcal{P}(n) \cup \mathcal{P}'(n)$, on tire aléatoirement $q$ individus dans $\mathcal{P}(n) \cup \mathcal{P}'(n)$. La note de $e$ est alors égale au nombre d'individus (parmi ces $q$) dont la fitness est inférieure à celle de $e$.

La sélection retient pour former la nouvelle population $\mathcal{P}(n+1)$, les $\mu$ éléments ayant obtenu les meilleures notes.

Les performances de représentants de ces deux familles d'algorithmes (ES et EP), et d'un algorithme génétique (utilisant un codage binaire) sont comparées sur des problèmes d'optimisation de fonctions test de $\mathbb{R}^n$ dans $\mathbb{R}$, dans [BS93].

### 4.7.3 Programmation Génétique

La programmation génétique est un type d'algorithme évolutionnaire dû à J.-R. Koza [Koz92], dans lequel les individus sont des arbres représentant des programmes.

Ces arbres sont manipulés directement par les opérateurs de mutation et de croisement. Un croisement consiste à échanger des sous-arbres des deux individus que l'on cherche à croiser, une mutation à remplacer un sous-arbre d'un individu à muter par un sous arbre généré aléatoirement.

Dans le cadre d'un algorithme génétique, l'algorithme manipule des objets (*génotypes) qui codent des données (*phénotypes). Dans le cadre de la programmation génétique, les phénotypes sont les arbres et sont manipulés directement par les opérateurs génétiques.

L'évaluation d'un arbre consiste à effectuer le programme donné par cet arbre et à évaluer la qualité du résultat obtenu par rapport au problème à résoudre. Le déroulement de l'algorithme est similaire à celui d'un algorithme génétique.

Une difficulté importante de cette méthode est de fixer *a priori* le type de nœuds (opérateurs logiques, arithmétiques ou autres) qui serviront à construire les arbres manipulés.

# Chapitre 5

# Évitement réactif par réseaux de neurones

## 5.1 Introduction

Des méthodes de génération de trajectoires sans conflit telles que celles qui sont présentées dans les chapitres 2 et 3 sont susceptibles d'être utilisées dans le cadre d'un système d'évitement embarqué. Cependant, un tel système d'évitement devrait obligatoirement fonctionner selon une approche *séquentielle*, les trajectoires des avions sont générées successivement, selon un *ordre de priorité*, les trajectoires déjà générées imposant des contraintes aux trajectoires qui restent à générer.

Une telle approche pose des problèmes, notamment celui dû au fait qu'elle ne recherche pas de solution optimale au problème global de la génération des trajectoires de tous les avions impliqués dans le conflit. Cela peut entraîner qu'aucune solution ne soit trouvée, alors qu'un autre ordre de priorité, par exemple, ou une méthode centralisée, en trouverait une (voir sections 2.9.3 et 3.7.2).

Un système d'évitement réactif (définition 1.10), c'est-à-dire générant la trajectoire de chaque avion, par une commande continue, à mesure que les différents avions suivent effectivement les trajectoires générées pour eux séparément, éviterait les problèmes liés au choix d'un ordre de priorité, puisqu'elle n'oblige pas à générer toute la trajectoire d'un avion avant que celui-ci ne s'y engage.

L'évitement réactif consiste en effet à ce que chaque avion, compte tenu de sa perception de la situation, agisse à tout moment de manière à poursuivre sa route sans conflit, sans concertation avec les autres avions éventuellement impliqués dans un conflit potentiel avec lui.

En limitant les actions d'évitement à des modifications de cap, on peut donc envisager le rôle d'un système d'évitement réactif, à un instant donné de l'évitement, de la manière suivante : appliquer une fonction donnant, à partir de grandeurs caractérisant la situation d'un avion, une modification du cap de cet avion. Les *grandeurs* en question, à partir desquelles cette fonction calcule la modification du cap d'un avion, décrivent la situation de cet avion, mais doivent aussi contenir des informations sur les avions avec lesquels il risque d'entrer en conflit. Leur définition est une part importante du travail à réaliser dans le cadre d'un système d'évitement réactif. Celles que nous utilisons dans le cadre de la méthode présentée ici sont définies dans la section 5.3.3. Elles nécessitent bien entendu un échange d'informations entre avion, ou du moins une certaine perception, par chaque avion, de la situation des autres avions. Cette fonction doit être telle que si le cap d'un avion est modifié successivement en utilisant cette fonction[1], alors la trajectoire de cet avion sera une trajectoire sans conflit de longueur minimale.

On ne connaît pas d'expression analytique de cette fonction (la connaissance d'une telle fonction résoudrait d'ailleurs le problème de l'évitement). Dans le cas de conflits à deux avions, il existe une méthode d'optimisation locale, LANCELOT [CGT92] permettant d'obtenir une trajectoire optimale pour chacun de ces avions, donc la valeur de la meilleure modification du cap des avions à tout instant. Cependant cette méthode est extrêmement coûteuse en temps de calcul (plusieurs heures pour un cas de conflit) [Dur96]. Comme nous le verrons, nous serons amené à considérer, pour l'apprentissage des réseaux de neurones, de très nombreuses configurations de conflit différentes, ce qui interdit l'utilisation d'une méthode aussi longue. De plus, cette méthode n'est plus applicable dès que le conflit à résoudre implique plus de deux avions. Puisque aucune fonction rapidement calculable et répondant idéalement au critère donné ci-dessus, c'est-à-dire permettant de générer des trajectoires sans conflit de longueur minimale, n'est connue, on cherche à obtenir une fonction l'approchant, en ce sens qu'elle permettrait de répondre le mieux possible à ce critère.

Le problème posé par l'obtention d'une telle fonction présente les deux caractéristiques suivantes :

- On recherche une bonne capacité de généralisation : l'objectif visé est d'obtenir un dispositif permettant de générer des trajectoires sans conflit et de longueur minimale pour un cas de conflit *quelconque*[2].
- On a vu qu'on ne dispose pas de points de l'espace de départ tels qu'on connaisse la valeur de la fonction que l'on cherche à approcher en ces points. On ne peut donc pas opérer de régression sur des couples *entrée / sortie*.

Les réseaux de neurones, qui sont des approximateurs universels [HSW89], nous ont semblé préférables à d'autres approximateurs, tels que les polynomes par exemple, parce qu'ils permettent généralement d'obtenir une meilleure capacité de généralisation.

Le but de la méthode que nous présentons dans ce chapitre est donc de permettre de résoudre les conflits aériens selon le principe suivant : chaque avion est équipé d'un réseau de neurones. Nous supposons ici que tous les avions sont équipés de réseaux de neurones identiques. Tant qu'un conflit est prévu entre des avions, le cap de chacun de ces avions est modifié sur des pas de temps rapprochés. La modification du cap est donnée par la sortie du réseau de neurones, à partir de données traduisant la situation de l'avion et la connaissance qu'il peut avoir de la situation des autres avions impliqués dans le conflit.

Les réseaux de neurones que nous utilisons sont des réseaux multi-couches totalement connectés à une couche cachée (voir 5.2). Ils sont utilisés pour résoudre des conflits à deux ou trois avions. Les entrées du réseau sont données dans la section 5.3.3 (conflits à deux avions), et dans les sections 5.5.3 et 5.5.4 (conflits à trois avions).

[1] La succession de ces modifications correspondra dans la méthode présentée ici à des pas de temps.

[2] Nous nous limiterons cependant à des cas de conflit compatibles avec l'horizon temporel de l'évitement. Ces cas de conflit seront appelés cas de conflit admissibles (définition 5.4).

L'architecture des réseaux est fixe. Leur apprentissage consistera donc à modifier leurs poids.

Comme il n'est pas possible d'opérer une régression sur des couples *entrée / sortie*, on ne peut pas employer pour l'apprentissage des réseaux de neurones des méthodes classiques telles que la rétro-propagation du gradient (voir section 5.2). De plus, on ne peut pas évaluer la qualité d'une réponse isolée du réseau (c'est-à-dire à un pas de temps donné). On ne peut évaluer que la qualité des trajectoires des avions, à la fin de l'évitement, ces trajectoires résultant de plusieurs appels au réseau de neurones.

C'est pourquoi nous utilisons, pour réaliser l'apprentissage des réseaux de neurones, un algorithme génétique (voir chapitre 4). Une évaluation de la qualité de la réponse d'un réseau sur un cas de conflit est obtenue à travers une simulation de la trajectoire des avions, sur la durée de l'évitement. Un mode d'évaluation de réseaux de neurones pour la génération de trajectoires s'apparentant à celui-ci est utilisé dans [SR94].

La fonction d'adaptation d'un réseau utilisée par l'algorithme génétique (voir section 4.1.1) est calculée à partir de l'évaluation de la qualité du réseau sur un ensemble de configurations de conflit. C'est cet ensemble qui constituera la *base d'apprentissage*.

De premiers travaux menés au LOG dans cette direction, dans le cadre du DEA de Gervais Gaudière [Gau95], montrent qu'on peut obtenir assez facilement des réseaux de neurones permettant de générer des trajectoires sans conflit menant à des retards très faible pour un ou même plusieurs cas de conflit donnés, lorsque l'apprentissage du réseau est mené sur ces cas.

Un aspect important du problème à résoudre réside donc dans le fait d'obtenir, à travers l'apprentissage d'un réseau, une bonne capacité de généralisation, c'est-à-dire un bon comportement sur des cas de conflits quelconques. La base d'apprentissage choisie, la manière dont elle est utilisée pour le calcul de l'évaluation des réseau, jouent un rôle fondamental dans l'obtention de cette capacité de généralisation.

Nous avons testé et comparé trois manières de mener l'évaluation des réseaux : la première utilise une base d'apprentissage *fixe* (les configurations de conflit qui la composent sont les mêmes tout au long de l'apprentissage), la deuxième utilise une base d'apprentissage *renouvelée* (les configurations de conflit qui la composent sont renouvelées à chaque génération de l'algorithme génétique). Ces deux méthodes sont présentées dans la section 5.3.7 et comparées dans les sections 5.4 (conflits à deux avions) et 5.6 (conflits à trois avions). La troisième approche, utilisant la coévolution, est présentée dans le chapitre 6. Son principe est de faire évoluer, au moyen d'un algorithme génétique, non seulement les réseaux de neurones, mais aussi les configurations de conflit utilisées pour leur apprentissage.

## 5.2 Les réseaux de neurones

Nous présentons dans cette section les réseaux de neurones très brièvement. On pourra, pour plus de détails, se reporter notamment à [HKP91].

À l'origine des réseaux de neurones formels se trouve une modélisation mathématique s'inspirant du fonctionnement des neurones du cerveau humain due à McCulloch et Pitts [MP43]. Le fonctionnement du cerveau humain peut se décrire (très schématiquement) de la manière suivante : le cerveau est constitué de cellules qui sont appelées des *neurones*. Chaque neurone est composé d'un noyau, de connexions entrantes (les *dendrites*) et d'une connexion sortante (l'*axone*). L'influx nerveux parcourt un neurone des dendrites au noyau, du noyau à l'axone. On peut se représenter le rôle du noyau en considérant qu'il transmet un influx nerveux à l'axone si et seulement si la somme pondérée des influx lui parvenant des différentes dendrites (les poids pouvant être négatifs) est supérieure à un certain seuil (il s'agit là bien entendu d'une représentation très simplifiée). L'axone d'un neurone peut se subdiviser et transmettre à travers des *synapses* l'influx nerveux aux dendrites de plusieurs autres neurones (formant ainsi un *réseau*).

Les influx nerveux qui arrivent au cerveau sont aussi reçus par des dendrites. Ceux qui en sortent proviennent d'axones.

Le modèle mathématique du neurone, donné par McCulloch et Pitts utilisait une fonction seuil $f_\theta$ de $\mathbb{R}$ dans $\{0, 1\}$, définie à partir d'un seuil $\theta \in \mathbb{R}$ :

$$\begin{cases} f_\theta(x) = 1 & \text{si } x > \theta, \\ f_\theta(x) = 0 & \text{sinon.} \end{cases} \tag{5.1}$$

Le neurone donne une sortie $O \in \mathbb{R}$ à partir d'un nombre $n_e$ d'entrées, notées $I_i \in \mathbb{R}$, pour $i = 1..n_e$. Des poids $W_i \in \mathbb{R}$, sont associés à ces entrées ; ce sont les poids et la valeur du seuil $\theta$ qui caractérisent le neurone, la sortie étant donnée par :

$$O = f_\theta(\sum_{i=1}^{n_e} W_i I_i) \tag{5.2}$$

À l'intérieur d'un réseau de neurones, la sortie d'un neurone peut être donnée en entrée à un autre neurone.

Les réseaux que nous utilisons sont des *réseaux de neurones à sortie réelle*, dans lesquels la sortie d'un neurone n'est plus égale à 0 ou à 1, mais peut prendre des valeurs réelles dans l'intervalle [0,1]. La fonction seuil $f_\theta$ est donc remplacée par une fonction de $\mathbb{R}$ dans $[0, 1]$, appelée *fonction d'activation*. La fonction d'activation le plus généralement utilisée sera une fonction de type *sigmoïde* :

$$f_\theta(x) = \frac{1}{1 + e^{-(x-\theta)}} \tag{5.3}$$

Nous utiliserons des réseaux de neurones multi-couches, qui sont des cas particuliers de réseaux à sens unique :

**Définition 5.1 – Réseau à sens unique –** *Un réseau à sens unique est un réseau dont les connexions ne forment pas de boucles, c'est-à-dire tel que la sortie d'un neurone ne contribue pas (même après être passée à travers plusieurs autres neurones) à une de ses entrées.*

**Définition 5.2 – Réseau multi-couche –** *Un réseau multi-couche est composé d'une couche d'entrée, d'une couche de sortie, et de couches intermédiaires, appelées* couches cachées. *Si les couches de neurones sont numérotées de la couche d'entrée à la couche de sortie, les entrées de* *la couche $i+1$ sont données par des sorties de la couche $i$. On dira que le réseau est totalement connecté si* chaque *neurone de la couche $i+1$ prend en entrée l'ensemble des sorties de* tous *les neurones de la couche $i$.*

Une notion très importante concernant les réseaux de neurones est la notion d'apprentissage.

Quand l'architecture du réseau que l'on cherche à obtenir est fixée, l'apprentissage consiste à modifier les poids du réseau. Lorsqu'on connaît un nombre suffisant de couples *entrées / sortie*, on peut opérer sur ces couple une régression. Une méthode efficace et généralement utilisée est la rétro-propagation du gradient [LeC87] : on cherche à minimiser les erreurs quadratiques entre les sorties du réseau sur ces entrées et les *bonnes* sorties correspondantes. Cette minimisation est effectuée au moyen d'une méthode de gradient, appliquée de neurone en neurone, de la sortie vers les entrées, d'où le terme de rétro-propagation.

On peut aussi modifier l'architecture des réseaux au cours de l'apprentissage. Cela peut, par exemple, être réalisé au moyen d'algorithmes évolutionnaires (voir chapitre 4). On définit alors des individus codant l'architecture des réseaux de neurones, les opérateurs génétiques étant adaptés aux types d'architecture que l'on souhaite générer, en modifiant ponctuellement cette architecture (mutation) ou en échangeant des sous-parties de deux individus (croisement). De telles techniques d'apprentissage, utilisant le schéma d'évolution de la programmation évolutionnaire (voir section 4.7.2), sont par exemple utilisées dans [Fad98].

Les réseaux que nous utilisons sont des réseaux multi-couches à une couche cachée. Leur architecture est fixe, et l'apprentissage dans ce cas consiste à modifier les poids. Comme nous l'avons vu dans l'introduction, nous ne disposons pas de données permettant d'opérer une régression, par rétro-propagation du gradient, par exemple.

L'apprentissage des réseaux de neurones est réalisé au moyen d'un algorithme génétique (voir chapitre 4). Le principe d'un apprentissage de réseaux de neurones à architecture fixée au moyen d'un algorithme génétique est très simple. Les individus de la population codent les poids des réseaux de neurones. La fonction d'adaptation utilisée par l'algorithme génétique doit rendre compte de la qualité de la sortie de chaque réseau de la population. Cette méthode peut donc s'appliquer dès que l'on peut évaluer la qualité de la réponse apportée au problème par un réseau de neurones. Elle ne nécessite pas de connaître *a priori* la *bonne* sortie du réseau pour un jeu d'entrées donné.

## 5.3 Génération de trajectoires d'évitement

### 5.3.1 Hypothèses

Rappelons que l'on dit que deux avions sont *séparés* si la distance entre eux est supérieure à la *norme de séparation horizontale* $n_h$ (les avions volant dans un plan horizontal, on ne tient pas compte de la norme de séparation verticale). On dit qu'il y a entre deux avions un *conflit* s'ils ne sont pas séparés.

Nous nous limiterons dans un premier temps aux conflits n'impliquant que deux avions. Nous supposons que les deux avions volent dans un même plan horizontal, que la vitesse de chacun d'eux est constante au cours du temps. En l'absence de déviation due à l'évitement, le cap de chaque avion est également constant. L'évitement est recherché grâce à des modifications de caps des avions qui continuent de voler à vitesse constante, dans un plan horizontal.

La position occupée par un avion $t = 0$ est appelée sa *position initiale*. Une durée de l'évitement, $t_f$ est fixée. La position qu'occuperait l'avion en l'absence de déviation au temps $t = t_f$ est appelée *position finale* ou *destination*.

Nous supposons que chacun des deux avions connaît la position finale de l'autre avion, et qu'à tout moment (à chaque pas de temps, puisque comme nous allons le voir, le temps est discrétisé) il connaît sa position, sa vitesse, son cap et sa destination, ainsi que la position, la vitesse, le cap et la destination de l'autre avion. Cette connaissance pourra être obtenue, comme dans le cadre de la méthode de génération d'un ordre de priorité entre avions (section 2.8.1), sans communication entre avions deux à deux, mais par diffusion par chaque avion des informations le concernant.

### 5.3.2 Rôle du réseau de neurones

Le réseau utilisé est un réseau de neurones multi-couche totalement connecté à une couche cachée à sortie réelle (voir section 5.2). Il utilise 8 entrées décrites dans la section 5.3.3 et retourne une sortie réelle. Les poids du réseau de neurones sont modifiés au cours de l'apprentissage par un algorithme génétique (voir chapitre 4). La fonction d'activation est une sigmoïde, dont l'expression est donnée par l'équation 5.3. Aux entrées de chaque couche est ajoutée une entrée fixe, dont la valeur sera 1,0. Pour chaque neurone, le poids correspondant à cette entrée donne le seuil $\theta$ de la fonction d'activation.

La couche cachée des réseaux que nous utilisons comporte 25 neurones (nous avons fixé ce nombre de neurones de la couche cachée après avoir mené des essais avec différente valeurs pour ce nombre). En comptant les poids correspondant aux seuils de fonctions sigmoïde utilisées par les neurones, un total de 251 poids.

Le principe de l'utilisation des réseaux de neurones pour la génération de trajectoires d'évitement est le suivant :

- Le temps est discrétisé. Chaque pas de temps correspond à un nouveau calcul du cap de l'avion au moyen du réseau de neurones. La vitesse et l'altitude des avions étant constantes, les modifications de leurs trajectoires ne sont dues qu'à des changements de cap, qui eux-mêmes sont supposés être instantanés, et prendre place sur un pas de temps. On note $l_t$ la longueur d'un pas de temps, $N_t$ le nombre de pas de temps. Ici, pour plus de simplicité, nous utiliserons la même discrétisation du temps pour tous les avions impliqués dans le conflit. Cela n'est pas nécessaire au fonctionnement de la méthode. Il n'est notamment pas nécessaire que les instants auxquels chaque avion modifie son cap (c'est-à-dire les pas de temps) soient synchronisés. $N_t = int(t_f/l_t)$, où $int$ désigne la partie entière. Pour $k \in \{0, .., N_t\}$, $t_k = k\ t$est le $k$-i'eme pas de temps.
- Les trajectoires des deux avions sont générées au moyen de réseaux de neurones identiques (même architecture, mêmes poids). Chacun de ces réseaux de neurones génère de manière indépendante la trajectoire de chacun des avions. Il n'y a pas de coordination entre avions, autre que la connaissance, par chaque avion, de la position, de la vitesse et du cap de l'autre avion au pas de temps courant, ainsi que de son point de destination (position à $t = t_f$ en l'absence de déviation).

- chaque pas de temps, si le recours au réseau de neurones est justifié (c'est-à-dire s'il y a toujours un conflit prévu), celui-ci prend en entrée des paramètres décrivant la situation dans laquelle se trouve l'avion dont on cherche à générer une trajectoire, par rapport à l'autre avion et à sa destination. Sa sortie est utilisée pour calculer une modification du cap de l'avion. Les entrées et la sortie du réseau de neurones sont présentées et discutées dans la section 5.3.3.

Le réseau de neurones n'est pas systématiquement utilisé à chaque pas de temps. En effet, avant de calculer un cap modifié pour l'avion, on vérifie d'abord qu'un conflit aurait lieu entre les deux avions avant qu'ils n'atteignent leurs points de destination, dans le cas où, à partir du pas de temps courant, ils suivraient tous deux une route directe vers ces points de destinations. Les vitesses des avions étant constantes et l'avion pour lequel on doit générer une trajectoire connaissant la position courante et la destination de l'autre avion, cela revient à résoudre une équation du second degré[3].

S'il n'y a pas de conflit entre les deux avions avant qu'ils aient tous deux atteint leur point de destination, l'avion prend (ou garde) un cap direct vers sa destination. En effet, dans ce cas, on sait quel est le meilleur cap pour l'avion : la route directe vers la destination étant évidemment la plus rapide, c'est celle qui répond le mieux au problème de l'évitement si elle n'entraîne pas de conflit entre les avions.

Ceci ne nécessite pas de supposer une coordination entre les avions autre que la connaissance, par chaque avion, à chaque pas de temps, de la position, de la vitesse et de la destination de l'autre avion impliqué dans le conflit, la vérification qu'aucun conflit n'aura lieu entre les avions peut être effectuée séparément par chaque avion.

### 5.3.3 Entrées et sortie du réseau

Les entrées doivent traduire la situation dans laquelle se trouve un avion impliqué dans un conflit potentiel, de manière à fournir des renseignements permettant de choisir la meilleure modification de cap. Il est difficile de déterminer *a priori* quelles informations seront les plus utiles aux réseaux de neurones. Certaines des entrées choisies sont utilisées par des contrôleurs humains pour caractériser les conflits. Il faut rappeler toutefois que le mode de fonctionnement d'un réseau de neurones n'a aucune raison de traduire directement la manière dont un agent humain prendrait une décision. Nous avons expérimenté plusieurs vecteurs d'entrées différents, et profité de l'expérience apportée sur ce point par [Gau95]. Les entrées que nous présentons ici sont celles grâce auxquelles nous avons obtenu les meilleurs résultats, mais nous ne pouvons bien entendu pas affirmer qu'un autre jeu d'entrées serait forcément moins efficace.

Ces entrées n'utilisent pas les positions absolues des avions, mais leurs positions relatives, tant en termes de distances que d'angles. En effet, au moment du conflit, les positions et déplacements relatifs des avions sont seuls importants pour le choix de modifications éventuelles de leurs caps. Il est clair que deux conflits se déduisant l'un de l'autre par un déplacement du plan (translation ou rotation) seront résolus par les mêmes manœuvres.

[3] Au cas où les vitesses prévues pour les avions jusqu'à leurs destinations ne seraient pas constantes, cette séparation pourrait par exemple être vérifiée par une *simulation* des trajectoires des avions, en discrétisant le temps et en faisant varier les vitesses des avions sur les pas de temps.

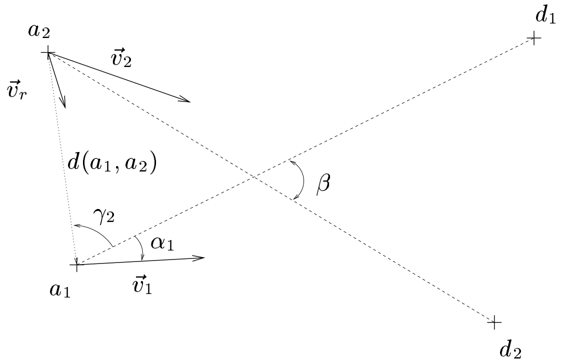



FIG. 5.1: *Entrées du réseau de neurones*

Dans la suite, nous appellerons *avion courant* l'avion pour lequel le réseau de neurones doit fournir une modification de cap (il sera noté $a_1$ dans le suite de cette section), et dans le cadre d'un conflit à deux avions, on appellera *intrus* l'autre avion (noté $a_2$) impliqué dans le conflit. Les positions des avions au *pas de temps courant* (le pas de temps auquel correspond le recours au réseau de neurones) seront appelées leurs *positions courantes*. La vitesse de l'avion $a_i$ est notée $v_i$, et son cap à un instant $t$ est noté $c_i(t)$.

La direction de référence, en ce qui concerne les caps et les angles, n'est pas le cap de l'avion courant, mais le cap qu'il aurait s'il suivait à partir de sa position courante une route directe vers sa destination, cap que nous appellerons son *cap direct vers sa destination*, ou plus simplement *cap direct*. Prendre comme direction de référence le cap de l'avion courant pourrait sembler équivalent, dès que la différence entre son cap courant et son cap direct vers sa destination fait partie des entrées du réseau. Cependant, on observe une meilleure efficacité du réseau en prenant le cap direct vers la destination comme direction de référence. Ce résultat peut s'expliquer de la manière suivante : ce cap direct correspond à la route la plus courte vers la destination, l'avion doit donc tenter de s'en approcher le plus possible pour minimiser sa déviation.

Les entrée du réseau de neurones sont *normalisées* de manière à être du même ordre de grandeur (elles seront comprises entre $-1$ et 1).

Les entrées du réseau traduisent les données suivantes (voir figure 5.1) :

- L'angle, noté $\alpha_1$, entre la route directe de l'avion vers sa destination et son cap.
- La différence entre les vitesses des avions.
- La distance entre les avions.
- La vitesse de rapprochement des deux avions.
- L'angle $\gamma_2$ sous lequel l'intrus est vu de l'avion courant, par rapport au cap direct de l'avion courant vers sa destination.
- L'angle $\beta$ entre les routes directes des avions de leurs positions courantes à leurs destinations.

Ces 6 données sont traduites par 8 entrées : les angles $\alpha_1$ et $\gamma_2$ ne sont pas donnés directement au réseau de neurones : chacun d'eux est traduit par deux entrées : son sinus et à son cosinus. Le fait d'utiliser directement les valeurs de angles introduit une discontinuité ne correspondant pas à la réalité. Si les angles sont donnés, par exemple, en degrés, avec des valeurs allant de 0 à 360, quand l'angle passe continûment de 1 à -1 degré, l'entrée correspondante passe de 1 à 359 degrés. Les sinus et cosinus des angles ne présentent pas cet inconvénient, et rendent bien compte de la périodicité des angles. L'angle $\beta$ est l'angle non orienté entre les routes directes des avions. Ce problème ne se pose donc pas pour lui : on prend toujours sa valeur entre 0 et 180 degré.

Remplacer les angles $\alpha_1$ et $\gamma_2$ par leurs sinus et cosinus augmente le nombre d'entrées, donc de poids, du réseau de neurones. La dimension de l'espace sur lequel l'algorithme génétique effectue la recherche est donné par le nombre de poids du réseau de neurones. Cette dimension se trouve donc augmentée par l'utilisation des sinus et cosinus à la place des valeurs des angles. Cependant, nous avons constaté, en comparant des réseaux ayant utilisé en entrée les valeurs des angles à des réseaux ayant utilisé leurs sinus et cosinus, que ceux-ci étaient sensiblement plus efficaces, et que de plus cette efficacité supérieure était obtenue après un temps d'apprentissage plus court.

Les 6 données énumérées ci-dessus sont donc traduites par les entrées du neurones de la manière suivante :

**L'angle $\alpha_1$ :** C'est l'angle entre la route directe de l'avion vers sa destination et son cap. Il est traduit par deux entrées :

$$E_1 = \sin \alpha_1$$

et

$$E_2 = \cos \alpha_1$$

$E_1$ donne la déviation de l'avion courant par rapport au cap qu'il devrait prendre, à partir du pas de temps courant, pour minimiser l'allongement de sa trajectoire.

**Différence de vitesses :** Il s'agit de la différence entre la vitesse de l'intrus et celle de l'avion courant, divisée par la vitesse de l'avion courant :

$$E_3 = \frac{v_2 - v_1}{v_1}$$

Le signe de $E_3$ indique si l'intrus est plus rapide ou plus lent que l'avion courant, sa valeur absolue traduit l'amplitude de la différence entre les vitesses des deux avions.

Nous rappelons que les vitesses des deux avions sont comprises entre $v_{min}$ et $v_{max}$. On a donc :

$$\frac{v_{min} - v_{max}}{v_{min}} \leq E_3 \leq \frac{v_{max} - v_{min}}{v_{min}} \tag{5.4}$$

$v_{min}$ et $v_{max}$ seront en général tels que $v_{max} - v_{min} > v_{min}$, et donc que l'entrée $E_3$ soit comprise entre -1 et 1. Ce sera en tout cas le cas dans les applications de cette méthode que nous présenterons dans la section 5.3.6.

Nous verrons dans la section 5.3.5 que cette différence entre les vitesses des avions est une donnée importante du problème de l'évitement.

**Distance entre les avions :** Cette distance, notée $d(a_1, a_2)$, a une signification très différente dans le cadre de l'évitement selon qu'elle est supérieure à la norme de séparation $n_h$ (les avions sont alors séparés), ou inférieure à $n_h$ (les avions ne sont pas séparés).

Cela pourrait être traduit par le signe de $d(a_1, a_2) - n_h$. Cependant, $d(a_1, a_2) - n_h$ ne rend pas compte de la différence fondamentale qui existe entre les deux situations, différence qui se retrouve dans la réaction que l'on souhaite voir l'avion adopter : si $d(a_1, a_2) < n_h$, les avions ne sont plus séparés, la minimisation de l'allongement de la trajectoire de l'avion est secondaire, la priorité étant donnée à faire augmenter la distance entre les deux avions de manière à ce que les avions soient de nouveau séparés le plus rapidement possible. On peut parler ici d'une certaine *rupture* dans les réactions souhaitables, rupture que traduirait l'inverse de $d(a_1, a_2) - n_h$.

Le fait de prendre $1/(d(a_1, a_2) - n_h)$ comme entré du réseau pose bien entendu un problème dû au fait que cette grandeur tend vers l'infini quand $d(a_1, a_2)$ tend vers $n_h$.

On définit $\delta_{dist}$ de la manière suivante : $\delta_{dist}$ est égal à la différence $d(a_1, a_2) - n_h$ exprimée en soixantième de mille nautique[4] si cette différence est non nulle, et égal à un si cette différence est nulle.

On prend donc comme quatrième entrée du réseau de neurones :

$$E_4 = \frac{1}{\delta_{dist}}$$

Puisque on a toujours $|\delta_{dist} \geq 1|$, on a :

$$-1 \leq E_4 \leq 1$$

On observe en effet de meilleurs résultats avec cette définition de $E_3$ qu'en prenant comme entrée $d(a_1, a_2) - n_h$. Ceci s'expliquant sans doute par le fait que la valeur de $E_3$ varie beaucoup quand $d(a_1, a_2)$ est proche de $n_h$, avec une discontinuité et un changement de signe pour $d(a_1, a_2) = n_h$, ce qui permet de mieux traduire la signification de $n_h$ dans le cadre du problème de l'évitement aérien.

**Vitesse de rapprochement des deux avions** On note $v_r$ la vitesse relative de l'intrus par rapport à l'avion courant. Les vitesses et les caps des deux avions étant connus, cette vitesse relative se calcule facilement.

On définit $v_{rmax} = 2\ v_{max}$ qui est la borne supérieure de cette vitesse relative, et on prend pour cinquième entrée du réseau de neurones :

$$E_5 = \frac{v_r}{v_{rmax}}$$

On a bien -1 $\leq E_5 \leq 1$. Si $E_5$ est négative, les avions s'approchent l'un de l'autre. Ils s'éloignent l'un de l'autre si $E_5$ est positive.

[4]Dans le programme traduisant l'algorithme présenté ici, les distances sont exprimées, pour des raisons de cohérence entre les unités utilisées, en soixantièmes de mille nautique. Il est d'ailleurs raisonnable, dans le cadre de l'application que nous avons en vue, de ne pas demander de précision supérieure au soixantième de mille nautique sur les positions des avions

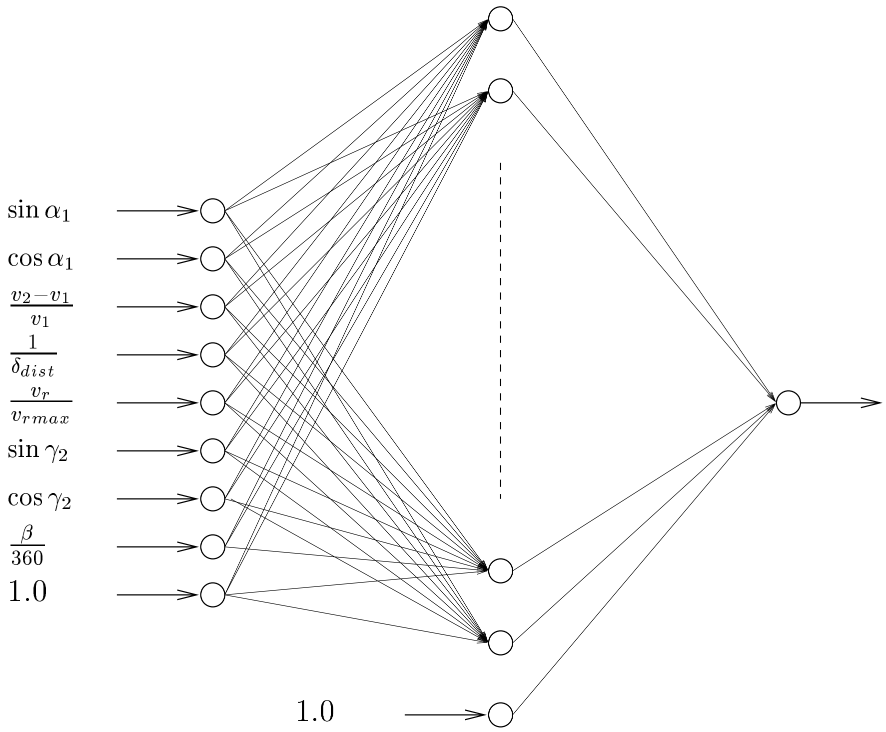



FIG. 5.2: *Architecture du réseau de neurones utilisé*

**Angle $\gamma_2$ :** C'est l'angle entre la droite reliant la position de l'avion courant à sa destination et celle reliant la position de l'avion courant à celle de l'intrus. Pour traduire la situation *vue de l'avion courant*, le gisement de l'intrus par rapport à l'avion courant (c'est-à-dire l'angle entre la droite reliant la position de l'avion courant à celle de l'intrus et le vecteur vitesse de l'avion courant) pourrait sembler plus indiqué, mais, comme nous l'avons vu, il s'est révélé préférable de prendre l'axe reliant l'avion courant à sa destination comme axe de référence. On a donc :

$$E_6 = \sin \gamma_2$$

et

$$E_7 = \cos \gamma_2$$

**Angle $\beta$ :** Il s'agit de l'angle non orienté $\beta$ entre les routes directes reliant les positions courantes des avions à leurs destinations.

$$E_8 = \frac{\beta}{360}$$

$\beta$ traduit une donnée importante de la géométrie du conflit. On constate par exemple que les situations dans lesquelles $\beta$ est faible au début du conflit sont celles qui sont, pour le réseau de neurones, les plus difficiles à résoudre.

La sortie du réseau de neurones est un réel compris entre 0 et 1, à partir duquel on calculera une modification du cap courant de l'avion.

On définit le taux de virage maximal de l'avion, c'est-à-dire sa vitesse maximale de changement de cap lors d'un virage, que l'on note $t_v$. En pratique nous prendrons un taux de virage maximal de trois degrés par secondes, correspondant au taux de virage standard de $360°$ en deux minutes.

Si l'on est au pas de temps $k$, c'est-à-dire à l'instant $t = t_k$, le cap courant de l'avion $a_i$ pour lequel le réseau génère une trajectoire est noté $c_i(t_k)$. Nous avons supposé que cet avion était l'avion $a_1$. On note $s_k$ la sortie du réseau de neurones obtenue avec les entrées correspondant à l'instant $t = t_k$.

On obtient à partir de $s_k$ la modification $\delta_{ck}$ du cap de l'avion courant :

$$\delta_{ck} = (s_k - 0,5) t_v l_t \tag{5.5}$$

Le taux de virage de l'avion sera donc toujours inférieur ou égal au taux de virage maximal.

Au pas de temps suivant, le cap de l'avion sera $c_1(t_{k+1}) = c_1(t_k) + \delta_{ck}$.

Nous avons vu (section 5.3.2) que si les routes directes des avions à partir de leurs positions au pas de temps courant n'entraînent pas de conflit, les avions prennent instantanément un cap direct vers leurs destinations. Cela peut amener un avion à effectuer un virage incompatible avec la limitation de leurs taux de virage. Dans le cadre d'une application plus réaliste, on pourra tenir compte sans aucun problème de cette limitation dans ce cas aussi, en simulant le virage de chacun des avions à un taux de virage standard avant qu'il ne puisse prendre une route directe.

### 5.3.4 Configurations de conflit

Nous allons présenter la notion de *configurations de conflit*. C'est sur ces configurations que le réseau de neurones sera utilisé, et c'est donc au moyen de telles configurations que l'apprentissage des réseaux sera mené. De même, les capacités de généralisation des réseaux obtenus après l'apprentissage seront testées sur un grand nombre de configurations de conflit. Ces configurations, aussi bien celles qui seront utilisées pour l'apprentissage que celles sur lesquelles seront testées les capacités de généralisation du réseau, seront générées aléatoirement. La description de cette génération aléatoire fera l'objet de la section 5.3.6.

Afin de pouvoir donner la définition d'une configuration de conflit, nous avons besoin de définir les termes *cas de conflit* et *cas de conflit admissible*.

**Définition 5.3 – Cas de conflit –** *un cas de conflit est la donnée d'une trajectoire pour chacun des avions (c'est-à-dire d'une position et d'un vecteur vitesse à un instant donné), ces trajectoires étant telles qu'il existe un instant t où les avions seront en conflit.*

Une distance d'alerte, notée $d_a$, est définie. Nous considérons qu'une trajectoire d'évitement doit être calculée pour chacun des deux avions dès que la distance entre eux est inférieure à la distance d'alerte $d_a$. Une fois que la distance entre les avions atteint la valeur de $d_a$, le réseau de neurones est utilisé, à chaque pas de temps, si les trajectoires directes des avions vers leurs destinations n'assurent pas la séparation (voir section 5.3.2), même si la distance entre les avions, du fait des déviations qu'ils subissent, est redevenue supérieure à $d_a$.

Pour un cas de conflit donné, on prendra l'origine du temps ($t = 0$) à l'instant précédent le conflit auquel la distance entre les deux avions est égale à $d_a$. Les *positions initiales* des avions sont donc celles qu'ils occupent à cet instant. Le cas où la distance entre les deux avions est constante au cours du temps n'a pas à être pris en compte : il n'y a pas de conflit.

Nous définissons aussi une durée, qui sera la longueur de la *fenêtre temporelle* sur laquelle sera menée l'évitement, et qui sera notée $t_f$.

Comme nous l'avons vu, les positions que les avions occuperaient au temps $t = t_f$ en l'absence de déviation sont appelées leurs *positions finales* ou *destinations*.

Nous ne souhaitons considérer que des cas de conflit tels que, en l'absence d'évitement :

1. Au temps $t = 0$, la distance entre les deux avions est égale à $d_a$.
2. Un conflit aura lieu entre les deux avions entre les temps $t = 0$ et $t = t_f$ : si ce n'est pas le cas, il est inutile de générer des trajectoires d'évitement pour les avions.
3. Les avions seront séparés au temps $t = t_f$, c'est-à-dire que la distance entre les positions finales des deux avions est supérieure à la norme de séparation $n_h$.

**Définition 5.4 – Cas de conflit admissible –** *Pour des valeurs de $d_a$ et $t_f$ fixées, nous appellerons cas de conflit admissible un cas de conflit respectant les trois conditions ci-dessus, c'est-à-dire tel que si le temps $t = 0$ est fixé à l'instant précédent le conflit auquel les avions sont distants l'un de l'autre de $d_a$, alors les avions seront en conflit à un instant $t < t_f$ et seront de nouveau séparés au temps $t = t_f$.*

La première de ces trois conditions est assurée par le choix de l'origine des temps.

Les autres conditions nécessitent que le temps $t_f$ soit suffisamment grand pour que les avions soient en conflit à un temps $t < t_f$ et de nouveau séparés au temps $t = t_f$. On ne peut pas choisir une valeur de $t_f$ telle que ces conditions soient satisfaites pour tous les cas de conflit. En effet, le temps mis par deux avions pour s'approcher l'un de l'autre de manière à ce que la distance entre eux passe de $d_a$ (avec $d_a > n_h$) à $n_h$ peut être arbitrairement grand : par exemple, dans le cas d'un rattrapage (un avion plus rapide suivant un avion plus lent, avec le même cap), ce temps tend vers l'infini quand la vitesse de l'avion le plus rapide tend vers celle de l'avion le plus lent.

Un cas de conflit admissible est déterminé par la position et le vecteur vitesse des avions au temps $t = 0$. Pour un cas de conflit $C_c$, les caractéristiques suivantes sont importantes, comme nous le verrons dans la section 5.4 :

- La vitesse $v_1$ de l'avion $a_1$, avec $v_{min} \leq v_1 \leq v_{max}$.
- La vitesse $v_2$ de l'avion $a_2$, avec $v_{min} \leq v_2 \leq v_{max}$.
- La distance minimale entre les avions, en l'absence de déviation, notée $d_{min}(C_c)$. On a $0 \leq d_{min}(C_c) < n_h$. On peut considérer que $d_{min}(C_c)$ mesure la *gravité* du conflit.
- L'angle $\theta_2(0)$ entre $\vec{v}_1(0)$ et $\vec{v}_2(0)$, les vecteurs vitesse des avions, au temps $t = 0$.

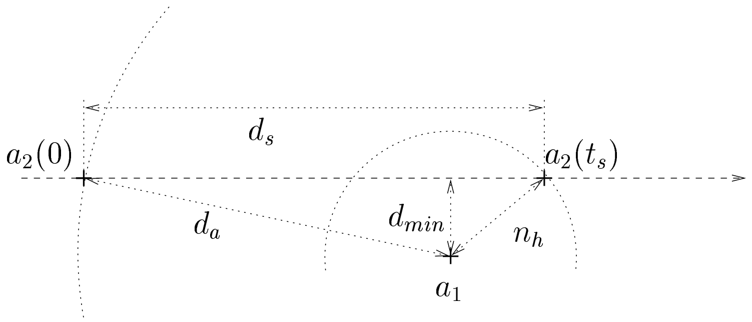



FIG. 5.3: *$d_{min}$ et $d_s$ pour un cas de conflit $C_c$*

Pour comprendre l'importance de la valeur de $t_f$, nous allons calculer, pour une valeur de $t_f$ donnée, la valeur minimale de $|v_1 - v_2|$ pour qu'un cas de conflit soit admissible, quelles que soient les valeurs de $\theta_2(0)$ et de $d_{min}(C_c)$ et la valeur minimale de $\theta_2(0)$ pour qu'un cas de conflit soit admissible, quelles que soient les valeurs de $v_1$, $v_2$, et $d_{min}(C_c)$.

On note $t_s(C_c)$ le temps de sortie de conflit. C'est le temps qui s'écoule entre le moment précédent le conflit où la distance entre les deux avions est égale à $d_a$ et le moment où le conflit entre les avions prend fin.

Plaçons-nous dans le référentiel lié à l'avion $a_1$. Dans ce référentiel, $\mathcal{R}_1 = (a_1, \vec{e}_{x_1}, \vec{e}_{y_1})$ est un repère du plan dans lequel volent les avions tel que $\vec{v}_1 = v_1\vec{e}_{x_1}$ On note, toujours dans ce référentiel, $d_s(C_c)$ la distance que parcourt l'avion $a_2$ entre $t = 0$ et $t = t_s(C_c)$ (voir figure 5.3) :

$$d_s(C_c) = \sqrt{{d_a}^2 - d_{min}(C_c)^2} + \sqrt{{n_h}^2 - d_{min}(C)^2} \tag{5.6}$$

La distance $d_s(C_c)$ est donc maximale pour les cas de conflit $C_c$ tels que $d_{min}(C_c) = 0$.

La vitesse de l'avion $a_2$ dans ce référentiel est notée $v_{rel}(C_c)$. Dans le repère $\mathcal{R}_1$, $\vec{v}_2$ le vecteur vitesse absolue de $a_2$ s'écrit :

$$\vec{v}_2 = v_2 \cos\theta_2(0)\vec{e}_{x_1} + v_2 \sin\theta_2(0)\vec{e}_{y_1} \tag{5.7}$$

On a :

$$\begin{aligned} v_{rel}(C_c) &= \sqrt{(v_2\cos\theta_2(0) - v_1)^2 + (v_2\sin\theta_2(0))^2} \\ &= \sqrt{{v_2}^2 + {v_1}^2 - 2v_1v_2\cos\theta_2(0)} \end{aligned}$$

Et on a :

$$t_s(C_c) = \frac{d_s(C_c)}{v_{rel}(C_c)} \tag{5.8}$$

C'est-à-dire, d'après les équations 5.6 et 5.3.4 :

$$t_s(C_c) = \frac{\sqrt{{d_a}^2 - d_{min}(C_c)^2} + \sqrt{{n_h}^2 - d_{min}(C_c)^2}}{\sqrt{{v_2}^2 + {v_1}^2 - 2v_1v_2\cos\theta_2(0)}} \tag{5.9}$$

D'après la définition 5.4, pour une valeur de $t_f$, le cas de conflit $C_c$ est admissible si et seulement si on a :

$$t_s(C_c) \leq t_f \tag{5.10}$$

Le choix d'une valeur de $t_f$ impose des contraintes sur les caractéristiques des cas de conflit admissibles :

- On ne peut pas avoir $\theta_2(0) = 0$ et $v_1 = v_2$. On aurait alors une distance constante entre les avions, et donc $t_s(C_c) = \infty$.
- Si on veut avoir des cas de conflit admissibles pour toutes les valeurs possibles de l'angle entre les vecteurs vitesse des avions ($\theta_2(O)$) et de la distance minimale entre les avions en l'absence de déviation ($d_{min}(C_c)$), cela entraîne une limitation sur les valeurs relatives de $v_1$ et $v_2$. Pour $v_1$ et $v_2$ fixés, $ts(C_c)$ est maximal pour $\theta_2(0) = 0$ et $d_{min}(C_c) = 0$ et dans ce cas :

$$t_s(C_c) = \frac{d_a + n_h}{|v_2 - v_1|} \tag{5.11}$$

  Pour que tout cas de conflit soit admissible, quelles que soient les valeurs de $\theta_2(O)$ et $d_{min}(C_c)$, on doit imposer sur les vitesses des avions la condition suivante :

$$|v_2 - v_1| \geq \frac{d_a + n_h}{t_f} \tag{5.12}$$

- Si on veut avoir des cas de conflit admissibles pour toutes les valeurs des vitesses $v_1$ et $v_2$ et de $d_{min}(C_c)$, cela entraîne une limitation sur l'angle entre les vecteurs vitesse des avions. La condition 5.10 donne :

$$\cos\theta_2(0) \leq \frac{1}{2v_1v_2}\left({v_1}^2 + {v_2}^2 - \left(\frac{\sqrt{{d_a}^2 - d_{min}(C_c)^2} + \sqrt{{n_h}^2 - d_{min}(C_c)^2}}{t_f}\right)^2\right) \tag{5.13}$$

  On doit donc avoir, pour que cette condition tiennent quelles que soient les valeurs de $v_1$, $v_2$ et $d_{min}$, la condition suivante sur $\theta_2(0)$ :

$$\cos\theta_2(0) \leq 1 - \frac{1}{2{v_{min}}^2}\left(\frac{d_a + n_h}{t_f}\right)^2 \tag{5.14}$$

Ainsi, selon la valeur choisie pour $t_f$, certains cas de conflits ne seront pas admissibles.

À partir de la notion de *cas de conflit admissible*, nous allons maintenant introduire celle de *configuration de conflit*, qui permettra de tenir compte du fait que deux cas de conflit admissibles différents peuvent être équivalents dans le cadre de la génération de trajectoires d'évitement par un réseau de neurones. Les entrées du réseau de neurones sont calculées à partir des caps et des positions des avions relativement à la position courante et à la destination de l'avion courant, c'est-à-dire l'avion pour lequel le réseau de neurones doit fournir une modification de cap (voir section 5.3.3).

Ainsi, deux cas de conflit se déduisant l'un de l'autre par un déplacement du plan (translation ou rotation) mèneront, à chaque pas de temps, aux mêmes entrées (et donc aux même sorties) du réseau de neurones et il serait donc inutile de prendre en compte chacun de ces deux cas, tant lors de l'apprentissage que lors de l'évaluation des capacités de généralisation d'un réseau de neurones.

Les trajectoires d'évitement des deux avions sont générées par des réseaux de neurones identiques. Deux cas de conflit se déduisant l'un de l'autre en changeant la numérotation des avions (c'est-à-dire tels qu'on passe de l'un à l'autre si on échange les positions et les vecteurs vitesse des deux avions à un instant donné) sont donc équivalents.

**Définition 5.5 – Configuration de conflit –** *Une relation d'équivalence entre les cas de conflit admissibles est définie, dans laquelle deux cas de conflit admissibles sont équivalents s'ils se déduisent l'un de l'autre par un déplacement du plan ou par un changement de numérotation des avions. On appelle configurations de conflit les classes d'équivalence associées à cette relation d'équivalence.*

Dans la suite de ce chapitre, pour plus de simplicité, nous dirons *configuration de conflit* pour désigner un *cas de conflit admissible représentant une configuration de conflit*. Quand il y aura risque d'ambiguïté nous parlerons des différents *représentants* d'une configuration de conflit.

### 5.3.5 Signe d'une configuration de conflit

On définit la notion de symétrie sur les configurations de conflit :

**Définition 5.6 – Symétrie sur les configurations de conflit –** *On dira qu'une configuration de conflit $\mathcal{C}_1$ est la configuration symétrique d'une configuration $\mathcal{C}_2$ s'il existe un cas de conflit admissible représentant la configuration $\mathcal{C}_1$ et se déduisant par une symétrie axiale du plan d'un cas de conflit admissible représentant la configuration $\mathcal{C}_2$.*

D'après les définitions des cas et des configurations de conflit (définitions 5.3, 5.4 et 5.5), il est clair que :

- La configuration symétrique d'une configuration donnée est unique. Dans toute la suite, la configuration de conflit symétrique d'une configuration de conflit $\mathcal{C}$ sera notée $S(\mathcal{C})$.
- Etant donné une configuration de conflit $\mathcal{C}$, la configuration de conflit symétrique de $\mathcal{C}$ est la configuration correspondant à l'image par une symétrie axiale quelconque du plan de n'importe lequel des cas de conflit admissibles représentant la configuration de conflit $\mathcal{C}$.
- On a $S(\mathcal{C}) = \mathcal{C}$ dans deux cas, et uniquement dans ces deux cas :
  - Si les vitesses des deux avions sont égales.
  - Si les droites portant les trajectoires des deux avions sont confondues.

Si des trajectoires d'évitement sont fournies par le réseau de neurones pour une configuration donnée, on peut en déduire de manière immédiate des trajectoires d'évitement *symétriques* pour sa configuration symétrique, en changeant, à chaque pas de temps, le signe de la modification de cap de chacun des deux avions impliqués dans le conflit.

La notion de symétrie, appliquée aux configurations de conflit, permettra donc de réduire la taille de l'espace des configurations sur lequel doit être réalisé l'apprentissage des réseaux de neurones.

Deux cas de conflit se déduisant l'un de l'autre par symétrie ne sont pas considérées comme équivalents, contrairement à deux cas de conflit se déduisant l'un de l'autre par un déplacement du plan. C'est pourquoi on distingue une configuration de conflit de sa configuration symétrique. Cela est dû au fait qu'il n'y a pas de manière intrinsèque, canonique, de séparer l'espace des configurations de conflit en deux sous-espaces, les éléments de l'un étant les symétriques des éléments de l'autre. Ainsi, nous allons définir deux *signes* différents pour effectuer cette division de l'espace des configurations de conflit.

Nous verrons d'ailleurs que le choix de diviser cet espace pour utiliser la symétrie ou de ne le pas le diviser, et, si on le divise, le choix de l'un ou l'autre des *signes* pour le diviser, influenceront les résultats obtenus (voir sections 5.4).

**Définition 5.7 – Signe d'une configuration de conflit –** *On appellera signe d'une configuration de conflit une fonction* $Sg$ *de l'ensemble des configurations de conflit dans* $\{-1, 0, 1\}$ *telle que* $Sg(S(\mathcal{C})) = -Sg(\mathcal{C})$. *Pour une fonction signe donnée, on dira qu'une configuration de conflit est positive si* $Sg(\mathcal{C}) \geq 0$.

Notons que $S(\mathcal{C}) = \mathcal{C}$ implique $Sg(\mathcal{C}) = 0$, mais que l'implication inverse n'est pas vraie.

Un réseau de neurones permettant de générer des trajectoires d'évitement pour des configurations de conflit positives pourra être utilisé pour générer des trajectoires d'évitement pour des configurations de conflit quelconques : si une configuration de conflit n'est pas positive, on obtient des trajectoires d'évitement à partir des trajectoires d'évitement obtenues sur sa configuration symétrique. C'est de cette manière que nous procéderons. Nous allons définir dans cette section deux fonctions définissant un signe sur les configurations de conflit. Nous comparerons l'influence du choix de l'une ou de l'autre de ces fonctions pour l'apprentissage des réseaux de neurones sur l'efficacité et le comportement de ces réseaux dans la section 5.4.

En pratique, nous ne manipulerons pas de configuration de conflit au sens de la définition 5.5, mais des cas de conflit admissibles particuliers représentant ces configurations. Nous définirons donc des signes sur les cas de conflit admissibles, signes qui sont les extensions des signes sur les configurations de conflit : le signe d'un cas de conflit admissible sera le signe de la configuration de conflit correspondante. Il faudra donc vérifier que le signe d'une configuration de conflit est indépendant du représentant de cette configuration sur lequel il a été calculé.

Pour définir les fonctions signe, nous devons d'abord préciser ce que signifie, pour un avion, qu'être à un instant donné *à droite* ou *à gauche* d'un autre.

Considérons une configuration de conflit $\mathcal{C}$ et un instant $t$. Pour un représentant donné de cette configuration, nous notons $\vec{v}_1$ et $\vec{v}_2$ les vecteurs vitesse respectifs de $a_1$ et $a_2$ à l'instant $t$ et $\vec{x}_{1,2}$ le vecteur joignant la position de $a_1$ à celle de $a_2$ et $\beta_{1,2}$ l'angle entre les vecteurs $\vec{v}_1$ et $\vec{x}_{1,2}$ (voir figure 5.4).

On dira que l'avion $a_2$ est

- à gauche de l'avion $a_1$ si $\sin\beta_{1,2} > 0$ (figure 5.4 a),
- à droite de $a_1$, si $\sin\beta_{1,2} < 0$ (figure 5.4 b),

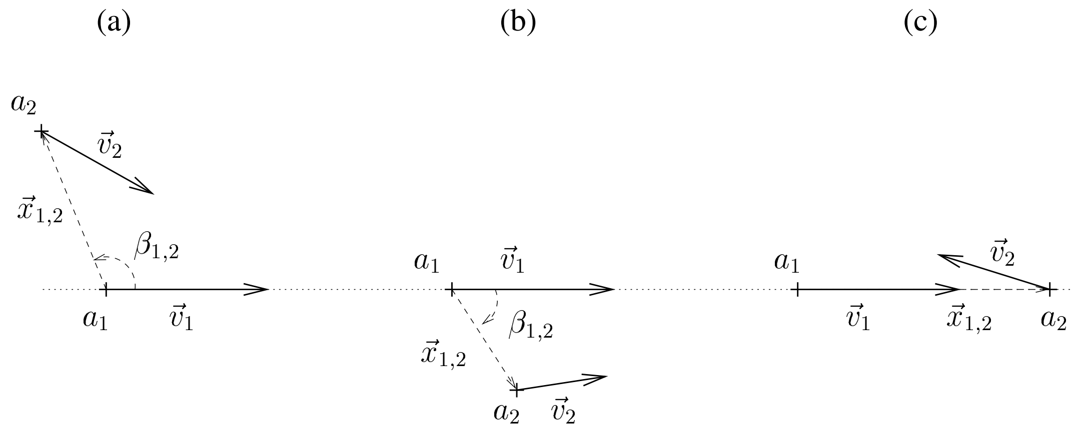



FIG. 5.4: *Position relative des avions*

- ni à droite ni à gauche de $a_1$ si $\sin\beta_{1,2} = 0$ (figure 5.4 c).

Le signe de $\sin\beta_{1,2}$ peut s'obtenir à partir du produit vectoriel de $\vec{v}_1$ par $\vec{x}_{1,2}$ : c'est le signe de $(\vec{v}_1 \wedge \vec{x}_{1,2}).\vec{e}_z$.

Les différents représentants d'une même configuration se déduisant les uns des autres par des déplacements du plan (laissant les angles invariants), ces définitions sont indépendantes du représentant de la configuration $\mathcal{C}$ utilisé.

Il faut noter que le fait que $a_1$ soit à gauche de $a_2$ n'entraîne pas forcément que $a_1$ soit à droite (ou à gauche) de $a_2$.

La première fonction signe que nous allons définir utilise la vitesse des avions. Elle est notée $Sg_v$. Pour une configuration de conflit $\mathcal{C}$, on calcule $Sg_v(\mathcal{C})$ de la manière suivante :

- si $v_1 = v_2$ alors $Sg_v(\mathcal{C}) = 0$ ;
- si $v_1 \neq v_2$, alors :
    - si l'avion le plus rapide est à droite de l'avion le plus lent à l'instant $t = 0$, alors $Sg_v(\mathcal{C}) = 1$,
    - si l'avion le plus rapide est à gauche de l'avion le plus lent à l'instant $t = 0$, alors $Sg_v(\mathcal{C}) = -1$,
    - si l'avion le plus rapide n'est ni à gauche ni à droite de l'avion le plus lent à l'instant $t = 0$, alors $Sg_v(\mathcal{C}) = 0$.

Cette définition de $Sg_v(\mathcal{C})$ ne dépend pas de la représentation de $\mathcal{C}$ utilisée, puisque le fait qu'un avion soit à droite, à gauche, ou ni à droite ni à gauche de l'autre n'en dépend pas. De plus, elle définit bien un signe au sens de la définition 5.7. En effet, on obtient un représentant $S(\mathcal{C})$ (la configuration symétrique de $\mathcal{C}$) à partir d'un représentant de $\mathcal{C}$ par une symétrie axiale du plan, qui inverse le signe des angles.

La seconde fonction signe que nous avons utilisée a une définition très simple, mais correspond à des caractéristiques plus complexes de la configuration de conflit considérée, caractéristiques qui sont liées à l'ordre de passage des avions au point d'intersection de leurs trajectoires. Cette fonction signe est notée $Sg_p$.

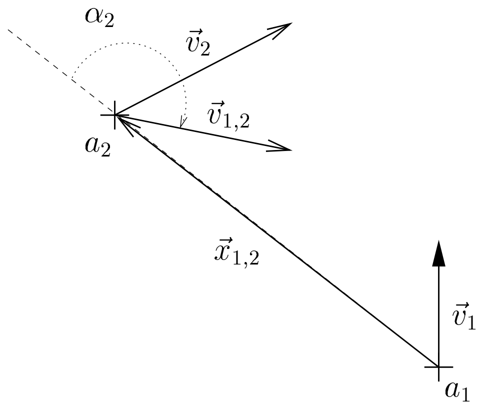



FIG. 5.5: *Vecteurs vitesse et position relative*

On considère un représentant d'une configuration $\mathcal{C}$. On se place à l'instant $t = 0$ (les avions sont alors distants l'un de l'autre de la distance d'alerte $d_a$). On note toujours $\vec{x}_{1,2}$ le vecteur joignant la position de $a_1$ au temps $t = 0$ à celle de $a_2$ (position relative de $a_2$ par rapport à $a_1$). On note de plus $\vec{v}_{1,2}$ le vecteur vitesse relative de l'avion $a_2$ par rapport à l'avion $a_1$ et $\alpha_{1,2}$ l'angle entre les vecteurs $\vec{x}_{1,2}$ et $\vec{v}_{1,2}$ (voir figure 5.5). On calcule la composante verticale du produit vectoriel de $\vec{x}_{1,2}$ par $\vec{v}_{1,2}$, que l'on note $p_{1,2}$ (les deux avions volant dans le même plan horizontal, ce produit vectoriel est réduit à sa composante verticale).

$$p_{1,2} = (\vec{x}_{1,2} \wedge \vec{v}_{1,2}).\vec{e}_z \tag{5.15}$$

On définit alors $Sg_p(\mathcal{C})$ de la manière suivante :

- Si $p_{1,2} > 0$, $Sg_p(\mathcal{C}) = 1$
- Si $p_{1,2} < 0$, $Sg_p(\mathcal{C}) = -1$
- Si $p_{1,2} = 0$, $Sg_p(\mathcal{C}) = 0$

Si on définit $\vec{x}_{2,1}$ le vecteur joignant la position de $a_2$ au temps $t = 0$ à celle de $a_1$, $\vec{v}_{2,1}$ le vecteur vitesse relative de $a_1$ par rapport à $a_2$, et $p_{2,1} = (\vec{x}_{2,1} \wedge \vec{v}_{2,1}).\vec{e}_z$, on a $\vec{x}_{2,1} = -\vec{x}_{1,2}$ et $\vec{v}_{2,1} = -\vec{v}_{1,2}$ et donc $p_{2,1} = p_{1,2}$.

Si pour calculer $Sg_p(\mathcal{C})$ on utilise $p_{i,j}$, on dira qu'on utilise l'avion $a_i$ comme référence pour le calcul de $Sg_p(\mathcal{C})$. Puisque $p_{2,1} = p_{1,2}$, la valeur de $Sg_p(\mathcal{C})$ est indépendante de l'avion utilisé comme référence. La définition de $Sg_p$ donnée ci-dessus est donc indépendante de la numérotation des avions.

Comme pour $Sg_v(\mathcal{C})$, $Sg_p(\mathcal{C})$, ainsi définie, est indépendante du représentant de $\mathcal{C}$ utilisé pour la calculer car les déplacements du plan conservent les angles. De même, le fait que $Sg_p$ définisse un signe sur les configurations de conflit est dû à ce que que les symétries axiales du plan inversent le signe des angles.

On sait qu'à l'instant $t = 0$ la distance entre les avions est égale à la distance d'alerte $d_a$, donc $\vec{x}_{i,j} \neq \vec{0}$. De plus les avions se rapprochent l'un de l'autre donc $\vec{v}_{i,j} \neq \vec{0}$. Donc, d'après la définition de $p_{i,j}$, le signe de $p_{i,j}$ est égal au signe de $\sin \alpha_{i,j}$ (angle entre $\vec{x}_{i,j}$, et $\vec{v}_{i,j}$). Cela implique que $Sg_p(\mathcal{C}) = 0$ si et seulement si les trajectoires des avions sont telles que, en l'absence de déviation, ils passeraient en même temps par le même point, ce qui signifie qu'il y aurait alors une collision.

Pour pouvoir interpréter $Sg_p(\mathcal{C})$, nous devons considérer l'ordre de passage des avions au point d'intersection de leurs trajectoires, en l'absence de déviation.

Les trajectoires non déviées des avions sont portées par des droites, puisque les caps sont constants. Une configuration de conflit $\mathcal{C}$ donnée correspond à l'un des trois cas de figure suivant :

1. Les vecteurs vitesse des avions sont colinéaires, et les droites portant leurs trajectoires sont parallèles ou confondues.
2. Les droites portant les trajectoires des avions se coupent, et en l'absence de déviation les deux avions atteindraient en même temps leur point d'intersection (il y aurait alors collision).
3. Ces droites se coupent, et les deux avions n'atteignent pas en même temps leur point d'intersection : on dira que l'avion qui atteint ce point le premier *passe devant* l'autre (l'autre avion passant *derrière* lui).

Nous allons donner une interprétation de $Sg_p(\mathcal{C})$ dans chacun des trois cas ci-dessus. Elle aura, comme nous le verrons dans la section 5.4 une signification particulièrement importante quand les trajectoires des deux avions se croisent et que les deux avions n'atteignent pas au même moment le point d'intersection (troisième des cas de figure ci-dessus) :

**Trajectoires parallèles :** supposons d'abord que les trajectoires des deux avions sont portées par des droites parallèles distinctes et que leurs vecteurs vitesse sont de sens contraires. Les trajectoires non déviées des avions n'entraînent pas de collision, donc $Sg_p(\mathcal{C})$ est non nul. De plus, on a, à partir de la définition de $p_{i,j}$, les résultats suivants :

- Si $Sg_p(\mathcal{C}) = -1$, chacun des deux avions est à droite de l'autre (figure 5.6 a).
- Si $Sg_p(\mathcal{C}) = 1$, chacun des deux avions est à gauche de l'autre (figure 5.6 b).

Si les trajectoires des avions sont portées par des droites parallèles, et que les vecteurs vitesse des deux avions sont de même sens, alors, puisque les avions s'approchent l'un de l'autre au temps $t = 0$, un des deux avions doit être plus rapide que l'autre, et on a :

- Si $Sg_p(\mathcal{C}) = -1$, l'avion le plus rapide est à droite de l'avion le plus lent (figure 5.6 d).
- Si $Sg_p(\mathcal{C}) = 1$, l'avion le plus rapide est à gauche de l'avion le plus lent (figure 5.6 c).

Si les avions volent sur la même droite, c'est-à-dire volent droit l'un sur l'autre, alors on a toujours $Sg_p(\mathcal{C}) = 0$.

**Collision :** si les avions, en l'absence de déviation, volent sur la même droites, c'est-à-dire volent droit l'un sur l'autre, ou s'ils volent sur des droites sécantes mais atteignent le point d'intersection au même moment, on a toujours $Sg_p(\mathcal{C}) = 0$

**Croisement sans collision :** on peut dans ce cas prendre comme avion de référence pour calculer $Sg_p(\mathcal{C})$ l'avion qui passe derrière l'autre, c'est-à-dire celui qui passe le deuxième au point d'intersection des trajectoires des deux avions. Rappelons que la valeur de $Sg_p(\mathcal{C})$ est indépendante de l'avion pris comme référence pour la calculer. Nous supposons que cet avion, qui passe derrière l'autre, est $a_1$. A partir de la définition de $p_{1,2}$ et de $\alpha_{1,2}$, on obtient dans ce cas une interprétation de $Sg_p(\mathcal{C})$. Nous donnons ici cette interprétation sans

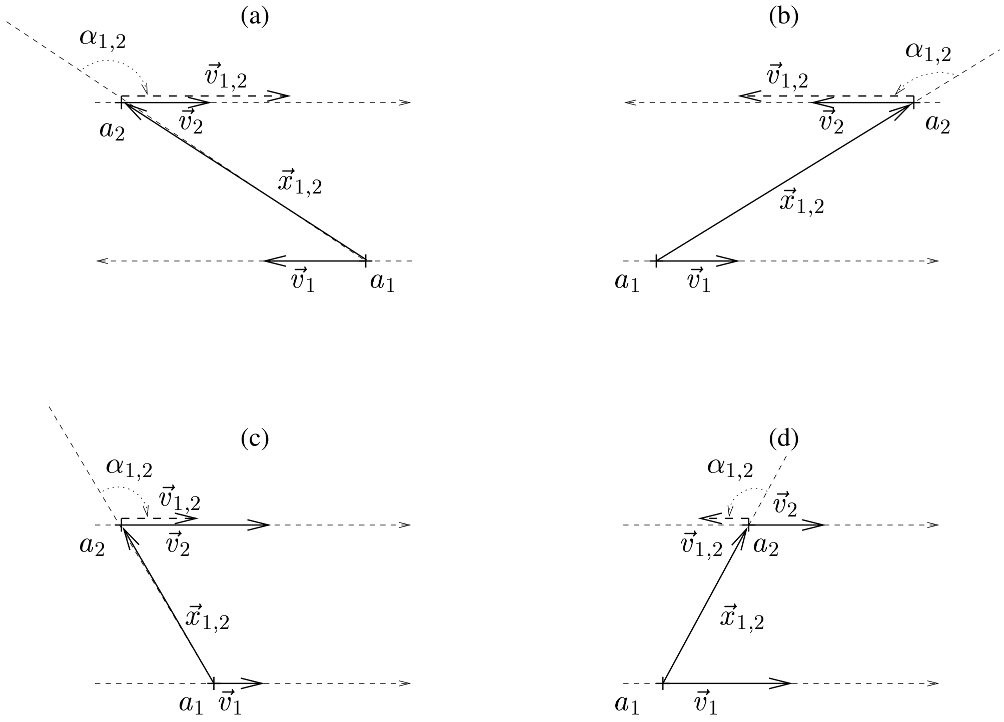



FIG. 5.6: *Trajectoires parallèles*

entrer dans les détails des calculs sur les angles qui permettent de l'obtenir. Ces calculs nécessitent de prendre en compte de nombreux cas différents, mais ne présentent aucune difficulté réelle.

Cette interprétation en terme d'ordre de passage des avions est illustrée par la figure 5.7. Dans chacun des quatre cas présentés, nous avons représenté les deux avions, à l'instant $t = 0$, dans le référentiel lié à l'avion $a_1$. La distance entre les deux avions est alors égale à $d_a$. Le vecteur position relative $\vec{x}_{1,2}$, le vecteur vitesse relative $\vec{v}_{1,2}$, et l'angle $\alpha_{1,2}$ sont représentés sur la figure 5.7. L'axe $a_1\vec{x}_1$ donne la direction du vecteur vitesse de $a_1$. Le trait en pointillé qui prolonge le vecteur vitesse relatif montre qu'il y aura conflit entre les avions (il coupe le cercle de rayon $n_h$) et que l'avion $a_2$ passe devant l'avion $a_1$ : dans le référentiel lié à l'avion $a_1$, la trajectoire de l'avion $a_2$, matérialisée par ce trait en pointillé coupe l'axe $a_1\vec{x}_1$ à gauche de l'avion $a_1$. Ce trait en pointillé indique aussi si l'avion $a_2$ coupera la trajectoire de $a_1$ après l'instant $t = 0$, ou s'il l'a coupée avant cet instant, d'après la position de cette intersection par rapport à celle de $a_2$ et au sens du vecteur $\vec{v}_{1,2}$.

- Si $Sg_p(\mathcal{C}) = -1$, alors trois cas sont possibles :

  1. L'avion qui passe devant (c'est-à-dire $a_2$) est à gauche de $a_1$ à $t = 0$ et atteindra le point d'intersection des trajectoires des avions après le temps $t = 0$ (figure 5.7 b),

  2. $a_2$ est à droite de $a_1$ à $t = 0$ mais est déjà passé, avant $t = 0$ au point d'intersection des trajectoires (figure 5.7 c), c'est-à-dire que si on considère les positions des avions avant que l'avion $a_2$ n'ait coupé la trajectoire de l'avion $a_1$, l'avion $a_2$ était alors à gauche de l'avion $a_1$.

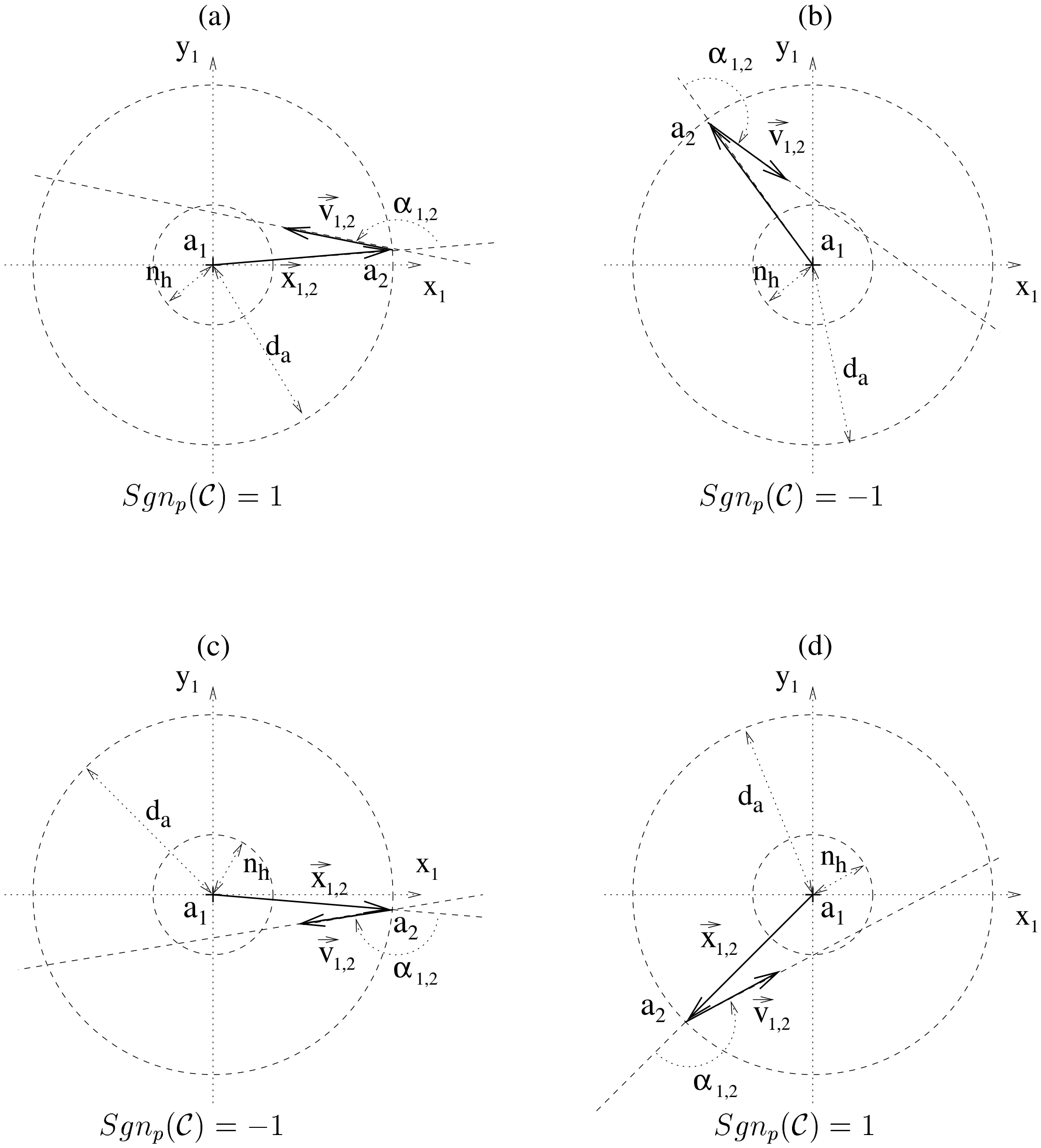



FIG. 5.7: *Signe d'une configuration basé sur l'ordre de passage*

3. Il se trouve en $t = 0$ au point d'intersection des trajectoires, il était avant $t = 0$ à gauche de l'avion $a_1$, et passe à droite.

On peut résumer ces trois cas en considérant que, indépendamment de la position des avions au temps $t = 0$, si $Sg_p(\mathcal{C}) = -1$, *l'avion qui passe devant l'autre vient de sa gauche*.

- Si $Sg_p(\mathcal{C}) = 1$, les trois cas symétriques sont alors possibles :
  1. $a_2$ à droite de $a_1$ à $t = 0$ et atteindra le point d'intersection des trajectoires des avions après le temps $t = 0$ (figure 5.7 d),
  2. $a_2$ est à gauche de $a_1$ à $t = 0$ mais est déjà passé, avant $t = 0$ au point d'intersection des trajectoires (figure 5.7 a).
  3. Il se trouve en $t = 0$ au point d'intersection des trajectoires, par rapport à $a_1$, il passe de droite à gauche.

  Ainsi, si $Sg_p(\mathcal{C}) = 1$, *l'avion qui passe devant l'autre vient de sa droite*.

En prenant comme référence l'avion qui passe devant l'autre pour le calcul de $Sg_p(\mathcal{C})$, on obtient la même valeur de $Sg_p(\mathcal{C})$, et de la même manière que précédemment, on obtient l'interprétation suivante de $Sg_p(\mathcal{C})$ :

- Si $Sg_p(\mathcal{C}) = -1$, l'avion qui passe derrière l'autre vient de la droite.
- Si $Sg_p(\mathcal{C}) = 1$, il vient de la gauche.

Nous avons réalisé l'apprentissage des réseaux de neurones sans utiliser de symétrie sur les configurations de conflit, c'est-à-dire en utilisant pour l'apprentissage des configurations de conflit de signes quelconques, puis en utilisant chacune des symétries définies par les fonctions signe $Sg_v$ et $Sg_p$, c'est-à-dire en utilisant exclusivement des configurations positives pour la fonction signe correspondante (définition 5.7).

Les réseaux dont l'apprentissage a été réalisé sur des configurations positives sont cependant testés sur des configurations de signes quelconques : dans le cas d'une configuration $\mathcal{C}$ non positive (de signe égal à -1), les modifications de caps à chaque pas de temps sont données par l'opposé de la modification de cap donnée par le réseau pour la configuration $S(\mathcal{C})$ (configuration symétrique de $\mathcal{C}$).

L'influence du signe choisi pour l'apprentissage sur l'efficacité et le comportement du réseau obtenu sera présentée dans les sections 5.4.2 et 5.4.3. Nous étudierons de manière plus qualitative l'influence de la fonction signe choisie sur le comportement du réseau dans la section 5.4.4, et nous verrons que choisir l'une ou l'autre de ces deux fonctions signe revient à aborder de deux manières différentes le problème de l'évitement.

### 5.3.6 Génération aléatoire de configurations de conflit

Nous présentons dans cette section la méthode que nous avons employée pour générer les configurations de conflit servant à l'apprentissage des réseaux de neurones.

Dans toute la suite, la valeur de $t_f$, dont dépendent les notions de cas de conflit admissible et de configuration de conflit sera fixée.

On génère aléatoirement des cas de conflit admissibles représentant des configurations de conflit qui seront utilisées pour l'apprentissage des réseaux de neurones. Nous cherchons à générer des cas de conflit admissibles tels qu'un réseau de neurones ayant réalisé son apprentissage sur eux soit à même de résoudre toutes les configurations de conflit auxquelles il sera confronté par la suite.

Cette capacité de généralisation du réseau sera vérifiée en testant le réseau de neurones obtenu à la fin de l'apprentissage sur un très grand nombre de configurations de conflit, elles aussi générées aléatoirement. La qualité du réseau de neurones, en termes de capacité de généralisation dépendra, comme nous le verrons dans la section 5.4, de la méthode d'apprentissage utilisée.

Les cas de conflit générés sont considérés comme des représentants de configurations de conflit et on ne cherche à générer qu'un seul représentant pour une configuration de conflit donnée. Les positions et les caps absolus des avions n'ont aucune importance. Nous cherchons donc à générer aléatoirement, pour une position et un cap arbitraire pour l'avion $a_1$ au temps $t = t_c$, qui sera l'instant auquel le conflit débutera entre les deux avions :

- La vitesse $v_1$ de l'avion $a_1$.
- La vitesse $v_2$ de l'avion $a_2$.
- Le gisement $\gamma_2(0)$ de l'avion $a_2$ par rapport à l'avion $a_1$.
- L'angle $\theta_2(0)$ entre $\vec{v}_1(0)$ et $\vec{v}_2(0)$, les vecteurs vitesse des avions, au temps $t = 0$.

Ces différents paramètres devront être tels qu'ils définissent un cas de conflit admissible, c'est-à-dire tels qu'ils mènent à un conflit entre les deux avions entre les temps $t = 0$ et $t = t_f$, et tels que les avions soient de nouveau séparés au temps $t = t_f$.

Nous considérerons que ces paramètres définissent une configuration de conflit, c'est-à-dire que si on considère deux tels cas de conflit, ils représentent deux configurations de conflit différentes si et seulement si un au moins des paramètres ci-dessus est différent d'un de ces deux cas de conflit à l'autre.

En toute rigueur, cela n'est pas vrai. Si pour deux cas de conflit admissibles, tous ces paramètres sont égaux, ils sont bien les représentants de la même configuration de conflit. Cependant, deux cas de conflit ayant les mêmes valeurs pour $v_1$ et $v_2$ mais des valeurs opposées pour $\gamma_2(0)$ et $\theta_2(0)$ représentent deux configurations de conflit symétriques. Ils représentent donc la même configuration $\mathcal{C}$ dans les cas où on a $\mathcal{C} = S(\mathcal{C})$, c'est-à-dire (voir section 5.7) :

- Si les vitesses des deux avions sont égales.
- Si les trajectoires des deux avions sont portées par deux droites confondues, c'est-à-dire si $\gamma_2(0) = 180$ et $\theta_2(0) = 0$ ou si $\gamma_2(0) = 0$ et $\theta_2(0) = 0$.

Cependant les valeurs des vitesses des avions, de $\theta_2(0)$ de $\gamma_2(0)$ sont générées aléatoirement. On peut donc considérer que les cas d'égalité ci-dessus ne se produiront jamais, ou du moins suffisamment rarement pour ne pas entraîner de redondance gênante dans les configurations utilisées pour l'apprentissage des réseaux.

Nous générons des cas de conflit admissibles de manière à ce que les configurations de conflit correspondantes permettent de couvrir l'ensemble des configurations de conflit (définition 5.5), au moment de l'utilisation du réseau. Cependant, il n'est pas nécessaire de couvrir la totalité de cet ensemble au moment de l'apprentissage : nous pourrons en effet utiliser la symétrie sur les configurations de conflit (définition 5.6). Si on souhaite l'utiliser, une fois choisie une des deux fonctions signe définies dans la section précédente ($Sg_v(\mathcal{C})$ ou $Sg_p(\mathcal{C})$), nous ne souhaitons générer que des configurations de conflits positives c'est-à-dire dont le signe est égal à 0 ou à 1 (définition 5.7). Pour cela nous procéderons simplement de la manière suivante : une configuration de conflit sera générée sans tenir compte de son signe, et si elle n'est pas positive, c'est sa configuration symétrique qui sera utilisée pour l'apprentissage.

On se place donc dans un repère orthonormé $(O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$. Les avions volent dans le plan $(\vec{e}_x, \vec{e}_y)$.

Pour générer une configuration de conflit, nous suivrons les étapes ci-dessous, après avoir fixé la valeur de $t_f$ :

1. On place $a_1$ en $(0, 0)$, avec le cap 0. On note $t_c$ l'instant correspondant à cette position de l'avion $a_1$.
2. On choisit aléatoirement les vitesses $v_1$ et $v_2$ des avions et une valeur pour $\alpha_2$, le cap de l'avion $a_2$.
3. À l'instant $t_c$, l'avion $a_2$ sera placé à une distance de l'avion $a_1$ égale à la norme de séparation $n_h$. On tire aléatoirement une valeur du gisement $\gamma$ de l'avion $a_2$ par rapport à l'avion $a_1$, de manière à ce que l'instant $t_c$ corresponde au début d'un conflit, et que le cas de conflit ainsi généré soit admissible.
4. On calcule (à rebours) la position des avions à l'instant, avant le conflit, où la distance entre eux est égale à $d_a$. L'origine des temps est fixée à cet instant, ce qui donne la valeur de $t_c$. Le cas de conflit généré est défini par la position à l'instant $t = 0$, la vitesse et le cap de chacun des deux avions.
5. On calcule, à partir des positions et des vecteurs vitesse des avions au temps $t = 0$, le signe de la configuration correspondant au cas de conflit que l'on a généré. Si cette configuration de conflit n'est pas positive (signe égal à -1), on prend le symétrique de ce cas de conflit autour de l'axe portant la trajectoire de l'avion $a_1$.

Nous allons développer l'étape 3 de la méthode décrite ci-dessus. Nous cherchons des critères pour que, pour des valeurs données de $v_1$, $v_2$ et $\alpha_2$, une valeur de $\gamma$ soit telle que le temps $t_c$ corresponde au début d'un conflit entre les avions, et que le cas de conflit généré soit admissible.

On se place dans le référentiel lié à l'avion $a_1$. On note, dans ce référentiel, $\mathcal{R}_1 = (a_1, \vec{e}_{x_1}, \vec{e}_{y_1})$ un repère du plan dans lequel volent les avions tel que $\vec{v}_1 = v_1 \vec{e}_{x_1}$.

La position au temps $t = t_c$ de l'avion $a_2$, dans ce repère, est donnée par le point de coordonnées $(n_h \cos\gamma, n_h \sin\gamma)$ et son vecteur vitesse est $\vec{v}_2 = (v_2 \cos\alpha_2)\vec{e}_{x_1} + (v_2 \sin\alpha_2)\vec{e}_{y_1}$.

On note $\vec{x}_r = (n_h \cos\gamma)\vec{e}_{x_1} + (n_h \sin\gamma)\vec{e}_{y_1}$ le vecteur donnant la position relative de $a_2$ par rapport à $a_1$, et $\vec{v}_r = (v_2 \cos\alpha_2 - v_1)\vec{e}_{x_1} + (v_2 \sin\alpha_2)\vec{e}_{y_1}$ le vecteur vitesse relative de $a_2$ par rapport à $a_1$ (voir figure 5.8).

Pour ces valeurs de $\alpha_2$, $v_1$, $v_2$ et $\gamma$, le temps $t_c$ correspond au début du conflit si et seulement si la distance entre les avions décroît (puisqu'elle est à cet instant égale à la norme de séparation), c'est-à-dire si et seulement si :

$$\vec{v}_r . \vec{x}_r < 0 \tag{5.16}$$

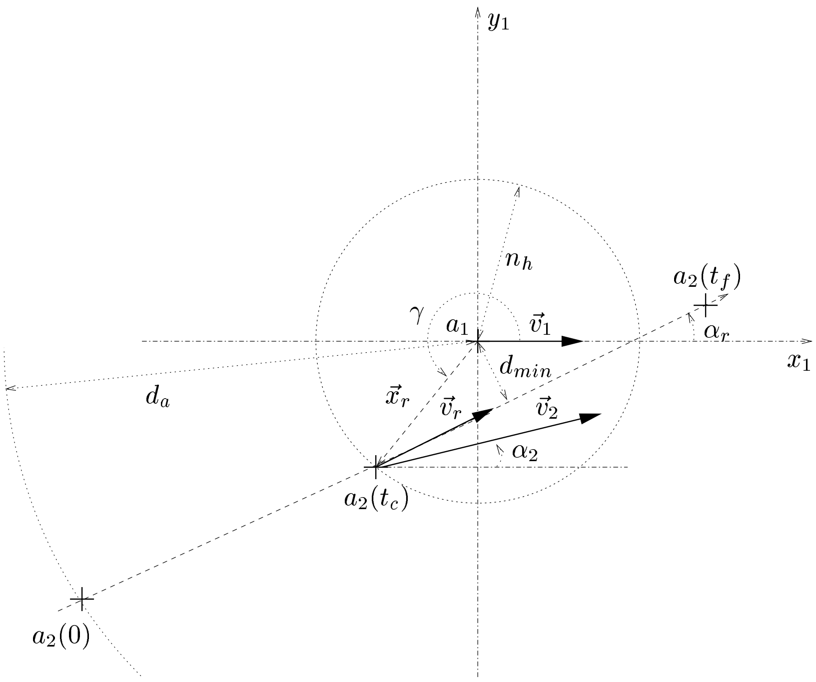



FIG. 5.8: *Génération aléatoire d'une configuration de conflit*

Ce qui donne :

$$v_2 \cos \alpha_2 \cos \gamma + v_2 \sin \alpha_2 \sin \gamma - v_1 \cos \gamma \leq 0 \tag{5.17}$$

C'est-à-dire :

$$\cos(\gamma - \alpha_2) \leq \frac{v_1}{v_2} \cos \gamma \tag{5.18}$$

On définit $t_s(C_c)$ le temps de sortie du conflit, c'est-à-dire le temps qui s'écoule entre le moment précédent le conflit où la distance entre les deux avions est égale à $d_a$ et le moment où le conflit entre les avions prend fin (voir section 5.3.4). Le conflit généré est admissible si et seulement si on a $t_s \leq t_f$.

On a l'équation 5.9 de la section 5.3.4 :

$$t_s = \frac{\sqrt{{d_a}^2 - d_{min}^2} + \sqrt{n_h^2 - d_{min}^2}}{\sqrt{v_2^2 + v_1^2 - 2v_1 v_2 \cos \theta_2(0)}} \tag{5.19}$$

où $\theta_2(0)$ est l'angle entre $\vec{v}_1$ et $\vec{v}_2$, les vecteurs vitesse des avions, au temps $t = 0$ et $d_{min}$ est la distance minimale entre les avions, en l'absence de déviation.

Comme le cap de l'avion $a_1$ est égal à 0, on a $\theta_2(0) = \alpha_2$ . De plus $d_{min} = |\cos(\alpha_r - \gamma)|$ , où $\alpha_r$ est l'angle entre le vecteur vitesse de l'avion courant $\vec{v}_1$ et le vecteur vitesse relatif de l'intrus

$\vec{v}_r$ (voir figure 5.8). La valeur de $\alpha_r$ se déduit donc des valeurs de $v_1$, $v_2$, $\alpha_2$ et $\gamma$. L'équation 5.19 donne ainsi directement un critère permettant de savoir si les valeurs de $\alpha_2$, $v_1$, $v_2$ et $\gamma$ choisie définissent un cas de conflit admissible.

En pratique, on tirera successivement des valeurs de $\gamma$ jusqu'à en obtenir une qui satisfasse ce critère. La valeur de $t_f$ que nous utiliserons sera suffisamment grande pour qu'on obtienne une telle valeur de $\gamma$ en un nombre limité de tirages aléatoires.

La position des avions à l'instant précédant le conflit où ils sont distants de $d_a$ est facile à déterminer : on connaît les positions et les vecteurs vitesse des avions au temps $t_c$, on note $\tau = t_c - t$, et $d(\tau)$ la distance entre les deux avions à l'instant $t = t_c - \tau$. La valeur de $t_c$ pour que les avions soient distants de $d_a$ au temps $t = 0$ est la racine positive de l'équation du second degré :

$$d(\tau)^2 - d_a^2 = 0 \tag{5.20}$$

La condition 5.16 assure l'existence d'une telle racine. Une fois la valeur de $t_c$ connue, on obtient, à partir de leurs positions en $t = t_c$, les positions initiales des avions $a_1$ et $a_2$, respectivement notées $(x_{01}, y_{01})$ et $(x_{02}, y_{02})$ :

$$\begin{cases} x_{01} = -v_1 \; t_c \\ y_{01} = 0 \end{cases} \tag{5.21}$$

et

$$\begin{cases} x_{02} = n_h \cos\gamma - (v_2 \cos\alpha_2) \; t_c \\ y_{02} = n_h \sin\gamma - (v_2 \sin\alpha_2) \; t_c \end{cases} \tag{5.22}$$

On notera que les valeurs de l'angle $\alpha_2$ et des vitesses $v_1$ et $v_2$ sont choisies d'abord et que, au cas où la valeur de $\gamma$ tirée aléatoirement est telle que le temps $t_c$ ne corresponde pas au début d'un conflit, ou que le cas de conflit généré ne soit pas admissible, on retire une valeur de $\gamma$ sans modifier les valeurs de $v_1$, $v_2$ et $\alpha_2$.

Ceci permet d'introduire des biais dans la répartition des valeurs de $\alpha_2$ et des vitesses des avions. En effet, les premiers essais que nous avons effectués ont montré que les configurations de conflit correspondant à des angles faibles entre les trajectoires des avions ou à des écarts faibles entre leurs vitesses étaient moins facilement résolues par les réseaux de neurones.

En générant plus de configurations de conflit, pour l'apprentissage des réseaux, correspondant à de tels cas, on obtient de meilleurs résultats en terme de capacités de généralisation des réseaux de neurones, c'est-à-dire que la proportion de configurations de conflit sur lesquelles l'apprentissage n'a pas été réalisé mais pour lesquelles le réseau de neurones génère des trajectoire sans conflit est plus grande.

La méthode que nous avons présentée dans cette section ne prétend pas générer des configurations de conflit uniformément réparties sur l'ensemble des configurations de conflit (au sens de la définition 5.5). Il faudrait d'ailleurs pour cela définir tout d'abord ce qu'on entend par *uniformément réparties sur l'ensemble des configurations de conflit*. Dans le cadre d'une éventuelle application au trafic réel de méthodes telles que celles qui sont présentées ici, chercher à définir, puis à obtenir une telle répartition uniforme des configurations de conflit n'aurait aucun sens. Ce qui serait très intéressant serait d'obtenir, pour tester le réseau de neurones obtenu, une répartition des configurations de conflit (en termes de vitesses et caps relatifs des avions, d'angles d'incidence des trajectoires, de distance minimale entre les avions en l'absence de déviation) identique à celle observée sur le trafic aérien réel dans les zones dans lesquelles cette technique serait susceptible d'être utilisée. Cela nécessiterait une étude statistique des cas de conflits réels qui dépasse le cadre du travail présenté ici.

Nous ne cherchons donc qu'à obtenir une répartition suffisamment significative des configurations de conflit qui seront générées aléatoirement. Nous sommes conscient que la méthode présentée ici introduit des biais dans la répartition de ces configurations (en plus des biais sur les vitesses et les angles $\alpha_2$ que nous introduisons volontairement). Nous n'avons pas cherché à les supprimer, pour les raisons que nous venons d'exposer.

Les configurations de conflit que nous utilisons ont été générées avec les valeurs numériques suivantes :

$$\begin{array}{lcl} n_h & = & 4\,NM \\ d_a & = & 16\,NM \\ v_{min} & = & 300\,kt \\ v_{max} & = & 500\,kt \\ t_f & = & 20\,mn \end{array}$$

La valeur de $t_f$ choisie peut sembler grande, mais elle doit l'être, pour permettre à une grande partie des cas de conflit d'être admissibles.

Ainsi, d'après la condition 5.12, un cas de conflit est admissible, quelles que soient les valeurs de $\theta_2(O)$ et $d_{min}(C_c)$, dès que l'on a $|v_2 - v_1| \leq 60\ kt$.

Et d'après la condition 5.14, pour qu'un cas de conflit soit admissible quelles que soient les valeurs de $\theta_2(O)$ et $d_{min}(C_c)$, $\alpha_2$ doit être supérieur à 11,5 degrés.

### 5.3.7 Évaluation des réseaux

Dans le cadre de l'apprentissage des réseaux de neurones par un algorithme génétique, une fonction d'adaptation doit être calculée pour chacun des réseaux présents dans la population (voir chapitre 4).

Chacune de ces évaluations sera réalisée au moyen d'une *base d'apprentissage*.

**Définition 5.8 – Base d'apprentissage –** *On appelle base d'apprentissage l'ensemble des configurations de conflit qui sont utilisées lors du calcul de la fonction d'adaptation d'un réseau de neurones.*

*On dira que l'apprentissage est réalisé sur une base d'apprentissage fixe quand les configurations de conflit présentes dans la base d'apprentissage sont les mêmes tout au long de l'évolution des réseaux.*

*On dira que l'apprentissage est réalisé sur une base d'apprentissage renouvelée, si, à chaque* *nouvelle génération de la population des réseaux, les configurations composant la base d'apprentissage sont de nouveau générées aléatoirement.*

Nous allons décrire dans cette section deux manières d'évaluer un réseau de neurones : l'une correspondant à une base d'apprentissage fixe, l'autre à une base renouvelée. Nous avons utilisé une base d'apprentissage renouvelée après avoir constaté que l'utilisation d'une base fixe ne menait pas à des résultats totalement satisfaisants en terme de capacités de généralisation des réseaux de neurones obtenus, comme nous le verrons dans les sections 5.4.2 et 5.4.3.

On note $N_a$ le nombre de configurations de conflit contenues dans la base d'apprentissage. Même dans le cas d'une base d'apprentissage renouvelée, le nombre de configurations de conflit contenues dans la base d'apprentissage est fixe. Les différentes configurations de cette base d'apprentissage sont notées $\mathcal{C}_i$, pour $i$ allant de 0 à $N_a - 1$.

Nous cherchons tout d'abord à évaluer la qualité du réseau de neurones sur une configuration $\mathcal{C}_i$ de la base d'apprentissage.

Considérons donc une configuration $\mathcal{C}_i$ et un réseau $\mathcal{R}$.

La qualité de la réponse apportée par le réseau de neurones au problème de l'évitement ne peut pas être évaluée à chaque calcul de la sortie du réseau pour un vecteur d'entrées donné (c'est-à-dire à chacun des pas de temps). En effet, la sortie du réseau permet de calculer une modification de cap pour l'avion courant (voir section 5.3.2). Nous ne disposons pas de moyen de juger directement, pour un pas de temps donné, de la qualité de la modification de cap obtenue. On ne peut en juger qu'une fois que tous les avions impliqués dans le conflit ont atteint leurs points de destination. Alors, on peut évaluer la qualité du réseau $\mathcal{R}$ sur la configuration $\mathcal{C}_i$ selon deux critères :

- Le respect des contraintes de séparation.
- L'allongement des trajectoires générées au moyen du réseau de neurones, par rapport aux trajectoires originelles des avions.

L'évaluation d'un réseau de neurones sur une configuration de conflit se fera donc en utilisant le réseau de neurones pour générer la trajectoire de chacun des deux avions et en évaluant la *qualité* des trajectoires obtenues.

On définit un entier noté $N_v(\mathcal{R}, \mathcal{C}_i)$, ou $N_v$ s'il n'y a pas de risque d'ambiguité sur le réseau et la configuration correspondants, d'abord initialisé à 0, qui comptera le nombre de pas de temps sur lesquels les contraintes de séparation sont violées, et un réel, noté $r(\mathcal{R}, \mathcal{C}_i)$, ou $r$, qui correspondra à la somme des retards des deux avions.

Les avions seront notés $a_j$, pour $j \in \{1, 2\}$. Leurs positions à un instant $t$ seront notées $p_j(t) = (x_j(t), y_j(t))$, leurs caps $c_j(t)$, et la distance entre eux sera notée $d(t)$, la longueur de leur trajectoire entre l'origine des temps et le pas de temps courant est notée $lg(t)$.

On calcule les valeurs de $(\mathcal{R}, \mathcal{C}_i)$ en procédant de la manière suivante :

1. *Les avions sont placés aux positions initiales données par la configuration $\mathcal{C}_i$, avec les caps et vitesses correspondants.*

   $t \longleftarrow 0$

   $N_v \longleftarrow 0$

   $r \longleftarrow 0$

2. *De $k = 0$ à $k = N_t$ (c'est-à-dire pour chaque pas de temps), on suit les étapes suivantes :*

(a) *On vérifie que si les avions suivaient à partir de leurs positions courantes des routes directes vers leurs destinations, il y aurait conflit entre eux. Si ce n'est pas le cas, on passe directement à l'étape 3.*

(b) *Si les avions sont en conflit au pas de temps courant, c'est-à-dire si la distance entre eux est inférieure à la norme de séparation $n_h$, on incrémente $N_v$ :*

$$N_v \longleftarrow N_v + 1$$

(c) *Pour chaque avion $a_j$, on calcule les entrées du réseau de neurones correspondant à la situation (voir section 5.3.3), et on obtient, en appliquant ces entrées, une modification de cap $\delta_{c_j}(t)$.*

(d) *On met à jour les caps et les positions des avions (correspondant au pas de temps suivant).*

$$x_j(t + l_t) \longleftarrow x_j(t) + l_t \ \cos(c_j(t) + \delta_{c_j}(t))$$
$$y_j(t + l_t) \longleftarrow y_j(t) + l_t \ \sin(c_j(t) + \delta_{c_j}(t))$$
$$c_j(t + l_t) \longleftarrow c_j(t) + \delta_{c_j}(t)$$

(e) *on avance d'un pas de temps :*

$$t \longleftarrow t + l_t$$

3. *Si les avions peuvent rejoindre leurs destinations sans conflit, on calcule les longueurs totales de leurs trajectoires entre leurs positions initiales et leurs destinations, en tenant compte des déviations dues aux étapes précédentes, et on en déduit $r(\mathcal{R}, \mathcal{C}_i)$ la somme des retards entraînés par ces déviations.*
4. *Si on a atteint le temps $t = t_f$ et que les avions ne peuvent toujours pas rejoindre leurs destinations sans conflit, on considère que l'évitement est un échec, et on ajoute une valeur de pénalisation au compteur des violations $N_v$. On prend, pour cette valeur de pénalisation le nombre total de pas de temps $N_t$ : c'est le nombre maximal possible de pas de temps sur lesquels les avions pourraient être en conflit.*

On définit $N_{vt}(\mathcal{R})$, la somme sur les configurations de la base d'apprentissage des nombres de pas de temps sur lesquels les avions n'ont pas été séparés :

$$N_{vt}(\mathcal{R}) = \sum_{i=1}^{N_a} N_v(\mathcal{R}, \mathcal{C}_i) \tag{5.23}$$

Et $r_t(\mathcal{R})$, la moyenne quadratique, sur ces configurations, des retards des avions :

$$r_t(\mathcal{R}) = \sqrt{\frac{\sum_{i=1}^{N_a} r(\mathcal{R}, \mathcal{C}_i)^2}{N_a}} \tag{5.24}$$

Dans le cas d'une évaluation sur une base d'apprentissage fixe, la fonction d'adaptation du réseau $\mathcal{R}$ est calculée directement à partir des valeurs des $N_v(\mathcal{R}, \mathcal{C}_i)$ et des $r(\mathcal{R}, \mathcal{C}_i)$. Cette fonction d'adaptation, notée $f_a(\mathcal{R})$, doit être telle que :

- Si $N_v \neq 0$, c'est-à-dire si sur une au moins des configurations de conflit présentes dans la base d'apprentissage, le réseau de neurones n'a pas permis aux avions d'atteindre leurs destinations sans conflit, la valeur de $f_a$ sur $\mathcal{R}$ doit être d'autant plus petite que le nombre total de pas de temps sur lesquels il y a eu conflit est grand, c'est-à-dire d'autant plus petite que $N_{vt}(\mathcal{R})$ est grand.

- Si $N_v = 0$, le réseau $\mathcal{R}$ a généré des trajectoires sans conflit pour toutes les configurations de conflit présentes dans la base d'apprentissage. La valeur de $f_a$ sur $\mathcal{R}$ doit alors être d'autant plus grande que les retards entraînés par les trajectoires générées par $\mathcal{R}$ sont faibles.
- La valeur de la fonction $f_a$ doit être toujours plus grande sur un réseau pour lequel $N_{vt} = 0$ (ayant généré des trajectoires sans conflit pour toutes les configurations de la base d'apprentissage) que pour un réseau pour lequel $N_{vt} \neq 0$ (ayant entraîné au moins un conflit).

Nous définissons donc $f_a(\mathcal{R})$ de deux manières différentes, selon que $\mathcal{R}$ génère ou non des trajectoires sans conflit pour toutes les configurations de la base d'apprentissage :

**Si $\boldsymbol{N_{vt}(\mathcal{R}) \neq 0}$ :**

$$f_a(\mathcal{R}) = \frac{1000}{1 + N_{vt}(\mathcal{R})} \tag{5.25}$$

**Si $\boldsymbol{N_{vt}(\mathcal{R}) = 0}$ :**

$$f_a(\mathcal{R}) = 1000 + \frac{1000}{1 + r_t(\mathcal{R})} \tag{5.26}$$

L'intérêt d'utiliser, dans le calcul de $f_a$, la moyenne quadratique, sur les configurations, des sommes des retards des deux avions, est de pénaliser les réseaux qui pourraient obtenir de bons résultats sur une grande majorité des configurations de conflit, mais des résultats très mauvais sur quelques unes d'entre elles.

Comme nous le verrons dans la section 5.4.2, cette fonction d'adaptation permet d'obtenir des réseaux de neurones qui obtiennent de bons résultats sur la base d'apprentissage, mais dont les capacités de généralisation laissent à désirer.

C'est pourquoi nous avons utilisé une base renouvelée. Le principe de la base d'apprentissage renouvelée est, comme nous l'avons vu (définition 5.8), d'utiliser, à chaque génération de la population de réseaux de neurones, de nouvelles configurations de conflit, à chaque fois générées aléatoirement.

Cependant, l'utilisation de la même fonction d'adaptation que celle qui est utilisée avec une base fixe mène à des résultats décevants : on constate alors une très grande instabilité de la population, c'est-à-dire que des réseaux de neurones ayant obtenus de très bons résultats pendant une ou plusieurs générations, disparaissaient de la population, pour avoir obtenu des résultats moins bons lors d'une génération ultérieure. Cette instabilité demeure, même si on ne renouvelle à chaque pas de temps qu'une fraction (40 %) des configurations de conflit de la base d'apprentissage. Le problème qui se pose, en fait, peut s'exprimer de la manière suivante : à une génération donnée, un réseau $\mathcal{R}_{j_1}$ a une meilleure fonction d'adaptation qu'un réseau $\mathcal{R}_{j_0}$ s'il obtient de meilleurs résultats sur les $N_a$ configurations de conflit actuellement présentes dans la base d'apprentissage. Cependant le réseau $\mathcal{R}_{j_0}$ peut avoir obtenu de bons résultats sur de nombreuses générations, c'est-à-dire avoir fourni des trajectoires sans conflit pour un nombre de configurations de conflit bien supérieur à $N_a$. Si le réseau $\mathcal{R}_{j_1}$ a obtenu de bons résultats sur les configurations de la base actuelle, mais n'a pas *fait ses preuves* sur un aussi grand nombre de bases d'apprentissage que le réseau $\mathcal{R}_{j_0}$, on peut considérer que ce dernier est plus *fiable*. Par *fiabilité* d'un réseau, on entend le fait d'être à même de proposer des trajectoires sans conflit pour la plus grande proportion possible des configurations de conflit qui lui seront proposées.

L'instabilité due à l'utilisation de bases d'apprentissage renouvelées est souvent combattue au moyen de techniques telles que la *relaxation* : on définit un coefficient de relaxation $\alpha_r$, et la fonction d'adaptation utilisée à la génération $k$, $f_r^{(k)}$, s'obtient à partir de la fonction d'adaptation calculée sur la base d'apprentissage courante, $f_a$, et de la valeur de $f_r^{(k-1)}$ de la manière suivante :

$$f_r^{(k)}(\mathcal{R}) = \alpha.f_r^{(k-1)}(\mathcal{R}) + (1-\alpha).f_a(\mathcal{R}) \tag{5.27}$$

Cependant la relaxation utilisée telle quelle ne correspond pas parfaitement à cette *fiabilité* que nous recherchons pour les réseaux obtenus.

Considérons de nouveau les deux réseaux, $\mathcal{R}_{j_0}$ et $\mathcal{R}_{j_1}$, ayant tous deux généré des trajectoires sans conflit sur les $N_a$ configurations de conflit présentes dans la base d'apprentissage au moment de l'évaluation de $\mathcal{R}_{j_0}$ et $\mathcal{R}_{j_1}$. Sur ces $N_a$ configurations d'apprentissage, les deux réseaux, du point de vue de la satisfaction des contraintes, auront eu 100 % de réussite. Supposons cependant que $\mathcal{R}_{j_0}$ ait été évalué au cours de $n_{j_0}$ générations, avec $n_{j_0} > 1$, en générant toujours des trajectoires sans conflit, alors que c'est la première fois que le réseau $\mathcal{R}_{j_1}$ est évalué. Ce taux de réussite de 100 %, obtenu par $\mathcal{R}_{j_0}$ sur $n_{j_0} N_a$ configurations est plus significatif que celui de $\mathcal{R}_{j_1}$ qui a été obtenu sur seulement $N_a$ configurations.

Dans [AD95], Jean-Marc Alliot décrit une situation comparable. Il s'agit d'améliorer, au moyen d'un algorithme génétique, un programme jouant à Othello. Le programme utilise un tableau de coefficients, dont les valeurs sont établies au moyen d'un algorithme génétique. L'algorithme génétique manipule une population de vecteurs réels, correspondant à différents tableaux de coefficients.

Ces tableaux de coefficients sont évalués en confrontant le programme qui les utilise à un programme de référence. L'adaptation d'un tableau de coefficients est donnée par la proportion de parties remportées sur le programme de référence à partir de différentes situations de départ générées aléatoirement, et sous différentes conditions. Dans ce cas encore, la fiabilité d'une telle évaluation dépend du nombre de confrontations subies par l'élément que l'on cherche à évaluer. Il est plus facile, par exemple, de remporter, par chance, 12 victoires sur 12 parties que 46 victoires sur 48 parties. Pour tenir compte de ce facteur, Jean-Marc Alliot introduit la notion de *fiabilité*.

Si on suppose que la probabilité de gagner une partie est $p$, la probabilité de remporter $m$ parties sur $n$ est donnée par :

$$P(p,m,n) = \binom{n}{m} p^m (1-p)^{(n-m)} \tag{5.28}$$

On cherche maintenant une estimation de la probabilité $p$, à partir du nombre de jeux remportés ($m$), sur le nombre de jeux joués ($n$). On fixe un taux de confiance $p_f \in [0,1]$. On montre qu'alors on peut obtenir une expression implicite d'une valeur $p_{m,n,p_f}$ telle que la probabilité que l'on ait $p \geq p_{m,n,p_f}$ soit égale à $p_f \in [0,1]$ :

$$\int_{p_{m,n,p_f}}^{1} P(p,m,n)dp = \frac{p_f}{n+1} \tag{5.29}$$

En fixant $p_f$ à 0,95, il définit ainsi la fonction d'adaptation d'un tableau de coefficients ayant permis de remporter $m$ parties sur $n$, qui est prise égale à $p_{m,n,p_f}$. Les valeurs de $p_{m,n,p_f}$ pour les différentes valeurs possibles de $n$ ayant été préalablement calculées et stockées dans un tableau.

Nous avons repris cette idée d'utiliser un indicateur de fiabilité des résultats obtenus, en fonction du nombre de configurations sur lesquelles les résultats ont été obtenus.

Nous avons fait, entre le problème de génération des trajectoires d'évitement par réseau de neurones et le problème de l'apprentissage des coefficients d'un programme d'Othello l'analogie suivante : l'équivalent d'une partie est la confrontation d'un réseau de neurones aux $N_a$ configurations de conflit présentes dans la base d'apprentissage au moment d'une évaluation. Remporter une partie correspond à fournir pour toutes les configurations présentes dans la base d'apprentissage des trajectoires sans conflit. Le nombre de configurations de conflit présentes dans la base d'apprentissage est donc important. Des essais effectués avec des nombres différents tendent à montrer que quand il y a plus de configurations de conflit dans la base d'apprentissage, un réseau fiable est plus rapidement obtenu, mais avec de moins bonnes performances en termes de retard. Si ce nombre est trop faible, il est difficile d'obtenir un réseau fiable. Pour des conflits à deux avions, parmi les différentes valeurs testées pour $N_a$, c'est avec $N_a = 50$ que nous avons obtenu les meilleurs résultats. On définit, pour chaque réseau de neurones $\mathcal{R}$ généré et manipulé par l'algorithme génétique, un compteur $n_s(\mathcal{R})$ donnant le nombre de générations successives sur lesquelles le réseau a fourni des trajectoires sans conflit. On appelle $n_s(\mathcal{R})$ le *nombre de succès* du réseau $\mathcal{R}$.

On obtient ainsi une mesure de fiabilité du réseau $\mathcal{R}$, donnée par $s(\mathcal{R}) = p_{n_s(\mathcal{R}),n_s(\mathcal{R}),p_f}$, définie par l'équation 5.29. On n'utilise cette notion de fiabilité que pour évaluer des réseaux n'ayant jamais échoué, c'est pourquoi, avec les notations de l'équation 5.29, on ne s'intéresse qu'aux cas $m = n = n_s(\mathcal{R})$. Comme dans le cas du programme d'Othello, on prend $p_f = 0,95$, et on calcule préalablement à l'utilisation de l'algorithme génétique, les valeurs de $p_{n_s,n_s,p_f}$, pour les différentes valeurs possibles de $n$ que l'on stocke dans un tableau, dans lequel elles seront lues pour calculer la fonction $f_a$. On a calculé ces valeurs pour $n_s$ allant jusqu'à 1 000 (c'est le nombre maximal de générations que nous avons fixé pour les algorithmes que nous utilisons). Par définition, $n_s$ est inférieur au nombre maximal de générations de la population de réseaux de neurones.

Nous utilisons donc cette valeur de $s(\mathcal{R})$ dans le calcul de la fonction d'adaptation du réseau $\mathcal{R}_j$. Cependant, nous souhaitons toujours, bien entendu, tenir compte des performances, en termes de retard et de respect des contraintes de séparation, du réseau $\mathcal{R}_j$ sur les configurations de conflit présentes dans la base d'apprentissage au moment de son évaluation.

Si le réseau dont on calcule la fonction d'adaptation a été confronté à plusieurs bases d'apprentissage différentes, l'information donnée par son comportement sur chacune des configurations des ces bases d'apprentissage est utilisée. Ainsi, le nombre de violations $N_{vt}(\mathcal{R})$ et la moyenne quadratique des retards $r_t(\mathcal{R})$ ne sont plus exactement ceux qui sont donnés par les équations 5.23 et 5.24. On note $r'_t(\mathcal{R})$ la moyenne quadratique de la somme des retards des deux avions, non plus sur les seules configurations présentes dans la base d'apprentissage au moment du calcul de la fonction d'adaptation du réseau $\mathcal{R}$, mais sur toutes les configurations auxquelles $\mathcal{R}$ a été confronté. De même, on note $N'_{vt}(\mathcal{R})$ la moyenne, sur les différentes bases d'apprentissage auxquelles $\mathcal{R}$ a été confronté, du nombre de pas de temps (total sur chaque base d'apprentissage), sur lesquels les contraintes de séparation ont été violées. On parle pour cela d'utilisation de *mémoire* dans le calcul de la fonction d'adaptation des réseaux.

La fonction d'évaluation doit cette fois remplir les conditions suivantes :

- Comme dans le cas de la base d'apprentissage fixe, la valeur de la fonction $f_a$ doit être toujours plus grande sur un réseau pour lequel $N'_{vt} = 0$ (qui a toujours généré des trajectoires sans conflit) que pour un réseau pour lequel $N'_{vt} \neq 0$ (qui a entraîné au moins un conflit).
- Pour un réseau $\mathcal{R}$ qui n'a pas entraîné de conflit, la valeur de $f_a(\mathcal{R})$ doit être une fonction croissante de la fiabilité $s(\mathcal{R})$, $f_a(\mathcal{R})$ et décroissante de la moyenne quadratique des retards entraînés $r'_t(\mathcal{R})$.
- Pour un réseau $\mathcal{R}$ qui a entraîné un conflit, la valeur de $f_a(\mathcal{R})$ doit être une fonction décroissante de $N'_{vt}$.

Pour satisfaire ces conditions, nous avons défini, dans le cadre de l'utilisation d'une base d'apprentissage renouvelée, la valeur de la fonction d'adaptation d'un réseau $\mathcal{R}$ de la manière suivante :

**Si $\boldsymbol{N'_{vt}(\mathcal{R}) \neq 0}$ :**

$$f_a(\mathcal{R}) = \frac{1000}{1 + (\mathcal{R})} \tag{5.30}$$

**Si $\boldsymbol{N'_{vt}(\mathcal{R}) = 0}$ :**

$$f_a(\mathcal{R}) = 1000 + s(\mathcal{R}).\frac{1000}{1 + r'_t(\mathcal{R})} \tag{5.31}$$

## 5.4 Résultats obtenus sur des conflits à deux avions

### 5.4.1 Paramètres de l'algorithme génétique

Les réseaux de neurones sont codés par leur poids : l'algorithme génétique utilise donc un codage réel (voir section 4.1.2). Les chromosomes contiennent 251 réels correspondant aux 251 poids des réseaux (voir section 5.3.2). Le croisement utilisé est le croisement barycentrique, avec des poids compris entre -0,5 et 1,5 (voir section 4.2.1), et la mutation est la mutation gaussienne (section 4.2.2)

Les paramètres de l'algorithme génétique utilisé pour l'apprentissage des réseaux de neurones sont les suivants :

- Nombre de chromosomes (c'est-à-dire de réseaux de neurones) dans la population : 500.
- Taux de croisement : 0,5.
- Taux de mutation : 0,2.
- Nombre de générations : 1 000.

  Le plus souvent, le réseau qui reste le meilleur à la fin de l'évolution des réseaux est cependant obtenu bien avant la millième génération, surtout dans le cas de l'utilisation d'une base d'apprentissage fixe. On remarque en effet que la stabilisation du meilleur élément (c'est-à-dire le fait que le meilleur élément reste inchangé sur un grand nombre de générations)

apparaît plus rapidement dans le cas de l'apprentissage sur une base fixe. De plus, quand l'apprentissage se fait au moyen d'une base renouvelée, il est important d'obtenir un réseau présent dans la population depuis un grand nombre de générations : cela signifie qu'il a *résisté* à un grand nombre de configurations de conflit différentes.
- Nombre de configurations de conflit dans la base d'apprentissage : 50.
- Le meilleur élément est conservé d'une génération à l'autre : l'algorithme utilise l'élitisme.
- La sélection est une sélection par le rang (voir section 4.2.3).
- L'algorithme utilise le sharing avec clustering adaptatif, avec un taux de sharing de 0,8 (voir section 4.4)

### 5.4.2 Apprentissage sur une base fixe d'exemples

Dans un premier temps, nous avons utilisé pour l'apprentissage du réseau de neurones des bases d'apprentissage fixe, comprenant 50 configurations de conflit générées aléatoirement selon la méthode décrite dans la section 5.3.6

Pour comparer l'influence de l'utilisation de la symétrie sur l'apprentissage et, au cas où la symétrie est utilisée, l'influence du signe choisi, nous avons mené l'apprentissage sur trois types de bases d'apprentissage :
- Une base d'apprentissage composée de configurations de conflit quelconques.
- Une base d'apprentissage composée de configurations d'apprentissage positives au sens de la fonction signe $Sg_v$ définie dans la section 5.3.5, c'est-à-dire des configurations de conflit dans lesquelles, au temps $t = 0$, l'avion le plus rapide est à droite de l'avion le plus lent (ou éventuellement ni à droite ni à gauche de l'avion le plus lent).
- Une base d'apprentissage composée de configurations de conflit positives au sens de la fonction signe $Sg_p$ (elle aussi décrite dans la section 5.3.5) c'est-à-dire de configurations telles que si les avions se croisent sans qu'il y ait collision, l'avion qui passe devant l'autre vient de sa droite.

On obtient très vite, dans chacun des trois cas ci-dessus, un réseau ne menant à aucun conflit sur les configurations de conflit de la base d'apprentissage. À la fin de l'évolution, on obtient un réseau menant de plus à des retards très faibles (de l'ordre de quelques secondes).

Cependant, il est intéressant de mesurer la capacité de généralisation des réseaux de neurones ainsi obtenus, c'est-à-dire leur comportement sur des configurations de conflit différentes de celles sur lesquelles leur apprentissage a été réalisé.

Nous avons donc testé chacun des trois réseaux de neurones que nous avons obtenus sur $10\,000$ configurations générées aléatoirement. Ces configurations sont, cette fois, pour chacun des trois réseaux, de signes quelconques. Si un réseau de neurones a réalisé son apprentissage sur des configurations de conflit positives pour une fonction signe donnée, on génère des trajectoires d'évitement pour une configuration de conflit non positive en appliquant le réseau de neurones à sa configuration symétrique, et on prend à chaque pas de temps l'opposé de la modification de cap donnée par le réseau pour cette configuration symétrique (voir section 5.3.5).

On constate que, du point de vue du respect des contraintes de séparation, aucun réseau de neurones ne généralise parfaitement : certaines des configurations de conflit mènent à des violations des contraintes. Cependant, on peut considérer que les résultats obtenus sont corrects : il n'y a violation des contraintes que dans 2 à 4 pour cent des configurations. De plus, comme le montre le tableau 5.1, une grande proportion des violations de contraintes mènent en fait à des distances minimales entre avions proche de la norme de séparation.

TAB. 5.1: *Base d'apprentissage fixe : violations des contraintes de séparation ; nombre de configurations (sur* $10\,000$*) pour lesquelles le conflit n'a pas été résolu, réparties sur les lignes selon la gravité de la violation de la contrainte de séparation.*

| Distance minimale | Configurations quelconques | Signe utilisant les vitesses | Signe utilisant les ordres de passages |
|---|---|---|---|
| de $3,75$ à $4$ $NM$ | 152 | 191 | 68 |
| de $3,5$ à $3,75$ $NM$ | 80 | 92 | 53 |
| de $3$ à $3,5$ $NM$ | 115 | 22 | 56 |
| de $2$ à $3$ $NM$ | 41 | 35 | 29 |
| de $1$ à $2$ $NM$ | 0 | 30 | 3 |
| de $0$ à $1$ $NM$ | 0 | 24 | 0 |
| moyenne | $3,5$ $NM$ | $3,3$ $NM$ | $3,5$ $NM$ |
| total | 388 | 394 | 209 |

Le tableau 5.1 montre en effet la répartition des violations de contraintes sur les configurations de conflit sur lesquelles les capacité de généralisation des réseaux de neurones ont été évaluées. Les colonnes correspondent aux trois réseaux testés. Les six premières lignes correspondent à des valeurs de la distance minimale entre les avions au cours de l'évitement. Chaque case donne le nombre de configurations pour lesquelles le réseau correspondant à la colonne a mené à une distance minimale entre avions correspondant à la ligne.

Sur les deux dernières lignes sont portées la moyenne de ces distances minimales sur les configurations pour lesquelles le conflit n'a pas été résolu, et le nombre de ces configurations.

Le tableau 5.2 montre, lui, les performances des réseaux de neurones, en termes de retards des avions, sur les configurations sur lesquelles ils ont généré des trajectoires sans conflit. Les valeurs des retards correspondent à la moyenne des retards des deux avions impliqués dans le conflit. Le tableau 5.2 est organisé selon le même principe que le tableau 5.1, les 5 premières lignes correspondent à des fourchettes de ce retard, les colonnes, à chacun des trois réseaux testés. Les nombres portés donnent le nombre de configurations sur lesquelles le réseau correspondant à la colonne mène à un retard correspondant à la ligne.

Les deux dernières lignes donnent, pour chaque réseau, la valeur moyenne des retards sur les configurations pour lesquelles il n'y a plus de conflit, et le nombre de ces configurations.

On constate, tant du point de vue du respect des contraintes de séparation que du retard sur les configuration respectant ces contraintes, que le réseau ayant appris sur des configurations positives pour la fonction signe $Sg_p$ (selon l'ordre de passage des avions) obtient de meilleurs résultats que celui qui a appris sur des configurations quelconques, qui lui-même obtient de meilleurs résultats que celui qui a appris sur des configurations positives pour $Sg_v$ (selon les vitesses). Nous verrons dans la section 5.4.4 que sur certains types de configurations, ce dernier réseau se montre cependant plus efficace que le réseau qui a appris sur des configurations positives pour $Sg_p$.

TAB. 5.2: *Base d'apprentissage fixe : retards quand la séparation est assurée ; nombre de configurations (sur* 10 000*) pour lesquelles les réseaux ont généré des trajectoires sans conflit, réparties sur les lignes selon le retard moyen par avion.*

| Retards | Configurations quelconques | Signe utilisant les vitesses | Signe utilisant les ordres de passages |
|---|---|---|---|
| de 0 à 5 $s$ | 5 581 | 3 901 | 7 123 |
| de 5 à 10 $s$ | 2 221 | 2 174 | 2 048 |
| de 10 à 20 $s$ | 1 737 | 2 924 | 598 |
| de 20 à 30 $s$ | 63 | 568 | 16 |
| de 30 à 60 $s$ | 10 | 39 | 16 |
| retard moyen | $5,1\ s$ | $8,0\ s$ | $3,6\ s$ |
| total | 9 612 | 9 606 | 9 791 |

### 5.4.3 Apprentissage sur une base renouvelée

Les défauts de généralisation constatés dans la section précédente nous ont poussé à utiliser, pour l'évolution de la population des réseaux de neurones, une base d'apprentissage renouvelée, et non plus fixe (voir section 5.3.7).

Nous avons, comme dans le cas de l'utilisation d'une base d'apprentissage fixe, obtenu trois réseaux de neurones, le premier ayant réalisé son apprentissage sur des configurations de conflit quelconques, le deuxième sur des configurations de conflit positives pour $Sg_v$, le troisième sur des configurations de conflit positives pour $Sg_p$.

Dans chacun de ces trois cas, les 50 configurations constituant la base d'apprentissage sont renouvelées à chaque génération, la fonction d'adaptation utilisée pour évaluer les réseaux de neurones de la population est la fonction $f_a$ donnée par les équations 5.30 et 5.31 de la section 5.3.7.

On constate tout d'abord que la généralisation est beaucoup plus efficace : les trois réseaux obtenus fournissent systématiquement des trajectoires sans conflit pour les 10 000 configurations sur lesquelles ils ont été testés. Les résultats obtenus, en termes de retard, par les trois réseaux sont donnés par le tableau 5.3, organisé de la même manière que le tableau 5.2.

Les retards entraînés par les différents réseaux sont légèrement plus importants que ceux auxquels menaient les réseaux dont l'apprentissage avait été réalisé sur des bases fixes (voir le tableau 5.2). On peut interpréter ce résultats en disant que la meilleure capacité de généralisation des réseaux, en termes de respect des contraintes de séparation, est obtenue au prix d'une certaine *prudence* préjudiciable à l'efficacité en terme de retards : les réseaux semblent s'assurer des *marges de sécurité*, en faisant faire aux avions des détours plus importants.

On remarque de plus, comme dans le cas de l'apprentissage sur des bases fixes, que l'utilisation de configurations positives pour $Sg_p$ est globalement plus efficace que l'utilisation de configurations quelconques, elle même plus efficace que l'utilisation de configurations positives pour $Sg_v$. Nous allons maintenant étudier de manière plus qualitative, l'influence, sur certaines configurations de conflit de la fonction signe utilisée.

TAB. 5.3: *Base d'apprentissage renouvelée : retards ; nombre de configurations (sur* 10 000*) pour lesquelles les réseaux ont généré des trajectoires sans conflit, réparties sur les lignes selon le retard moyen par avion.*

| Retards | Configurations quelconques | Signe utilisant les vitesses | Signe utilisant les ordres de passages |
|---|---|---|---|
| de 0 à 5 $s$ | 4 771 | 3 260 | 5 798 |
| de 5 à 10 $s$ | 2 060 | 1 678 | 2 578 |
| de 10 à 20 $s$ | 2 666 | 3 79 | 1 445 |
| de 20 à 30 $s$ | 353 | 1 341 | 151 |
| de 30 $s$ à 1 $mn$ | 132 | 395 | 26 |
| de 1 à 2 $mn$ | 18 | 47 | 2 |
| retard moyen | $7,5$ $s$ | $11,4$ $s$ | $5,2$ $s$ |
| total | 10 000 | 10 000 | 10 000 |

### 5.4.4 Comparaison des deux fonctions signe

Nous cherchons à comparer le comportement de deux réseaux de neurones. L'un d'eux a réalisé son apprentissage sur des configurations positives pour la fonction signe $Sg_v$ (l'avion le plus rapide vient de la droite), ce réseau sera appelé $res_v$. L'autre réseau, que nous appellerons $res_p$ a, lui, réalisé son apprentissage sur des configurations positives pour la fonction signe $Sg_p$ (l'avion qui passe devant l'autre vient de la droite).

Il s'agit des réseaux déjà testés dans le section 5.4.3 : leurs apprentissages ont été menés sur des bases renouvelées.

Nous avons testé ces deux réseaux sur 100 configurations de conflit, de manière à pouvoir comparer la forme de trajectoires qu'ils génèrent pour chacune de ces configurations. Celles-ci ont été générées aléatoirement selon la méthode décrite dans la section 5.3.6. Chacune d'entre elles a été traitée par chacun des deux réseaux, $res_v$ et $res_p$. Rappelons que si une configuration n'est pas positive pour le signe correspondant au réseau qui la traite, le réseau traite sa configuration symétrique et le signe de la modification du cap de l'avion est inversé pour chaque pas de temps.

On observe tout d'abord que les deux réseaux adoptent des stratégies différentes. Le réseau $res_v$ fait souvent tourner un avion dans un sens, et l'autre avion dans l'autre sens (comme par exemple sur la figure 5.9 b). Le réseau $res_p$, lui, fait le plus souvent tourner les deux avions dans le même sens.

On remarque aussi que les configurations sur lesquelles les deux réseaux mènent aux résultats les plus différents en terme de qualité correspondent à des angles faibles entre les trajectoires d'évitement des avions. Nous avons systématiquement observé que ce sont ces configurations qui sont les plus difficiles pour les réseaux de neurones[5]. Quand les angles d'incidence des avions sont plus grands, les configurations sont plus *faciles* pour les deux réseaux, aucun d'entre eux ne rencontre de problème majeur.

[5]Ce sont d'ailleurs systématiquement ces configurations qui posent le plus de problèmes quel que soit le système d'évitement envisagé. On pourra se reporter par exemple à [Dur96].

En considérant les configurations sur lesquelles les deux réseaux mènent à des valeurs très différentes des allongements des trajectoires, on constate que les configurations pour lesquelles le réseau $res_v$ est sensiblement meilleur que le réseau $res_p$ ont une au moins des caractéristiques suivantes :

- La différence entre les vitesses des avions est importante.
- La distance minimale entre les avions en l'absence d'évitement est petite.

Les configurations sur lesquelles le réseau de $res_p$ est sensiblement meilleur que le réseau $res_v$ ont, elles, une des caractéristiques suivantes :

- La distance minimale entre les avions, en l'absence d'évitement, est grande.
- La différence entre les vitesses des avions est faible.

Nous avons ainsi représenté sur les figures 5.9 à 5.11 les trajectoires proposées par chacun des deux réseaux $res_v$ et $res_p$ sur les trois configurations menant aux plus grandes différences sur les retards des avions, sur les 100 configurations utilisées pour cette étude : la première, la dix-neuvième et la trente-cinquième.

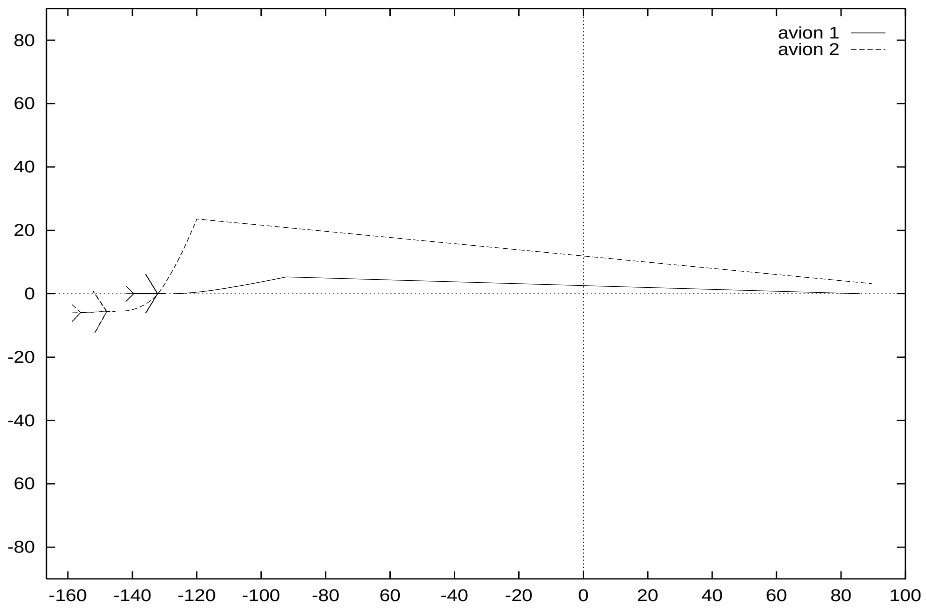
La figure 5.9 montre les trajectoires données, pour la configuration 19, par $res_p$ (a) et par $res_v$ (b). Cette configuration correspond à une différence de vitesse de 51 nœuds entre les deux avions, et à une distance minimale de 1,2 NM entre eux, en l'absence d'évitement. Cette distance minimale est relativement faible (les configurations de conflit sont générées de manière à ce qu'il y ait un conflit entre les avions, c'est-à-dire à ce que la distance minimale entre les avions soit inférieure à 4 NM), et l'écart entre les vitesses est relativement important. On observe un retard moyen de 10 secondes par avion quand les trajectoires sont générées par $res_v$ et de 55 secondes quand elles sont générées par $res_p$.

Cette différence dans l'efficacité des réseaux, avec toujours un avantage pour $res_v$, est encore plus marquée sur la configuration 35 : la différence de vitesses entre les avions est de 136 kt (la plus grande différence possible est de 200 kt ; la distance minimale entre les avions en l'absence de déviation est de 0,6 NM. Le retard moyen n'est alors que de 7 secondes par avion quand les trajectoires sont générées par $res_v$ (figure 5.10 b), alors qu'il est de 1 minute et 46 secondes quand elles sont générées par $res_p$ (figure 5.10 a).

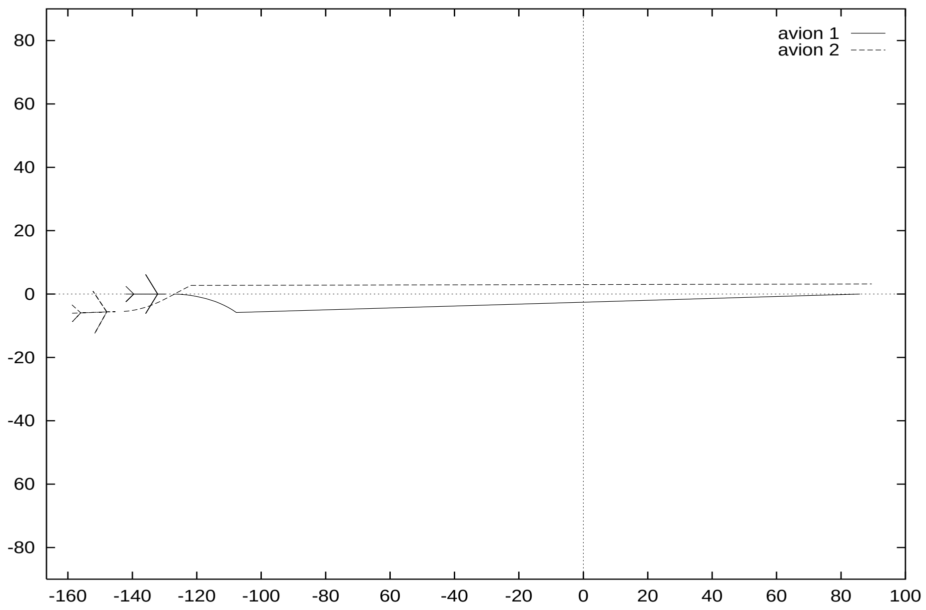
La configuration 1, elle, correspond à une différence de vitesses de 21 kt seulement, avec une distance minimale en l'absence d'évitement de 3,5 NM. On constate dans ce cas un retard de seulement 4 secondes par avion, avec $res_p$ (figure 5.11 a), mais de 1 minute 40 secondes avec $res_v$ (figure 5.11 b).

Ces comparaisons, menées sur un petit nombre de configurations, n'ont pour but que de tenter de comprendre les différences qualitatives dans le comportement des réseaux et non plus comme dans les sections précédentes dans leur efficacité globale.

On constate cependant sur ces configurations que le réseau $res_v$, qui a appris sur des configurations positives pour une fonction signe basée sur la position de l'avion le plus rapide fournit une meilleure solution si les différences de vitesses sont importantes. Le réseau $res_p$, qui a appris sur des configurations positives pour une fonction signe basée sur l'ordre de passage des avions fournit une meilleure solution quand les avions passent loin l'un de l'autre. On peut dire que chaque réseau fournit une meilleure réponse quand la fonction signe utilisée est plus adaptée à la configuration ; cela a plus de sens de parler de *l'avion le plus rapide* quand la différence entre les vitesses est grande, de *l'avion qui passe devant* quand les avions passent loin l'un de l'autre.

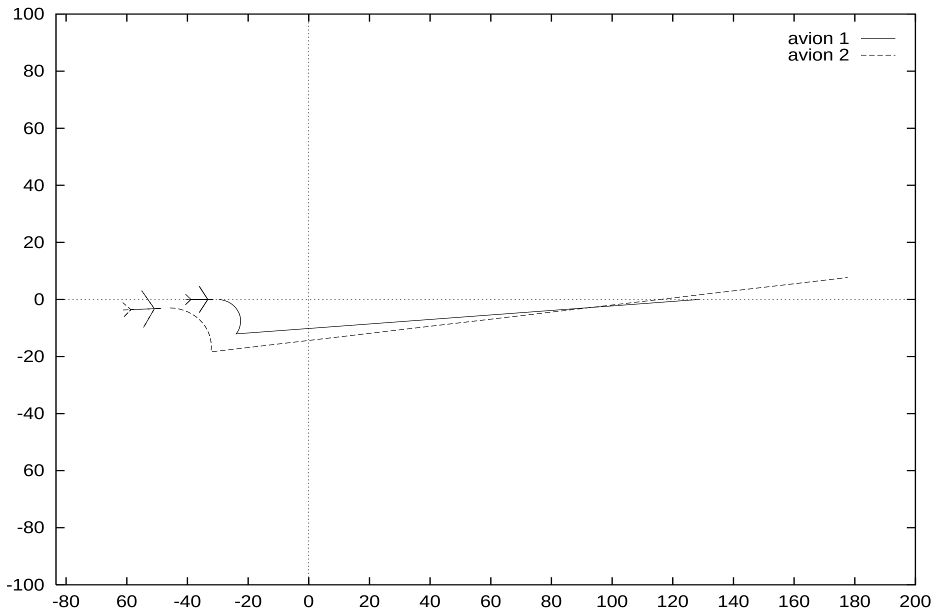



(a) Trajectoires générées par $res_p$.

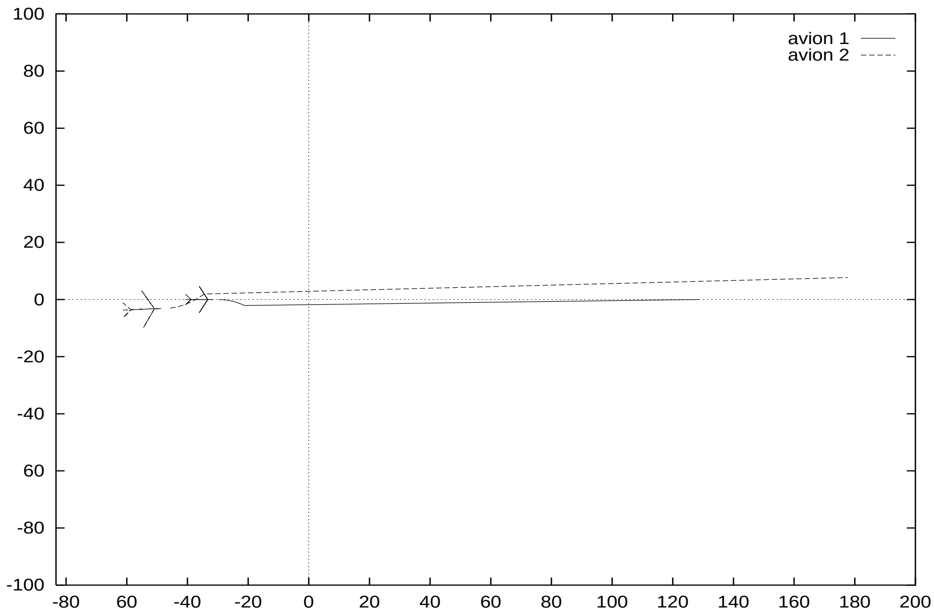



(b) Trajectoires générées par $res_v$.

FIG. 5.9: *Configuration 19*




(a) Trajectoires générées par $res_p$.




(b) Trajectoires générées par $res_v$.

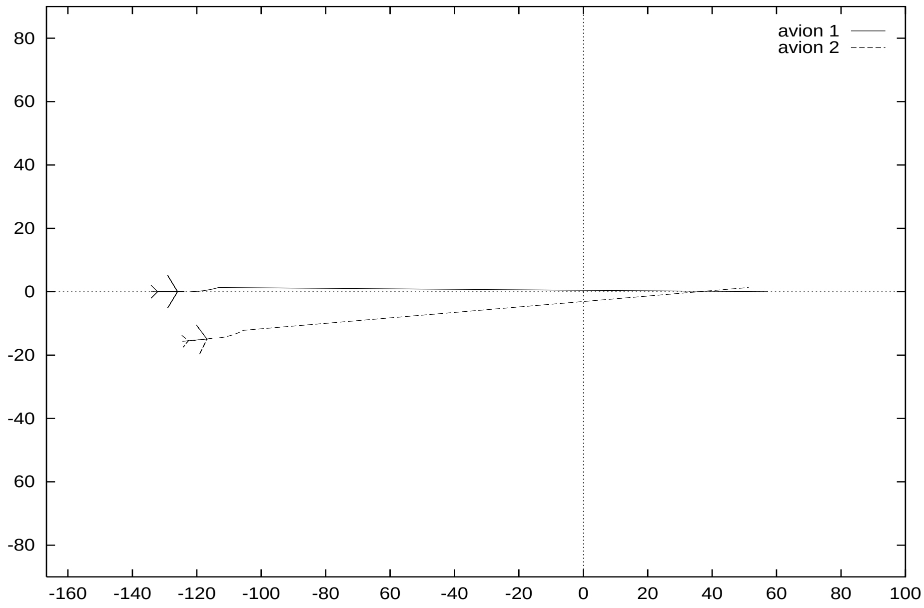
FIG. 5.10: *Configuration 35*

Si on observe la forme des trajectoires, on constate que celles qui sont générées par $res_v$ dévient les avions dans des sens différents, tandis que celles qui sont générées par $res_p$ les dévient dans le même sens. C'est le cas dans les trois cas illustrés ici, et c'est aussi le cas dans une très grande majorité des cas que nous avons pu observer. Ces stratégies seraient-elles les mieux adaptées pour chacun des cas ? Il nous semble que oui.

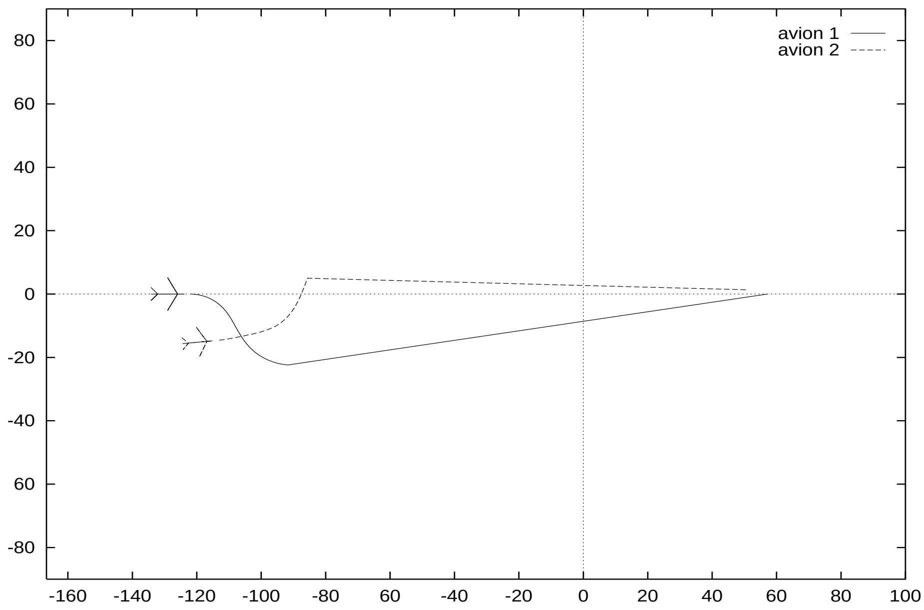
En effet, considérons une configuration dans laquelle les avions volent à des vitesses très différentes et se croiseraient, sans déviation, en passant très près l'un de l'autre, avec un angle faible entre leurs trajectoires. Au début de l'évitement, l'avion le plus rapide est alors *derrière* l'autre. Cela est d'ailleurs visible sur les figures 5.9 et 5.10. Il semble donc raisonnable de faire se croiser tout de suite les avions en faisant passer l'avion le plus rapide derrière l'avion le plus lent. C'est ce que fait le plus souvent le réseau $res_v$. Le fait de n'apprendre que sur des configurations positives pour $Sg_v$, c'est-à-dire sur des configurations dans lesquelles l'avion le plus rapide est à droite, revient donc à apprendre dans des cas où cette stratégie se traduit simplement par : l'avion le plus rapide, qui est à droite, tourne à gauche, l'avion le plus lent, qui est à gauche, tourne à droite. L'apprentissage de cette stratégie est favorisé par les configurations auxquelles les réseaux sont confrontés.

Considérons maintenant une configuration dans laquelle les réseaux volent à des vitesses proches mais passent loin de l'autre, toujours pour des angles faibles. Le fait que les vitesses soient proches fait qu'au début de l'évitement les deux avions sont plutôt l'un à côté de l'autre (l'avion qui passera devant l'autre étant déjà plus avancé). Le fait de dévier les avions dans des sens différents serait moins utile : si les vitesses sont proches cela ne ferait que repousser le problème, en revenant vers leurs destinations, les avions se retrouveraient de nouveau l'un à côté de l'autre. Pour que les avions puissent se croiser sans conflit, il est utile qu'ils se retrouvent, au moment où ils reprendront des caps directs vers leurs destinations *l'un derrière l'autre* : ils se croiseront alors naturellement. La meilleure stratégie est donc de les dévier tous deux dans la même direction. S'il passent assez loin l'un de l'autre, même en l'absence de déviation, il n'est pas nécessaire, pour qu'ils puissent se croiser sans conflit, de les dévier jusqu'à ce qu'ils se trouvent effectivement l'un derrière l'autre. C'est le cas sur la figure 5.11 : il suffit d'augmenter très légèrement la distance minimale entre eux. En l'absence de déviation, un des avions passerait devant l'autre. Il est souvent plus rapide de dévier les avions de manière à ce qu'au moment où ils se croisent, cet ordre de passage soit respecté, en déviant les avions dans la direction dans laquelle se trouve l'avion qui passe devant. Le fait de n'apprendre que sur des configurations de conflit positives pour $Sg_p$, telles que si les avions se croisent, l'avion qui passe devant vient de la droite, revient à traduire simplement cette stratégie par : les deux avions tournent à droite (puisque l'avion qui passe devant vient de la droite). Sur les figures que nous présentons, il arrive que dans ce cas les deux avions tournent à gauche. On constate que l'avion le plus avancé au début de l'évitement se trouvait en effet alors à gauche. Nous rappelons que les configurations utilisées pour tester les réseaux sont de signe quelconque. Si une configuration n'est pas positive pour $Sg_p$, le réseau $res_p$ fournit en fait des trajectoires pour sa configuration symétrique, desquelles on déduit des trajectoires pour la première configuration, en modifiant le signe des changements de caps. c'est, par exemple, le cas de la configuration 19 (figure 5.9).

Les résultats observés montrent donc que l'utilisation des symétries pour réduire l'ensemble des configurations à couvrir lors de l'apprentissage est utile, même si les résultats des sections 5.4.2 et 5.4.3 indiquent que globalement, l'utilisation du signe basé sur les différences de vitesses est moins adaptée aux configurations de conflit, telles du moins que nous les générons. Il est cependant




(a) Trajectoires générées par $res_p$.




(b) Trajectoires générées par $res_v$.

FIG. 5.11: *Configuration 1*

possible que ce ne soit plus le cas pour d'autres répartitions des configurations de conflit.

## 5.5 Résolution de conflits à trois avions

### 5.5.1 Configurations de conflit à trois avions

Nous présentons maintenant une tentative d'utiliser des techniques similaires à celle qui a été décrite dans les sections précédentes pour résoudre des conflits impliquant trois avions.

Nous supposons toujours que les avions volent à vitesse constante dans un même plan horizontal, et que leurs trajectoires originelles (en l'absence d'évitement) ne comportent pas de changements de cap. Nous supposerons de plus, comme dans la section 5.3.1 que chacun des trois avions connaît à chaque pas de temps la position, le cap, la vitesse et la destination de chacun des deux autres avions.

Par *conflit à trois avions*, on entend un *cluster* à trois avions. Rappelons qu'on appelle *cluster* la fermeture transitive de plusieurs conflits (définition 1.7).

Les trois avions ne sont donc pas forcément tous deux à deux en conflit au même instant.

Ainsi, considérons trois avions, $a_1$, $a_2$ et $a_3$. On parle d'un conflit impliquant ces trois avions quand deux de ces avions, par exemple $a_1$ et $a_2$, sont en conflit à partir d'un instant $t_{1,2}$, et que l'un au moins de ces deux avions (par exemple $a_2$) est en conflit avec le troisième avion ($a_3$), à un instant $t_{2,3}$ (avec, par exemple $t_{2,3} \geq t_{1,2}$) suffisamment proche de $t_{1,2}$ pour que les deux conflits ne puissent pas être résolus séparément, c'est-à-dire qu'on ne puisse pas assurer que les avions impliqués dans le premier conflit soient revenus sur leurs trajectoires d'origine avant le début de la résolution du deuxième conflit.

Supposons toujours que $a_1$ et $a_2$ entrent en conflit au temps $t = t_{1,2}$, et que $a_2$ entre en conflit avec $a_3$ au temps $t = t_{2,3}$, avec $t_{2,3} \geq t_{1,2}$. Comme dans la section 5.3.4, on place l'origine des temps à l'instant précédent le conflit où la distance entre les avions $a_1$ et $a_2$ est égale à la distance d'alerte $d_a$. On note $t'_0$ l'instant, précédent leur conflit où $a_2$ et $a_3$ sont distants de $d_a$. En notant toujours $t_f$ la fenêtre de temps sur laquelle prend place un évitement, on considérerait que les conflits successifs entre $a_1$ et $a_2$ d'une part, entre $a_2$ et $a_3$ d'autre part, forment un cluster dès que l'on a $t'_0 \leq t_f$.

Cependant nous ne considérerons pas tous les clusters répondant à ce critère. En effet, le temps nécessaire à la résolution d'un conflit est très souvent bien inférieur à $t_f$. On constate que les conflits à trois avions présentant de réelles difficultés pour les réseaux de neurones que nous avons testés sont ceux pour lesquels les différents conflits à deux avions constituant le *cluster* sont les plus rapprochés dans le temps. C'est pourquoi nous introduisons la notions de *conflits liés*.

Rappelons que les instants où les conflits *commencent*, appelés aussi *débuts* des conflits (correspondant aux instants $t_{1,2}$ et $t_{2,3}$ dont il est question ci-dessus), sont les instants auxquels les avions impliqués dans les conflits correspondant cesseraient d'être séparés en l'absence de déviation.

**Définition 5.9 – Conflits liés –** *On fixe une durée $t_l$, avec $t_l < t_f$. Pour trois avions $a_i$, $a_j$ et $a_k$, on considère deux conflits, l'un entre les avions $a_i$ et $a_j$, et l'autre entre les avions $a_j$ et $a_k$ d'autre* *part, commençant respectivement aux temps $t = t_{i,j}$ et $t = t_{j,k}$. On dit que ces deux conflits sont liés si on a :* $|t_{j,k} - t_{i,j}| \leq t_l$.

Dans la suite de ce chapitre, on parlera de conflit à trois avions quand un (au moins) de ces trois avions est impliqué dans un conflit avec chacun des deux autres avions et que ces conflits sont liés au sens de la définition 5.9.

On définit la distance minimale entre trois avions de la manière suivante, $d(a_i, a_j)$ étant la distance entre les avions $a_i$ et $a_j$ :

**Définition 5.10 – Distance minimale –** *Pour trois avions $a_i$, $a_j$ et $a_k$, on définit la distance minimale entre eux, notée $d_{min}(a_i, a_j, a_k)$, comme la plus petite des distances entre deux de ces trois avions :*

$$d_{min}(a_i, a_j, a_k) = \min\{d(a_i, a_j), d(a_j, a_k), d(a_i, a_k)\}$$

On reprend les définitions 5.3 et 5.4 de la section 5.3.4, pour définir, dans le cas d'un conflit impliquant trois avions, un cas de conflit et un cas de conflit admissible :

**Définition 5.11 – Cas de conflit à trois avions –** *un cas de conflit est la donnée d'une trajectoire pour chacun des avions (c'est-à-dire d'une position et d'un vecteur vitesse à un instant donné), ces trajectoires étant telles que ces avions sont impliqués dans un conflit à trois avions, c'est-à-dire que l'un au moins d'entre eux est impliqué dans un conflit avec chacun des deux autres et que ces conflits sont liés (définition 5.9).*

Un cas de conflit est défini par la position le cap et la vitesse de chacun des trois avions à un instant donné.

De même que dans le cadre de la résolution de conflits n'impliquant que deux avions, nous ne souhaitons pas considérer tous les cas de conflit possibles, mais seulement ceux qui auront lieu sur une durée limitée.

Le fait d'utiliser des méthodes d'évitement embarquées pour traiter des conflits impliquant plus de deux avions pose des problèmes spécifiques. On suppose en général que chaque avion tient compte de la présence des éventuels intrus à partir du moment où ceux-ci sont en deçà d'un certain *horizon*, que cet horizon soit défini par une distance (l'avion tient compte de l'intrus quand celui-ci s'approche d'une distance inférieure à une distance d'alerte, comme c'était le cas dans les sections précédentes), ou par une durée (l'avion tient compte de l'intrus si un conflit avec lui est prévu à l'intérieur d'une certaine durée).

Dans ce cas, un avion peut être impliqué dans un conflit avec un deuxième avion, quand un troisième avion apparaît, qu'il doit prendre en compte, alors qu'il est déjà sur une trajectoire déviée pour éviter le deuxième avion. Ce problème est abordé dans [Gra98]. Notre but ici, est simplement d'étudier la possibilité d'étendre l'utilisation des réseaux de neurones décrits dans les sections précédentes à des conflits à trois avions. Pour plus de simplicité, nous considérons dans la première approche que nous présentons ici, que la génération de trajectoires d'évitement commence au même instant pour les trois avions. C'est ce que nous supposons aussi dans le cas de l'utilisation de l'algorithme $A^*$ (voir chapitre 3) et de méthodes d'intervalles (voir chapitre 2) à des techniques d'évitement séquentielles. La manière dont s'organisera, dans le cadre d'une application à des cas de conflit issus du trafic réel, la prise en compte des intrus par chaque avion pourra être développée dans un deuxième temps, comme cela a été fait dans [Gra98] à partir des méthodes séquentielles utilisant l'algorithme $A^*$ que nous décrivons dans le chapitre 3.

Le début de l'évitement, c'est-à-dire l'instant à partir duquel chacun des trois avions pourra être dévié, est fixé à l'instant précédent le conflit où la distance minimale entre les avions est égale à la distance d'alerte $d_a$. L'origine des temps sera fixée à cet instant.

Nous fixons, comme dans le cas de conflit à deux avions, une durée sur laquelle aura lieu l'évitement, notée $t_f$. Nous souhaitons, de même que dans les cas d'un conflit à deux avions, ne considérer que des cas de conflit tels que les avions soient de nouveau deux à deux séparés à l'instant $t = t_f$.

**Définition 5.12 – Cas de conflit admissible à trois avions –** *Pour des valeurs de $d_a$ et $t_f$ fixées, nous appellerons cas de conflit admissible à trois avions un cas de conflit à trois avions tel que si le temps $t = 0$ est fixé à l'instant précédent le conflit auquel la distance minimale entre les avions est égale à $d_a$, alors les avions seront impliqués dans un conflit à trois avions entre les temps $t = 0$ et $t = t_f$ et seront de nouveau séparés au temps $t = t_f$.*

Les entrées des réseaux de neurones (sections 5.5.3 et 5.5.4) sont les mêmes que dans le cas d'un conflit à deux avions. Elles n'utilisent donc que des valeurs relatives des positions et des caps. Deux cas de conflit se déduisant l'un de l'autre par un déplacement du plan sont donc, ici encore, traités de manière identique par les réseaux de neurones. De plus les trajectoires des trois avions sont générés par des réseaux de neurones identiques. Deux cas de conflit se déduisant l'un de l'autre par un changement de numérotation des avions sont donc équivalents. On définit donc de la même manière que pour des conflits impliquant deux avions, la notion de *configuration de conflit* :

**Définition 5.13 – Configuration de conflit à trois avions –** *Une relation d'équivalence sur les cas de conflit admissibles à trois avions est définie, dans laquelle deux cas de conflit admissibles à trois avions sont équivalents s'ils se déduisent l'un de l'autre par un déplacement du plan ou par un changement de numérotation des avions. On appelle configurations de conflit à trois avions les classes d'équivalence associées à cette relation d'équivalence.*

La symétrie entre deux configurations de conflits pourrait être utilisée, comme dans le cas de conflits à deux avions, pour réduire l'espace des configurations de conflit. Cependant, les fonctions *signe* définies sur les configurations de conflit à deux avions ne sont pas généralisables aux configurations de conflit à trois avions. Les apprentissages des réseaux de neurones que nous allons décrire dans la suite de ce chapitre sont donc réalisés sur des configurations de conflit quelconques, générées aléatoirement.

### 5.5.2 Génération aléatoire de configurations de conflit à trois avions

La méthode que nous employons pour générer aléatoirement des configurations de conflit à trois avions reprend celle que nous avons utilisée pour la génération aléatoire de configurations de conflit à deux avions, qui est décrite dans la section 5.3.6. Comme dans la section 5.3.6, nous générons en fait des cas de conflit admissibles, qui seront des représentants de configurations de conflit.

On se place dans un repère orthonormé $(O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$. Les avions volent dans le plan $(\vec{e}_x, \vec{e}_y)$.

On suit les étapes suivantes, après avoir fixé les valeurs de $t_f$ et de $t_l$ :

1. On place $a_1$ en (0,0), avec le cap 0. On choisit aléatoirement la valeur $v_1$ de la vitesse de l'avion $a_1$. On note $t_{1,2}$ l'instant correspondant à cette position de l'avion $a_1$.
2. On choisit aléatoirement la vitesse $v_2$ et le cap $\alpha_2$ de l'avion $a_2$, ainsi que la vitesse $v_3$ et le cap $\alpha_3$ de l'avion $a_3$.
3. L'avion $a_2$ sera placé à cet instant $t_{1,2}$ à une distance égale à la norme de séparation $n_h$ de l'avion $a_1$. On tire aléatoirement une valeur $\gamma_2$ de son gisement par rapport à $a_1$ à l'instant $t_{1,2}$, de manière à ce que cet instant corresponde au début d'un conflit entre $a_1$ et $a_2$.
4. On choisit aléatoirement un instant $t_{1,3}$, tel que $|t_{1,2} - t_{1,3}| \leq l_t$.
5. On place les avions $a_1$ et $a_2$ dans leurs positions correspondant à $t = t_{1,3}$.
6. On reprend pour l'avion $a_3$ l'étape 3 : on le place à $t = t_{1,3}$ à une distance $n_h$ de l'avion $a_1$, après avoir tiré aléatoirement une valeur $\gamma_3$ de son gisement par rapport à $a_1$, de manière à ce que l'instant $t_{1,3}$ corresponde au début d'un conflit entre $a_1$ et $a_3$.
7. On vérifie que le cas de conflit à trois avions ainsi généré est admissible. Si ce n'est pas le cas, on revient à l'étape 3.
8. On calcule la position des avions à l'instant, précédent les conflits, auquel la distance minimale entre les avions est égale à la distance d'alerte $d_a$. L'origine des temps est fixée à cet instant, ce qui donne les valeurs de $t_{1,2}$ et de $t_{1,3}$. Le cas de conflit généré est défini par les positions des avions à l'instant $t = 0$, leurs vitesses et leurs caps.

Les étapes 3 et 6 sont menées comme dans le cadre de la génération aléatoire de configurations de conflit à deux avions telle qu'elle est décrite dans la section 5.3.6.

Les résultats que nous présentons dans la suite de ce chapitre ont été obtenus en utilisant pour la génération aléatoire de configurations de conflit les valeurs numériques suivantes :

$$\begin{aligned} n_h &= 4\,NM \\ d_a &= 16\,NM \\ v_{min} &= 300\,kt \\ v_{max} &= 500\,kt \\ t_f &= 20\,mn \end{aligned}$$

Nous nous sommes limités à des conflits liés assez rapprochés dans le temps, après avoir constaté lors de premiers essais que c'étaient ces conflits qui présentaient de réelles difficultés. Nous avons donc pris pour valeur de $t_l$, correspondant à l'écart temporel maximal entre deux conflits impliquant un même avion pour que ces conflits soient considérés comme étant liés (voir définition 5.9) :

$$l_c = 2\,mn \tag{5.32}$$

### 5.5.3 Réseaux de neurones à un intrus

Dans un premier temps, nous avons utilisé, pour la résolution de conflit à trois avions, des réseaux de neurones identiques, pour ce qui est de leurs entrées et de leur architecture, aux réseaux de neurones qui sont décrits dans la section 5.3.2. Ces réseaux sont utilisés pour générer la trajectoire d'un avion, en fonction de la situation de cet avion par rapport à un *intrus*, c'est-à-dire un deuxième avion susceptible d'entrer en conflit avec lui. C'est la manière la plus simple et la plus directe d'adapter les techniques développées dans les sections précédentes à la résolution de conflits à trois avions.

Pour chaque avion, le principe de cette utilisation du réseau de neurones est le suivant : à chaque pas de temps, la trajectoire de l'avion est générée au moyen d'un réseau de neurones semblable à ceux qui sont décrits dans la section 5.3.2, dont les entrées sont celles qui sont décrites dans la section 5.3.3, calculées à partir des position, vitesse, cap et destination d'un seul des deux intrus. Le choix de cet intrus peut se faire de différentes manières, nous en avons utilisé et testé deux, qui correspondent à deux façons différentes d'aborder les conflits à trois avions.

**Définition 5.14 – Intrus menaçant –** *On considère un avion $a_i$, impliqué dans un conflit à trois avions, et on se place à un instant $t$ donné. On dit d'un autre avion impliqué dans ce conflit, $a_k$, qu'il est un intrus menaçant pour $a_i$ à l'instant $t$ si le fait que $a_i$ et $a_k$ tentent tous deux de rejoindre leurs destinations, en suivant un cap direct, à partir de leurs positions à l'instant $t$, entraînerait un conflit entre ces deux avions.*

Il est clair, d'après cette définition, que si $a_k$ est menaçant pour $a_i$, $a_i$ est menaçant pour $a_k$.

Cette définition permet de distinguer les deux utilisations différentes que nous avons faites de réseaux de neurones à un intrus pour résoudre un conflit à trois avions. L'éventuelle modification de cap de l'avion courant est donnée, à chaque pas de temps, par un réseau de neurones, dont les entrées peuvent être définies des deux manières suivantes :

- On calcule les entrées du réseau de neurones à partir des données correspondant à l'intrus menaçant le plus proche de l'avion courant. On ne tient pas compte d'un intrus qui ne serait pas menaçant. On dira qu'on utilise alors un réseau à un intrus menaçant.
- On calcule ces entrées à partir de celui des deux intrus qui est le plus proche de l'avion courant au pas de temps courant, qu'il soit ou non menaçant. On dira qu'on utilise alors un réseau à un intrus quelconque.

Ces deux méthodes peuvent mener à des problèmes de discontinuité, en ce sens qu'il peut arriver que d'un pas de temps à l'autre, l'intrus le plus proche (ou l'intrus menaçant le plus proche) ne soit plus le même. Ces problèmes nous semblent inévitable si on veut ne tenir compte que d'un intrus.

Pour les conflits à deux avions, lorsque les deux avions peuvent rejoindre leurs destinations sans conflit, à partir de leurs positions courantes, ils prennent un cap direct vers cette destination (voir section 5.3.2). Le fait de prendre en compte les deux intrus systématiquement, ou seulement les éventuels intrus menaçants, pour déterminer à partir de quel avion sont calculées les entrées du réseau de neurones permettant de générer une trajectoire pour l'avion courant, mène à deux traductions différentes de cette idée dans le cadre de la résolution de conflits à trois avions :

**Prise en compte des seuls intrus menaçants :** Supposons que l'avion courant soit l'avion $a_i$. Au pas de temps courant, trois possibilités se présentent :

- Les deux autres avions sont menaçants pour $a_i$. Dans ce cas, le changement de cap de l'avion $a_i$ est donné par la sortie du réseau de neurones, dont les entrées sont calculées à partir des position, cap et destination de l'avion le plus proche.

- Un seul des deux autres avions est menaçant : le changement de cap est donné par la sortie du réseau, dont les entrées sont calculées à partir de cet intrus menaçant.

- Aucun des deux autres avions n'est menaçant. Dans ce cas $a_i$ ne tient compte de la présence d'aucun autre avion : il prend donc un cap direct vers sa destination.

Supposons maintenant qu'au pas de temps courant, ni $a_k$ ni $a_j$ ne soient menaçants pour $a_i$, mais soient menaçants l'un pour l'autre. L'avion $a_i$ prend alors un cap direct vers sa destination, mais $a_k$ et $a_j$ ne le font pas. L'avion $a_i$ peut se retrouver de nouveau dans une situation telle qu'il ne puisse plus rejoindre sa destination sans entrer en conflit avec un des autres avions (son retour sans conflit n'était assuré que si les autres avions prenaient aussi un cap direct vers leurs trajectoires). Il devra alors de nouveau avoir recours au réseau de neurones. L'argument utilisé dans la section 5.3.2, selon lequel le retour direct assurait à l'avion la trajectoire optimale à partir de se position courante (la plus courte des trajectoires sans conflit) ne tient plus, on n'est plus assuré que la trajectoire suivie par l'avion sera une ligne droite, donc la plus courte des trajectoires possibles, puisque l'avion peut être dévié de nouveau.

**Prise en compte systématique des deux intrus :** Puisque l'avion courant prend systématiquement en compte les deux autres avions, même s'ils ne sont pas menaçants, il nous a semblé plus naturel de considérer qu'il ne prend un cap direct vers sa position finale que quand les trois avions peuvent rejoindre directement leurs positions finales en restant deux à deux séparés. Dans ce cas les trois avions prennent ce cap de retour simultanément. Ils peuvent donc atteindre leurs destinations respectives sans conflit, la route directe est en effet la route optimale à partir du pas de temps courant. Comme pour les conflits à deux avions, ce retour direct ne nécessite pas de coordination d'action entre les avions, si on suppose que chacun des trois avions connaît à chaque instant la position, le cap, la vitesse, et la destination de chacun des deux autres avions impliqués dans le conflit.

Le fait de ne prendre en compte que les avions menaçants pourrait, au vu de ce qui précède, paraître une méthode inadaptée au problème à résoudre. Elle présente cependant un avantage par rapport aux deux autres méthodes que nous présentons ici (réseau à un intrus avec prise en compte des deux intrus, menaçants ou non, et réseau à deux intrus). Elle ne nécessite pas que le conflit soit identifié comme un conflit à trois avions. À chaque pas de temps, l'avion courant détermine quels sont les intrus menaçants et tient compte du plus proche d'entre eux. Il adopte en fait la même stratégie qu'il soit impliqué dans un conflit à deux trois, ou même plus d'avions. Cette méthode nous semble donc plus généralisable. Il s'agit cependant là d'un avantage qui ne pourra pas être immédiatement ni simplement mis en pratique : l'apprentissage d'un réseau de neurones qui permettrait, dans le cadre de cette stratégie, de fournir des trajectoires d'évitement satisfaisantes quelles que soient le nombre d'intrus éventuels ne manquera pas de poser des problèmes.

### 5.5.4 Réseaux de neurones à deux intrus

Une autre façon d'étendre à la résolution de conflits à trois avions les méthodes présentées dans les sections 5.3 et 5.4 est d'utiliser un réseau prenant en entrée des paramètres décrivant la situation de cet avion courant, non plus par rapport à un seul des deux intrus, mais par rapport à ces deux intrus.

À chaque pas de temps, l'intrus le plus proche de l'avion courant, que nous appellerons *premier intrus* fournira des entrées du réseau de neurones, et l'autre intrus (le plus éloigné de l'avion courant), que nous appellerons *second intrus* en fournira d'autres. Nous avons donc utilisé un réseau reprenant le principe de celui qui est présenté dans la section 5.3.3, mais en ajoutant les entrées relatives au second intrus.

Rappelons que le réseau de neurones décrit dans la section 5.3.3 comportait, en plus d'une entrée fixée à 1.0 (correspondant à des poids donnant les seuils des fonctions sigmoïdes utilisées par les neurones de la couche cachée), 8 entrées :

- Les deux premières étaient calculées à partir de l'angle entre la route directe de l'avion vers sa destination et son cap.
- La troisième, à partir de la différence de vitesses entre les deux avions.
- La quatrième, à partir de la distance entre les deux avions.
- La cinquième, à partir de la vitesse relative de l'intrus par rapport à l'avion courant.
- La sixième et la septième, à partir de l'angle entre le cap direct de l'avion courant vers sa destination et l'axe allant de l'avion courant à l'intrus.
- La huitième, à partir de l'angle entre les trajectoires directes des deux avions.

Les deux premières entrées sont donc calculées uniquement à partir de caractéristiques de l'avion courant ; les six autres utilisent des caractéristique de l'intrus.

Le réseau de neurones à deux intrus comporte ces 6 entrées pour chacun des deux intrus. Il utilise donc 14 entrées : les deux premières, n'utilisant que des caractéristiques de l'avion courant, sont identiques à celles des réseaux utilisés pour des conflits à deux avions. Les six suivantes sont semblables à celles de ces réseaux, et sont calculées à partir du premier intrus, tandis que les six dernières sont calculées à partir du second intrus.

Le fait de distinguer l'avion le plus proche de l'avion le moins proche pour faire correspondre certaines des entrées du réseau de neurones à l'un et les autres entrées à l'autre peut introduire une forme de discontinuité, si l'avion qui était le plus proche de l'avion courant à un pas de temps ne l'est plus au pas de temps suivant. Ce choix nous a semblé plus naturel pourtant, car il donne un critère pour faire correspondre un avion à un jeu d'entrée. Deux autres versions d'un réseau à deux intrus mériteraient cependant d'être envisagées :

- On pourrait affecter certaines entrées à l'un, quelconque, des deux intrus, les autres entrées, à l'autre, et s'y tenir tout au long de l'évitement. Cela éviterait d'éventuelles discontinuités, mais obligerait à ne plus considérer que deux configurations de conflits se déduisant l'une de l'autre par un changement de numérotation des avions sont identiques (voir définition 5.13).
- On pourrait utiliser des réseaux dont les poids correspondant aux entrés traduisant la situation du premier intrus soient systématiquement identiques à ceux correspondant aux entrées

traduisant celle du premier intrus. Une telle technique s'apparenterait aux *convolutional neural networks* introduits par Y. LeCun pour la reconnaissance d'écriture [LB94] La présence de poids identiques dans différentes parties du réseaux de neurones permettant de rendre la réponse du réseau moins sensibles aux modifications locales (translations, rotations distorsions) de l'image donnée en entrée. Outre d'éviter ce problème de discontinuité, cela permettrait de réduire le nombre de poids à optimiser.

Les réseaux de neurones sont ici encore des réseaux multi-couches à une couche cachée. Cette couche cachée contient 30 neurones. Le réseau prenant 14 entrés, cela donne, en comptant les poids correspondant aux seuils de fonctions sigmoïdes utilisées par les neurones, un total de 481 poids. Les chromosomes manipulés par l'algorithme génétique sont donc des vecteurs de 481 réels (contre 251 dans le cas de réseaux à un intrus, voir section 5.3.2). La sortie du réseau de neurones est utilisée pour calculer une modification de cap pour l'avion courant de la même façon que dans le cas des conflits à deux avions (voir équation 5.5).

À chaque pas de temps, on vérifie que, si les trois avions suivaient une route directe à partir de leurs positions courantes, ils n'atteindraient pas leurs destinations sans conflit. S'ils peuvent atteindre leurs destinations en suivant ces routes directes, ils prennent un cap direct vers ces destinations. Sinon, chaque avion utilise le réseau de neurones pour calculer une modification de son cap. Ces conditions d'utilisation sont donc identiques à celles du réseau à un intrus quelconque décrit dans la section 5.5.3.

## 5.6 Résultats obtenus sur les conflits à trois avions

Nous avons testé les trois méthodes décrites dans les section 5.5.3 et 5.5.4, en réalisant, pour chacune de ces trois méthodes, l'apprentissage successivement au moyen d'une base d'apprentissage fixe et d'une base d'apprentissage renouvelée (voir section 5.3.7). Dans le cas de l'apprentissage sur une base fixe, le nombre de configurations de conflit présentes dans la base a été fixé à $N_a = 100$. Dans le cas d'une base d'apprentissage renouvelée, le nombre de configurations présentes dans la base a été fixé à $N_a = 25$ (sur des premiers tests effectuées pour différentes valeurs de $N_a$, les meilleurs résultats ont été obtenus avec celle-ci).

Les autres paramètres de l'algorithme génétique sont les mêmes que dans le cadre des conflit à deux avions (voir section 5.4.1).

Dans un premier temps, nous avons utilisé, pour l'apprentissage de chacun des trois types de réseaux de neurones que nous souhaitions tester, une base d'apprentissage fixe, composée de 100 configurations de conflit.

Les réseaux ainsi obtenus ont des capacités de généralisation très médiocres : sur $10\,000$ configurations de conflit générées aléatoirement, plusieurs centaines mènent à des violations des contraintes de séparation. Ce résultat, connaissant les résultats obtenus avec des bases d'apprentissage fixes pour des conflits à deux avions (voir section 5.4.2), n'a rien de surprenant.

Cependant, les résultats obtenus sur la base d'apprentissage elle-même sont bons.

Le tableau 5.4 montre les résultats obtenus *sur les 100 configurations de la base d'apprentissage*. Sur ces 100 configurations, tous les conflits ont été résolus. Ces résultats permettent non pas de juger de la qualité des réseaux de neurones en terme de généralisation, mais de sa capacité à traiter des configurations de conflits sur lesquelles l'apprentissage est réalisé. Un comportement correct sur ces configurations est nécessaire pour que l'on puisse considérer que les réseaux de neurones ont la capacité répondre au problème posé. Les apprentissages de ces trois réseaux ont été réalisés sur la même base d'apprentissage, c'est-à-dire sur les 100 mêmes configurations de conflit.

TAB. 5.4: *Base d'apprentissage fixe : retards sur les configurations d'apprentissage ; nombre de configurations (sur les 100 présentes dans la base d'apprentissage) pour lesquelles les réseaux ont généré des trajectoires sans conflit, réparties sur les lignes selon le retard moyen par avion.*

| Retards | Réseau à un intrus menaçant | Réseau à un intrus quelconque | Réseau à deux intrus |
|---|---|---|---|
| de 0 à 5 $s$ | 2 | 2 | 2 |
| de 5 à 10 $s$ | 4 | 5 | 4 |
| de 10 à 20 $s$ | 26 | 18 | 18 |
| de 20 à 30 $s$ | 31 | 25 | 21 |
| de 30 à 60 $s$ | 26 | 25 | 36 |
| de 1 à 2 $mn$ | 11 | 21 | 15 |
| de 2 à 3 $mn$ | 0 | 4 | 4 |
| retard moyen | $30,4\ s$ | $44,5\ s$ | $40,3\ s$ |

Le tableau 5.4 est organisé selon le même principe que le tableau 5.1 : les 7 premières lignes correspondent à des fourchettes de ce retard, les colonnes, à chacun des trois réseaux testés, c'est-à-dire, ici, le réseau à un intrus menaçant, le réseau à un intrus quelconque et le réseau à deux intrus. Les nombres portés donnent le nombre de configurations sur lesquelles le réseau correspondant à la colonne mène à un retard correspondant à la ligne.

La dernière ligne donne, pour chaque réseau, la valeur moyenne des retards sur les 100 configurations d'apprentissage.

Les retards obtenus sont, même sur les configurations de conflit sur lesquelles l'apprentissage des réseaux a été mené, bien plus importants que les retards qui étaient obtenus pour des conflits à deux avions. Cependant, ces retards restent raisonnables.

Les trois réseaux testés mènent à des retards du même ordre de grandeur. On constate que c'est le réseau à un intrus menaçant qui mène aux retards les plus faibles. La stratégie correspondant à ce réseau, en terme de retour direct vers la destination peut se résumer de la manière suivante : l'avion $a_i$ prend un cap direct vers sa destination s'il peut rejoindre cette destination sans entrer en conflit avec aucun des deux autres avions, au cas où tous les avions prennent un cap direct vers leurs destinations. Comme on l'a vu dans la section 5.5.3, il est possible qu'alors les deux autres avions soient l'un pour l'autre menaçants, et qu'ils ne prennent pas un cap direct vers leurs destinations. L'un d'eux peut ainsi redevenir menaçant pour $a_i$, qui devra alors de nouveau faire appel au réseau de neurones pour calculer une modification de son cap. Rien ne garantit l'optimalité de la trajectoire de l'avion $a_1$. L'effet négatif de cette stratégie est qu'après avoir pris un cap direct, l'avion $a_i$ peut être amené à être dévié de nouveau. Cependant, cette stratégie a aussi un effet positif : elle permet à l'avion $a_1$ de prendre un cap direct plus tôt que la stratégie correspondant aux autres réseaux, pour lesquels les trois avions ne prennent un cap direct vers leurs destinations que dans le cas où ils peuvent tous trois atteindre ces destinations en restant deux à deux séparés.

Sur les configurations de la base d'apprentissage, l'effet positif de cette stratégie semble l'emporter sur l'effet négatif. Cela peut s'expliquer par le fait que sur ces configurations, les réseaux de neurones fournissent des trajectoires relativement bonnes pour les avions, en ce sens que ceux-ci ne sont que légèrement déviés de leurs routes directes. Un avion prend un cap direct vers sa destination dès qu'il peut l'atteindre sans conflit, à supposer que les autres avions prennent aussi des caps directs vers leurs destinations. Si les deux autres avions sont déviés, il est tout de même plus probable que le premier avion atteigne en effet sans conflit sa destination si ces déviations sont légères, c'est-à-dire si les trajectoires de ces deux autres avions restent très proches des trajectoires directes, qui assurent qu'ils restent séparés du premier.

Ces résultats, obtenus sur la base d'apprentissage elle-même, montrent que les réseaux, même s'ils mènent à des retards sensiblement plus importants que dans le cas de conflits à deux avions, peuvent résoudre de manière correcte des conflits à trois avions. Cependant, contrairement à ce qu'on observe dans le cas de conflit à deux avions, l'utilisation d'une base d'apprentissage renouvelée ne permet pas d'obtenir des résultats véritablement satisfaisants en termes de capacité de généralisation des réseaux. Nous présentons les résultats obtenus dans les tableaux 5.5 et 5.6.

TAB. 5.5: *Base d'apprentissage renouvelée : violations des contraintes de séparation ; nombre de configurations (sur* $10\,000$*) pour lesquels le conflit n'a pas été résolu, réparties sur les lignes selon la gravité de la violation des contraintes de séparation.*

| Distance minimale | Réseau à un intrus menaçant | Réseau à un intrus quelconque | Réseau à deux intrus |
|---|---|---|---|
| de $3,75$ à $4$ $NM$ | 8 | 9 | 1 |
| de $3,5$ à $3,75$ $NM$ | 16 | 5 | 3 |
| de $3$ à $3,5$ $NM$ | 12 | 1 | 0 |
| de $2$ à $3$ $NM$ | 5 | 2 | 1 |
| de $1$ à $2$ $NM$ | 1 | 0 | 1 |
| de $0$ à $1$ $NM$ | 0 | 0 | 0 |
| moyenne | $3,4$ $NM$ | $3,6$ $NM$ | $3,2$ $NM$ |
| total | 42 | 17 | 6 |

Comme on le voit sur le tableau 5.5, les capacités de généralisation des réseaux obtenus, en termes de respect des contraintes de séparation ne sont pas parfaites. Dans le cas du réseau à deux intrus, les résultats sont cependant assez bons, puisqu'il n'y a violation des contraintes de séparation que dans 6 cas sur $10\,000$. Même dans le cas du réseau à un intrus, avec lequel on obtient les moins bons résultats (cependant acceptables : violation des contraintes dans $42$ cas sur $10\,000$), on n'a une distance minimale entre les avions inférieure à 3 NM que dans 6 cas sur $10\,000$.

Cependant, ces résultats ne suffisent pas à conclure que les capacités de généralisation des réseaux sont bonnes. En effet, mêmes si les trajectoires proposées par les réseaux de neurones respectent les contraintes de séparation, elles peuvent mener à des retards très importants, correspondant à des détours injustifiés. De telles trajectoires ne peuvent être considérées comme des solutions correctes du problème de l'évitement.

TAB. 5.6: *Base d'apprentissage renouvelée : retards quand la séparation est assurée ; nombre de configurations (sur* $10\,000$*) pour lesquelles les réseaux ont généré des trajectoires sans conflit, réparties sur les lignes selon le retard moyen par avion.*

| Retards | Réseau à un intrus menaçant | Réseau à un intrus quelconque | Réseau à deux intrus |
|---|---|---|---|
| de 0 à 5 $s$ | 211 | 204 | 159 |
| de 5 à 10 $s$ | 463 | 391 | 282 |
| de 10 à 20 $s$ | 1 719 | 1 276 | 952 |
| de 20 à 30 $s$ | 2 485 | 1 724 | 1 608 |
| de 30 à 60 $s$ | 3 591 | 3 646 | 3 728 |
| de 1 à 2 $mn$ | 1 089 | 2 137 | 1 969 |
| de 2 à 3 $mn$ | 256 | 315 | 478 |
| de 3 à 4 $mn$ | 89 | 110 | 550 |
| de 4 à 5 $mn$ | 31 | 51 | 78 |
| de 5 à 10 $mn$ | 24 | 114 | 136 |
| de plus de 10 $mn$ | 0 | 15 | 54 |
| retard moyen | $40,1\ s$ | $53,6\ s$ | $67,9\ s$ |
| total | 9 958 | 9 983 | 9 994 |

Le tableau 5.6 montre les retards obtenus sur les configurations sur lesquelles les contraintes de séparation étaient respectées. On voit que dans le cas du réseau à deux intrus, plus de 800 configurations sur 10 000 mènent à un retard moyen par avion supérieur à 3 minutes, et presque 200 mènent à un retard de plus de 5 minutes, dont 54, à un retard supérieur à dix minutes. Ces retards apparaissent sur un trajet qui, en l'absence d'évitement, est effectué en 20 minutes.

La situation est moins mauvaise en ce qui concerne le réseau à un intrus quelconque : 296 configurations mènent à un retard de plus de 3 minutes, 15, à un retard de plus de 10 minutes, et sensiblement meilleure dans le cas d'un réseau à un intrus menaçant : 144 retards supérieurs à 3 minutes, aucun supérieur à dix minutes.

Nous pouvons interpréter ces résultats de la manière suivante :

**Réseau à un intrus menaçant :** Ce réseau se caractérise par de meilleurs résultats que les autres sur les configurations d'apprentissage, pour une base d'apprentissage fixe, et, pour une base d'apprentissage renouvelée, de moins bons résultats en termes de violation des contraintes mais de meilleurs résultats en termes de retard.

On a vu qu'on pouvait expliquer ainsi les meilleurs résultats obtenus sur les configurations d'apprentissage : tant que les déviations des avions restaient légères, le fait qu'un avion prenne un cap de retour direct en se basant sur les trajectoires de retour direct des autres avions pouvait se révéler utile parce que leurs trajectoires déviées restaient proche de leurs trajectoires directes.

Pour ce qui est des capacités de généralisation, les bons résultats en termes de retard nous semblent dûs au fait que le réseau de neurones n'a à prendre en compte que la présence d'un autre avion, proche, et menaçant. Les différentes configurations de conflit à trois avions mènent de ce fait à des situations moins variées, pour le réseau de neurones, que dans le cas d'un réseau à deux intrus.

Le plus grand nombre de violations des contraintes de séparation peut être dû au fait que, quand un avion a pris un cap direct vers sa destination et doit, ensuite, avoir de nouveau recours au réseau de neurones parce qu'un avion redevient menaçant envers lui, il est possible qu'alors cet autre avion soit très proche de lui, c'est-à-dire que la distance entre les deux avions ne soit que très légèrement supérieure à la norme de séparation. Il peut être alors très difficile d'éviter le conflit.

**Réseau à un intrus quelconque :** La différence entre le comportement du réseau à un intrus menaçant et le réseau à un intrus quelconque est que ce dernier est meilleur du point de vue du respect des contraintes de séparation, moins bon du point de vue des retards.

Ces meilleurs résultats du point de vue du respect des contraintes de séparation peuvent s'expliquer par le fait que les avions ne prennent un cap direct vers leurs destinations que quand ils sont assurés de pouvoir en effet atteindre ces destinations sans avoir à être de nouveau déviés. Il n'arrive pas, comme dans le cas d'un conflit à un avion menaçant, que deux avions redeviennent menaçants l'un pour l'autre en étant trop proches l'un de l'autre.

Les retards plus importants peuvent s'expliquer par le fait que les avions continuent plus longtemps de suivre une trajectoire fournie par les réseaux de neurones. Si les réseaux de neurones fournissaient réellement des trajectoires optimales, cela ne serait pas un problème, mais il est clair que la réponse apportée par le réseau de neurones, dans le cas d'un conflit à trois avions, reste très imparfaite.

**Réseau à deux intrus :** Le réseau de neurones prend systématiquement en entrées des données correspondant aux deux avions. La diversité des situations des avions les uns par rapport aux autres est donc beaucoup plus importante, l'espace sur lequel la capacité de généralisation du réseau de neurones est testée est plus vaste, la généralisation est donc plus difficile à obtenir que dans le cas de conflits à deux avions.

De plus la taille plus importante des réseau de neurones correspond à un plus grand nombre de poids à faire évoluer. Les *chromosomes* manipulés par l'algorithme génétique correspondent à ces poids : ce sont dans le cas du réseau à deux intrus des vecteurs de dimension 481, alors que dans le cas des réseaux de neurones à un intrus ce sont des vecteurs de dimension 251. Cet accroissement de la dimension des *chromosomes* peut expliquer une moins grande efficacité de l'algorithme génétique.

Le nombre faible de configurations de conflit sur lesquelles les contraintes de séparation sont violées peut s'expliquer par le fait que les avions sont toujours systématiquement pris en compte par le réseau.

On retrouve ainsi, seulement dans le cas du réseau à deux intrus, une caractéristique de l'utilisation des réseaux de neurones pour la résolution de conflit à deux avions. Dans ce dernier cas, les trajectoires des deux avions sont générées par des réseaux de neurones identiques. La trajectoire de chaque avion dépend à travers un réseau de neurones, de celle de l'autre, qui, elle, dépend de la sienne, à travers un réseau de neurones identique. Sans coordination d'action, les deux avions ont des trajectoires *cohérentes*. Rappelons, par exemple, le comportement des réseaux utilisés pour résoudre des conflits à deux avions et ayant réalisé leur apprentissage sur des configurations de conflits positives pour une fonction signe donnée (section 5.4.4). Les réseaux de neurones semblent alors véritablement suivre une stratégie comme *les deux avions tournent dans la même direction*, ou *les deux avions tournent dans* *des directions opposées*. L'efficacité des modifications de cap d'un avion dépend du fait que celles de l'autre avion suivent la même *stratégie*.

Dans le cas du réseau à deux intrus, le trajectoire de chaque avion dépend de celle des deux autres, d'une manière comparable. Il est possible qu'on retrouve cette *cohérence* des trajectoires, ce qui peut expliquer que même si les résultats obtenus sont médiocres en termes d'efficacité, la séparation est presque systématiquement assurée (dans $99,96\%$ des cas).

Dans le cas des réseaux à un intrus, la trajectoire de l'avion $a_i$ peut dépendre, pendant un certain temps de celle de l'avion $a_j$, puis de celle de l'avion $a_k$. Quand l'avion $a_i$ tient compte de l'avion $a_k$ pour modifier son cap, celui-ci peut tenir compte de l'avion $a_j$, et non pas de l'avion $a_i$. La cohérence des trajectoires suivies par les avions, dans le cas d'un conflit à deux avions du fait qu'ils *tiennent compte l'un de l'autre*, à travers des réseaux identiques, n'a plus lieu d'être, même si les réseaux utilisés par les avions sont encore identiques.

## 5.7 Conclusion

Comme nous l'avons vu en introduction, un des principaux objectifs à atteindre pour les méthodes présentées dans ce chapitre était d'obtenir un système possédant de bonnes capacités de généralisation, c'est-à-dire qui soit capable de fournir des trajectoires d'évitement de bonne qualité pour des cas de conflit admissibles quelconques.

Le renouvellement aléatoire des configurations de conflit, avec utilisation des notions de fiabilité et de mémoire pour le calcul de la fonction d'adaptation des réseaux de neurones (section 5.3.7), permet en effet, dans le cas de la résolution de conflit à deux avions, d'obtenir de bons résultats en termes de capacité de généralisation des réseaux de neurones.

Dans le cadre de la résolution de conflits impliquant trois avions, les résultats obtenus sont moins bons. Il semble même que dans ce cas on se heurte à un problème dû à l'utilisation de la notion de fiabilité, qui favorise les réseaux de neurones ayant résolu le plus grand nombre de conflits différents : on obtient en effet des réseaux permettant d'éviter les conflits sur un très grand nombre de configurations générées aléatoirement, mais parfois au prix d'un retard très important.

La coévolution, que nous présentons dans le chapitre suivant, nous a semblé être un moyen alternatif d'obtenir une bonne capacité de généralisation des réseaux de neurones, éventuellement sans utilisation de la fiabilité des réseaux de neurones dans le calcul de leur fonction d'adaptation. Une première approche de cette méthode d'apprentissage, sur des cas de conflits à deux avions est présentée dans le chapitre suivant. Comme nous le verrons, cette première tentative n'est pas apparue suffisamment concluante pour l'étendre à la résolution de cas de conflit à trois avions.

Les bons résultats obtenus dans le cadre des conflits à deux avions sont cependant suffisamment encourageants pour que l'axe de recherche représenté par l'utilisation des réseaux de neurones pour l'évitement aérien continue d'être exploré. Une des prochaines étapes dans cette direction nous semblent, en plus de continuer à tenter de résoudre des conflits impliquant un plus grand nombre d'avions, devoir être l'évaluation de la robustesse de telles méthodes, notamment à travers l'étude du comportement d'un réseau de neurones devant maintenir un avion séparé d'un ou de plusieurs autres avions dont la trajectoire n'est pas une trajectoire d'évitement générée par un réseau de neurones identique.

Les problèmes rencontrés par les réseaux de neurones à un intrus (voir section 5.6) dans le cadre de la résolution de conflits à trois avions montrent en effet que les réseaux de neurones obtiennent de moins bons résultats lorsque les actions des différents avions ne sont plus cohérentes (dans le cadre de la résolution de conflits à trois avions au moyen de réseaux à un intrus, cette absence de cohérence n'est pas due au fait que les trajectoires des deux avions ne sont pas générées par des réseaux identiques, mais au fait qu'elles sont générées par des réseaux ayant des entrées qui ne sont plus cohérentes : un avion peut tenir compte d'un deuxième avion, qui, lui, tient compte d'un troisième avion).

Il serait intéressant aussi d'étudier le comportement du réseau de neurones quand les entrées dont il dispose (traduisant se situation et celle des autres avions impliquées dans le conflit) sont bruitées ou incomplètes.

Les modes d'apprentissage des réseaux devront éventuellement être adaptés de manière à assurer un bon comportement des réseaux dans ces cas (intrus ne suivant pas de trajectoires *cohérentes*, données bruitées ou incomplètes).

Enfin, de telles méthodes, comme celles proposées par Karim Zeghal, présentent l'inconvénient de générer des trajectoires qui non seulement seraient difficiles à transmettre et à suivre (changements de cap continus, ou presque), mais qui de plus seraient pénibles pour les passagers des avions. Un travail de simplification des trajectoires proposées (tout en assurant encore la séparation des avions) sera donc nécessaire si l'on envisage d'utiliser ce type de méthodes dans le cadre de systèmes d'évitement embarqués.

# Chapitre 6

# Coévolution des réseaux de neurones et des configurations de conflit

## 6.1 Introduction

Un des problèmes que pose l'utilisation de réseaux de neurones pour l'évitement est l'obtention de bonnes capacités de généralisation (voir chapitre 5). L'introduction des notions de mémoire et de fiabilité, conjointement à un renouvellement aléatoire de la population (voir section 5.3.7, permet d'obtenir, du moins sur des conflits n'impliquant que deux avions, des résultats très satisfaisants (voir section 5.4.3).

Cette méthode d'apprentissage n'a cependant pas permis d'obtenir des résultats pleinement convaincants sur des conflits impliquant trois avions. Ils nous a semblé intéressant d'essayer aussi une autre méthode pour obtenir une bonne capacité de généralisation des réseaux, la *coévolution*. Le principe le plus général de la coévolution peut être décrit comme suit.

Deux populations évoluent parallèlement. Par *évoluent*, on entend qu'elles sont soumises toutes les deux aux étapes successives d'un algorithme génétique : sélection, mutation, éventuellement croisement (voir section 4.1.1). Ainsi, appliquer ce principe au problème de l'évitement consistera à faire évoluer :

- Une population de réseaux de neurones.
- Une population de configurations de conflit.

Les fonctions d'évaluation des éléments d'une population utilisent les éléments de l'autre population : le *but* des réseaux est d'obtenir de bons résultats sur les configurations de conflit, et celui des configurations est d'être difficiles pour les réseaux de neurones, c'est-à-dire qu'ils obtiennent sur elles de mauvais résultats.

Nous avons tenté d'appliquer ce principe à la résolution de conflits impliquant deux avions, avec bien entendu l'arrière pensée de l'adapter à la résolution de conflit à trois avions, problème sur lequel le renouvellement aléatoire des configurations d'apprentissage avec mémoire est moins performant que dans le cas de conflits à deux avions. Cependant, les résultats que nous avons obtenus sur les conflits à deux avions avec les deux versions d'un algorithmes de coévolution que nous avons utilisées (décrites dans les sections 6.3.2 et 6.3.3) ne nous ont pas paru suffisamment probants pour que nous envisagions de les appliquer au problème, plus complexe, de la résolution de conflits à trois avions (voir section 6.4.2).

Les travaux présentés ici sont une première approche de l'application de la coévolution au problème de l'apprentissage de réseaux de neurones pour l'évitement. Nous les avons menés pour tenter de comprendre quel rôle pourrait avoir la coévolution sur ce problème. Ils ne prétendent en aucune manière apporter de réponse définitive sur ce point. D'autres pistes sans doute pourraient être explorées pour introduire une forme de coévolution dans l'apprentissage de tels réseaux de neurones.

## 6.2 Quelques travaux existant sur la coévolution

Le principe général de la coévolution décrit dans la section précédente se prête à des réalisations pratiques pouvant être très variées. Il n'existe pas d'algorithme de coévolution *canonique*. Nous présentons dans les sections suivantes différents travaux ayant utilisé ce principe. Ces descriptions rapides permettront de donner une image plus concrète de ce que peut être la coévolution.

### 6.2.1 Travaux de W. D. Hillis

Hillis a été, à notre connaissance, un des premiers à utiliser des techniques de coévolution du type *proie-prédateur*. Ce nom provient du fait qu'il s'agit de faire évoluer simultanément, mais de manière concurrente, deux populations. Hillis emploie le mot de *parasites* pour désigner les éléments de la population concurrente, qui en exploitant les failles éventuelles des éléments de la première population, les poussent à s'améliorer.

Dans son article de 1992, *Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure* [Hil92], Hillis présente une méthode dont le but est de générer des réseaux de tri, parvenant à trier un nombre $n$ donné de nombres au moyen du plus petit nombre possible de comparaisons successives. Les éléments de la première population sont des reseaux de tri, qui évoluent au moyen d'un algorithme particulier. Nous développerons les caractéristiques de cet algorithme qui nous semblent jouer un rôle important dans l'utilisation par Hillis de la coévolution. Les *parasites* sont des ensembles de *cas tests*, c'est-à-dire des séquences d'entiers (de 0 et de 1 puisque un réseau de tri qui trie correctement toute séquence de 0 et de 1 de taille $n$ trie correctement toute sequences de nombres de taille $n$).

Hillis utilise un algorithme génétique de type *diploïde* : les chromosomes sont doubles, et le croisement s'effectue en formant un chromosome au moyen de deux demi-chromosomes. Cela permet d'utiliser les notions de gènes dominants ou récessifs dans le but, entre autres, d'obtenir des individus de taille variable, donc ici des réseaux de tri utilisant un nombre variable de comparaisons. Cet aspect de l'algorithme ne semble pas jouer un rôle majeur dans l'utilisation de la coévolution.

La répartition *géographique* de la population joue, elle, semble-t-il, un rôle important dans l'adaptation à la coévolution de l'algorithme de Hillis.

Les individus sont initialement répartis aléatoirement sur une grille bidimensionnelle. Les opérateurs de sélection et de croisement sont locaux : les individus sont croisés avec des éléments proches d'eux sur la grille, la sélection consiste à comparer deux éléments voisins et à remplacer celui des deux dont la fonction d'adaptation est la plus faible par celui dont la fonction d'adaptation est la meilleure : le moins bon disparaît, le meilleur est présent deux fois sur la grille (mais sera modifié ensuite par les croisements et les mutations). Une succession aléatoire d'échanges de lignes ou de colonnes adjacentes est appliquée de manière à ce que les individus se déplacent lentement sur la grille, en suivant une marche aléatoire.

Quand Hillis utilise pour l'apprentissage des réseaux de tri l'évolution *simple*, c'est-à-dire sans coévolution, les réseaux sont évalués au moyen d'un échantillon de cas tests. Les réseaux ont une adaptation d'autant plus grande qu'ils parviennent à ordonner correctement une plus grande proportion de ces cas tests

La coévolution consiste dans ce cadre a faire évoluer, en même temps que la population de réseaux, une population de *parasites*. Un élément de cette population représente un petit nombre (10 ou 20) de cas tests. Les parasites et les réseaux de tri évoluent sur la même grille. Un réseau (resp. un parasite) utilise pour calculer sa fonction d'évaluation le parasite (resp. le réseau) situé sur le même point de la grille que lui.

L'adaptation d'un réseau est alors donnée par le nombre des cas tests contenus dans le parasite correspondant au même emplacement sur la grille, que ce réseau ordonne correctement. Inversement l'adaptation d'un parasite est donnée par le nombre des cas test (parmi ceux qu'il contient) que le réseau correspondant ne parvient pas à ordonner correctement.

L'importance dans leurs évolutions respectives et dans leur interaction de l'emplacement des réseaux et des cas tests joue sans doute un rôle majeur dans l'efficacité des mécanismes de la coévolution telle qu'elle est utilisée par Hillis. Les parasites peuvent ainsi s'adapter aux réseaux qui évoluent dans la même région de la grille qu'eux (si des réseaux semblables sont concentrés sur la même région de la grille, ce qui est possible compte tenu de la sélection et du croisement utilisés, ils formeront une cible plus facile pour les *parasites*, tandis que les déplacements aléatoires des éléments des deux populations sur la grille permet de compenser les effets de cette adaptation, qui sans cela pourrait se révéler néfaste).

Hillis obtient grâce à la coévolution de meilleurs résultats que ceux qu'il avait obtenus au moyen de l'évolution simple des réseaux, tant en termes d'efficacité des réseaux pour un plus petit nombre de comparaisons, qu'en terme de rapidité d'obtention de ces réseaux.

Il voit à ce résultats deux explications. D'abord, les réseaux imparfaits (qui ne trient pas correctement toutes les séquences de $n$ entiers), qui pourraient constituer des optima locaux sont la proie de parasites adaptés à leurs faiblesses qui se développent dans leur voisinage, et ainsi ne sont pas viables (on voit ici le rôle de la localisation des éléments des deux populations sur une grille). De plus, les cas tests les plus difficiles dominent rapidement dans la population des parasites, ce qui évite des évaluations inutiles de réseaux sur des cas tests trop faciles.

### 6.2.2 Travaux de Jan Paredis

Jan Paredis [Par94, Par96b, Par96a] utilise aussi des méthodes de coévolution du type *prédateur-proie*. Il utilise conjointement un algorithme de coévolution (CGA : Co-evolutionary Genetic Algorithm), et une technique d'apprentissage, qu'il appelle *Life Time Fitness Evaluation* (LTFE). Il donne dans *coevolutionary computation* [Par96a] une présentation générale de son algorithme

CGA tel qu'il peut être appliqué à ce qu'il appelle des *test-solution problems*. c'est-à-dire des problèmes dans lesquels on cherche à obtenir des solutions générales à appliquer à un grand nombre de cas différents, et pour lesquels on peut disposer d'un (grand) nombre de cas tests. La recherche de réseaux de tri décrite dans l'article de Hillis est un cas particulier de problème solution-test, et Jan Paredis développe dans les différents articles cités ici l'application de son algorithme a différents problèmes tels que la recherche de réseaux neuronaux de classification (classification neural networks) [Par96a], ou des problèmes de satisfaction de contraintes (CSP) [Par94, Par96a].

Paredis utilise, de même que Hillis, deux populations : une population de solutions, et une population de tests, qui évoluent simultanément. Chacune d'entre elles utilise l'autre population pour l'évaluation de ses éléments

L'évaluation des éléments des deux populations se fait au moyen de la technique LTFE. Le plus souvent, dans le cas de l'application a un problème du type *solution-test* de l'évolution simple, chaque évaluation d'un individu se fait en une fois, au moment de sa naissance (par croisement ou par mutation), en mesurant les performances de cet individu sur tous les tests disponibles ou sur une partie de ces tests. La LTFE consiste a étaler cette évaluation sur la vie de l'individu, ou du moins sur une partie de sa vie. L'évaluation d'une solution dépend ainsi de son histoire (récente) : au cours de sa vie une solution *rencontre* des tests. Chaque fois qu'un test est rencontré, la solution marque 1 point si elle satisfait ce test, 0 si elle ne le satisfait pas. Une solution est évaluée selon le nombre de points qu'elle a obtenu lors des $N$ dernières rencontre (Paredis suggère de prendre $N = 20$).

Dans le cadre du CGA, les tests sont évalués de la même manière, mais un test marque 1 point si la solution rencontrée ne le satisfait pas, 0 si elle le satisfait. Les solutions et les tests susceptibles de se rencontrer sont sélectionnés en fonction de leur adaptation courante (issue de leurs $N$ dernières rencontres). Périodiquement (après un certains nombre de ces rencontres), un individu nouveau est créé, dans chacune des populations ou seulement dans la population des solutions selon les cas (voir application aux réseaux de classification [Par96a]), à partir d'individus sélectionnés eux aussi en fonction de leur adaptation courante. Le nouvel individu est alors inséré à la population après avoir été évalué au moyen de $N$ rencontres avec des individus sélectionnés dans la population adverse. Lors de la création des populations initiales, les individus sont évalués au moyen de $N$ rencontres avec des individus aléatoirement choisis dans la population adverse.

Paredis applique cet algorithme à l'apprentissage de réseaux neuronaux appliqués à la classification et à un problème de satisfaction de contraintes (CSP). La suite de cette section décrit l'application aux réseaux neuronaux pour la classification [Par96a] :

Pour l'apprentissage de réseaux de neurones de classification, les solutions sont des réseaux de neurones, les tests des points à classifier. Il n'y a pas de création de nouveaux tests : l'algorithme permet d'évaluer les tests existant, au moyen des rencontres. La sélection des individus de la population de tests qui participeront à ces rencontres permet ensuite de donner aux points tests un poids d'autant plus fort dans l'apprentissage des réseaux qu'ils ont mis un plus grand nombre de réseaux en échec : en effet un réseau a d'autant plus de chance de rencontrer un test donné que ce test aura obtenu un score plus élevé lors des $N$ dernières rencontre auxquelles il aura participé.

Paredis obtient au moyen de son algorithme de coévolution des résultats très sensiblement meilleurs que ceux qu'il avait obtenus par un algorithme génétique classique : les réseaux qu'il obtient ont un taux de réussite plus élevé, et sont obtenus plus rapidement (en terme de nombre de réseaux générés avant d'obtenir le meilleur).

Paredis explique ce gain d'efficacité par les effets conjugués de la coévolution et de la technique d'évaluation LTFE : celle-ci permet de gagner du temps en repérant plus vite les solutions qui sont sensiblement meilleures ou moins bonnes que les autres, et la coévolution permet de concentrer l'apprentissage des réseaux sur les points difficiles.

On peut considérer que la principale différence entre l'algorithme proposé par Jan Paredis et celui de Hillis est que dans l'algorithme de Paredis les individus des deux populations (solutions et tests) ne sont pas localisés, alors que dans l'algorithme de Hillis, les points sont localisés sur une grille. Les tests, ou parasites peuvent alors, chez Hillis, s'adapter à un petit groupe de solutions, se spécialiser en quelque sorte. Chez Paredis, les tests ne s'adaptent pas à la population globale des solutions, mais plus précisément aux solutions qui sont les plus adaptées aux tests courants, puisque les solutions qui participent aux rencontres sont sélectionnées.

Cela peut mener à une certaine instabilité, due à des spécialisations successives des deux populations, l'une devenant difficile pour l'autre, qui à son tour devient difficile pour la première.

Ce problème dû à l'instabilité, à l'hostilité même de l'environnement est mis en évidence par J. Paredis dans [Par97], sous le nom de *problème de la reine rouge*[1]. Les éléments de la population des réseaux doivent s'adapter à un environnement qui évolue trop rapidement pour qu'ils aient le temps de s'y adapter.

### 6.2.3 Motivations du recours à la coévolution

Considérons la méthode décrite dans le chapitre 5. Nous nous limitons aux conflits à deux avions. Le but est d'obtenir un réseau de neurones qui, s'il est utilisé dans les conditions décrites dans la section 5.4, génère, *quelle que soit la configuration de conflit* (voir définition 5.3.4), une trajectoire pour chacun des avions impliqués dans le conflit, de telle sorte que :

1. Les deux avions restent séparés tout au long de l'évitement.
2. Les retards engendrés par les déviations nécessaires à la séparation des avions sont les plus faibles possibles.

L'expression *quelle que soit la configuration de conflit* signifie que l'important est d'obtenir un réseau ayant une bonne capacité de généralisation. C'est-à-dire un réseau de neurone dont les résultats tant en terme de séparation qu'en terme de retards, restent bons sur le plus grand nombre possible de configurations de conflit, idéalement sur toutes.

Il est clair que la séparation des avions est la première condition à remplir pour que les performances du réseau sur une configuration soient jugées bonnes. Cependant, on a vu (section 5.6) que, même en respectant les contraintes de séparation, la réponse apportée par le réseau de neurones pouvait être jugée mauvaise si le retard entraîné était trop important.

On peut considérer qu'une configuration sur laquelle le réseau ne satisfait pas ces deux critères (séparation des avions et retards raisonnables) est une configuration difficile pour le réseau.

C'est là que la coévolution nous a semblé pouvoir jouer un rôle utile. Rappelons que les réseaux utilisés sont obtenus au moyen d'un algorithme génétique. Une population de réseaux,

[1] D'après un personnage d'*Alice au pays de merveilles* de L. Caroll : *Here, you see, it takes all the running you can do, to keep in the same place. If you want to go somewhere else, you must run at least twice as fast as that !*

codés par leurs poids, évolue. Ces réseaux sont à chaque génération testés sur des configurations de conflit. Supposons qu'on puisse obtenir que parmi les configurations de conflit utilisées se trouvent justement les plus difficiles pour les réseaux qu'elles permettent d'évaluer. Si les meilleurs réseaux résistent toujours, c'est-à-dire donnent toujours de bons résultats sur ces configurations, ils en seront d'autant plus fiables : si le comportement d'un réseau sur les configurations les plus difficiles pour lui reste bon, il devrait l'être sur toutes les configurations.

## 6.3 Algorithme de coévolution utilisé

Nous avons d'abord utilisé une approche très simple de la coévolution. Cette approche est décrite dans la section 6.3.2. Les résultats décevants obtenus nous ont poussé à modifier la sélection utilisée pour les configurations de conflit, de manière à la rendre mieux adaptée au rôle que nous souhaitions voir jouer à l'évolution des configurations de conflit dans le cadre de l'apprentissage des réseaux de neurones. Ce nouveau mode de sélection est décrit dans la section 6.3.3. Les résultats obtenus avec lui sont exposés dans la section 6.4.

La première étape de l'utilisation d'un algorithme de coévolution pour l'apprentissage de réseaux de neurones dans le cadre de l'évitement aérien est de définir un codage des configurations de conflit qui permette de manipuler une population de ces configurations, ainsi que les opérateurs génétiques qui sont appliqués à cette population.

Les configurations de conflit utilisées répondent, comme dans le chapitre 5, à la définition 5.5. L'initialisation de la population des configurations de conflit s'effectue en générant aléatoirement des configurations de conflit selon la méthode décrite dans la section 5.3.6.

### 6.3.1 L'évolution des configurations de conflit

Nous présentons dans cette section le codage utilisé pour les configurations de conflit, ainsi que l'opérateur de mutation que nous utilisons. Le codage des réseaux de neurones est identique à celui utilisé dans le cadre de la méthode présentée dans la section 5. Il en est de même pour les opérateurs de croisement et de mutation appliqués à la population des réseaux de neurones.

Les avions impliqués dans le conflit sont notés $a_1$ et $a_2$. La vitesse de l'avion $a_i$ est notée $v_i$, son cap initial est noté $c_i$. Nous fixons, comme dans la section 5.3.4, le cap de l'avion $a_1$ : $c_1 = 0$. Une configuration de conflit est alors définie par les données suivantes :

1. La vitesse $v_1$ de l'avion $a_1$.
2. La vitesse $v_2$ de l'avion $a_2$.
3. Le cap $c_2$ de l'avion $a_2$.
4. Le gisement de l'avion $a_2$ par rapport à l'avion $a_1$. Au début de l'évitement, la distance entre les avions est égale à la distance d'alerte $d_a$ définie dans la section 5.3.4, le gisement suffit donc à donner la position relative des avions.

Pour faire évoluer, par un algorithme génétique, une population de configurations, il faut définir des opérateurs de mutation et éventuellement de croisement sur ces configurations.

La mutation porte sur une des quatre données énumérées ci-dessus et définissant une configuration de conflit : vitesses des avions, cap de l'avion $a_2$ et gisement de l'avion $a_2$ par rapport à l'avion $a_1$.

Lorsqu'un individu de la population des configurations doit être muté, on tire aléatoirement (avec distribution uniforme) une des mutations suivantes :

**Mutation de la vitesse :** On choisit aléatoirement un des deux avions. Sa vitesse est modifiée en lui ajoutant une variable aléatoire gaussienne de moyenne nulle et d'écart-type $\sigma_v$. Les vitesses des avions sont comprises entre 300 et 500 kt, et on prend pour la mutation des vitesses : $\sigma_v = 20\ kts$.

**Mutation du cap :** Le cap de l'avion $a_2$ est modifié, en lui ajoutant une variable aléatoire gaussienne de moyenne nulle et d'écart-type $\sigma_c = 10°$.

**Mutation de la position relative des avions :** La distance entre les avions étant égale, au début de l'évitement,à la distance d'alerte $d_a$, la position relative des avions est traduite par le gisement de l'avion $a_2$ par rapport à l'avion $a_1$. Ce gisement est modifié en lui ajoutant une variable aléatoire gaussienne de moyenne nulle et d'écart-type $\sigma_g = 10°$.

Ces modifications peuvent entraîner que les nouvelles trajectoires des avions ne définissent plus un cas de conflit admissible, au sens de la définition 5.5. Il est possible qu'un des deux cas de figure suivants se présente :

- Il n'y a plus de conflit entre les avions avant la fin de la durée de l'évitement $t_f$.
- La distance entre les positions finales des avions en l'absence d'évitement n'est pas supérieure à la norme de séparation $n_h$.

Pour assurer que les nouveaux caps, les nouvelles vitesses, ou les nouvelles positions initiales ainsi obtenues correspondent à des cas de conflit admissibles, et donc puissent définir des configurations de conflit, on procède simplement de la manière suivante :

1. On choisit le type de mutation à effectuer (cap, vitesse ou gisement).
2. On opère cette mutation (ajout d'une valeur aléatoire).
3. Si les trajectoires obtenues ne définissent pas un cas de conflit admissible, retour à l'étape 2.

En pratique, les faibles écarts-types utilisés pour les mutations gaussiennes entraînent que des cas de conflit admissibles sont toujours obtenus après un très faible nombre de tentatives, le plus souvent dès la première.

La définition d'un croisement sur les configurations de conflit pose par contre des problèmes. On peut imaginer un croisement consistant à échanger certaines grandeurs caractéristiques entre deux configurations de conflit (cap de l'avion $a_2$, vitesse d'un des avions, gisement de $a_2$ par rapport à $a_1$), ou à prendre des barycentres d'une de ces grandeurs. Cependant, il n'y a alors aucune raison pour que les trajectoires ainsi obtenues définissent des cas de conflit, et encore moins des cas de conflit admissibles. Nous n'utilisons donc pas d'opérateur de croisement.

De même, aucune distance naturelle n'existant sur l'espace des configurations de conflit, nous n'utilisons pas de sharing sur la population des configurations de conflit.

Pour éviter la convergence prématurée de la population des configurations de conflit due à l'absence de croisement (pouvant jouer un rôle d'exploration *en début d'évolution*) et surtout de sharing, une *petite* fraction de cette population sera renouvelée aléatoirement à chaque génération.

### 6.3.2 Premier algorithme

Dans cette section, notre première tentative d'appliquer la coévolution à l'apprentissage de réseaux de neurones pour l'évitement est décrite. L'algorithme utilisé n'est pas très différent d'un algorithme génétique classique.

On considère deux populations : une population de réseaux de neurones, une population de configurations de conflit. Les deux populations évoluent au moyen d'un algorithme génétique, et n'interagissent qu'au moment de l'évaluation de leurs éléments.

Dans la version la plus simple, l'algorithme se déroule donc de la manière suivante :

1. *Création des populations initiales de réseaux et de configurations de conflit.*
2. *Évaluation des réseaux et des configurations de conflit.*
3. *Application des opérateurs génétiques de sélection, de croisement, de mutation à chaque population.*
4. *Si le critère d'arrêt (nombre maximal de générations atteint) n'est pas vérifié, retour en 2.*

La génération aléatoire des réseaux et des configurations de conflit se fait comme au chapitre précédent. Pour les réseaux, les opérateurs de croisement et de mutation sont identiques à ceux qui sont utilisés dans le chapitre 5. Pour les configurations de conflit, la mutation a été décrite dans la section précédente. L'algorithme n'utilise pas de croisement, mais le renouvellement aléatoire d'une petite fraction de la population.

**Notation –** *Dans toute la suite de ce chapitre, les différentes variables, grandeurs, entités ou fonctions pouvant se rapporter soit aux réseaux de neurones, soit aux configurations de conflit porteront le même nom dans les deux cas, mais avec un "r" en exposant quand elles se rapporteront aux réseaux de neurones, et un "c" quand elles se rapporteront aux configurations de conflit.*

Ainsi, la population des réseaux de neurones sera notée $\mathcal{P}^r$ et celle des configurations de conflit, $\mathcal{P}^c$. Le nombre de réseaux de neurones dans $\mathcal{P}^r$ est noté $N^r$, celui de configurations de conflit dans $\mathcal{P}^c$, $N^c$.

Nous allons maintenant détailler la deuxième étape de l'algorithme décrit ci-dessus.

L'évaluation des réseaux se fait sur les configurations de conflit présentes dans la population $\mathcal{P}^c$ à l'instant de l'évaluation, selon la méthode donnée dans la section 5.3.7. Nous la rappelons brièvement.

On cherche à calculer $f_a^r(\mathcal{R})$, la fonction d'évaluation d'un réseau $\mathcal{R}$. On définit $N_{vt}^r(\mathcal{R})$, la somme sur les configurations présentes dans la population $\mathcal{P}$ des nombres de pas de temps sur lesquels les avions n'ont pas été séparés :

$$N_{vt}^r(\mathcal{R}) = \sum_{i=1}^{N^c} N_v^r(\mathcal{R}, \mathcal{C}_i) \tag{6.1}$$

et $r_t^r(\mathcal{R})$, la moyenne quadratique, sur ces configurations, des retards des avions :

$$r_t^r(\mathcal{R}) = \sqrt{\frac{\sum_{i=1}^{N^c} r(\mathcal{R}, \mathcal{C}_i)^2}{N^c}} \tag{6.2}$$

Dans le cadre de cette première approche de la coévolution, nous n'avons pas utilisé les notions de fiabilité et de mémoire introduites dans la section 5.3.7. On définit $f_a^r(\mathcal{R})$ de deux manières différentes, selon que $\mathcal{R}$ génère ou non des trajectoires sans conflit pour toutes les configurations de la base d'apprentissage :

**Si $N_{vt}^r(\mathcal{R}) \neq 0$ :**

$$f_a^r(\mathcal{R}) = \frac{1000}{1 + N_{vt}^r(\mathcal{R})} \tag{6.3}$$

**Si $N_{vt}^r(\mathcal{R}) = 0$ :**

$$f_a^r(\mathcal{R}) = 1000 + \frac{1000}{1 + r_t^r(\mathcal{R})} \tag{6.4}$$

L'évaluation des configurations de conflit se fait de manière similaire.

On cherche à calculer la fonction d'évaluation d'une configuration $\mathcal{C}$, notée $f_a^c(\mathcal{C})$. On définit $N_{vt}^c(\mathcal{C})$, la somme, sur tous les réseaux présents dans $\mathcal{P}^r$, du nombre de violations sur la configuration $\mathcal{C}$ :

$$N_{vt}^c(\mathcal{C}) = \sum_{i=1}^{N^r} N_v^r(\mathcal{R}_i, \mathcal{C}, ) \tag{6.5}$$

Et $r_t^c(\mathcal{C})$, la moyenne quadratique, sur ces réseaux, de la somme des retards des avions pour cette configuration :

$$r_t^c(\mathcal{C}) = \sqrt{\frac{\sum_{i=1}^{N^r} r(\mathcal{R}, \mathcal{C}_i)^2}{N^r}} \tag{6.6}$$

La fonction d'évaluation $f_a^c$ doit, cette fois, satisfaire les conditions suivantes :

- Si $N_{vt}^c \neq 0$, c'est-à-dire si au moins un des réseaux de la population $\mathcal{P}^r$ n'est pas capable de fournir des trajectoires sans conflit, la valeur de la fonction d'évaluation de $\mathcal{C}$ doit être d'autant plus grande que $N_{vt}^c$, le nombre total de violations sur $\mathcal{C}$, est grand.
- Si $N_{vt}^c = 0$, c'est-à-dire si tous les réseaux de $\mathcal{P}^r$ fournissent des trajectoires sans conflit pour la configuration $\mathcal{C}$, la fonction d'évaluation de $\mathcal{C}$ doit être d'autant plus grande que les retards obtenus par les réseaux sont grands.
- Une configuration de conflit sur laquelle tous les réseaux respectent les séparations doit avoir une fonction d'évaluation moins bonne qu'une configuration de conflit sur laquelle un réseau n'aura pas fourni de trajectoire sans conflit.

Ces conditions sont en fait les conditions inverses de celles que devaient remplir la fonction d'évaluation des réseaux (voir section 5.3.7).

On définit alors, $f_a^c$ la fonction d'évaluation des configurations de la manière suivante :

**Si $N_{vt}^c(\mathcal{C}) \neq 0$ :**

$$f_a^c(\mathcal{C}) = N_{vt}^c(\mathcal{C}) \tag{6.7}$$

**Si $N^c_{vt}(\mathcal{C}) = 0$ :**

$$f^c_a(\mathcal{C}) = \min(\frac{r^c_t(\mathcal{C})}{t_f}, 1) \tag{6.8}$$

Si tous les réseaux mènent, sur la configuration $\mathcal{C}$, à un retard moyen égal à la moitié de la durée de l'évitement $t_f$, alors $r^c_t(\mathcal{C}) = t_f$. On considère alors que la configuration $\mathcal{C}$ est aussi difficile que si elle entraînait une violation des contraintes de séparation et qu'il en va de même si $r^c_t(\mathcal{C}) > t_f$. C'est ce qui explique l'expression de $f^c_a(\mathcal{C})$ dans le cas où $N^c_{vt}(\mathcal{C}) = 0$.

Les grandeurs numériques utilisées sont les suivantes.

Pour les réseaux de neurones :

- Nombre d'éléments dans la population : $N^r = 500$.
- Taux de mutation : 0,2
- Taux de croisement : 0,5

Pour les configurations de conflit :

- Nombre d'éléments dans la population : $N^r = 50$.
- Taux de mutation : 0,3
- Taux de renouvellement : 0,2

Les résultats obtenus au moyen de cette première version de l'algorithme ne sont pas réellement satisfaisants. La figure 6.1 montre l'évolution de la fiabilité des réseaux de neurones au cours des générations, obtenus pour une exécution de l'algorithme. Le nombre de générations donné en abscisse compte le nombre total de générations effectuées, c'est-à-dire en comptant les générations de configurations de conflit et les générations de réseaux de neurones (donc deux générations correspondent à une génération de réseaux de neurones). Toutes les 50 générations, le meilleur réseaux de neurones de la population $\mathcal{P}^r$ (en termes de fonction d'adaptation, c'est-à-dire de performances sur les configurations présentes dans la population $\mathcal{P}^c$ courante) a été testé sur 10 000 configurations de conflit générées aléatoirement. Le nombre de conflits non résolus est indiqué, pour chacun de ces réseaux de neurones successivement testé, par le point correspondant de la courbe de la figure 6.1.

Ces résultats sont certes meilleurs, en terme de nombre de conflits résolus, que les résultats obtenus avec un algorithme génétique sans coévolution sur une base d'apprentissage fixe (voir chapitre 5) présentés dans la section 5.4.2. Ils sont cependant bien moins bons que ceux qui ont été obtenus avec une base d'apprentissage renouvelée, en utilisant les notions de mémoire et de fiabilité des réseaux, présentés dans la section 5.4.3. La dégradation observée, après obtention de bons résultats, peut s'expliquer par le phénomène de la *reine rouge*, mis en évidence par Paredis. Les bons réseaux obtenus, en termes de capacités généralisation, sont mis en échec par des configurations difficiles, et s'y adaptent, au prix d'une dégradation de leurs capacités de généralisation.

La principale différence de qualité entre les résultats obtenus avec ces différentes méthodes réside dans le nombre de conflits non résolus. Les retards obtenus par le meilleur réseau à la fin de l'apprentissage (1000 générations), avec cette version de la coévolution, pour les configurations pour lesquelles la séparation des avions est assurée, sont légèrement supérieurs à ceux obtenus par l'apprentissage sur une base renouvelée (section 5.4.3), mais restent du même ordre de grandeur (retard moyen de près de 9 minutes, contre 7,5 pour l'apprentissage sur une base renouvelée).

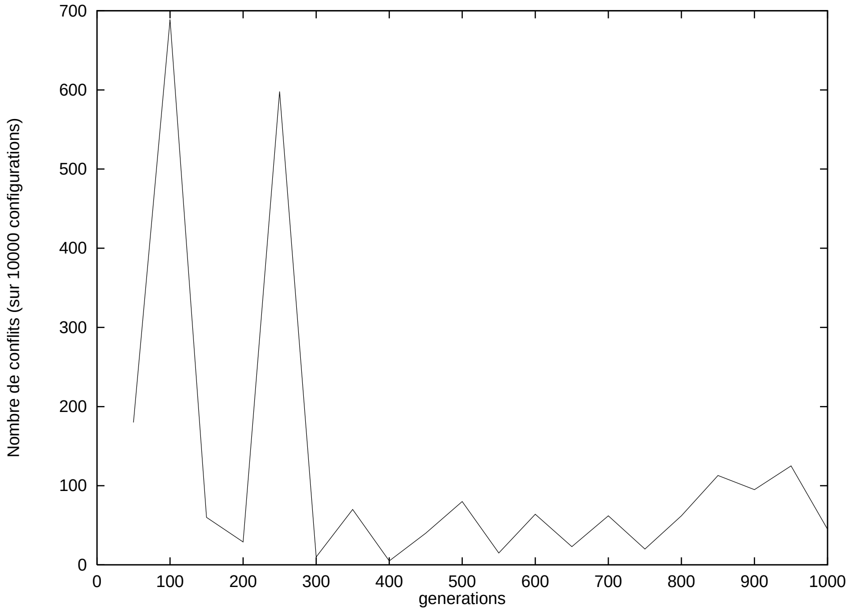



FIG. 6.1: *Évolution de la fiabilité, au cours des générations, du meilleur réseau obtenu avec la coévolution* simple *: nombre de conflits non résolus sur* 10 000 *configurations de conflit*

Ces performances plutôt décevantes sont à notre avis dues aux raisons suivantes.

L'algorithme présenté dans cette section traite les réseaux de neurones et les configurations de manière *symétrique* : chaque réseau de neurone est évalué sur l'ensemble des configurations de conflit présentes dans $\mathcal{P}^c$, chaque configuration de conflit est évalué sur l'ensemble des réseaux de neurones présents dans $\mathcal{P}^r$, la sélection des réseaux de neurones et des configurations se fait de la même manière (sélection par le rang).

Dans l'application de la coévolution au problème de l'évitement, cependant, une différence fondamentale existe entre les réseaux de neurones et les configurations de conflit. Le principe d'un algorithme génétique est de favoriser le développement des meilleurs éléments de la population, au moyen de l'évaluation (permettant de distinguer ces meilleurs éléments) et de la sélection.

La question se pose donc de savoir ce qu'est, dans le cadre du problème à résoudre, un *bon réseau* et une *bonne configuration*.

Un bon réseau est un réseau ayant une bonne capacité de généralisation, c'est-à-dire un réseau qui restera bon sur toutes les configurations, quelles qu'elles soient, qu'il sera amené à rencontrer par la suite. Si on génère aléatoirement une configuration de conflit, on veut qu'un réseau qu'on a considéré comme bon obtienne sur elle de bons résultats.

Une bonne configuration est ce que nous avons appelé une configuration *difficile*, mais on ne veut pas qu'une configuration qu'on a considérée comme bonne soit difficile pour tout réseau généré aléatoirement. Plus exactement, un réseau généré aléatoirement a peu de chance de fournir des trajectoires de bonne qualité sur une configuration de conflit et la *qualité* de cette configuration de conflit pour l'apprentissage des réseaux de neurones n'a rien à y voir. Une *bonne* configuration sera une configuration qui permettra de mettre en évidence les faiblesses d'un réseau qu'on croyait bon. Un bonne configuration est bonne parce qu'elle est difficile sur les réseaux présents dans la population. On peut considérer qu'elle est d'autant meilleure, même, qu'elle est plus difficile pour les meilleurs réseaux présents dans la population. En effet, c'est alors qu'elle joue le mieux le rôle que nous souhaitions lui voir jouer dans la section 6.2.3 : empêcher qu'on considère comme fiable un réseau qui ne l'est pas. Ce sont ces réflexions qui nous ont amené à utiliser la méthode de sélection *par l'ordre des configurations sur les réseaux ordonnés* décrite dans la section 6.3.3.

De plus, on peut retrouver dans le cas de la coévolution un problème qui se posait dans le cas de l'évaluation des réseaux sur une base d'apprentissage renouvelée (voir section 5.3.7) : le fait qu'un réseau soit meilleur ou moins bon qu'un autre sur la population des configurations à une génération donnée ne doit pas faire oublier son comportement sur les configurations de conflit auxquelles il a été confronté dans le passé. L'utilisation de la fonction d'évaluation utilisant une mémoire des comportements du réseaux sur les configurations auxquelles il avait été confronté, couplée à la notion de fiabilité (voir section 5.3.7) semble donc pouvoir être utile. C'est pourquoi nous l'avons testée avec la méthode de sélection par l'ordre des configurations sur les réseaux ordonnés. Cependant, nous avons bien entendu testé aussi cet algorithme de coévolution sans mémoire. Nous avons vu dans le chapitre 3 que ce mode d'utilisation de la mémoire pouvait pousser les réseaux de neurones à faire des détours exagérés pour éviter tout conflit (surtout pour des conflits à 3 avions). Idéalement, le recours à la coévolution, s'il permet de ne pas utiliser cette mémoire, pourrait éviter cet effet indésirable.

### 6.3.3 Sélection par l'ordre des configurations de conflit sur les réseaux ordonnés

L'idée qui nous a motivé était cette fois d'utiliser pour l'évaluation des configurations de conflit des informations plus précises quant aux comportements sur elles des différents réseaux.

Le principe de base de cette méthode, qui sera développée dans la suite de cette section, est le suivant :

- Les réseaux sont toujours évalués selon leur comportement sur l'ensemble des configurations. Nous allons décrire et comparer deux versions légèrement différentes de l'algorithme présenté ici. Dans l'une d'elles, les réseaux sont évalués sur les seules configurations de conflit présentes dans $\mathcal{P}^c$, comme c'est le cas dans l'algorithme simple présenté dans la section précédente. Dans le cadre de l'autre version, on prend en compte, pour le calcul de la fonction d'évaluation d'un réseau toutes les configurations auxquelles il a été confronté, avec utilisation des notions de fiabilité et de mémoire introduites dans la section 5.3.7. Pour distinguer cette deuxième version on dira qu'elle utilise la mémoire pour l'évaluation des réseaux.
- Les configurations de conflit sont sélectionnées en suivant l'idée selon laquelle une configuration est d'autant meilleure qu'elle est plus difficile pour un bon réseau. Cette idée est traduite par l'introduction d'un nouveau mode de sélection.

L'évaluation des réseaux de neurones peut se faire, comme nous l'avons dit ci-dessus, de deux façons : avec ou sans utilisation de la mémoire. Sans utilisation de la mémoire, elle se fait donc selon la méthode décrite dans la section précédente (équation 6.3 et 6.4).

L'utilisation de la mémoire se fait selon une méthode similaire à celle qui est décrite dans la section 5.3.7 dans le cas de l'apprentissage sur une base renouvelée.

Le calcul de la *fiabilité* ne se fait pas cependant de manière tout à fait identique. Rappelons que la fiabilité était obtenue à partir du nombre de bases d'apprentissage différentes auxquelles le réseau avait été confronté.

À partir de l'équation

$$\int_{p_{m,n,p_f}}^{1} P(p,m,n)dp = \frac{p_f}{n+1} \tag{6.9}$$

Donnant une expression implicite de $p_{m,n,p_f}$, on a calculé avant l'exécution du programme traduisant l'algorithme les valeurs de $p_{n,n,p_f}$ pour $p_f = 0,95$, et pour des valeurs de $n$ allant de 1 à 1 000.

Dans la méthode décrite dans la section 5.3.7, on définit $n_s(\mathcal{R})$ le nombre de générations sur lesquelles le réseau a fourni des trajectoires sans conflit, à partir duquel on calcule la *fiabilité* de $\mathcal{R}$ : $s(\mathcal{R}) = p_{n_s(\mathcal{R}),n_s(\mathcal{R}),p_f}$.

Cependant, dans le cadre de l'utilisation de la coévolution, les configurations de conflit ne sont pas toutes renouvelées. On note $N_s(\mathcal{R})$ le nombre de configurations différentes auxquelles un réseau $\mathcal{R}$ ayant toujours fourni des trajectoires sans conflit a été confronté. On définit le nombre de succès du réseau, toujours noté $n_s(\mathcal{R})$, par :

$$n_s(\mathcal{R}) = \frac{N_s(\mathcal{R})}{N^c} \tag{6.10}$$

$N^c$ est le nombre de configurations de conflit présentes dans $\mathcal{P}^c$. On remarque qu'en cas de renouvellement systématique de toutes les configurations de conflit, cette expression de $n_s(\mathcal{R})$ est équivalente à celle qui a été donnée dans la section 5.3.7.

On note $r_t^r(\mathcal{R})$ la moyenne quadratique de la somme des retards des deux avions sur toutes les configurations différentes auxquelles $\mathcal{R}$ a été confronté. On note $N_v^r(\mathcal{R})$ la somme, sur les configurations différentes auxquelles $\mathcal{R}$ a été confronté, du nombre de pas de temps sur lesquels les contraintes de séparation ont été violées, et on note $N_{vt}^r(\mathcal{R}) = N_v^r(\mathcal{R})/N^c$.

On retrouve encore, dans le cas d'un renouvellement total des configurations de conflit, les mêmes expressions pour $N_v^r(\mathcal{R})$ et $r_t^r(\mathcal{R})$ que dans la section 5.3.7.

La fonction d'évaluation du réseau $\mathcal{R}$ se calcule alors de la manière suivante :

**Si $\boldsymbol{N_{vt}(\mathcal{R}) \neq 0}$ :**

$$f_a^r(\mathcal{R}) = \frac{1000}{1 + N_{vt}(\mathcal{R})} \tag{6.11}$$

**Si $\boldsymbol{N_{vt}^r(\mathcal{R}) = 0}$ :**

$$f_a^r(\mathcal{R}) = 1000 + s(\mathcal{R}).\frac{1000}{1 + r_t^r(\mathcal{R})} \tag{6.12}$$

Le principe que nous voulons traduire pour l'évaluation des configurations utilise la notion de *difficulté*.

On définit la *difficulté* d'une configuration $\mathcal{C}$ pour le réseau $\mathcal{R}$, notée $D(\mathcal{C},\mathcal{R})$ de la manière suivante :

$$D(\mathcal{C},\mathcal{R}) = N^c_{vt}(\mathcal{C}) + \min(\frac{r^c_t(\mathcal{C})}{t_f}, 1) \tag{6.13}$$

La difficulté d'une configuration pour un réseau est donc d'autant plus grande que les trajectoires générées par ce réseau sur cette configuration de conflit mènent à un plus grand nombre de pas de temps sur lesquels les avions ne sont pas séparés. Pour un même nombre de pas de temps sur lesquels les avions ne sont pas séparés, la difficulté est d'autant plus grande que les retards des avions sont grands.

On dira dans la suite qu'une configuration $\mathcal{C}_i$ est plus difficile qu'une configuration $\mathcal{C}_j$ sur un réseau $\mathcal{R}$ quand $D(\mathcal{C}_i,\mathcal{R}) > D(\mathcal{C}_j,\mathcal{R})$.

Pour traduire le fait que l'on souhaite sélectionner les configurations de conflit selon leur difficulté pour les meilleurs réseaux, nous utiliserons un classement des configurations, obtenus, pour chaque génération, à partir de la position des configurations dans un tableau que nous noterons $T$. Ce tableau comporte $N^r$ colonnes et $N^c$ lignes et est construit de la manière suivante :

- À chaque colonne de ce tableau correspond un réseau de neurones. Les colonnes sont ordonnées par ordre décroissant des valeurs de la fonction d'évaluation sur les réseaux.
- À l'intérieur de chaque colonne, chaque case correspond à une configuration. Dans chaque colonne, les configurations sont rangées par ordre décroissant de *difficulté* sur le réseau correspondant à la colonne.
- On ordonne les configurations selon l'ordre dans lequel on les rencontre dans ce tableau en le parcourant ligne par ligne : la première configuration est celle qui est la plus difficile sur l'exemple dont la fonction d'adaptation est la plus grande, la deuxième, celle qui est la plus difficile pour le deuxième exemple (par ordre décroissant de $f^r_a$), si elle est différente de la première, etc.

  Quand on a ordonné $n$ configurations, la configuration qui vient en position $n+1$ selon cet ordre est la première que l'on rencontre en parcourant le tableau ligne par ligne, par ordre croissant de colonnes, et qui n'a pas encore été classée.

Un exemple de cette manière d'ordonner des configurations est donné par le tableau 6.1.

| | $\mathcal{R}_2$ | $\mathcal{R}_1$ | $\mathcal{R}_0$ | $\mathcal{R}_3$ |
|---|---|---|---|---|
| 1 | $\mathcal{C}_3$ | $\mathcal{C}_3$ | $\mathcal{C}_1$ | $\mathcal{C}_2$ |
| 2 | $\mathcal{C}_1$ | $\mathcal{C}_4$ | $\mathcal{C}_2$ | $\mathcal{C}_3$ |
| 3 | $\mathcal{C}_4$ | $\mathcal{C}_2$ | $\mathcal{C}_3$ | $\mathcal{C}_0$ |
| 4 | $\mathcal{C}_0$ | $\mathcal{C}_1$ | $\mathcal{C}_0$ | $\mathcal{C}_4$ |
| 5 | $\mathcal{C}_2$ | $\mathcal{C}_0$ | $\mathcal{C}_4$ | $\mathcal{C}_1$ |
| | 1 | 2 | 3 | 4 |

TAB. 6.1: *Ordre des exemples dans le tableau*

La population des réseaux contient quatre éléments et celle des configurations en contient 5. Le réseau $\mathcal{R}_2$ est celui pour lequel la valeur de la fonction d'évaluation, calculée sur la population des configurations est la plus grande, puis viennent, par ordre de fonction d'évaluation décroissante les réseaux $\mathcal{R}_1$, $\mathcal{R}_0$ et $\mathcal{R}_3$. Si on compare les difficultés des différentes configurations sur un réseau, par exemple $\mathcal{R}_1$, la configuration la plus difficile est $\mathcal{C}_3$, puis, par ordre décroissant de difficulté, viennent $\mathcal{C}_4, \mathcal{C}_2, \mathcal{C}_1$ et $\mathcal{C}_0$ (deuxième colonne).

L'ordre obtenu sur les configurations de conflits est alors le suivant : $\mathcal{C}_3, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_4$ et $\mathcal{C}_0$.

## 6.4 Résultats

### 6.4.1 Caractéristiques des algorithmes utilisés

Nous présentons dans cette section deux variantes de l'algorithme de coévolution avec sélection des configurations selon leur ordre sur les réseaux ordonnés (voir section 6.3.3).

Les étapes de l'algorithme sont, dans les deux cas, celles de l'algorithme de coévolution simple décrit dans la section 6.3.2, c'est-à-dire la succession d'une génération de réseaux de neurones et d'une génération de configurations de conflit.

La sélection des exemples se fait dans les deux cas par la méthode de la sélection par le rang, c'est-à-dire par un *linear ranking* suivi d'une sélection par reste stochastique (voir section 4.2.3), après avoir ordonné les configurations de conflit selon la méthode décrite dans la section précédente (ordre sur les réseaux ordonnés).

La différence entre les deux versions de l'algorithme tiendra dans le calcul de la fonction d'adaptation des réseaux de neurones.

Nous utilisons dans une des deux versions la fonction d'évaluation n'utilisant pas les notions de mémoire et de fiabilité décrites dans la section 6.3.2. Cette version de l'algorithme sera dans la suite appelée *algorithme de coévolution sans mémoire*.

Dans l'autre version nous utilisons la fonction d'évaluation décrite dans la section précédente et utilisant la notion de mémoire. Cette version sera appelée *algorithme de coévolution avec mémoire*.

Dans les deux cas, l'algorithme utilisera, pour les réseaux de neurones, le sharing par clustering adaptatif (voir section 4.4.3), avec un taux de sharing égal à 0,8, et une sélection par le rang (voir section 4.2.3).

Les grandeurs caractéristiques des algorithmes utilisés seront les mêmes pour les deux versions de l'algorithme.

Pour les réseaux de neurones :

- Nombre d'éléments dans la population : $N^r = 500$.
- Taux de mutation : 0,2
- Taux de croisement : 0,5

Pour les configurations de conflit :

- Nombre d'éléments dans la population : $N^r = 50$.
- Taux de mutation : 0,3
- Taux de renouvellement : 0,2.

L'algorithme s'arrête quand $1\,000$ générations ont été effectuées, c'est-à-dire $500$ générations de réseaux de neurones et $500$ générations de configurations de conflit.

Cela correspond à un temps de calcul de près d'une dizaine d'heures sur un ordinateur (PC) équipé d'un processeur *pentium 2* de $300\ MHz$.

### 6.4.2 Résultats obtenus

On constate tout d'abord que l'algorithme de coévolution sans mémoire testé ici (avec sélection des configurations de conflit selon leur ordre sur les réseaux ordonnés) permet d'obtenir des réseaux ayant de meilleures capacités de généralisation que l'algorithme de coévolution *simple* présenté dans la section 6.3.2, comme le montre la figure 6.2.

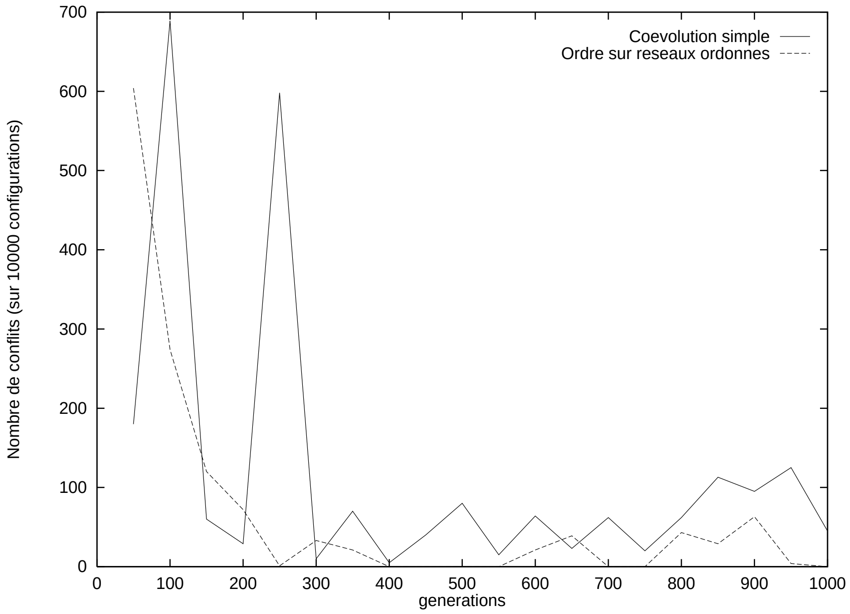



FIG. 6.2: *Évolution de la fiabilité, au cours des générations, du meilleur réseau obtenu avec la coévolution* simple*, et avec la sélection des configurations de conflit selon leur ordre sur les réseaux ordonnés : nombre de conflits non résolus sur* $10\,000$ *configurations de conflit*

Cette figure est obtenue de la même manière que la figure 6.1 (elle compare la courbe présentée sur la figure 6.1 à la courbe équivalente obtenue pour une exécution du programme de coévolution avec sélection des configurations selon leur ordre sur les réseaux ordonnés). Pour chacun des deux algorithmes comparés, toutes les 50 générations, le meilleur réseau de neurones de la population $\mathcal{P}^r$ (en termes de fonction d'adaptation, c'est-à-dire de performances sur les configurations présentes dans la population $\mathcal{P}^c$ courante) a été testé sur $10\,000$ configurations de conflits générées aléatoirement. Le nombre de conflits non résolus est indiqué, pour chacun de ces réseaux de neurones successivement testés, par un point de la courbe correspondant à l'algorithme considéré.

Les différences entre les fiabilités obtenues ne sont cependant pas très marquées. Les courbes de la figure 6.2 correspondent de plus à une seule exécution des programmes traduisant les algorithmes comparés. La longueur de l'exécution de ces programmes empêche de pouvoir les lancer un nombre suffisant de fois pour faire réellement des statistiques fiables sur les résultats obtenus. Cependant, en lançant plusieurs fois les programmes (avec des initialisations différentes du générateur de nombres aléatoires) nous avons toujours obtenu des résultats semblables à ceux de la figure 6.2.

Les performances des réseaux de neurones obtenus par l'algorithme de coévolution, en termes de résolution de conflits générés aléatoirement, sont donc relativement bonnes : on obtient souvent une résolution de tous les conflits sur 10 000 configurations, avec une pointe à 63 conflits non résolus à la génération 900, ce qui reste un taux d'échec assez faible (0,6 %).

Pour s'assurer que ces capacités de généralisation ne sont pas dues tout simplement au renouvellement partiel de la population de réseaux de neurones assuré par l'algorithme de coévolution, nous avons comparé l'évolution de la fiabilité des réseaux obtenus au cours des générations par l'algorithme de coévolution sans mémoire (sélection par l'ordre des configurations sur les réseaux ordonnés), avec des réseaux obtenus avec un renouvellement total des configurations de conflit de la base d'apprentissage à chaque génération, et avec des réseaux obtenus avec un renouvellement partiel (50 %) de ces configurations de conflit, sans utilisation de la mémoire pour le calcul de la fonction d'adaptation des réseaux.

La figure 6.3 montre les résultats ainsi obtenus.

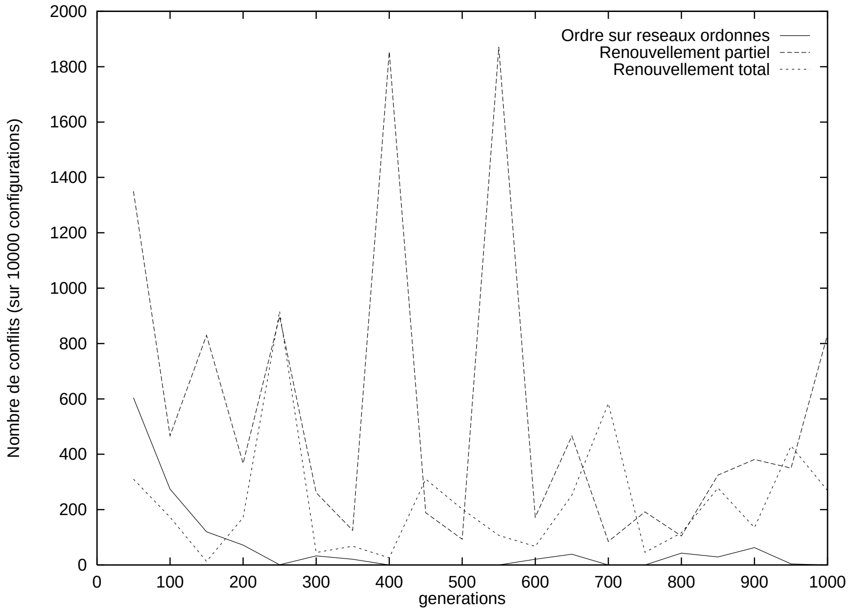



FIG. 6.3: *Évolution de la fiabilité, au cours des générations, du meilleur réseau obtenu avec la coévolution (sélection des configurations de conflit selon leur ordre sur les réseaux ordonnés), le renouvellement total et le renouvellement partiel de la base d'apprentissage (sans mémoire) : nombre de conflits non résolus sur* 10 000 *configurations de conflit*

On voit donc que la coévolution a un effet positif sur la fiabilité des réseaux obtenus, en termes de résolution de conflits générés aléatoirement, qui ne peut pas être attribué au seul renouvellement partiel des configurations de conflit.

Les effets sur les performances de l'algorithme de coévolution de l'utilisation de la mémoire et de la notion de fiabilité dans le calcul de la fonction des réseaux de neurones ne sont pas exactement ceux que l'on pouvait attendre : si en effet les réseaux obtenus avec la mémoire dans le cadre de la coévolution permettent de résoudre plus de conflits que ceux obtenus avec la coévolution sans mémoire, ils n'atteignent pas la fiabilité systématiquement observée avec l'apprentissage par algorithme génétique simple, une base d'apprentissage renouvelée et l'utilisation de la mémoire dans le calcul de la fonction d'adaptation des réseaux (voir section 5.4.3).

Ce résultat est visible sur la figure 6.4, qui montre l'évolution de la fiabilité des meilleurs réseaux obtenus avec la coévolution sans mémoire, la coévolution avec mémoire, et le renouvellement de la base d'apprentissage avec mémoire.

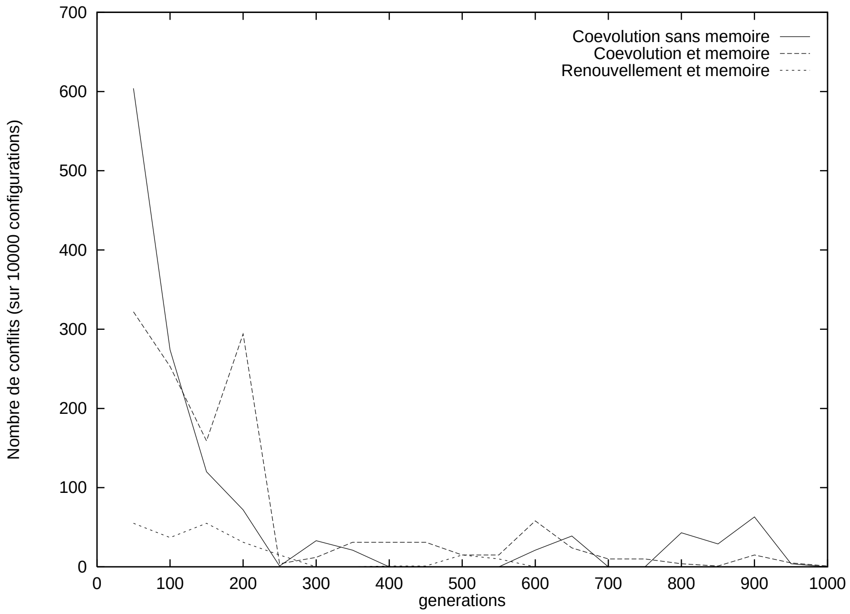



FIG. 6.4: *Évolution de la fiabilité, au cours des générations, du meilleur réseau obtenu avec la coévolution (sélection des configurations de conflit selon leur ordre sur les réseaux ordonnés) avec et sans mémoire, et le renouvellement total de la base d'apprentissage avec mémoire : nombre de conflits non résolus sur* 10 000 *configurations de conflit*

Le tableau 6.2 montre, les retards entraînés par le meilleur réseau obtenu au bout de 1000 générations par chacune de ces trois méthodes.

On voit que les différences de retards obtenus ne sont pas très importantes d'une méthode à l'autre, même si on note un léger avantage pour l'algorithme génétique sans coévolution, avec renouvellement de la base d'apprentissage et utilisation de la mémoire pour le calcul de la fonction d'évaluation des réseaux.

À partir de ces résultats, nous proposons une première interprétation de l'effet de la coévolution sur l'apprentissage de réseaux de neurones utilisés pour l'évitement aérien.

La coévolution permet d'obtenir, sans utilisation des notions de fiabilité et de mémoire, une capacité de généralisation bien meilleure, en termes de résolution de conflit, que l'algorithme génétique avec base d'apprentissage fixe (voir section 5.4.2). Cela nous semble dû au fait que la coévolution permet de concentrer l'apprentissage des réseaux de neurones sur les configurations

TAB. 6.2: *Comparaison de la coévolution avec et sans mémoire, et de l'évolution simple avec renouvellement de la base d'apprentissage et mémoire : retards quand la séparation est assurée ; nombre de configurations (sur* $10\,000$*) pour lesquelles les réseaux ont généré des trajectoires sans conflit, réparties sur les lignes selon le retard moyen par avion.*

| Retards | Renouvellement des configurations | Coévolution sans mémoire | Coévolution avec mémoire |
|---|---|---|---|
| de $0$ à $5$ $s$ | $4\,771$ | $3\,688$ | $3\,368$ |
| de $5$ à $10$ $s$ | $2\,060$ | $3\,152$ | $3\,345$ |
| de $10$ à $20$ $s$ | $2\,704$ | $2\,704$ | $2\,726$ |
| de $20$ à $30$ $s$ | $353$ | $395$ | $472$ |
| de $30$ $s$ à $1$ $mn$ | $132$ | $56$ | $83$ |
| de $1$ à $2$ $mn$ | $18$ | $5$ | $5$ |
| retard moyen | $7,5$ $s$ | $8,64$ $s$ | $8,3$ $s$ |
| total | $10\,000$ | $10\,000$ | $9\,999$ |

de conflit qui posent le plus de problèmes.

La reproduction, en favorisant les plus difficiles parmi ces configurations de conflit, assure qu'elles pèseront plus dans l'apprentissage d'un réseau de neurone, en les maintenant de génération en génération. Ainsi, un réseau de neurone qui ne sait pas résoudre ces conflits les plus difficiles ne pourra jamais avoir une bonne fonction d'adaptation.

L'effet de la reproduction des configurations de conflit n'est pas seulement de maintenir une certaine stabilité de la population des configurations de conflit, malgré la mutation et le renouvellement aléatoire d'une fraction de cette population, mais d'assurer que les plus stables parmi les configurations de conflit soient justement les plus difficiles.

Cependant, on constate que l'efficacité de la coévolution pour l'obtention d'une bonne capacité de généralisation des réseaux demeure malgré tout limitée. Ce problème apparaît même (moins gravement cependant) dans le cas de l'utilisation de la mémoire.

En effet, quand elle est couplée à l'utilisation de la mémoire et de la fiabilité dans le calcul de la fonction d'adaptation des réseaux de neurones, la coévolution donne de moins bons résultats que l'algorithme génétique simple avec renouvellement de la base d'apprentissage, et la même fonction d'adaptation pour les réseaux de neurones.

La coévolution semble alors avoir un rôle négatif. En fait, l'utilisation de la mémoire est couplée à une mesure de la fiabilité : on considère qu'un réseau est d'autant plus fiable qu'il a résolu un plus grand nombre de conflits différents. Mais si les conflits cherchent à être les plus difficiles pour les réseaux présents dans la population, même un réseau qui serait assez fiable statistiquement peut échouer sur une configuration très difficile pour lui. L'utilisation de la mémoire favorise en fait les réseaux ayant une plus longue survie. Si les configurations de conflit visent à être le plus difficile possible, la fiabilité d'un réseau n'est plus traduite par sa durée de survie.

Cela correspond à un problème plus général posé par la coévolution, qui est que le changement de l'environnement, ici dû à l'évolution de la population de configurations sur laquelle les réseaux sont évalués, est ressenti très durement par la population des réseaux, puisque justement l'évolution des configurations vise à déstabiliser les réseaux. Si cette évolution des configurations de conflit est efficace, un réseau, même de bonne qualité aura peu de chance de survivre. C'est d'ailleurs ce qu'on constate sur la figure 6.4 : la coévolution, même sans mémoire, permet d'obtenir des réseaux très fiables (résolution des 10 000 conflits tests), mais ne les conserve pas.

On retrouve le type d'instabilité désigné par Paredis sous le terme de *problème de la reine rouge* [Par97].

En plus des résultats relativement décevants en termes de respect des contraintes de séparation, la coévolution entraîne des retards au moins aussi importants que le renouvellement de la base d'apprentissage avec mémoire. On a vu (section 5.4.3) que cette dernière méthode payait sa très bonne capacité de généralisation par une augmentation des retards. Cette augmentation des retards reste raisonnable dans le cas de conflit à deux avions, mais peut devenir trop importante, dans un certain nombre de cas de conflits à 3 avions. Le fait que la coévolution mène à une augmentation des retards par rapport à cette méthode pour des conflits à 2 avions ne laisse rien présager de bon, de ce point de vue, quant à son comportement sur des conflits à 3 avions.

Cette augmentation des retards entraînée par la coévolution n'est d'ailleurs en fait pas trop surprenante : ayant réalisé leur apprentissage sur des configurations de conflits *difficiles*, c'est-à-dire entraînant des retards importants, les réseaux de neurones ont tendance à faire subir aux avions d'importantes déviations, mêmes sur des configurations de conflit pour lesquelles cela ne serait pas nécessaire.

## 6.5 Conclusion

Cette première approche nous a permis surtout de constater que les principaux effets indésirables de la coévolution évoqués dans la littérature sur ce sujet (difficultés d'adaptation à un environnement changeant, et adaptation aux adversaires les plus difficiles, mêmes s'ils sont peu fréquents, menant à une moins bonne réponse sur la majorité des adversaires), se retrouvaient dans l'application de la coévolution au problème de l'apprentissage de réseaux de neurones pour l'évitement aérien.

Ces problèmes sont souvent rencontrés, et donc souvent attaqués. Considérons tout d'abord les deux approches de la coévolution que nous avons décrites dans la section 6.2.

Dans l'approche de Jan Paredis, l'instabilité propre à la coévolution peut-être contrebalancée par l'utilisation de la *Life Time Fitness Evaluation*. Dans le problème de l'évitement, on peut considérer que la mémoire, telle que nous l'avons définie et utilisée peut jouer un rôle comparable. Elle permet certes une amélioration par rapport à la coévolution sans mémoire, mais la coévolution avec mémoire reste moins performante que l'algorithme génétique simple avec base d'apprentissage renouvelée et mémoire.

Jan Paredis a également joué sur les rythmes d'évolution des deux populations (dans le but d'obtenir des rythmes d'évolution compatibles entre les populations, avec l'idée que l'évolution de deux populations différentes ne se faisait pas obligatoirement à la même vitesse). Dans notre cas, cela revient, par exemple, à ce qu'on n'ait plus une génération de réseaux entre deux générations de configurations, mais $n_g^r$ générations de réseaux, puis $n_g^c$ générations de configurations de conflits. Nous avons fait quelques tentatives pour différentes valeurs de $n_g^r$ sans jamais obtenir d'améliorations notables.

Dans l'approche de W.D. Hillis, c'est la répartition *géographique* des éléments des deux populations, couplée au caractère local des opérateurs génétiques et de l'évaluation des éléments qui introduit une certaine stabilité. Cette approche très intéressante nous a cependant paru trop complexe pour être employée dans le cadre d'une première approche de la coévolution.

De plus, dans le cadre de ces travaux, les *parasites* sont constitués de plusieurs exemples d'apprentissage. Chaque réseau de tri est évalué sur le groupe d'exemples d'apprentissage constituant le parasite occupant le même site que lui. L'équivalent ici serait de faire évoluer, conjointement à la population des réseaux de neurones, une population de petites *bases d'apprentissage*, constituées chacune d'un certain nombre de configurations de conflit, et non plus une population de configurations de conflit. Cette approche pourrait sans doute être intéressante dans le cadre de l'apprentissage de réseaux de neurones pour l'évitement. Elle pourrait permettre de maintenir une certaine diversité dans les configurations de conflits utilisées pour l'apprentissage : un *parasite* composé de plusieurs configurations pourrait être plus difficile pour un réseau donné si les configurations qu'il contient sont très différentes les unes des autres. Cependant elle risque d'entraîner une augmentation du nombre de confrontations réseau / configuration nécessaires à l'apprentissage et donc d'augmenter fortement la durée de l'apprentissage.

Un moyen plus simple de favoriser la diversité des configurations de conflit serait d'utiliser une forme de sharing (voir section 4.4) dans le cadre de leur évolution. Il faudrait pour cela définir une distance sur l'espace des configurations. Aucune distance naturelle n'existe sur cet espace. La définition d'une telle distance et d'une fonction de sharing serait pourtant très utile dans le cadre d'une étude plus approfondie de la coévolution sur ce problème.

D'autres idées intéressantes existent dans la littérature sur la coévolution. Deux techniques qui pourraient être adaptées à notre problème sont décrites dans [RB97] : le *competitive fitness sharing*, et le *hall of fame*. Le *competitive fitness sharing* consiste à moduler la part de la fonction d'adaptation d'un individu qui lui vient du fait qu'il a vaincu un *parasite* donné, selon le nombre d'individus de la population ayant vaincu ce même parasite : le gain apporté à la fonction d'adaptation d'un réseau de neurones par le fait qu'il résout un des conflits de la population $\mathcal{P}^c$ serait d'autant plus fort que ce conflit serait résolu par un nombre plus petit de réseaux de neurones de la population $\mathcal{P}^r$. Cette idée est intéressante, mais on peut craindre cependant qu'elle favorise l'adaptation des réseaux aux configurations les plus difficiles au détriment de leur performance sur les plus courantes. Le *hall of fame* offre peut-être un moyen de maintenir une certaine stabilité dans un monde changeant et de pallier le problème de la *reine rouge* : les meilleurs éléments rencontrés à chaque génération sont conservés, non pas seulement d'une génération sur l'autre, comme c'est le cas quand on utilise l'élitisme, mais sur une durée plus longue, voire sur toute la durée de l'évitement.

On trouve dans [JP98] une voie pour atténuer les effets néfastes des modifications trop brutales de l'environnement : plutôt que de favoriser l'apparition de parasites difficiles, on chercher à favoriser une augmentation graduelle de leur difficulté. Cette idée serait difficile à appliquer ici : pour un réseau donné, la présence d'une seule configuration pour laquelle il échoue fait chuter la valeur de sa fonction d'adaptation, de telle manière que le réseau a alors peu de chance d'y survivre. Il est difficile alors d'envisager de graduer la difficulté des configurations pour les réseaux. Cette idée mériterait d'être envisagée, mais nécessiterait de modifier le calcul de la fonction d'adaptation des réseaux.

Une telle modification du calcul de cette fonction d'adaptation, permettant de ne pas rendre fatale une violation de contrainte isolée, est une voie qui mériterait sans doute d'être explorée non seulement dans le cadre de la coévolution, mais aussi dans celui de l'évolution *simple*.

Toutes ces techniques permettraient peut-être d'améliorer les résultats obtenus avec la coévolution. La première tentative présentée ici a semblé montrer que l'utilisation de la coévolution ne pourrait que rendre plus complexe un problème qui en lui même l'est déjà suffisamment, et nécessiter de très nombreux réglages, raffinements, adaptations de l'algorithme pour avoir une chance de mener à une amélioration des performances. L'emploi de techniques de coévolution nous semble n'être pas, en lui même, le meilleur moyen de rendre plus performant l'apprentissage des réseaux de neurones pour l'évitement aérien. L'utilisation d'un algorithme d'évolution simple, tel que l'algorithme génétique utilisé dans la méthode présentée dans le chapitre 5, avec une gestion dirigée de la base d'apprentissage, nous semble pouvoir mener à de meilleurs résultats. La recherche d'un tel mode de gestion de la base d'apprentissage constitue donc un axe de recherche qui nous semble très intéressant pour continuer l'étude des méthodes présentées ici et dans le chapitre 5. Dans ce cadre, une poursuite des travaux que nous avons engagés sur l'application de la coévolution à l'apprentissage des réseaux de neurones pour l'évitement, en permettant de mieux comprendre le rôle des configurations de conflit présentes dans la base d'apprentissage, et l'influence des modifications de cette base sur l'apprentissage des réseaux, serait probablement très utile.

# Conclusion

Les méthodes qui ont été présentées ici visent toutes à générer des trajectoires d'évitement pour des avions impliqués dans des conflits potentiels. La génération de telles trajectoires serait un des deux rôles principaux d'un système d'évitement automatique (l'autre étant la pré-détection des conflits). Nous avons distingué dans le chapitre 1 deux types de systèmes d'évitement automatiques : les systèmes *centralisés* (définition 1.9) et les systèmes *embarqués* (définition 1.10). Parmi les méthodes de génération de trajectoires que nous avons développées, les approches globales de la génération de trajectoire par méthodes d'intervalles (chapitre 2) et par l'algorithme $A^*$ (chapitre 3) correspondent à des systèmes d'évitement centralisés. Les approches séquentielles de ces méthodes (chapitres 2 et 3) et la génération de trajectoires d'évitement au moyen de réseaux de neurones (chapitres 5 et 6) correspondent à deux types différents de systèmes d'évitement *embarqués*.

Les résultats obtenus dans les chapitres 2 et 3 montrent que les approches globales des méthodes d'intervalles et de l'algorithme $A^*$ ne sont pas adaptées à la génération de trajectoires sans conflit dans le cadre d'un système d'évitement. En effet, elles ne sont plus applicables dès que les conflits à résoudre impliquent plus de deux avions : problème de temps de calcul pour les méthodes d'intervalles (section 2.9.2) et d'encombrement mémoire pour l'algorithme $A^*$ (section 3.7.1).

Il serait intéressant de conserver de la méthode utilisant l'algorithme $A^*$ la représentation de l'ensemble des trajectoires possibles par un arbre, mais d'essayer d'autres algorithmes de recherche pour obtenir un chemin optimal dans cet arbre, tel que l'algorithme de recherche en approfondissement itératif, décrit dans [Kor85], qui permettrait de réduire (légèrement) les besoins en mémoire.

L'approche globale des méthodes d'intervalles présentées dans le chapitre 2, offre l'avantage de permettre d'obtenir une solution légèrement sous optimale, pour un nombre relativement important d'avions, en un temps de calcul réduit, grâce à l'élagage (alors que l'équivalent de cette technique pour l'algorithme $A^*$ est beaucoup moins efficace). Cela ne permet toujours pas, cependant, l'utilisation de ces méthodes en *temps réel* : le temps de calcul nécessaire à l'obtention d'une solution, même avec de grandes valeurs de la coupure d'élagage, ne peut pas être estimé *a priori* ; on ne peut donc pas garantir l'obtention d'une solution à l'intérieur d'un délai fixé.

La complexité des problèmes d'optimisation posés par une approche globale de la génération de trajectoires d'évitement optimales est telle que ces problèmes sont très difficiles pour des méthodes d'optimisation déterministes. Ils peuvent être cependant résolus par l'utilisation de méthodes d'optimisation stochastiques [Dur96].

À l'intérieur des systèmes d'évitement *embarqués*, on peut distinguer deux sous-groupes différents, les systèmes d'évitement *réactifs* (les trajectoires des avions sont générées au moyen d'une commande continue ou de commandes successives et rapprochées) et les systèmes d'évitement *séquentiels* (les trajectoires sont générées l'une après l'autre, selon un ordre de priorité entre les avions).

Le principal inconvénient des systèmes séquentiels est que la qualité des trajectoires obtenues, et même leur existence, dépendent de l'ordre dans lequel elles sont générées, c'est-à-dire de l'ordre de priorité fixé entre les avions (voir sections 2.9.3 et 3.7.2). Le seul moyen d'éviter cet inconvénient est de générer les trajectoires d'évitement simultanément. Il faut alors avoir recours à un système réactif, puisque la trajectoire d'un avion doit alors pouvoir être modifiée à tout moment (on ne dispose pas d'information sur les *intentions* des autres avions qui permettrait de générer une trajectoire avec une certaine anticipation, puisque les trajectoires à venir des autres avions ne sont pas encore établies). Cependant, un système réactif posera aussi des problèmes : outre que les trajectoires générées seraient difficiles à suivre, et incompatibles avec le confort des passagers, le fait de ne pas connaître avec certitude les intentions des autres avions fait qu'une violation des contraintes de séparation peut apparaître trop tard pour être évitée. Un système séquentiel permet une anticipation : si aucune solution n'est trouvée, à cause par exemple d'un ordre de priorité entre avions inadapté au conflit, il est encore temps de mettre l'avion sur une trajectoire de secours (quitte à lui faire subir une déviation plus importante). Pour ces raisons, l'approche séquentielle d'un système d'évitement embarqué nous semble plus fiable que l'approche réactive.

La méthode présentée dans le chapitre 5, utilisant des réseaux de neurones pour la génération de trajectoires d'évitement de manière réactive, nous semble difficilement utilisable telle quelle dans le cadre d'un système d'évitement automatique, les techniques mises en œuvre rendant la fiabilité de ces méthodes difficile à établir. Elle offre cependant l'avantage, par rapport à d'autres méthodes réactives (voir section 1.4.4), de rechercher des trajectoires optimales. L'obtention de ces trajectoires optimales, ou du moins de trajectoires s'en approchant, pour des configurations de conflit quelconques, n'est cependant pas garantie, surtout dans le cas de conflits à trois avions ou plus. Le développement de modes d'apprentissage adaptés reste un axe de recherche important dans ce cadre. Les techniques de coévolution ne semblent pas, en l'état actuel de nos recherches, pouvoir mener à une amélioration des résultats obtenus par les réseaux de neurones.

Les approches séquentielles de la génération de trajectoire par méthodes d'intervalles (chapitre 2) et par l'algorithme $A^*$ (chapitre 3) correspondent à des systèmes d'évitement embarqué de type séquentiel, et sont, à ce titre, plus intéressantes. Dans ce cadre, l'utilisation de l'algorithme $A^*$ nous semble pouvoir être plus utile que les méthodes d'intervalles. En effet, l'algorithme $A^*$ ne nécessite pas certaines hypothèses fortes (telles que les vitesses constantes) et est plus facilement adaptable à différents types de trajectoires d'évitement, et à l'utilisation de manœuvres d'évitement dans le plan vertical. Elle pourrait être utilisée dans le cadre d'un système d'évitement embarqué, comme le montrent des premiers résultats très encourageants [Gra98], et serait, du fait de sa souplesse, très utile pour la mise au point d'un tel système (comparaison de différents types de trajectoires d'évitement ou de différents ordres de priorité).

# Bibliographie


[ABDM96] Alliot (Jean-Marc), Bosc (Jean-François), Durand (Nicolas) et Maugis (Lionel). – *Cats : a complete air trafic simulator*. – Rapport technique, CENA, 1996.

[AD95] Alliot (Jean-Marc) et Durand (Nicolas). – A genetic algorithm to improve an othello program, *In : Artificial Evolution*. – Springer Verlag, 1995.

[AL92] Angerand (Luc) et LeJeannic (Hervé). – *Bilan du projet SAINTEX*. – Rapport technique, CENA, 1992. CENA/R92009.

[All96] Alliot (Jean-Marc). – Techniques d'optimisation stochastique appliquées aux problèmes du trafic aérien, mai 1996. Thèse d'habilitation à diriger des recherches, INPT.

[AS92] Alliot (Jean-Marc) et Schiex (Thomas). – *Intelligence Artificielle et Informatique Théorique*. – Cepadues, 1992. ISBN : 2-85428-324-4.

[BG91] Bridges (C.L) et Goldberg (D.E). – An analysis of multipoint crossover, *In : Proceedings of the Foundation Of Genetic Algorithms*. FOGA. – 1991.

[Bos97] Bosc (Jean-François). – *Techniques d'évitement réactif et simulation du trafic aérien*. – Thèse de Doctorat, INPT, 1997.

[BS93] Bäck (T) et Schwefel (H.-P.). – An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, vol. 1, n° 1, 1993, pp. 1–24.

[Cat90] Catoni (O.). – *Large deviations for Annealing*. – Thèse de Doctorat, Université de Paris XI, 1990.

[Cel90] Celio (Joseph C.). – *Controller perspective of AERA2*. – Rapport technique, MITRE, February 1990. MP-88W00015.

[Cer94] Cerf (R). – *Une Théorie Asymptotique des Algorithmes Génétiques*. – Thèse de Doctorat, Université Montpellier II (France), 1994.

[CGT92] Conn (A.R.), Gould (Nick) et Toint (Ph. L.). – *A comprehensive description of LANCELOT*. – Rapport technique, IBM T.J. Watson research center, 1992. Report 91/10.

[Del95] Delahaye (Daniel). – *Optimisation de la sectorisation de l'espace arien par algorithmes gntiques*. – Thèse de Doctorat, ENSAE, 1995.

[DHN97] Duong (Vu N.), Hoffman (Eric) et Nicolaon (Jean-Pierre). – Initial results of investigation into autonomous aircraft concept (freer-1), *In : Procceedings of the 1rst USA/Europe Seminar*. – 1997.

[Dur96] Durand (Nicolas). – *Optimisation de trajectoires pour la résolution de conflits en route*. – Thèse de Doctorat, INPT, 1996.

[Eur97] Eurocontrol. – *ATM strategy for 2000+*. – Rapport technique, European organisation for the safety of air navigation, 1997.

[Fad98] Fadda (Alessandro). – *Étude de problèmes inverses par algorithmes d'évolution et réseaux de neurones*. – Thèse de Doctorat, Ecole Polytechnique, 1998.

[FFAF92] Fogel (D.B.), Fogel (L.J.), Atmar (W.) et Fogel (G.B). – Hierarchics methods of evolutionary programming, *In : Proceedings of the 1rst annual conference on Evolutionary Programming*, éd. par Fogel (D.B.) et Atmar (W.). Evolutionary Programming Society. – 1992.

[FMT93] Fron (Xavier), Maudry (Bernard) et Tumelin (Jean-Claude). – *Arc 2000 : Automatic Radar Control*. – Rapport technique, Eurocontrol, 1993.

[FOW66] Fogel (L.J), Owens (A.J) et Walsh (M.J). – *Artificial Intelligence Through Simulated Evolution*. – Wiley and sons. NY, 1966.

[FW83] Freidlin (M.I) et Wentzell (A.D). – *Random Perturbations of Dynamical Systems*. – Springer-verlag, New-York, 1983.

[Gau95] Gaudière (Gervais). – *Résolutions de conflits aériens par réseaux de neurones*. – DEA, Ecole Nationale de l'Aviation Civile (ENAC), 1995.

[Gol89] Goldberg (D.E). – *Genetic Algorithms in Search, Optimization and Machine Learning*. – Reading MA Addison Wesley, 1989.

[Gol91] Goldberg (D.E). – Real-coded genetic algorithms, virtual alphabets and blocking. *Complex Systems*, vol. 5, 1991, pp. 139–167.

[Gra98] Granger (Géraud). – *Résolution de conflits embarquée dans les espaces de faible densité*. – DEA, Ecole Nationale de l'Aviation Civile (ENAC), 1998.

[Han68] Hanson (R.J.). – *Interval arithmetic as a closed arithmetic system on a computer*. – Rapport technique, Jet Propulsion Laboratory, 1968.

[Han92] Hansen (Eldon). – *Global Optimization using Interval Analysis*. – New York, Marcel Dekker, inc, 1992, *Monographs and textbooks in pure and applied mathematics*.

[Hil92] Hillis (W. D.). – Coevolving parasites improve simulated evolution as an optimisation procedure, *In : Artificial Life II*, éd. par Langton (C.G.), Taylor (C.), Farmer (J.D.) et Rasmussen (S.). – Addison-Wesley, 1992.

[HKP91] Hertz (J.), Krogh (A.) et Palmer (R). – *Introduction to the theory of neural computation*. – Addison Wesley, 1991.

[HNR68] Hart (P.E), Nilsson (N.J) et Raphael (B.). – A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on SSC*, vol. 4, 1668.

[Hol62] Holland (John). – Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machinery*, vol. 3, 1962.

[HSW89] Hornik (K.), Stinchcombe (M.) et White (H.). – Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989, pp. 359–366.

[IR92] Ingber (L) et Rosen (B). – Genetic algorithms and very fast simulated re-annealing. *Mathematical Computer Modeling*, vol. 16, n° 11, 1992, pp. 87–100.

[JP98] Juillé (Hugues) et Pollack (Jordan B.). – Coevolving the ideal trainer : Application to the discovery of cellular automata rules, *In : Proceedings of the third annual Genetic Conference*. – 1998.

[K+89] Krella (Fred) et al. – *Arc 2000 scenario (version 4.3).* – Rapport technique, Eurocontrol, April 1989.

[Kah68] Kahan (W.M.). – *A more complete interval arithmeticc.* – Lecture notes for a summer course, University of Michigan, 1968.

[Kor85] Korf (Richard E.). – Depth first iterative deepening : an optimal admissible tree search. *Artificial Intelligence*, vol. 27, 1985, pp. 97–109.

[Koz92] Koza (J.R). – *Genetic Programming.* – MIT press, 1992.

[LB94] LeCun (Yann) et Bengio (Yoshua). – World-level training of a handwritten word recognizer based on convolutional neural networks, *In : Proceedings of the international conference on pattern recognition*, éd. par IAPR. IEEE, pp. 88–92. – 1994.

[LeC87] LeCun (Y.). – *Modèles connexionnistes de l'apprentissage.* – Thèse de Doctorat, Université Pierre et Marie Curie, Paris VI, 1987.

[MDA95] Médioni (Frédéric), Durand (Nicolas) et Alliot (Jean-Marc). – Air trafic conflicts resolution by genetic algorithms, *In : Artificial Evolution.* – Springer Verlag, 1995.

[Méd94] Médioni (Frédéric). – *Optimisation de la résolution de conflits par algorithmes stochastiques et programmation linéaire.* – DEA, Ecole Nationale de l'Aviation Civile (ENAC), 1994.

[MG92] Mahfoud (S.W) et Goldberg (D.E). – *Parallel Recombinative Simulated Annealing : A Genetic Algorithm.* – Illigal report n° 92002, Urbana, IL 61801-2996, University of Illinois, April 1992.

[MG94] Meckiff (Dr. C.) et Gibbs (Dr. P.). – *PHARE : Highly Interactive Problem Solver.* – Rapport technique, Eurocontrol, 1994.

[Mic92] Michalewicz (Z). – *Genetic algorithms + Data Structures = Evolution Programs.* – Springer-verlag, 1992.

[MJ91] Michalewicz (Z) et Janikov (C.Z). – Handling constraints in genetic algorithms, *In : Proceedings of the Fourth International Conference on Genetic Algorithm.* ICGA. – 1991.

[Moo66] Moore (R.E.). – *Interval Analysis.* – London, Prentice Hall International, INC., 1966, *Prentice Hall series in Automated Computation.*

[MP43] McCulloch (W.S.) et Pitts (W.A.). – A logical calculus immanent in nervous activity. *Bulletin of mathematics and biophysics*, vol. 5, 1943.

[Nie89a] Niedringhaus (W.P.). – *Automated Planning Function for AERA3 : Manoeuver Option Manager.* – Rapport technique, FAA, 1989. DOT/FAA/DS-89/21.

[Nie89b] Niedringhaus (W.P.). – *A mathematical formulation for planning automated aircraft separation for AERA3.* – Rapport technique, FAA, 1989. DOT/FAA/DS-89/20.

[OD98] Oussedik (Sofiane) et Delahaye (Daniel). – Reduction of air traffic congestion by genetic algorithms, *In : Parallel Problem Solving from Nature V*, éd. par Eiben (Agoston E.), Bäck (Thomas), Schoenauer (Marc) et Schwefel (Hans-Paul). – Springer-Verlag, 1998.

[PA91] Perry (Tekla S.) et Adam (John A.). – Improving the world's largest, most advanced system ! *IEEE Spectrum*, February 1991.

[Par94] Paredis (Jan). – Co-evolutionary constraint satisfaction, *In : International Conference on Evolutionary Computation, Parallel Problem Solving from Nature III*, éd. par Davido (Y.), Schwefel (H.-P.) et Manner (R.). – Springer-Verlag, 1994.

[Par96a] Paredis (Jan). – Coevolutionary computation. *Artificial Life Journal*, vol. 2, n° 4, 1996. – MIT Pres / Bradford books.

[Par96b] Paredis (Jan). – Coevolutionary life-time learning, *In : International Conference on Evolutionary Computation, Parallel Problem Solving from Nature IV*, éd. par Voigt (H.M.), Ebeling (M.), Rechenberg (I.) et Schwefel (H.P.). – Springer-Verlag, 1996.

[Par97] Paredis (Jan). – Coevolving cellular automata : Be aware of the red queen !, *In : Proceedings of the 7th International Conference on Genetic Algorithms*, éd. par Bäck (T.). – Morgan Kaufman, 1997.

[RB97] Rosin (Christopher D.) et Belew (Richard K.). – New methods for competitive coevolution. *Evolutionary Computation*, vol. 5, n° 1, 1997, pp. 1–29.

[RR84] Ratschek (Helmut) et Rokne (Jon). – *Computer Methods for the Range of Functions*. – Chichester, Ellis Horwood, 1984.

[RR95] Ratschek (Helmut) et Rokne (Jon). – Interval methods. *In : Handbook of Global Optimization*, éd. par Horst (Reiner) et Pardalos (Panos M.). – Kluwer Academic Publishers, Dordrecht (NL), 1995.

[RTC95] RTCA (board of directors select commitee). – *Report on Free Flight*. – Rapport technique, Washington, RTCA, 1995.

[Sch87] Schwefel (H.P.). – Collective phenomena in evolutionary systems, *In : 31st annual meeting of internationall society for general system research*, pp. 1025–1033. – 1987.

[SPSS83] Schuster (E. M.), Petroski (F. R.), Sciambi (R. K.) et Stokrp (M. MC). – *AERA 2 Functional Design and Performance Description*. – Rapport technique, MITRE, September 1983. MtR-83W136.

[SR94] Schoenauer (Marc) et Ronald (Edmund). – Neuro-genetic truck backer-upper controller, *In : Proceedings of WCCI 94, IEEE conference on evolutionary computation*. – IEEE press, 1994.

[Wri91] Wright (A.H). – Genetic algorithms for real parameter optimization, *In : Proceeding of the Foundation Of Genetic Algorithms*. FOGA. – 1991.

[YG93] Yin (X) et Germay (N). – A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization, *In : Proceedings of the Artificial Neural Nets and Genetic Algorithms*. – 1993.

[Zeg93] Zeghal (Karim). – *Techniques réactives pour l'évitement*. – Rapport technique, ONERA, June 1993.

[Zeg94] Zeghal (Karim). – *Vers une thèorie de la coordination d'actions. Application à la navigation aérienne*. – Thèse de Doctorat, Université Paris VI, 1994.