Algorithmes Génétiques et autres méthodes d'optimisation appliqués à la gestion de trafic aérien

Nicolas Durand

▶ **To cite this version:**

Nicolas Durand. Algorithmes Génétiques et autres méthodes d'optimisation appliqués à la gestion de trafic aérien. Optimisation et contrôle [math.OC]. INPT, 2004. tel-01293722

HAL Id: tel-01293722
https://theses.hal.science/tel-01293722

Submitted on 25 Mar 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

HABILITATION À DIRIGER DES RECHERCHES

présentée devant

L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

École doctorale : Informatique et Télécommunications

par

Nicolas DURAND

Laboratoire d'Optimisation Globale
CENA - ENAC

Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion du trafic aérien

---

soutenue le 2 novembre 2004 devant le jury composé de :

M.	Patrick SIARRY	Université Paris 12	(Rapporteur et Président)
M.	Jin-Kao HAO	Université d'Angers	(Rapporteur)
M.	Vojin TOSIC	Université de Belgrade	(Rapporteur)
M.	Joseph NOAILLES	ENSEEIHT	Correspondant
M.	Alain PRINTEMPS	CENA	Examinateur
M.	Jean-Marc GAROT	Eurocontrol	Examinateur

Résumé :

Ce document présente différentes méthodes d'optimisation appliquées à la gestion du trafic aérien. La première partie est consacrée aux algorithmes génétiques et propose un opérateur de croisement adapté aux problèmes partiellement séparables. Une application est proposée : l'optimisation de la circulation des avions sur l'aéroport. La deuxième partie traite le problème de résolution de conflits aériens avec différentes modélisations (approches centralisées ou autonômes) et différentes méthodes d'optimisation : algorithmes génétiques, branch and bound par intervalles, réseaux de neurones, programmation semi-définie, hybridation d'algorithmes génétiques et de méthodes déterministes (programmation linéaire, algorithmes A^*).

Mots-clés :

algorithmes génétiques, gestion du trafic aérien, résolution de conflits, séparabilité partielle, réseaux de neurones, branch and bound par intervalles, programmation linéaire, programmation semi-définie.

Abstract :

This document discribes different optimization methods applied to the air traffic management domain. The first part details genetic algorithms and introduces a crossover operator adapted to partially separated problems. The operator is tested on an airport ground traffic optimization problem. In the second part, the conflict resolution problem is optimized with different (centralized or autonomous) models and different algorithms : genetic algorithms, branch and bound, neural networks, semidefinite programming, hybridization of genetic algorithms and deterministic methods such as linear programming or A^* algorithms.

Keywords :

genetic algorithms, air trafic management, conflict resolution, partial separability, neural networks, branch and bound, linear programming, semidefinite programming.

Table des matières

Glossaire **9**

I Les algorithmes génétiques **15**

1 AG : généralités et améliorations **19**
1.1 Principes généraux . . . 19
1.2 Description détaillée . . . 20
1.2.1 Codage des données . . . 20
1.2.2 Génération aléatoire de la population initiale . . . 21
1.2.3 Gestion des contraintes . . . 21
1.2.4 Opérateur de croisement . . . 21
1.2.5 Opérateur de mutation . . . 22
1.2.6 Principes de sélection . . . 24
1.3 Améliorations classiques . . . 25
1.3.1 Introduction . . . 25
1.3.2 *Scaling* . . . 25
1.3.3 *Sharing* . . . 28
1.3.4 *Sharing* clusterisé . . . 30
1.3.5 Algorithmes génétiques et recuit simulé . . . 30
1.3.6 Recherche multiobjectif . . . 32
1.3.7 Association des AG avec des méthodes locales . . . 33
1.4 Parallélisme . . . 34
1.4.1 Parallélisme par îlots . . . 34
1.4.2 Parallélisation des calculs . . . 35
1.4.3 Conclusion . . . 35
1.5 Résultats de convergence théorique des algorithmes génétiques . . . 35

2 Elaboration d'un croisement adapté aux problèmes partiellement séparables **37**
2.1 Séparabilité partielle . . . 38
2.1.1 Définition . . . 38
2.1.2 Opérateur de croisement adapté : . . . 38
2.2 Etude théorique sur un exemple simple . . . 39
2.2.1 Probabilité d'amélioration . . . 40
2.2.2 Vitesse de convergence avec l'opérateur de croisement adapté . . . 43
2.2.3 Probabilité de sélection . . . 43
2.3 Résultats expérimentaux . . . 44

2.3.1 Algorithme génétique sans *sharing* . . . 44
2.3.2 Algorithme génétique avec *sharing* . . . 49
2.3.3 Influence du *sharing* . . . 49
2.4 Exemple de fonctions de test . . . 49
2.4.1 Une fonction totalement séparée : la fonction de Corana . . . 49
2.4.2 La fonction de Griewank . . . 52
2.5 Conclusion . . . 53

3 Exemple d'application : optimisation de la circulation des aéronefs sur un aéroport 55
3.1 Introduction . . . 55
3.1.1 Modélisation . . . 55
3.1.2 Fonction de coût . . . 56
3.1.3 L'aéroport . . . 56
3.1.4 Le trafic . . . 56
3.2 Simulation . . . 58
3.2.1 BB : Méthode 1 contre n . . . 58
3.2.2 GA et GA+BB : algorithmes génétiques . . . 59
3.3 Résultats . . . 61
3.3.1 Simulations . . . 61
3.3.2 Comparaison des stratégies . . . 61
3.3.3 Remarques . . . 63
3.4 Conclusion . . . 63

II La résolution des conflits en route 65

4 Le problème de résolution de conflits 67
4.1 Complexité du problème de résolution de conflit . . . 69
4.2 Approche centralisée, approche autonome . . . 69
4.3 Les projets d'automatisation existants . . . 70
4.3.1 La tentative américaine abandonnée, AERA . . . 70
4.3.2 ARC2000 et ses dérivés . . . 71
4.3.3 Le projet SAINTEX . . . 72
4.3.4 Le Projet FREER . . . 73
4.3.5 Les méthodes de forces répulsives. . . . 74
4.3.6 Résolution par programmation linéaire en nombres entiers . . . 76

5 Approches centralisées 77
5.1 Résolution de conflits par algorithmes génétiques . . . 77
5.1.1 Modélisation du problème . . . 77
5.1.2 Modélisation des manœuvres . . . 78
5.1.3 Modélisation des incertitudes . . . 79
5.1.4 La fonction d'évaluation . . . 79
5.1.5 L'opérateur de croisement adapté . . . 80
5.1.6 L'opérateur de mutation . . . 80
5.1.7 Autres paramètres . . . 80
5.1.8 Prise en compte de l'effet horizon . . . 82

5.1.9 Conflit à 5 avions . . . 82
5.1.10 Conflit à 20 avions . . . 82
5.2 AG et programmation linéaire . . . 84
5.2.1 Hypothèses . . . 84
5.2.2 Choix de la modélisation . . . 84
5.2.3 Mise en œuvre de l'algorithme génétique . . . 88
5.2.4 Conflit à 5 avions . . . 89
5.2.5 Conflit à 6 avions . . . 89
5.3 *Branch and Bound* par intervalles . . . 91
5.3.1 L'analyse d'intervalles . . . 91
5.3.2 *Branch and bound* par intervalles . . . 93
5.3.3 Problème à résoudre . . . 95
5.3.4 Conséquences de l'introduction des intervalles . . . 97
5.3.5 Déroulement de l'algorithme . . . 99
5.3.6 Approche globale . . . 100
5.3.7 Approche séquentielle . . . 102
5.3.8 Conclusion . . . 103
5.4 Programmation semi-définie (SDP) . . . 105
5.4.1 Le modèle . . . 105
5.4.2 Algorithmes d'optimisation convexe . . . 110
5.4.3 Application à la résolution de conflits . . . 111
5.4.4 Méthode de relaxation de problèmes quadratiques par SDP . . . 112
5.4.5 Application au problème de résolution de conflits. . . . 117
5.5 Conclusion . . . 120

6 Approches autonomes ou embarquées de résolution de conflits 121
6.1 Méthodes neuronales . . . 121
6.1.1 Les réseaux de neurones . . . 122
6.1.2 Génération de trajectoires d'évitement . . . 124
6.1.3 Évaluation des réseaux . . . 126
6.1.4 Résultats obtenus sur des conflits à deux avions . . . 129
6.1.5 Résolution de conflits à trois avions . . . 132
6.1.6 Résultats obtenus sur les conflits à trois avions . . . 132
6.1.7 Conclusion . . . 132
6.2 Méthode d'allocation de jetons et A^* . . . 135
6.2.1 Ordonnancement des avions . . . 135
6.2.2 Résolution d'un conflit à l'aide d'un algorithme A * . . . 139
6.2.3 Résultats sur 5 avions . . . 141
6.3 Conclusion . . . 142

A Convergence théorique des AGs 147
A.1 Théorie des schémas . . . 147
A.1.1 Définitions fondamentales . . . 147
A.1.2 Effets de la reproduction . . . 148
A.1.3 Effets des croisements . . . 149
A.1.4 Effets des mutations . . . 149
A.1.5 Conclusion sur le codage binaire . . . 149

A.2 Modélisation par chaînes de Markov 150
A.2.1 Description formelle rapide d'un algorithme génétique 150
A.2.2 Modélisation 151
A.2.3 Application de la théorie de Freidlin et Wentzell 156

B Rappels d'optimisation convexe **163**
B.1 Optimisation convexe sans contrainte 163
B.1.1 Algorithme de Newton modifié 163
B.1.2 Méthode de gradient conjugué 164
B.1.3 BFGS 164
B.2 Opti convexe sous contraintes d'inégalités 167
B.2.1 Méthode proximale entropique EPM 167
B.2.2 Méthodes de multiplicateurs 168

Glossaire

ADS-B : Automatic Dependent Suveillance - Broadcast. Programme de surveillance ATC par Data Link.

AERA : Advanced EN-Route ATC. Projet d'automatisation de contrôle en route.

AMPF : Airspace Manager Planning Functions. Un composant de AERA utilisé pour planifier l'occupation de l'espace aérien.

ASAS : Airborne Separation Assurance System. Système d'aide au maintien de séparation embarqué.

ASF : Automated Separation Function. Un composant de AERA utilisé pour séparer les paires d'avions.

ATC : Air Traffic Control. Contrôle du trafic aérien.

ATFM : Air Traffic Flow Management. Gestion des flux de trafic aérien.

ATMS : Air Traffic Management System. Système de gestion du trafic aérien.

BADA : Base of Aircraft DAta. Base de données avion utilisée pour la simulation de vol.

CASA : Computer Assisted Slot Allocation. Système d'allocation de créneau.

CATS : Complete Air Traffic Simulator. Simulateur de trafic aérien.

CAUTRA : Coordonnateur AUtomatique TRafic Aérien

CENA : Centre d'Etudes de la navigation Aérienne.

CFMU : Central Flow Management Unit. Cellule européenne de gestion des flux de trafic.

CNS : Communication Navigation and Surveillance.

Data-Link : Liaison de données. Terme généralement employé dans le domaine aéronautique pour désigner les liaisons de données sol-air ou air-air, afin de renseigner soit les systèmes sols, soit les autres avions sur l'état et/ou les intentions d'un avion donné.

DGAC : Direction Générale de l'Aviation Civile.

EATMS : European Air Traffic Management System. Futur système européen.

EFR : Extended Flight Rules. Règles de vol étendues utilisées dans Free-R.

ENAC : Ecole Nationale de l'Aviation Civile.

EOBT : Estimated Off-Block Time. Heure estimée du départ du parking.

FAA : Federal Aviation Administration. Organisme américain de l'administration de l'aviation.

FANS : Futur Air Navigation System. Comité issu de l'OACI chargé de traiter des concepts de gestion de trafic.

FFA : Free-Flight Airspace. Espace aérien réservé au Free-Flight.

FMP : Flow Management Position. Cellule de gestion des flux du trafic aérien d'un centre de contrôle travaillant en collaboration avec la CFMU.

FMS : Flight Management System. Système embarqué de gestion du vol (paramètres, poussée, consommation, type de croisière...)

FREER : Free Route Experimental Encounter Resolution. Projet de vol free-flight d'Eurocontrol.

GPS : Global Positionning System. Système de positionnement par satellite provenant du département de la défense américain.

HIPS : Highly Interactive Problem Solver

IFR : Instrument Flight Rules. Règles de vol aux instruments.

MOM : Maneuver Option Manager. Un composant de AERA utilisé pour gérer un ensemble de manoeuvre.

OACI : Organisation de l'Aviation Civile Internationale (ICAO International Civil Aviation Organization).

STD : Standard Traveled Distance. Distance moyenne parcourue.

STIP : Système de Traitement Initial Plan de vol.

STT : Standard Transit Time. Temps de transit moyen.

TCAS : Traffic alert and Collision Avoidance System. Système anti-abordage embarqué à bord des avions.

TMA : Terminale Maneuvering Area. Région de contrôle terminale. Région de contrôle établie en principe au carrefour de routes ou aux environs d'un ou plusieurs aérodromes importants.

VFR : Visual Flight Rules. Règles de vol à vue.

Introduction

La recherche dans le domaine de la gestion du trafic aérien est encore au stade du balbutiement. Pendant longtemps on a cru qu'en augmentant le nombre de contrôleurs aériens, de secteurs de contrôle, de pistes et de taxiways sur les aéroports, on allait pouvoir absorber l'augmentation du trafic. On s'est ensuite intéressé à réduire les espacements entre avions, augmenter les cadences sur les pistes, réorganiser localement le réseau de routes ou les grandes approches. C'est seulement vers la fin des années 80 que les premiers projets de recherche ont débuté dans le domaine de la gestion du trafic aérien. Ils furent d'abord lancés par des équipes issues du monde opérationnel ayant une bonne connaissance de l'environnement des problèmes, mais peu de compétences dans les domaines mathématiques et informatiques. Des approches plus théoriques sont ensuite apparues, menées par des chercheurs issus du monde universitaire, parfois tentés d'oublier certaines contraintes opérationnelles liées aux problèmes de gestion du trafic aérien, au profit des outils qu'ils maîtrisaient. La grande difficulté dans toute tentative d'optimisation d'un problème opérationnel consiste à le simplifier suffisamment pour permettre d'en extraire un modèle mathématique sur lequel on va pouvoir utiliser des méthodes d'optimisation évoluées, sans pour autant rendre impossible la prise en compte ultérieure des contraintes opérationnelles que l'on aura mises de côté pendant la phase de modélisation.

Anecdote : en 2003, j'étais relecteur dans deux conférences scientifiques, l'une (5^{th} USA / Europe ATM R & D Seminar[1]) consacrée aux recherches menées dans le monde de la gestion du trafic aérien, l'autre (EA03[2]) consacrée aux méthodes d'optimisation à base d'algorithmes évolutionnaires. Familier des deux domaines, je reste toutefois étonné par le fossé qui subsiste entre les préoccupations de deux mondes. Ainsi, je n'ai pas trouvé dans les articles soumis à ATM R & D 2003 les éléments nécessaires à la compréhension de la complexité des problèmes étudiés, ni même parfois une évocation, même succincte, des algorithmes utilisés dans le cadre de la résolution de ces problèmes, comme si les auteurs pensaient que ces aspects ne constituaient qu'un détail. A l'inverse, dans les articles présentés à EA03, j'aurais parfois aimé que les auteurs ne se limitent pas à tester leur nouvel opérateur de croisement uniquement sur le "one-max" problème ou le problème de la "voie royale", dont on ne sait pas trop bien pourquoi ils seraient représentatifs de quelque difficulté que ce soit.

A la fin de ma première année de thèse, après avoir désespérément tenté d'appliquer la théorie de la commande optimale sur un problème de résolution de conflits aériens, mon directeur de thèse et moi nous sommes retrouvés face à un dilemme : oublier les contraintes de vol d'un avion et pouvoir ainsi poursuivre dans la voie dans laquelle nous étions engagés, ou se souvenir que ma thèse était financée par la direction générale de l'aviation civile et s'attaquer au problème différemment, quitte à laisser tomber quelques beaux résultats de commande optimale que nous aurions pu obtenir. C'est cette dernière voie que j'ai choisie, et qui m'a conduit à me spécialiser depuis dans le domaine des algorithmes génétiques, tout en comparant les résultats obtenus avec d'autres algorithmes d'optimisation globale.

[1] Air Traffic Management Research and Development

[2] Evolution artificielle 2003

Le Laboratoire d'Optimisation Globale (LOG) dont je suis directeur adjoint a été créé en 1996 sous la direction de Jean-Marc Alliot. J'ai participé à sa création avec Jean-Marc Alliot et Pascal Brisset à la fin de ma thèse. Il réunit une dizaine de chercheurs du Centre d'Etudes de la Navigation Aérienne (CENA) et de l'Ecole Nationale de l'Aviation Civile (ENAC). Les activités du LOG se concentrent sur la modélisation et l'optimisation de problèmes de gestion du trafic aérien.

Mes activités de recherches menées au LOG se partagent en deux grands domaines.

- **Modélisation de la gestion du trafic aérien :** Il s'agit d'identifier, de simplifier et de modéliser des problèmes de gestion du trafic aérien pour pouvoir les résoudre grâce à des *méthodes d'optimisation*. La modélisation pose de nombreux problèmes. Il faut *séparer* le problème global en sous-problèmes de tailles raisonnables, *identifier les contraintes essentielles* du problème et celles qui pourront être traitées dans une phase plus avancée de la modélisation, modéliser les problèmes d'*incertitudes* liées au domaine d'application et prendre en compte les contraintes de fonctionnement du système en *temps réel*. Parmi les problèmes modélisés, on peut citer le problème de résolution de conflits aériens dans différents contextes (résolution centralisée, résolution autonôme embarquée, en routes directes ou balisées), mais également, le problème du roulage des avions sur la plateforme aéroportuaire, de l'organisation du réseau de routes, de la sectorisation ou encore des regroupements de secteurs aériens.
- **Algorithmes de résolution numérique :** Les problèmes précédemment cités sont généralement à *variables mixtes* (continues et discrètes). Ce sont des problèmes d'*optimisation fortement combinatoire* ayant de nombreux minima locaux. Leur *grande taille* et l'*absence d'expression analytique* des fonctions d'évaluation (résultats de simulations) rendent généralement impossible leur résolution par des méthodes déterministes classiques. Enfin, les contraintes d'incertitude et de temps réel rendent cruciale la recherche de solution admissibles dans un temps raisonnable. Initialement, les *algorithmes génétiques* se sont révélés très efficaces pour la résolution des problèmes de conflits en route ou pour le roulage au sol. La nature *partiellement séparable* de ces problèmes m'a permis de développer des *opérateurs de croisement et de mutation adaptés* permettant d'améliorer largement les performances des algorithmes génétiques. J'étudie également d'autres approches parmi lesquelles des méthodes de *réseaux de neurones*, de *Branch and Bound par intervalles*, de *programmation semi-définie*, de *colonies de fourmis*... L'*hybridation* de méthodes stochastiques et de méthodes déterministes est plus particulièrement explorée.

Depuis 1996, j'ai co-encadré avec Jean-Marc Alliot les thèses de plusieurs étudiants :

- Frédéric MÉDIONI (CMAPX Ecole Polytechnique 1998) " Méthode d'optimisation pour l'évitement aérien : systèmes centralisés, systèmes embarqués"
- Géraud GRANGER (CMAPX Ecole Polytechnique 2001) "Détection et résolution de conflits aériens : modélisations et analyse"
- Jean-Baptiste GOTTELAND (LIMA ENSEEIHT 2004 (soutenance prévue en novembre)) "Optimisation de la circulation des aéronefs au sol"
- David GIANAZZA (LIMA ENSEEIHT 2004 (soutenance prévue en novembre)) "Optimisation des flux de trafic aériens"
- Nicolas ARCHAMBAULT (LIMA ENSEEIHT 2^{eme} année de thèse) "Développement d'algorithmes et d'outils de prévision de trajectoires et de détection / résolution de conflits pour le contrôle aérien"
- Charles-Edmond BICHOT (LIMA ENSEEIHT 1^{ere} année de thèse) "Optimisation des zones de qualification de l'espace aérien européen"

J'ai également co-encadré avec Jean-Marc Alliot les stages de DEA et de DRU des étudiants suivants :

- Christophe BONTEMPS (DEA IFP ENSEEIHT 1997) : "Prévision stochastique de trajectoires : procédures paramétriques et non-paramétriques"
- Boris BARTOLOMÉ (DEA IFP ENSEEIHT 1998) : "Optimisation de la matrice d'entrelacement des Turbo-Codes convolutifs par algorithmes génétiques"
- Géraud GRANGER (DEA IFP ENSEEIHT 1998) : "Résolution de conflits embarquée dans les espaces de faible densité"
- Pierre DODIN (DEA Maths Appliquées PARIS VI 1999) : "Résolution de conflits via la programmation semi-définie"
- Brankica PESIC (DRU PS ENSEEIHT 2000) : "Optimisation de la circulation des aéronefs au sol sur la plate-forme de Roissy"
- Catherine RONFLE-NADAUD (DEA PS ENSEEIHT 2001) : "Optimisation des trajectoires d'avions au sol"
- Nicolas ARCHAMBAULT (DEA IARFA PARIS VI 2003) : "Optimisations du solveur de conflit du simulateur de Trafic CATS"
- Charles-Edmond BICHOT (DEA SI LAAS 2004) : "Optimisation du découpage de l'espace aérien par diverses métaheuristiques"

J'ai par ailleurs été plusieurs fois relecteur pour les conférences *Evolution Artificielle*, PPSN, GECCO et CEC. Enfin je suis membre des comités de programme d'ATM R & D 2003 et 2005 (Air Traffic Management Research and Development) USA/Europe, d'ICRAT 2004 (International Conference on Research in Air Transportation) et d'RIVF 2004 et 2005 (Journées de Recherche en Informatique Vietnamiennes et Francophones).

La première partie de ce document est consacrée aux algorithmes génétiques et notamment à leur adaptation à des problèmes d'optimisation partiellement séparables, souvent rencontrés dans le monde opérationnel de l'aviation civile. Le travail réalisé au cours de ma thèse de doctorat m'a permis par la suite de développer un opérateur de croisement efficace, adapté à certains problèmes d'optimisation dit "partiellement séparables" pour lesquels la fonction à minimiser est la somme de termes positifs ne faisant chacun intervenir qu'une partie des variables du problème. A titre d'exemple, une application permettant d'optimiser le roulage des avions sur une plate-forme aéroportuaire est présentée à la fin de la première partie.

La deuxième partie aborde les problèmes de résolution de conflits aériens et les diverses approches envisagées pour le résoudre. Après un état de l'art, différentes modélisations utilisant des méthodes d'optimisation diverses sont comparées. On aborde entre autres des méthodes utilisant la programmation linéaire, la programmation semi-définie, des réseaux de neurones, des méthodes d'intervalles, avec à chaque fois, une modélisation spécifique du problème de résolution de conflits dont on détaille les avantages et les inconvénients.

Ces dix dernières années ont donné lieu à plusieurs publications dont la liste figure à la fin de ce document.

Première partie

Les algorithmes génétiques

Jean-Marc Alliot et moi-même avons commencé à programmer un algorithme génétique en 1993, pour optimiser le problème de résolution de conflits aériens en route. Les difficultés rencontrées sur ce problème avec des méthodes d'optimisation classique nous ont rapidement conduit vers cette famille d'algorithmes capable de gérer des problèmes combinatoires de grande taille à variable mixte et dont l'expression de la fonction de coût est le résultat d'une simulation sans expression analytique. C'est par abus de langage que j'ai souhaité conserver le terme *d'algorithmes génétiques* pour désigner ce que les théoriciens du domaine appellent désormais les *algorithmes évolutionnaires*.

Ces derniers font partie de ce que l'on appelle les *métaheursistiques* au même titre que les méthodes de *recuit simulé*, de *recherche avec tabous*, de *colonies de fourmis*, d'*essaims particulaires*, …, grâce auquelles on tente de résoudre des problèmes d'optimisation difficiles. Plus intéressé par la résolution de problèmes pratiques que par une classification générale des problèmes d'optimisation et de méthodes suceptibles de les résoudre, je laisserai le lecteur intéressé par une vision plus globale du domaine se reporter à l'ouvrage de Patrick Siarry [DPST03] consacré aux métaheuristiques pour l'optimisation difficile. Cette partie est divisée en trois chapitres : le premier présente les algorithmes génétiques, leur mise en oeuvre, les principales améliorations classiques et quelques résultats de convergence théorique. Le deuxième chapitre présente un opérateur de croisement qui permet d'accélerer la convergence des algorithmes génétiques sur des problèmes partiellement séparables. Cet opérateur est utilisé dans l'exemple d'application présenté dans la troisième partie, consacrée à l'optimisation de la circulation des aéronefs sur une plate-forme aéroportuaire. J'ai volontairement placé cet exemple d'application dans la première partie de mon document car c'est sans doute la modélisation la plus réaliste d'un problème réel que nous ayons pu entreprendre au LOG. Il fait ainsi le lien entre les activités de modélisation et d'optimisation que nous faisons coexister au sein de notre laboratoire.

Chapitre 1

AG : généralités et améliorations

1.1 Principes généraux

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle[1] : croisements, mutations, sélection, etc. Les algorithmes génétiques ont déjà une histoire relativement ancienne, puisque les premiers travaux de John Holland sur les systèmes adaptatifs remontent à 1962 [Hol62]. L'ouvrage de David Goldberg [Gol89c] a largement contribué à les vulgariser.

Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants :

1. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. Le choix du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très employés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs, pour l'optimisation de problèmes à variables continues.
2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.
3. Une fonction à optimiser. Celle-ci prend ses valeurs dans $\mathbb{R}^+$ et est appelée *fitness* ou fonction d'évaluation de l'individu. Celle-ci est utilisée pour selectionner et reproduire les meilleurs individus de la population.
4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.
5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la figure 1.1.

[1] Il est intéressant de trouver dans l'œuvre d'un spécialiste de zoologie, Richard Dawkins [Daw89], un exemple informatique tendant à prouver la validité de l'hypothèse darwinienne de la sélection naturelle. La méthode utilisée est presque semblable aux techniques génétiques.

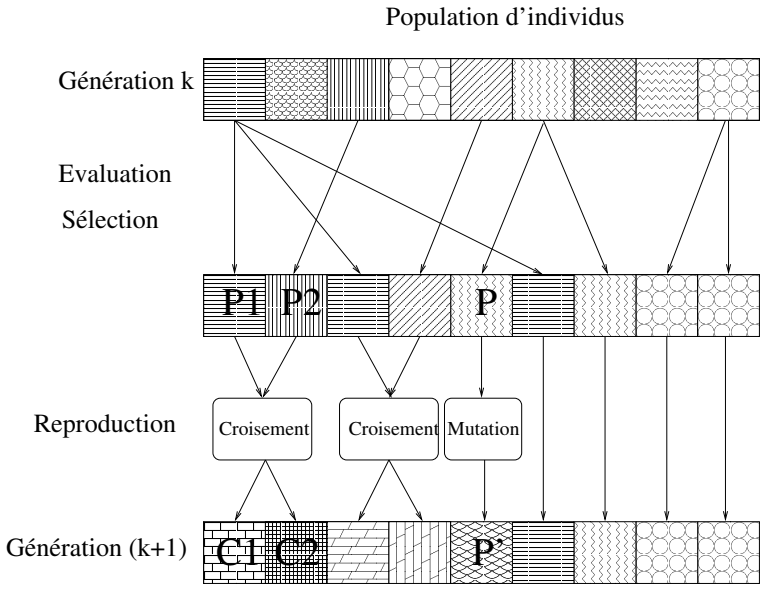

FIG. 1.1 – Principe général des algorithmes génétiques

On commence par engendrer une population d'individus de façon aléatoire. Pour passer d'une génération k à la génération $k+1$, les trois opérations suivantes sont répétées pour tous les éléments de la population k. Des couples de parents P_1 et P_2 sont sélectionnés en fonction de leurs adaptations. L'opérateur de croisement leur est appliqué avec une probabilité P_c (généralement autour de 0.6) et engendre des couples d'enfants C_1 et C_2. D'autres éléments P sont sélectionnés en fonction de leur adaptation. L'opérateur de mutation leur est appliqué avec la probabilité P_m (P_m est généralement très inférieur à P_c) et engendre des individus mutés P'. Les enfants (C_1,C_2) et les individus mutés P' sont ensuite évalués avant insertion dans la nouvelle population (la figure 1.1 présente le cas où les enfants et les individus mutés remplacent les parents). Différents critères d'arrêt de l'algorithme peuvent être choisis :

- Le nombre de générations que l'on souhaite exécuter peut être fixé a priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.
- L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement.

1.2 Description détaillée

1.2.1 Codage des données

Historiquement, le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace d'état. Ce type de codage a pour intérêt de permettre de créer des opérateurs de croisement et de mutation simples. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus.

Cependant, ce type de codage n'est pas toujours bon :

- deux éléments voisins en terme de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un

codage de Gray.

– 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.

Les algorithmes génétiques utilisant des vecteurs réels [Gol91, Wri91] évitent ce problème en conservant les variables du problème dans le codage de l'élément de population, sans passer par le codage binaire intermédiaire. Certains les appellent RCGA (*Real Coded Genetic Algorithms*, d'autres parlent d'algorithmes évolutionnaires. La structure du problème est conservée dans le codage.

1.2.2 Génération aléatoire de la population initiale

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 d'engendrer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état, en veillant à ce que les individus produits respectent les contraintes [MJ91]. Si par contre, des informations a priori sur le problème sont disponibles, il paraît bien évidemment naturel d'engendrer les individus dans un sous-domaine particulier afin d'accélérer la convergence. Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités.

1.2.3 Gestion des contraintes

Un élément de population qui viole une contrainte se verra attribuer une mauvaise *fitness* et aura une probabilité forte d'être éliminé par le processus de sélection.

Il peut cependant 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 *fitness* 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.

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.

1.2.4 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.

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 C_1 et C_2 (voir figure 1.2).

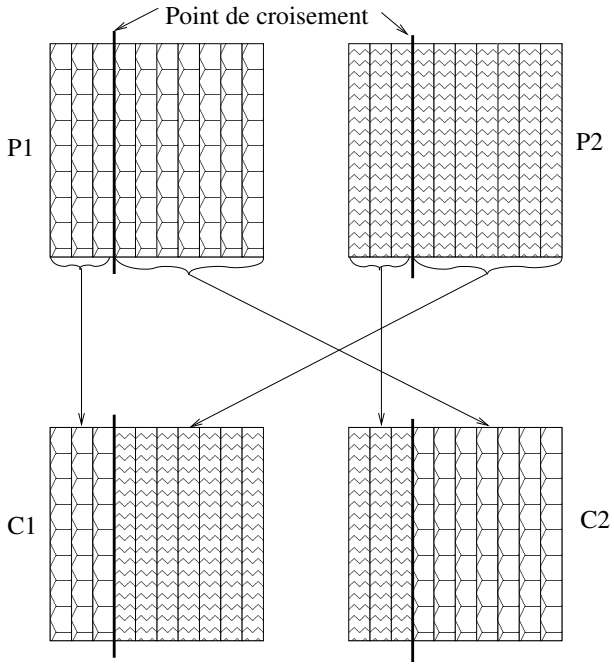

FIG. 1.2 – Croisement à 1 point

On peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaînes mais en 3, 4, etc [BG91]. (voir figure 1.3).

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets. Pour les problèmes continus, un croisement "barycentrique" est souvent utilisé : deux gènes $P_1(i)$ et $P_2(i)$ sont sélectionnés dans chacun des parents à la même position i. Ils définissent deux nouveaux gènes $C_1(i)$ 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 α 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 d'engendrer des points entre, ou à l'extérieur des deux gènes considérés).

Dans le cas particulier d'un chromosome matriciel constitué par la concaténation de vecteurs, on peut étendre ce principe de croisement aux vecteurs constituant les gènes (voir figure 1.4) :

$$\begin{cases} \vec{C_1}(i) = \alpha \vec{P_1}(i) + (1-\alpha) \vec{P_2}(i) \\ \vec{C_2}(i) = (1-\alpha) \vec{P_1}(i) + \alpha \vec{P_2}(i) \end{cases}$$

On peut imaginer et tester des opérateurs de croisement plus ou moins complexes sur un problème donné mais l'efficacité de ces derniers est souvent liée intrinsèquement au problème.

1.2.5 Opérateur de mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les

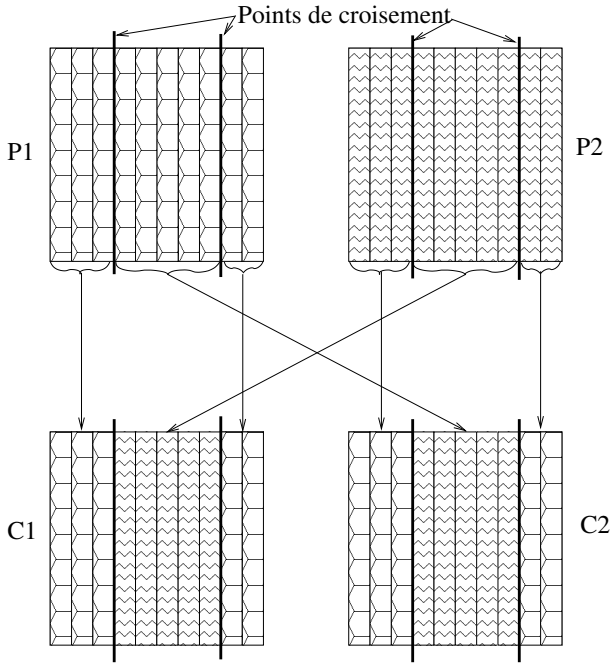

FIG. 1.3 – Croisement à 2 points

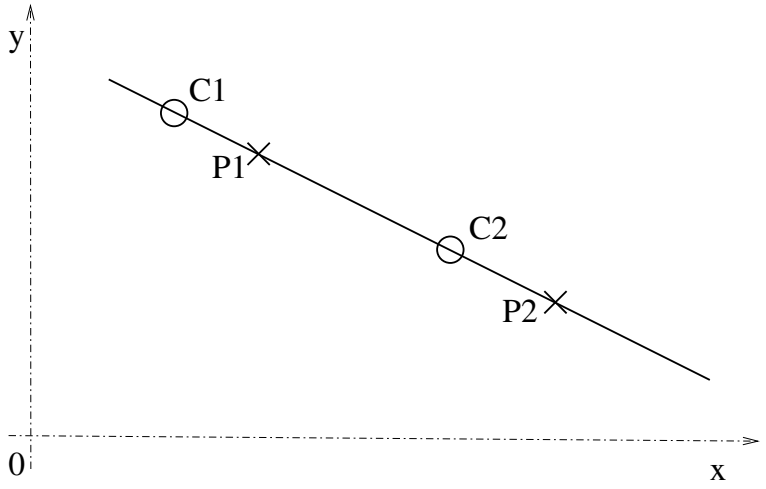

FIG. 1.4 – Croisement barycentrique

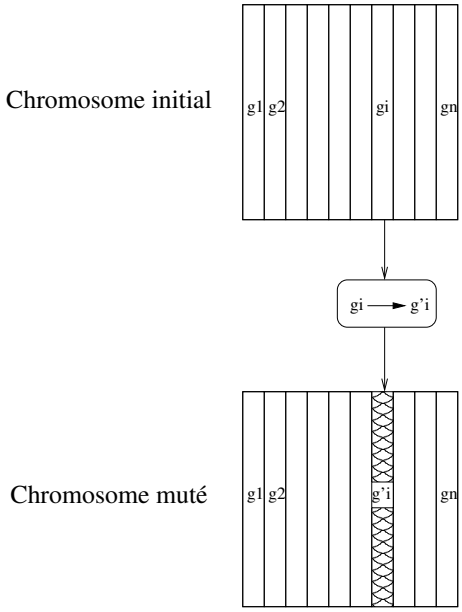

FIG. 1.5 – Principe de l'opérateur de mutation

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 implémentations 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 (voir figure 1.5). Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale.

Dans les problèmes continus, on procède un peu de la même manière en tirant aléatoirement un gène dans le chromosome, auquel on ajoute un bruit généralement gaussien. L'écart-type de ce bruit est difficile à choisir a priori.

1.2.6 Principes de sélection

A 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 peu d'hypothèses requises permet de traiter des problèmes très complexes. La fonction à optimiser peut ainsi être le résultat d'une simulation.

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent. Les deux principes de sélection suivants sont les plus couramment utilisés :

– *Roulette wheel selection* [Gol89c] ;
– *Stochastic remainder without replacement selection* [Gol89c] ;

Le principe de *Roulette wheel selection*[2] consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa *fitness*. On reproduit ici le principe de tirage aléatoire utilisé dans les roulettes de casinos avec une structure linéaire. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1. On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on "regarde" quel est le segment sélectionné. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent choisis que les petits. Lorsque la dimension de la population est réduite, il est difficile d'obtenir en pratique l'espérance mathématique de sélection en raison du peu de tirages effectués. Un biais de sélection plus ou moins fort existe suivant la dimension de la population.

La *Stochastic remainder without replacement selection* évite ce genre de problème et donne de bons résultats pour nos applications. Décrivons ce principe de sélection :

- 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* $r_i - e(r_i)$.

Lorsque des populations de faible taille sont utilisées, ce principe de sélection s'avère généralement efficace dans les applications pratiques.

1.3 Améliorations classiques

1.3.1 Introduction

Les processus de sélection présentés 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. Ainsi, dans l'exemple de la figure 1.6 le second mode M_2 risque d'être le seul représentant pour la génération suivante et seule la mutation pourra aider à atteindre l'objectif global M_1, au prix de nombreux essais successifs.

Pour éviter ce comportement, il existe d'autres modes de sélection (*ranking*) ainsi que des principes (*scaling, sharing*) qui empêchent les individus "forts" d'éliminer complètement les plus "faibles". On peut également modifier le processus de sélection en introduisant des tournois entre parents et enfants, basés sur une technique proche du recuit simulé.

Enfin, on peut également introduire des recherches multiobjectif, en utilisant la notion de dominance lors de la sélection.

1.3.2 *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*, on peut envisager le *scaling* linéaire et le *scaling* exponentiel. Soit f_r la *fitness* avant *scaling* et f_s la *fitness* modifiée par le *scaling*.

Scaling linéaire

Dans ce cas la fonction de *scaling* est définie de la façon suivante [Mic92a] :

[2] Dans la littérature, cette méthode porte parfois le nom de méthode de Monte-Carlo.

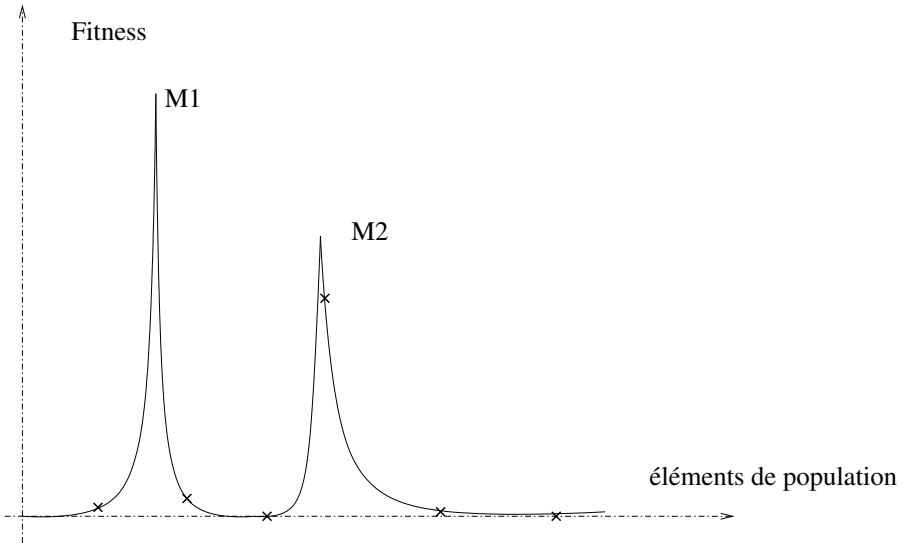

FIG. 1.6 – Exemple où les sélections classiques risquent de ne reproduire qu'un individu

$$f_s = a f_r + b$$

$$a = \frac{max' - min'}{max - min};\ b = \frac{min'.max - min.max'}{max - min}.$$

En règle générale, le coefficient a est inférieur à un, 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.

La figure 1.7 donne un exemple de l'influence de la fonction de *scaling* sur la pression de sélection.

Scaling exponentiel

Il est défini de la façon suivante [Mic92a] (voir figure 1.8) :

$$f_s = (f_r)^{k(n)}$$

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 :

$$k = \left(\tan \left[\left(\frac{n}{N+1} \right) \frac{\pi}{2} \right] \right)^p$$

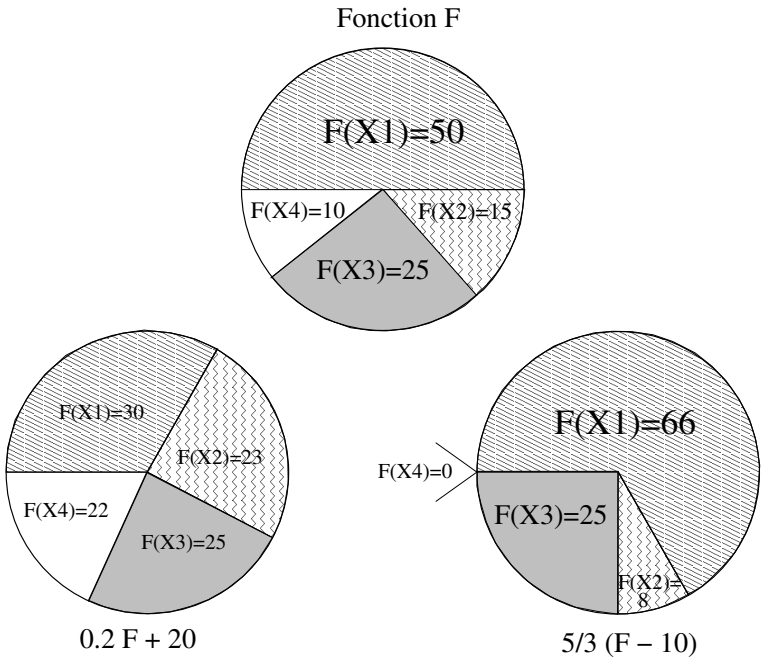

FIG. 1.7 – Influence de la fonction de *scaling* sur la pression de sélection.

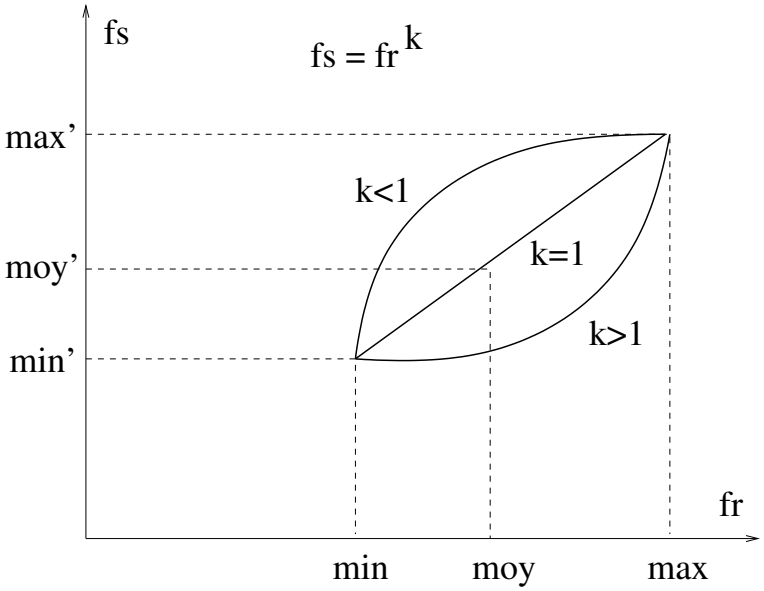

FIG. 1.8 – Fonction de *scaling* exponentielle

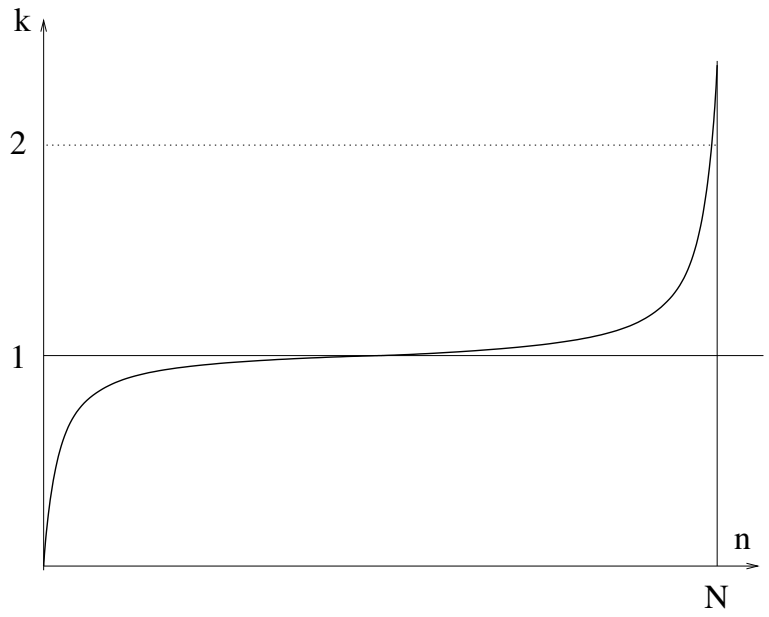

FIG. 1.9 – Allure de l'évolution de k en fonction des générations

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 est donnée par la figure 1.9.

Dans le cas des fonctions objectifs multi-modes présentant des optima quasi-équivalents, cette technique de *scaling*, en amplifiant les écarts de *fitness* en fin de convergence, va effectivement favoriser le mode dominant, mais aussi masquer les modes sous-optimaux, qui peuvent tout de même présenter un intérêt. Le *scaling* permet donc une bonne exploration de l'espace d'état, mais ne favorise pas la répartition des individus sur les différents modes de la fonction objectif.

1.3.3 *Sharing*

Introduction

L'objectif du *sharing* est de répartir les individus de la population sur tous les sommets de la fonction à optimiser. La figure 1.10 présente deux exemples de répartitions de populations dans le cas d'une fonction à cinq sommets : le premier sans *sharing*, le second avec *sharing*.

Principe

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 :

$$f'_i = \frac{f_i}{m'_i};\ m'_i = \sum_{j=1}^{N} S(d(x_i, x_j))$$

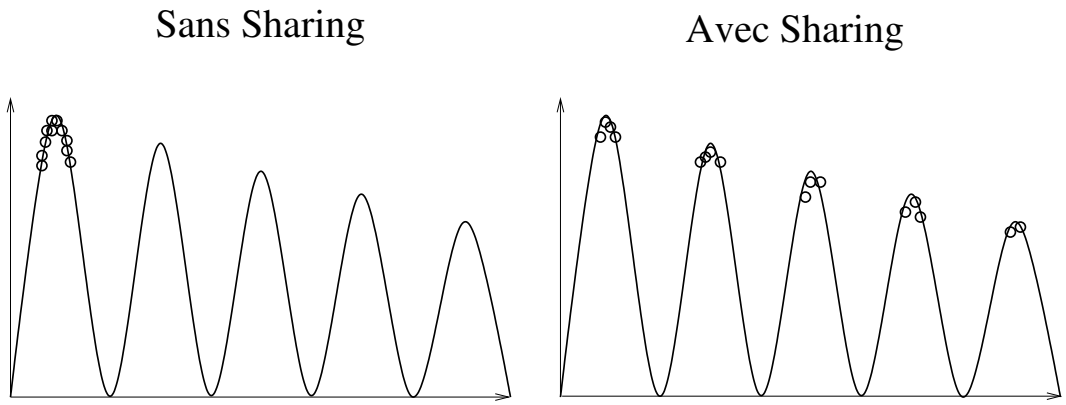

FIG. 1.10 – Objectif du *sharing*

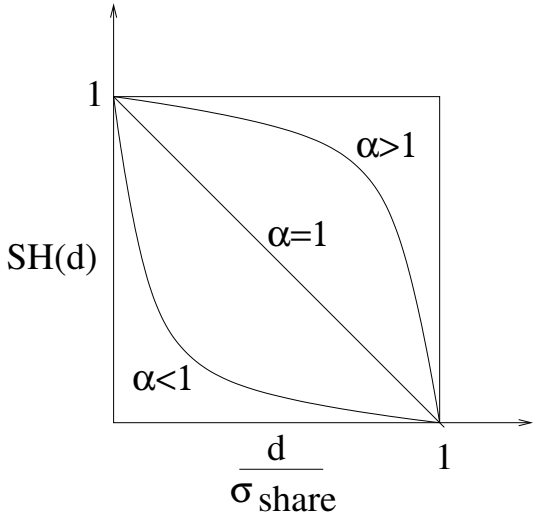

FIG. 1.11 – Allure de $S(\frac{d}{\sigma_{share}})$

avec

$$S(d) = 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} \text{ si } d < \sigma_{share}$$

$$S(d) = 0 \text{ si } d > \sigma_{share}$$

Le paramètre σ_{share} permet de délimiter le voisinage d'un point et dépend du problème traité. La figure 1.11 donne l'allure de $S(d)$ pour différentes valeurs de α. Suivant la valeur donnée à α 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".

1.3.4 *Sharing* clusterisé

Pour effectuer ce type de *sharing* [YG93a], on commence par identifier les différents groupes 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 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.
- α : 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 $O(N \log N)$[YG93a] 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, distance qu'il est très difficile d'estimer. Nous présentons dans la section **??** une technique permettant de calculer automatiquement ces quantités.

1.3.5 Algorithmes génétiques et recuit simulé

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 [IR92b]. 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 d'hybridation entre les algorithmes génétiques et le recuit simulé, les travaux les plus intéressants étant ceux de Mahfoud et de Goldberg [MG92].

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 1.12). On applique ensuite l'opérateur de croisement classique qui

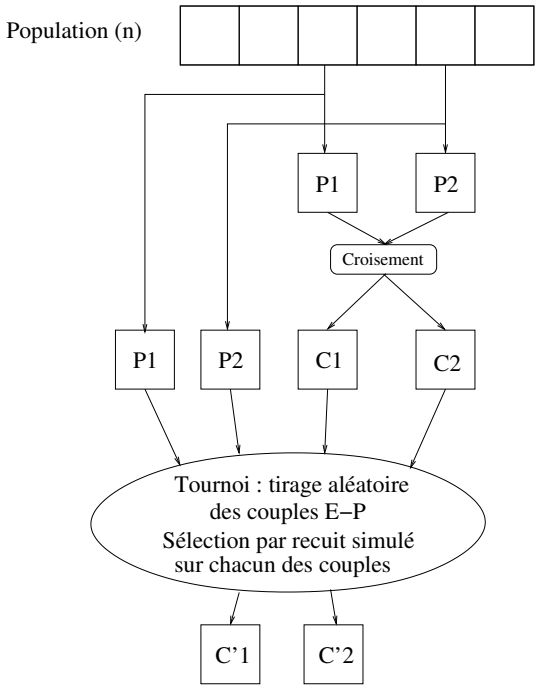

FIG. 1.12 – Principe du croisement avec recuit simulé

génère deux enfants C_1 et C_2. Un tournoi est alors effectué entre les parents et les enfants, pour lequel 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'.

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.

$$\begin{cases} C_s = -\frac{\Delta f_{max}}{ln\left(\frac{1}{k-1}\right)} & k = 0.75 \quad \text{"Température initiale"} \\ C_x = -\frac{\Delta f_{max}}{ln\left(\frac{1}{k-1}\right)} & k = 0.99 \quad \text{"Température de basculement"} \\ C_f = -\frac{\Delta f_{min}}{ln\left(\frac{1}{k-1}\right)} & k = 0.99 \quad \text{"Température finale"} \end{cases}$$

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.

Ce même principe de recuit simulé appliqué sur l'opérateur de mutation réduit le brassage de la population provoqué par la mutation, en limitant l'espace exploré aux zones qui améliorent statistiquement la *fitness* en "interdisant" les domaines qui la dégradent. L'exploration du domaine admissible est fragilisée.

1.3.6 Recherche multiobjectif

Introduction

Dans le cadre de la recherche multiobjectif, on cherche à optimiser une fonction suivant plusieurs critères, dont certains peuvent d'ailleurs être antagonistes. On définit alors la classique notion de dominance : on dit que le point A domine le point B si, $\forall i, f_i(A) \geq f_i(B)$ et $\exists i, f_i(A) > f_i(B)$, où les f_i représentent les critères à maximiser. L'ensemble des points qui ne sont dominés par aucun autre point forme le front de Pareto. Tout point du front de Pareto est "optimal", dans la mesure où on ne peut améliorer la valeur d'un critère pour ce point sans diminuer la valeur d'au moins un autre critère.

Les algorithmes génétiques peuvent permettre de trouver l'ensemble de la surface de Pareto, car il est possible de répartir la population de l'algorithme génétique sur ladite surface.

Technique employée

La technique employée dérive directement des travaux de Jeffrey Horn et Nicholas Nafpliotis ([HN93]). Le principal changement induit concerne le processus de sélection : en multiobjectif, comment décider qu'un individu est meilleur qu'un autre[3] ? On introduit alors une variante de la notion de dominance, que l'on définit ainsi : on peut par exemple décider que l'élément E_i domine E_j si le nombre des valeurs contenues dans son vecteur d'adaptation qui sont supérieures aux valeurs correspondantes dans E_j dépasse un certain seuil. A partir de là, la technique proposée pour effectuer la sélection est simple : on tire deux individus au hasard, ainsi qu'une sous-population[4] à laquelle ils n'appartiennent pas, et qui va servir à les comparer.

Trois cas se présentent alors :

- si le premier élément domine tous ceux de la sous-population et que ce n'est pas le cas pour le second, alors le premier sera sélectionné.
- inversement, si seul le second domine l'ensemble de la sous-population, alors c'est lui qui sera conservé.
- restent maintenant deux possibilités : soit les deux sont dominés, soit les deux dominent. On ne peut se prononcer sans ajouter un autre test, c'est pourquoi, dans ce cas, il est fait usage d'un nouveau type de *sharing*, qui opère sur l'espace objectif.

[3] En ce qui concerne les autres opérateurs de base, à savoir le croisement et la mutation, il n'y a aucune modification, car ceux-ci travaillent dans l'espace d'état, et non dans l'espace résultat.

[4] La sous-population tirée aura une taille proportionnelle à celle de la population de départ. Seule une partie des individus est utilisée, ce qui permet de réduire le temps de calcul.

Le *sharing* va conduire à sélectionner celui des deux individus qui a le moins de voisins proches, autrement dit on élimine celui qui se situe dans une zone d'agrégation, pour conserver celui qui est dans une région moins dense.

Encore une fois, le terme *voisin proche* n'a pas de signification précise, mais il est possible de définir un voisinage (aussi appelé "niche") correct, en se servant de la distance de Holder :

$$d_H(E_i, E_j) = \left(\sum_{k=1}^{n} |f_i^k - f_j^k|^p \right)^{\frac{1}{p}}$$

f_i^k désignant la k-ième composante du vecteur adaptation de l'élément i. Le paramètre p permet de faire varier la forme et la taille du voisinage. A l'intérieur des voisinages ainsi définis dans l'espace objectif, il suffit de compter le nombre d'individus pour favoriser les zones les moins denses et, de cette façon, maintenir la diversité de la population. Ainsi la figure 1.13 montre comment les voisinages sont choisis autour des individus de la région de Pareto, lorsque ceux-ci ne peuvent être départagés sans *sharing*.

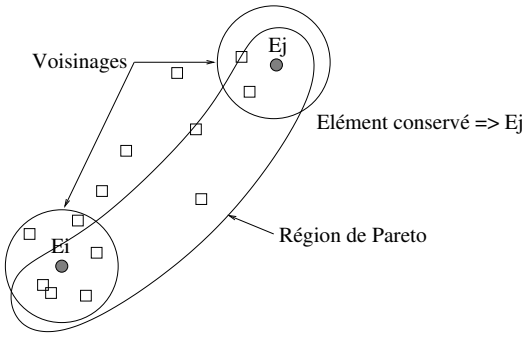

FIG. 1.13 – Surface de Pareto et voisinages

1.3.7 Association des AG avec des méthodes locales

La grande force des algorithmes génétiques est leur capacité à trouver la zone de l'espace des solutions contenant l'optimum de la fonction. En revanche, ils sont inefficaces lorsqu'il s'agit de trouver la valeur exacte de l'optimum dans cette zone. Or, c'est précisément ce que les algorithmes locaux d'optimisation réalisent le mieux.

Il est donc naturel de penser à associer un algorithme local à l'algorithme génétique de façon à trouver la valeur exacte de l'optimum. On peut aisément le faire en appliquant après l'algorithme génétique un algorithme local sur le meilleur élément trouvé. Cette technique est d'autant plus efficace que l'on utilise simultanément du *clustering*, et que l'algorithme local est appliqué à chaque meilleur élément de chaque *cluster*. En effet, on constate souvent que le meilleur élément trouvé par l'algorithme génétique ne reste pas le meilleur élément, après amélioration par l'algorithme local de tous les meilleurs éléments de *clusters*.

Une autre technique consiste à utiliser un algorithme local associé à l'algorithme génétique pour calculer la *fitness* d'un élément. On peut, par exemple dans un espace fortement combinatoire, rechercher avec l'algorithme génétique les zones intéressantes de l'espace en générant les éléments, l'adaptation de chaque élément étant calculée par un programme d'optimisation local (linéaire par exemple). Un exemple de ce type est donné dans la section 5.2.

En fait, l'association AG–méthodes locales est une quasi-nécessité. Les deux méthodes sont complémentaires et ont des champs d'application différents. L'algorithme génétique permet de faire "disparaître" la combinatoire du problème, laissant alors le champ libre aux méthodes locales, dans chacune des zones connexes qu'il pense susceptible de contenir l'optimum global.

1.4 Parallélisme

L'intérêt de la parallélisation des algorithmes génétiques est de gagner en temps de calcul. Il existe pour cela au moins deux méthodes utilisées classiquement (on pourra se reporter à [BD93], [SPF93], [CJ91], [Muh89] et [PLG87] pour plus de précisions sur les modèles de parallélisme utilisables dans les AG) :

- la première consiste à diviser la population de taille n en N sous-populations et à les répartir sur l'ensemble des machines dont on dispose ;
- la seconde maintient la population totale sur une seule machine mais se sert des autres pour y confier les évaluations, afin qu'elles se fassent en même temps.

Dans les deux cas, il est nécessaire d'avoir un mécanisme de communication inter-processus. Les résultats présentés ici ont été obtenus en utilisant PVM. PVM a été réalisé par le *Oak Ridge National Laboratory*. Il permet à un réseau hétérogène de machines d'apparaître comme une seule entité ayant plusieurs processeurs capables de communiquer entre eux (comparable à un ordinateur à mémoire distribuée). La communication entre les divers composants de cette machine virtuelle se fait par l'envoi de paquets contenant des données ou encore des messages de contrôle. Pour plus de détails, on peut se reporter à [GBD$^+$94].

1.4.1 Parallélisme par îlots

L'idée ici est de faire fonctionner plusieurs algorithmes génétiques en parallèle avec une population *réduite* pour chacun. Le programme maître lance N occurrences d'algorithmes génétiques appelés esclaves sur des machines différentes en transférant à chacune les paramètres nécessaires à leur bon fonctionnement comme le montre la figure 1.14.

Ensuite chaque processus fait évoluer sa population indépendamment jusqu'à ce qu'il décide (selon une probabilité fixée à l'avance) de rassembler ses meilleurs individus pour en transmettre une certaine quantité (proportionnelle à la taille de la population) à une autre machine de son choix. La machine réceptrice intègre alors ces nouveaux éléments dans sa propre population en éliminant les moins bons de ses individus. L'intérêt du parallélisme par îlots est qu'il offre la possibilité de travailler sur de grandes populations (n_0) tout en donnant des résultats dans un temps raisonnable, puisque la durée nécessaire est à peu de choses près celle qu'il faudrait pour une population de taille $\frac{n_0}{N}$, si N est le nombre d'ordinateurs disponibles et si l'on néglige les temps de communication.

La méthode introduit un *clustering* forcé car chaque îlot peut être considéré comme un *cluster* subdivisé en petits groupes. Chaque machine a la possibilité de converger vers des optima qui seront différents de ceux calculés sur les autres, ce qui correspond au comportement introduit avec le *clustering*. D'autre part, le surcoût de temps passé pour les communications n'est a priori pas excessif, puisqu'il n'y a de gros paquets de données à transmettre que de temps en temps. Il faut tout de même garder à l'esprit qu'une subdivision en sous-populations de taille trop réduite risque de conduire à faire tourner des algorithmes génétiques non fiables statistiquement. En effet, il faut quand même qu'une population contienne suffisamment d'individus pour que l'espace d'état puisse être exploré de façon correcte, afin que les résultats aient une certaine valeur.

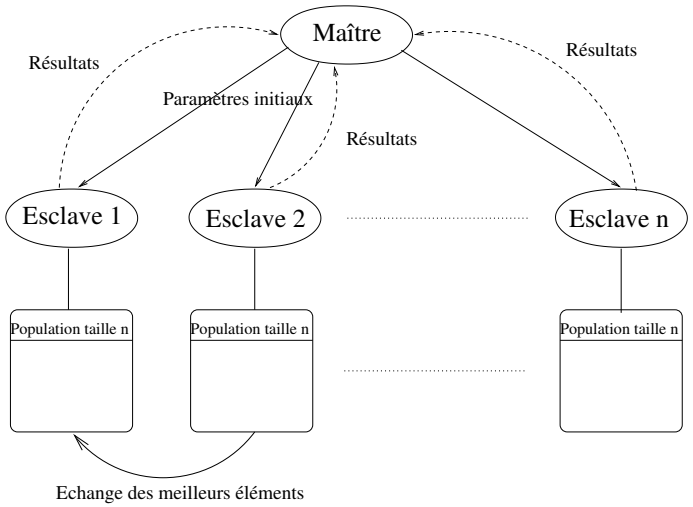

FIG. 1.14 – Principe de fonctionnement du parallélisme par îlots

1.4.2 Parallélisation des calculs

Contrairement à la méthode qui vient d'être décrite, qui divise la population totale, on utilise ici des "démons" de calcul de *fitness* dont la seule fonction est de recevoir un individu et de retourner son adaptation. Pour utiliser la puissance de calcul parallèle offerte de manière optimale, il faut retarder au maximum les calculs pour les envoyer en blocs aux démons et faire en sorte qu'aucun ne reste inactif. Le principe se trouve résumé sur la figure 1.15 : le programme maître se charge de faire la sélection, les croisements, etc... : en d'autres termes, il fait évoluer la population, puis répartit les calculs dont il a besoin sur l'ensemble des démons. Enfin, dès qu'il a reçu tous les résultats, l'algorithme commence une nouvelle génération.

Il faut noter que ce mécanisme demande un grand nombre de communications pour transmettre les données et les évaluations. La méthode n'est donc intéressante que si le temps passé pour un calcul d'adaptation est grand devant le temps de communication, elle sera par conséquent utilisée pour des problèmes dont les évaluations de *fitness* prennent du temps, on pense essentiellement à des cas faisant appel à des réseaux de neurones, ou à de gros calculs matriciels.

1.4.3 Conclusion

La parallélisation est extrêmement efficace pour accélérer les temps de résolution des algorithmes génétiques. Il faut certes bien étudier le problème afin d'utiliser le bon mécanisme de parallélisme, mais les gains en temps sont alors importants.

1.5 Résultats de convergence théorique des algorithmes génétiques

Les résultats théoriques sur les algorithmes génétiques sont nombreux. Sont évoqués en annexe A, la théorie des schémas développée par Goldberg [Gol89a], et dont l'intérêt scientifique a été très

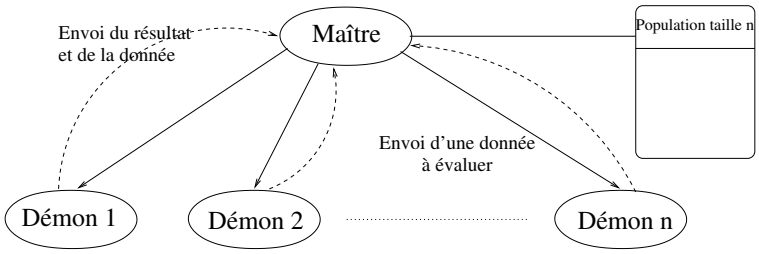

FIG. 1.15 – Principe de fonctionnement de la parallélisation des calculs

largement mis en cause depuis. On trouvera également un résumé des travaux de Cerf [Cer94] réalisés au milieu années 90 et qui ont permis, grâce à une modélisation par chaîne de Markov, d'apporter des éléments de convergence théorique importants, mais sans réelle utilité pratique. Je citerais en guise de conclusion les *No Free Lunch Theorems* [WM97] qui montrent qu'il n'est pas possible de définir une modélisation, des opérateurs génétiques adaptés à tous types de problèmes. Nous nous intéressons, dans la partie suivante, à une classe de problèmes partiellement séparables pour laquelle un opérateur de croisement adapté permet d'accélerer considérablement la convergence de l'algorithme génétique.

Chapitre 2

Elaboration d'un croisement adapté aux problèmes partiellement séparables

Introduction

Un algorithme génétique est d'autant plus efficace que les opérateurs de croisement et de mutation qu'il utilise favorisent la création d'enfants mieux adaptés que leurs parents. La gestion du trafic aérien fait apparaître de nombreux problèmes "partiellement séparables" pour lesquels on peut élaborer des opérateurs croisement plus performants que ceux décrits dans le chapitre précédent. Après une définition rapide de la séparabilité partielle et de l'opérateur de croisement adapté, une étude théorique de convergence est proposée sur un exemple simple, validée ensuite expérimentalement. L'opérateur est ensuite testé sur plusieurs fonctions tests.

On traite dans ce chapitre de problèmes de minimisation pour lesquels le critère à optimiser est la somme de termes positifs ne faisant chacun intervenir qu'une partie des variables.

Griewank et Toint [GT82] furent les premiers à définir les fonctions partiellement séparables il y a une trentaine d'années pour développer des méthodes d'optmisation locale non-linéaires.

Les opérateurs de croisement classiques décrits dans la littérature [Gol89a, Mic92b, Hol75] génèrent deux individus enfants à partir de deux individus parents, dans l'espoir que ces enfants "combineront les bonnes caractéristiques" de leurs parents, sans les détruire.

Les premiers opérateurs utilisés sur des chaînes de bits opéraient une coupure aléatoire en un point de la chaîne sur chacun des deux parents. Les schémas courts avaient de fait plus de chances de survie que les grands, ce qui donnait un rôle très important au codage des données. L'utilisation d'opérateurs de croisement multi-points, les codages de Gray pour les variables binaires, ou les croisements arithmétiques, ne permettent pas de reconnaître les "bons schémas" et de les favoriser.

De nombreux travaux de recherche ont été effectués pour favoriser les "bons" croisements et "bonnes mutations", à savoir ceux qui protègent les "bons schémas" [CW96]. On peut citer par exemple des méthodes d'apprentissage de règles [RSS95].

Lorsque la fonction à optimiser formule explicitement des liens privilégiés entre certaines variables, on peut en tenir compte pour influencer les opérations de croisement et de mutation afin qu'elles tiennent compte de cette structure particulière. C'est le cas pour les problèmes de séparation d'avions en vol ou sur une plate-forme aéroportuaire.

2.1 Séparabilité partielle

2.1.1 Définition

Les problèmes partiellement séparables traités dans la suite de ce chapitre ont les caractéristiques suivantes : la fonction F à minimiser dépend de n variables x_1, x_2, .., x_n et est la somme de m fonctions positives F_i, qui ne dépendent chacune que d'un sous-ensemble de variables.

Ces fonctions peuvent s'exprimer :

$$F(x_1, x_2, .., x_n) = \sum_{i=1}^{m} F_i(x_{j_1}, x_{j_2}, .., x_{j_{n_i}})$$

2.1.2 Opérateur de croisement adapté :

Aucun codage des données n'est requis pour mettre en oeuvre l'opérateur de croisement adapté. L'élément de population de l'algorithme génétique est représenté directement par ses variables, qui peuvent être réelles, entières ou de toute autre type.

L'idée intuitive est la suivante : pour un problème complètement séparable, le minimum global est obtenu lorsque l'on minimise la fonction sur chacune des variables séparément. La fonction à minimiser peut s'écrire :

$$F(x_1, x_2, .., x_n) = \sum_{i=1}^{n} F_i(x_i)$$

Il suffit de minimiser séparément chacune des fonctions F_i pour obtenir le minimum global de la fonction.

L'opérateur de croisement qui choisit, pour chaque variable x_i, celle des deux parents qui minimise la fonction F_i, permet de construire un individu toujours meilleur (au sens large) que ses deux parents.

On adapte cette stratégie aux fonctions partiellement séparables. Pour créer un individu enfant à partir de deux parents, l'idée est de choisir, pour chaque variable, celle qui minimise la somme des fonctions partielles F_i dans laquelle elle intervient.

On définit pour cela tout d'abord une fonction d'adaptation locale ou *fitness locale* $G_k(x_1, x_2, .., x_n)$, associée à chaque variable x_k comme suit :

$$G_k(x_1, x_2, .., x_n) = \sum_{i \in S_k} \frac{F_i(x_{j_1}, x_{j_2}, .., x_{j_{n_i}})}{n_i}$$

où S_k est l'ensemble des indices i tels que x_k est une variable de F_i et n_i est le nombre de variables de F_i.

La *fitness* locale associée à une variable isole la contribution de cette variable dans la *fitness* globale.

On observe la propriété suivante sur les *fitness* locales :

$$\sum_{k=1}^{n} G_k(x_1, x_2, .., x_n) = F(x_1, x_2, .., x_n)$$

Le principe de l'opérateur de croisement adapté est le suivant :

Si

$$G_k(parent_1) < G_k(parent_2) - \Delta$$

alors on retient pour l'enfant 1 la variable x_k du parent 1. Sinon, si

$$G_k(parent_1) > G_k(parent_2) + \Delta$$

alors on retient pour l'enfant 1 la variable x_k du parent 2. Enfin, si

$$|G_k(parent_1) - G_k(parent_2)| \leq \Delta$$

alors on choisit aléatoirement la variable x_k du parent 1 ou du parent 2 pour l'enfant 1. On peut également choisir une combinaison linéaire des variables x_k des deux parents lorsque l'on utilise des variables réelles. Si l'on utilise la même stratégie pour l'enfant 2 que pour l'enfant 1, on peut produire deux enfants semblables, surtout si Δ est petit. On peut éviter ce problème en utilisant une nouvelle paire de parents pour fabriquer chaque enfant.

Soit la fonction complètement séparable :

$$F(x_1, x_2, x_3) = x_1 + x_2 + x_3$$

avec x_1, x_2 et x_3 entiers de l'intervalle $[0, 2]$. La *fitness* locale de la variable k s'exprime : $G_k(x_1, x_2, x_3) = x_k$. Si l'on applique l'opérateur de croisement sur les individus $(1\ ,\ 0\ ,\ 2)$ et $(2\ ,\ 1\ ,\ 0)$ qui ont la même *fitness* $F = 3$. Avec $\Delta = 0$, l'enfant 1 s'écrit $(1\ ,\ 0\ ,\ 0)$: $F = 1$. Avec $\Delta = 1$, l'enfant 2 peut être $(2\ ,\ 1\ ,\ 0)$, $(2\ ,\ 0\ ,\ 0)$, $(1\ ,\ 1\ ,\ 0)$, ou $(1\ ,\ 0\ ,\ 0)$.

Les *fitness* des enfants sont toujours meilleures que celles des parents quand $\Delta = 0$ ce qui n'est pas le cas avec un opérateur de croisement classique.

Pour mesurer l'intérêt de l'opérateur de croisement adapté, on introduit dans le paragraphe suivant un exemple simple de fonction partiellement séparable.

2.2 Etude théorique sur un exemple simple

Soit la fonction suivante :

$$F(x_1, x_2, .., x_n) = \sum_{0<i\neq j\leq n} \delta(x_i, x_j) \tag{2.1}$$

$(x_1, x_2, .., x_n)$ est une chaîne de bits. $\delta(x_i, x_j) = 1$ si $x_i \neq x_j$ et 0 si $x_i = x_j$.

On remarque que cette fonction est partiellement séparable et a deux minima globaux, à savoir $(1, 1, 1, .., 1)$ et $(0, 0, 0, .., 0)$.

Pour $x = (x_1, x_2, ..x_n)$, on définit la *fitness* locale $G_k(x)$ comme suit :

$$G_k(x) = \frac{1}{2} \sum_{i=1}^{n} \delta(x_k, x_i)$$

Soit $I(x)$ le nombre de bits égaux à 1 dans x. Il est facile d'établir que :

$$\begin{aligned} F(x) &= I(x)(n - I(x)) \\ G_k(x) &= \frac{I(x)}{2} \quad si\ x_k = 0 \\ &= \frac{n - I(x)}{2}\ si\ x_k = 1 \end{aligned}$$

Dans la suite de ce paragraphe, l'opérateur de croisement classique utilisé est un croisement à n points qui consiste à choisir pour chaque bit de l'enfant C, au hasard, celui provenant du parent A_1 ou du parent A_2.

Dans le paragraphe 2.2.1, on compare pour les croisements adaptés et classiques les probabilités que l'enfant ait une meilleure *fitness* que les parents. On compare dans le paragraphe 2.2.2 la vitesse de convergence de l'algorithme génétique lorsque l'on utilise uniquement les opérateurs de croisement sans sélection ni mutation. Dans le paragraphe 2.2.3, un opérateur de sélection est modélisé de façon à comparer les deux opérateurs de croisement.

2.2.1 Probabilité d'amélioration

Pour la fonction (2.1), on peut calculer pour tous les couples de parents possibles la probabilité que l'enfant soit meilleur que ses parents pour le croisement classique ou adapté.

Soit $P_{1-1}(i,j,k)$ la probabilité de trouver k bits valant 1 à la même position chez les deux parents A_1 et A_2, lorsque $I(A_1)=i$ et $I(A_2)=j$. On a $P_{1-1}(i,j,k)=P_{1-1}(j,i,k)$, et l'on supposera donc par la suite que $i \leq j$.

On peut montrer par un simple calcul que :

– si $k > i$, alors :

$$P_{1-1}(i,j,k)=0$$

– si $k \leq i$, alors :

$$P_{1-1}(i,j,k)=C_i^k \prod_{l=0}^{k-1} \frac{j-l}{n-l} \prod_{l=k}^{i-1} \frac{(n-l)-(j-k)}{n-l}$$

L'opérateur de croisement classique utilisé est le croisement à n points qui consiste à choisir les bits de A_1 ou A_2 avec une probabilité $\frac{1}{2}$ de sorte que l'ordre des bits dans la chaîne n'a pas d'importance.

Pour l'opérateur de croisement adapté (resp classique), soit $P_a(i,j,k)$ (resp $P_c(i,j,k)$) la probabilité que si $I(A_1)=i$ et $I(A_2)=j$ alors $I(C)=k$. Puisque $P_a(i,j,k)=P_a(j,i,k)$ et $P_c(i,j,k)=P_c(j,i,k)$, on supposera par la suite que $i \leq j$. On peut alors calculer que pour l'opérateur de croisement classique :

$$P_c(i,j,k)=\sum_{l=\max(0,\frac{i+j+1-n}{2})}^{\min(k,i+j-k)} P_{1-1}(i,j,l)\,\frac{C_{i+j-2\,l}^{k-l}}{2^{i+j-2\,l}}$$

Pour l'opérateur de croisement adapté (si $m=\min(k,n-k)$) :

$$\begin{aligned}
i+j<n: \quad P_a(i,j,k) &= P_{1-1}(i,j,k)\\
i+j>n: \quad P_a(i,j,k) &= P_{1-1}(n-i,n-j,n-k)\\
i+j=n: \quad P_a(i,j,k) &= \sum_{l=0}^{m} P_{1-1}(i,j,l)\,\frac{C_{i+j-2\,l}^{k-l}}{2^{i+j-2\,l}}
\end{aligned}$$

Puisque $P_{1-1}(i,j,k)=0$ si $k>\min(i,j)$, alors :

– si $i+j<n$ et $k>\min(i,j)$, alors $P_a(i,j,k)=0$

– si $i+j>n$ et $k<\max(i,j)$, alors $P_a(i,j,k)=0$

En conséquence :

– si $i+j<n$ et $P_a(i,j,k)>0$,
alors $k<\min(i,j,n-i,n-j)$

– si $i+j>n$ et $P_a(i,j,k)>0$,
alors $k>\max(i,j,n-i,n-j)$

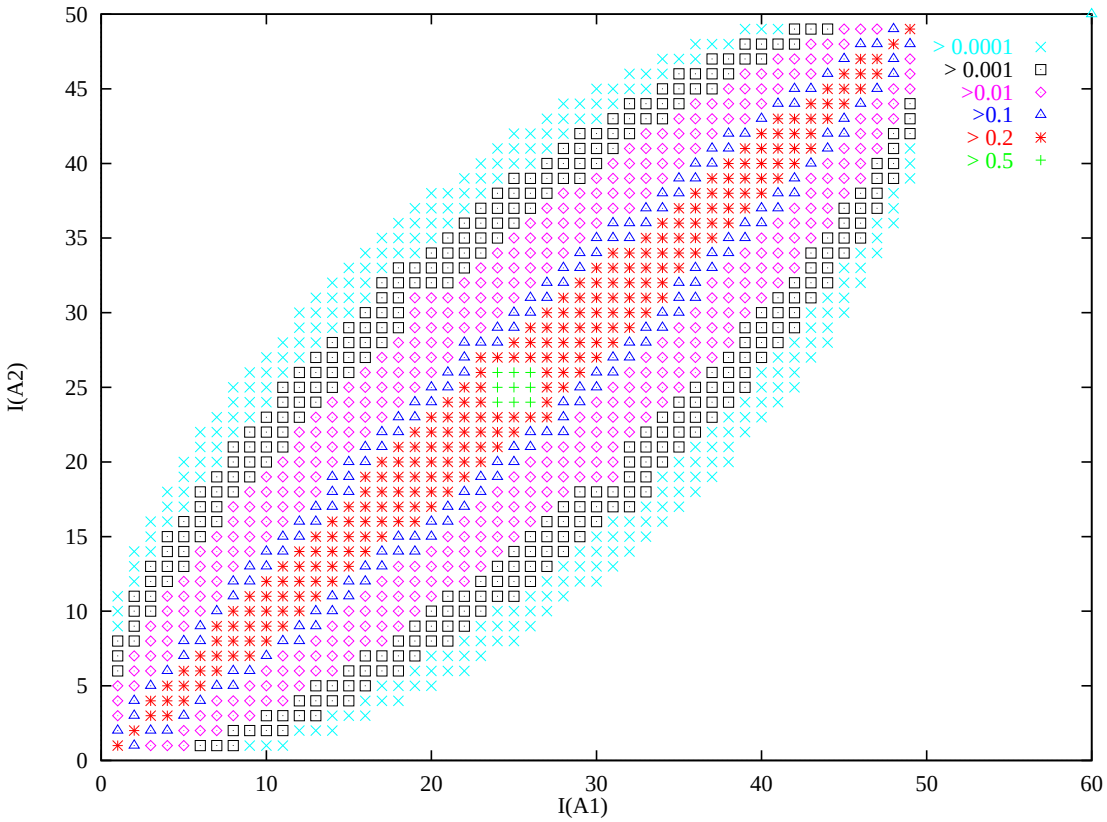

FIG. 2.1 – Probabilité que la *fitness* de l'enfant soit meilleure que celle de ses deux parents ($Prob(F(C) > \max[F(A_1), F(A_2)]$) en fonction de $[I(A_1), I(A_2)]$ - croisement classique - $n = 50$.

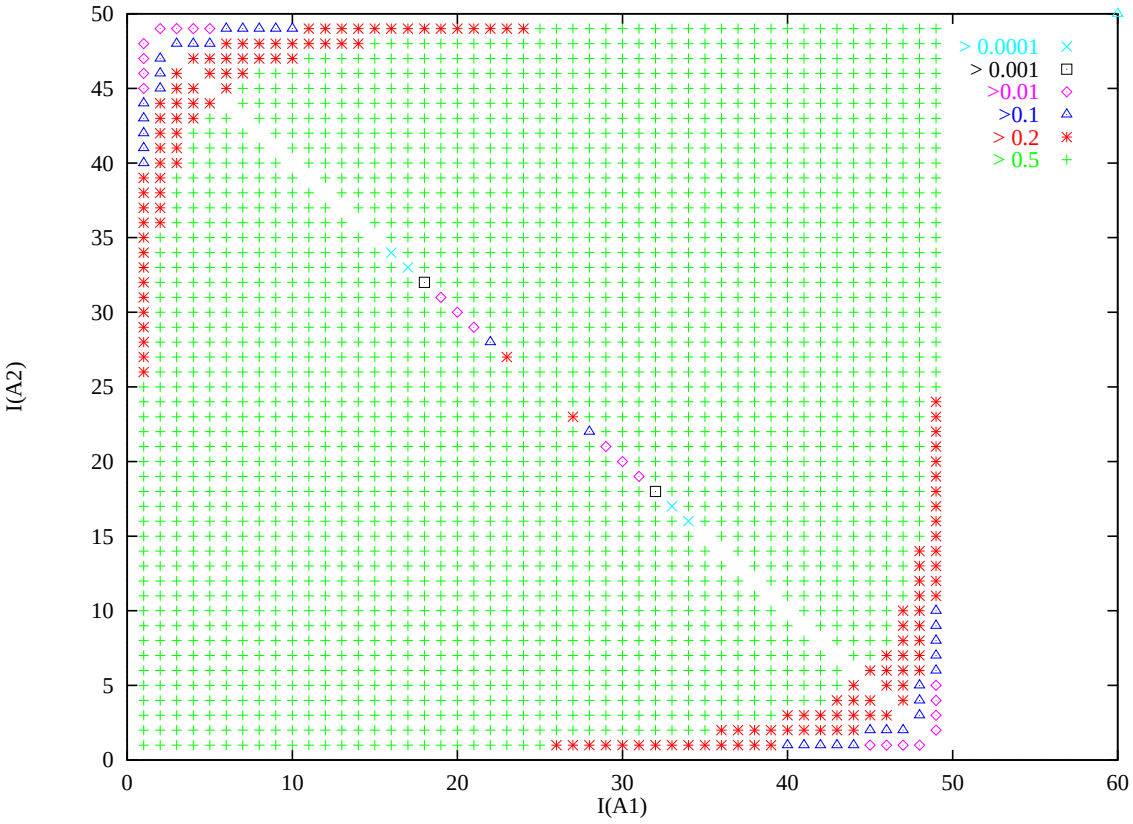

FIG. 2.2 – Probabilité que la *fitness* de l'enfant soit meilleure que celle de ses deux parents ($Prob(F(C) > \max[F(A_1), F(A_2)]$) en fonction de $[I(A_1), I(A_2)]$ - croisement adapté - $n = 50$.

Ainsi, si $i + j \neq n$, alors $F(C) \geq \max[F(A_1), F(A_2)]$. Si $i + j = n$, les *fitness* locales associées à chaque variable des deux parents sont identiques et les opérateurs de croisement adapté et classique sont identiques dans ce cas.

Les figures 2.1 et 2.2 donnent la probabilité pour un enfant d'avoir une meilleure *fitness* que ses parents, et ce, pour toutes les combinaisions de parents possibles.

Pour cet exemple, l'opérateur de croisement adapté est bien plus efficace que l'opérateur de croisement classique. Le petit carré au centre de la figure 2.1 représente une probabilité d'augmentation de (*fitness* supérieure à 0.5. Le carré devient très large dans la figure 2.2.

2.2.2 Vitesse de convergence avec l'opérateur de croisement adapté

P_a et P_c étant connus, on peut évaluer la probabilité de trouver une solution optimale après m générations, avec un algorithme génétique utilisant ni sélection ni mutation, mais seulement l'opérateur de croisement adapté.

Soit $Q_a(m, k)$ la probabilité que $I(C) = k$ où C est un élément de population choisi au hasard à la génération m. A la génération 0 :

$$Q_a(0, k) = \frac{C_n^k}{2^n}$$

A la génération $m + 1$:

$$Q_a(m + 1, k) = \sum_{i=0}^{n} \sum_{j=0}^{n} Q_a(m, i)\, Q_a(m, j)\, P_a(i, j, k)$$

La figure 2.3 donne l'évolution de $Q_a(m, 0) = Q_a(m, n)$ pour différentes valeurs de n. On observe que la probabilité de trouver les solutions $(0, 0, 0.., 0)$ ou $(1, 1, 1.., 1)$ se stabilise autour de $\frac{1}{3}$ après 7 générations. Lorsque les *fitness* locales sont égales, on utilise l'opérateur de croisement classique. Si l'on avait choisi, en cas d'égalité des *fitness* locales, de privilégier un des deux parents, les probabilités d'atteindre les solutions $(0, 0, 0.., 0)$ ou $(1, 1, 1.., 1)$ se seraient stabilisées à $\frac{1}{2}$. On notera que la taille du problème n a peu d'influence sur la vitesse de convergence.

En utilisant un opérateur de croisement classique dans les mêmes conditions, on ne peut espérer faire converger l'algorithme génétique. On a :

$$\forall m \in N,\ Q_c(m, k) = \frac{C_n^k}{2^n}$$

2.2.3 Probabilité de sélection

La fonction *fitness* prend ses valeurs dans l'intervalle $[0, n^2/4]$. On peut donc décider de sélectionner les individus proportionnellement à $n^2/4 - F$. Considérons un algorithme génétique utilisant l'opérateur de sélection qui reproduit chaque individu avec une probabilité proportionnelle à $n^2/4 - F$, et qui applique l'opérateur de croisement adapté ou classique.

Soient $S_a(m, k)$ et $S_c(m, k)$ les probabilités que $I(C) = k$ où C est un élément quelconque de la population à la génération m, après application de l'opérateur de sélection.

Un simple calcul permet de montrer que :

$$\begin{aligned} S_a(m + 1, k) &= \frac{s_a(m + 1, k)}{\sum_{k=0}^{n} s_a(m + 1, k)} \\ S_c(m + 1, k) &= \frac{s_c(m + 1, k)}{\sum_{k=0}^{n} s_c(m + 1, k)} \end{aligned}$$

avec

$$H(k) = \frac{n^2}{4} - k\,(n-k)$$

$$s_a(m+1,k) = \sum_{i=0}^{n}\sum_{j=0}^{n} H(k)\,S_a(m,i)\,S_a(m,j)\,P_a(i,j,k)$$

$$s_c(m+1,k) = \sum_{i=0}^{n}\sum_{j=0}^{n} H(k)\,S_c(m,i)\,S_c(m,j)\,P_c(i,j,k)$$

Sur la figure 2.4, les quatre courbes du haut représentent $S_a(m,0)$ pour des individus de taille 20, 50, 100 et 200 bits. Les quatre courbes du bas représentent $S_c(m,0)$ pour des individus de taille 20, 50, 100 et 200.

L'efficacité de l'opérateur de croisement adapté apparaît clairement, particulièrement pour les problèmes de grande taille. La vitesse de convergence ne dépend que très peu de la taille du problème dans le cas du croisement adapté, alors qu'elle est très dépendante de la taille du problème, avec l'opérateur de croisement classique.

2.3 Résultats expérimentaux

Dans l'étude précédente, la taille de la population n'était pas limitée. Dans la pratique, la taille de la population a une réelle influence sur la vitesse de convergence de l'algorithme génétique. De plus, dans l'exemple précédent, la fonction a deux optima globaux, et l'on n'a pas mesuré en pratique la capacité de l'algorithme génétique à atteindre les deux optima. C'est ce que nous tenterons de faire dans cette section.

Considérons la même fonction que précédemment :

$$F(x_1,..,x_n) = \sum_{1\le i\neq j\le n} \delta(x_i,x_j) \tag{2.2}$$

mais avec x_i entier dans l'intervalle $[0,4]$.

Cette fonction a 5 optima définis par $\forall(i,j)\; x_i = x_j$.

Dans le paragraphe 2.3.1, les opérateurs de croisement classiques et adaptés sont comparés avec un algorithme génétique semblable au modèle théorique.

Le nombre d'optima trouvés et la vitesse de convergence sont comparés. Dans le paragraphe 2.3.2, un opérateur de *sharing* est ajouté de façon à trouver le plus d'optima possibles. Les efficacités des croisements adaptés et classiques sont également mesurées.

2.3.1 Algorithme génétique sans *sharing*

La fonction (2.2) est optimisée avec un algorithme génétique exécuté 200 fois. (nombre de générations : 1000, taille de la population : 500, proportion de croisements : 50%, type de sélection : *stochastic reminder without replacement*, élitisme, pas de *sharing*, pas de mutation).

Les 100 premières exécutions sont réalisées avec le croisement classique à n points[1]. Les 100 exécutions suivantes sont réalisées avec le croisement adapté.

La figure 2.5 donne le nombre d'exécutions pour lesquelles on a trouvé 1, 2, 3 ou 4 optima avec le croisement adapté. Il est important de noter que les 5 optima ne sont jamais trouvés. La figure montre

[1] L'ordre des variables n'a donc pas d'importance

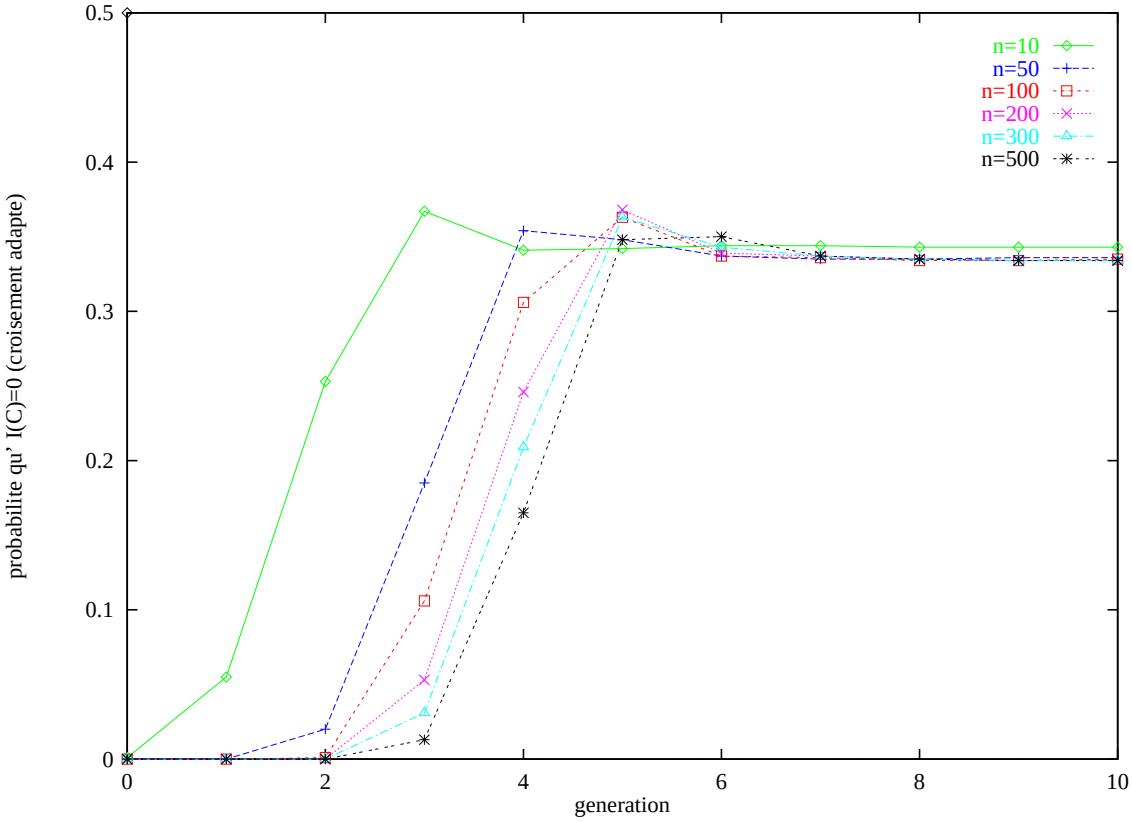

FIG. 2.3 – $Q_a(m, 0)$ en fonction de la génération m pour différentes valeurs de n.

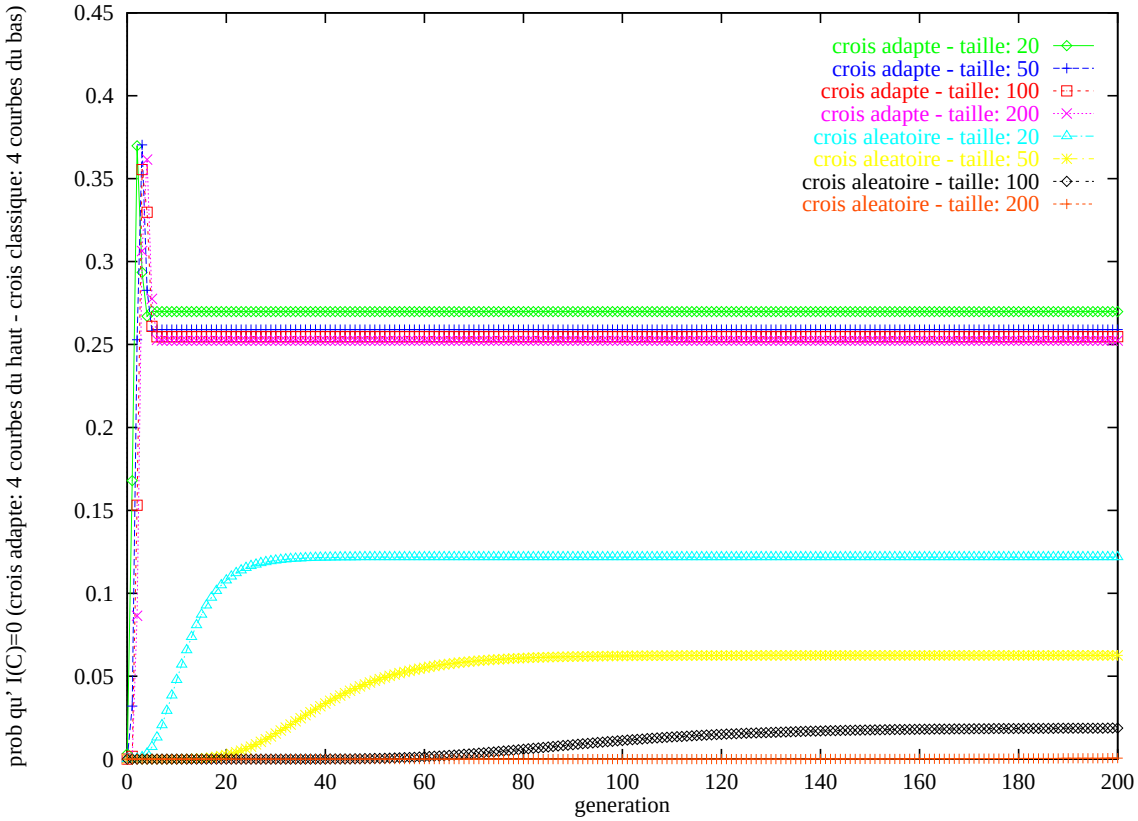

FIG. 2.4 – $S_a(m, 0)$ et $S_c(m, 0)$ en fonction de la génération m pour différentes valeurs de n.

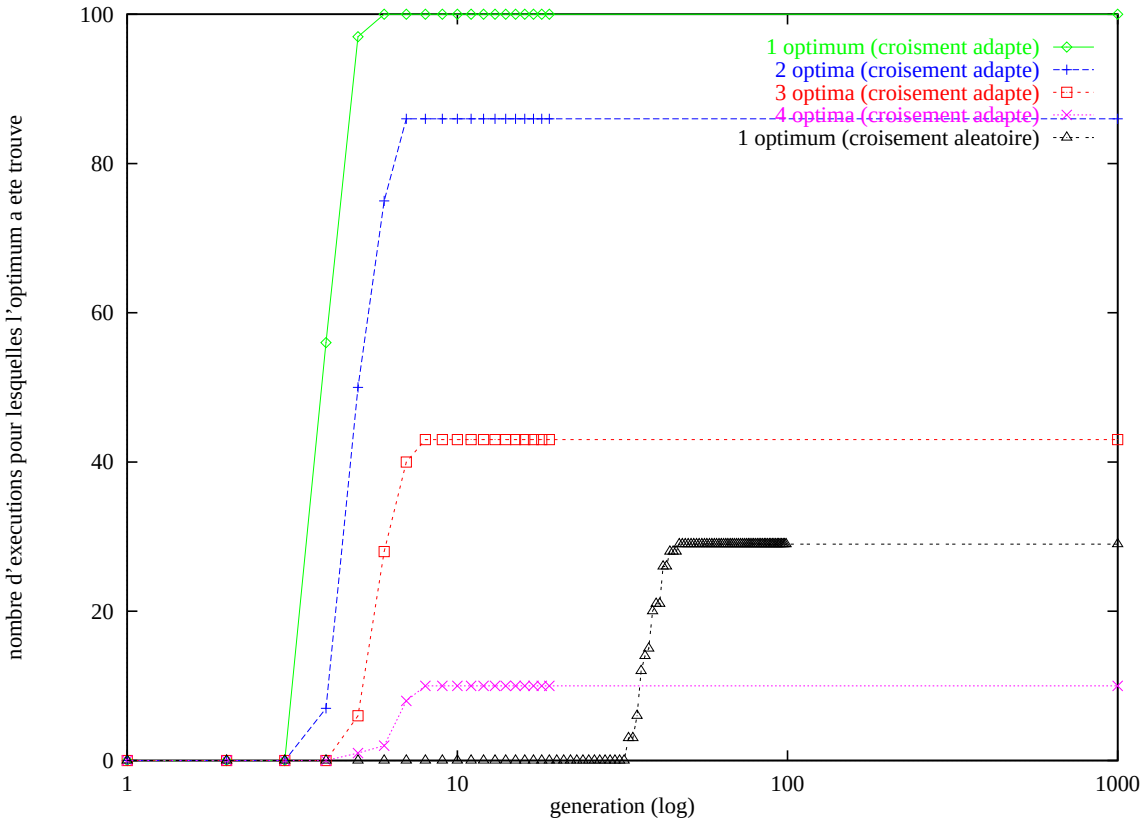

FIG. 2.5 – AG sans *sharing*. Nombre d'exécutions du programme pour lesquelles 1, 2, 3 ou 4 optima ont été obtenus avec le croisement adapté, 1 optimum a été obtenu avec le croisement classique.

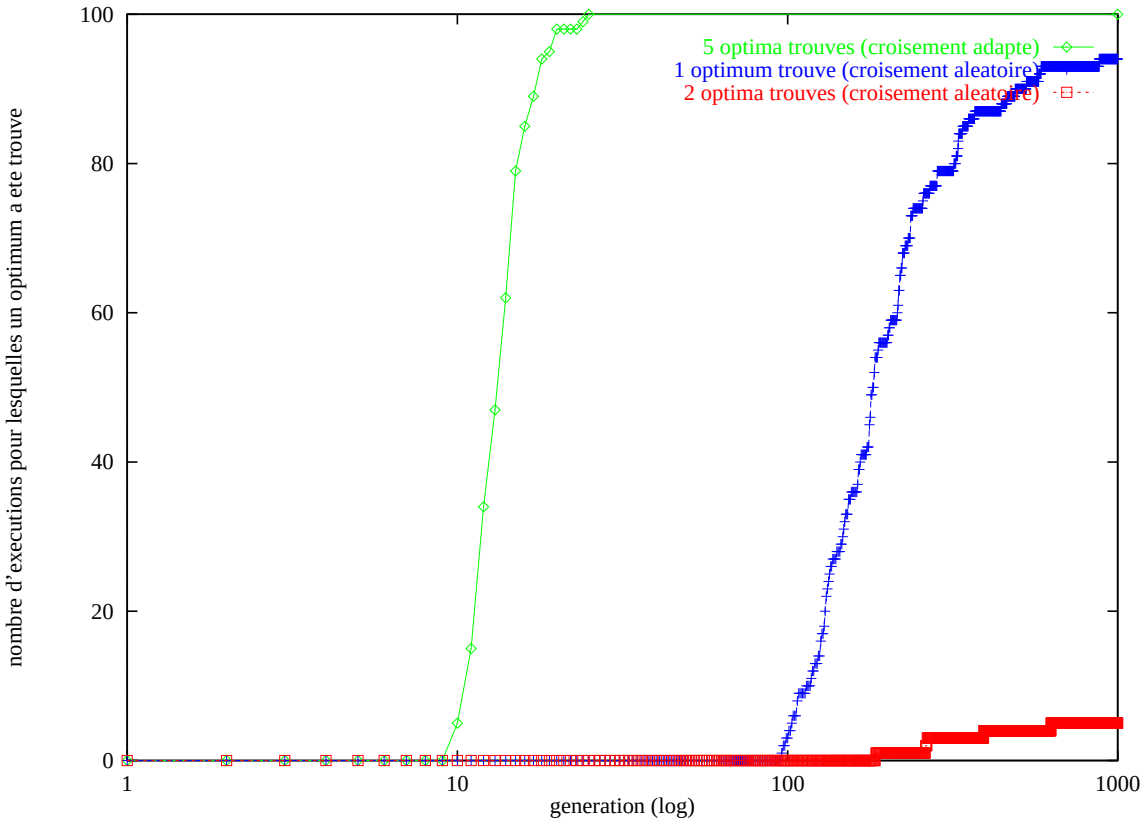

FIG. 2.6 – AG avec *sharing*.Nombre d'exécutions du programme pour lesquelles on trouve 5 optima avec le croisement adapté, 1 ou 2 optima avec le croisement classique, en fonction de la génération.

également le nombre d'exécutions pour lesquelles on trouve 1 optimum avec le croisement classique. L'algorithme génétique ne trouve jamais deux optima avec le croisement classique. Le croisement adapté permet de trouver rapidement plusieurs optima avant que la population ne s'uniformise.

2.3.2 Algorithme génétique avec *sharing*

On applique le même algorithme génétique que précédemment, sur la même fonction, mais avec un opérateur de *sharing*. Si $h(C_1, C_2)$ est la distance de Hamming entre les individus C_1 et C_2 et n la taille des individus, alors on définit :

$$Dist(C_1, C_2) = \begin{array}{l} 1\,,\ \text{si } h(C_1, C_2) < \dfrac{n}{2} \\ 0\,,\ \text{sinon} \end{array}$$

et $N(C_i)$ le nombre d'individus C_j de la population tels que $Dist(C_i, C_j) = 0$. L'opérateur de *sharing* modifie la *fitness* des individus avant la sélection en divisant la *fitness* de chaque individu C_i par $N(C_i)$.

La figure 2.6 donne le nombre d'exécutions pour lesquelles on trouve 5 optima en utilisant l'opérateur de croisement adapté avec le *sharing*.

La figure représente également le nombre d'exécutions pour lesquelles 1 ou 2 optimas sont trouvés en utilisant l'opérateur de croisement classique avec le *sharing* (on n'en trouve jamais plus de deux).

2.3.3 Influence du *sharing*

Un "bon" croisement peut être défini comme un croisement pour lequel la *fitness* de l'enfant est meilleure que celle des deux parents. Pour mesurer l'influence du *sharing* sur la qualité du croisement, on a représenté figure 2.7 (resp 2.8) le pourcentage de "bons" croisements avec l'opérateur de croisement classique (resp adapté) avec ou sans *sharing*.

La figure 2.7 montre que le *sharing* retarde la convergence vers l'optimum lorsque l'on utilise le croisement classique.

En comparant les figures 2.5 et 2.6, on observe que la vitesse de convergence est pénalisée par le *sharing*. Toutefois, il est difficile de dire si le *sharing* a une influence sur l'efficacité de l'opérateur de croisement.

La figure 2.8 montre que le croisement adapté est d'autant plus efficace que la population est diversifiée. La figure 2.6 montre que les 5 optima sont toujours trouvés avant la vingtième génération. Le *sharing* semble avoir moins d'influence sur la vitesse de convergence (le ralentissement de la convergence dû à la diversification de la population semble être compensé par une meilleure efficacité de l'opérateur de croisement adapté).

2.4 Exemple de fonctions de test

2.4.1 Une fonction totalement séparée : la fonction de Corana

La fonction de Corana [CMMR87] est classiquement utilisée pour tester des algorithmes d'optimisation globale. Elle a la particularité d'être totalement séparée.

On utilise ici la restriction introduite par Ingber [IR92a]. La fonction est définie sur $[-10000, 10000]^N$:

$$f_0(x) =$$

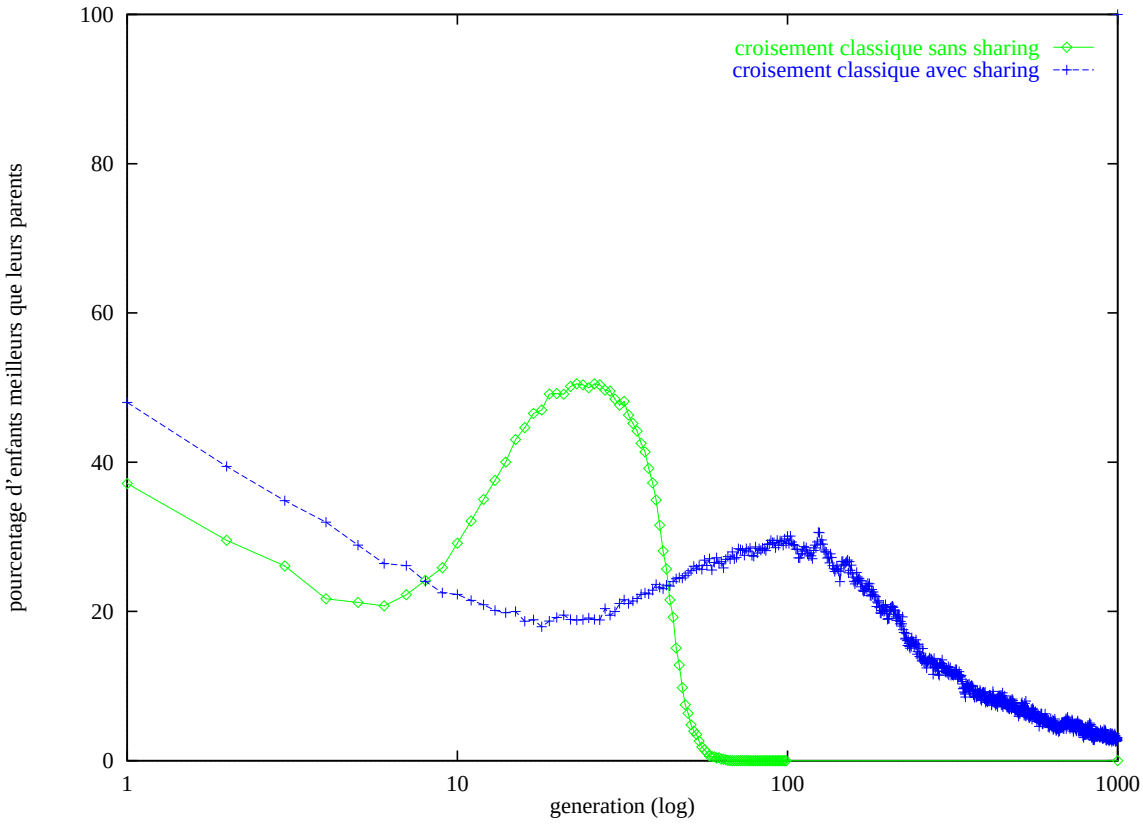

FIG. 2.7 – Croisement classique : pourcentage d'enfants meilleurs que leurs deux parents (sans *sharing* et avec *sharing*).

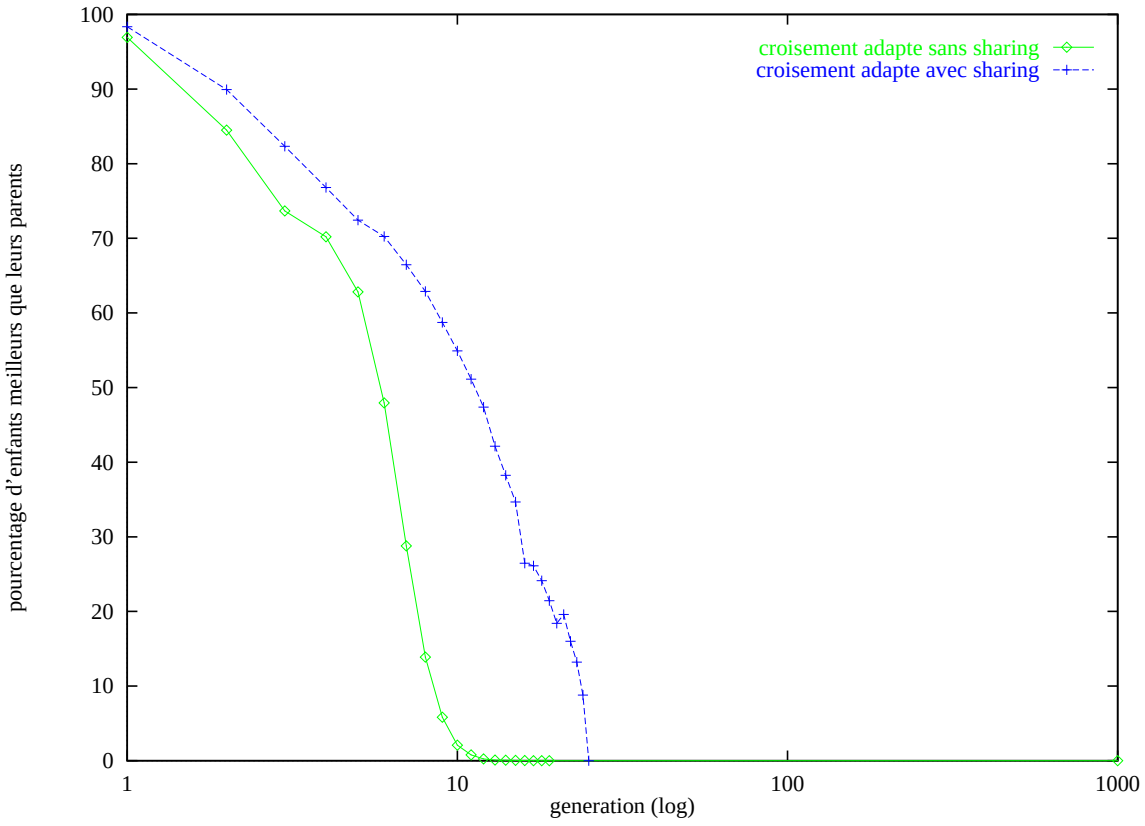

FIG. 2.8 – Croisement adapté : pourcentage d'enfants meilleurs que leurs deux parents (sans *sharing* et avec *sharing*).

	min	max	moy	σ
classique	76	294	179.44	178.47
adapté	42	204	92.33	91.81
classique / *sharing*	79	357	248.87	247.32
adapté / *sharing*	42	186	96.57	95.95

TAB. 2.1 – Convergence pour la fonction de Griewank

$$\sum_{i=1}^{N} \left\{ \begin{array}{ll} 0.15\, d_i (0.05\, S(z_i) + z_i)^2 & \text{pour } |x_i - z_i| < 0.05 \\ d_i\, x_i^2 & \text{sinon} \end{array} \right\}$$

$$\begin{aligned} z_i &= 0.2\, \lfloor |x_i/0.2| + 0.49999 \rfloor\, S(x_i) \\ S(z_i) &= \left\{ \begin{array}{ll} 1 & \text{si } z_i > 0 \\ 0 & \text{si } z_i = 0 \\ -1 & \text{si } z_i < 0 \end{array} \right. \\ d_{i \bmod 4} &= \{1.0, 1000.0, 10.0, 100.0\} \end{aligned}$$

Cette fonction a 10^{5N} optima locaux et tous les points de $[-0.05, 0.05]^N$ sont des optima globaux. Cette fonction a été utilisée par Ingber pour tester VFSR[2], jusqu'à maintenant la meilleure méthode de recuit simulé pour résoudre ce problème de minimisation globale.

La fonction étant totalement séparée, on définit très facilement une *fitness* locale pour chaque variable :

$$G_i(x_i) = \left\{ \begin{array}{ll} 0.15\, d_i (0.05\, S(z_i) + z_i)^2 & , |x_i - z_i| < 0.05 \\ d_i\, x_i^2 & \text{sinon} \end{array} \right.$$

Comme on pouvait s'y attendre, le croisement adapté est très efficace. VFSR et l'algorithme génétique classique ne parviennent pas à trouver d'optimum pour $N > 24$. Avec le croisement adapté, l'optimum est toujours trouvé, même pour $N = 1000$.

2.4.2 La fonction de Griewank

Cette fonction est plus intéressante que la fonction de Corana, car elle n'est pas complètement séparable. Elle est définie par :

$$F(x_1, .., x_n) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$$

On se limite dans cet exemple à $n = 10$. Cette fonction n'a qu'un optimum global : $x = (0, \ldots, 0)$. On définit la *fitness* locale par :

$$G_i(x_1, .., x_{10}) = \frac{1}{4000}\, x_i^2 - \prod_{i=1}^{10} \cos(\frac{x_i}{\sqrt{i}})$$

4 séries de 100 tests ont été exécutées : (croisement classique sans *sharing*), (croisement classique, avec *sharing*), (croisement adapté, sans *sharing*), (croisement adapté, avec *sharing*). On estime

[2] *Very Fast Simulated Reannealing*

que l'algorithme génétique a trouvé l'optimum lorsqu'une méthode d'optimisation locale appliquée au meilleur élément permet d'atteindre l'optimum. Le tableau 2.1 donne le nombre minimal, maximal, moyen de générations nécessaires pour trouver l'optimum, ainsi que l'écart-type trouvé dans chaque série de tests. On observe que l'opérateur de croisement adapté est largement plus efficace que l'opérateur de croisement classique.

2.5 Conclusion

L'opérateur de croisement adapté introduit dans ce chapitre peut être utilisé pour de nombreux problèmes d'optimisation dès lors que la fonction à optimiser présente un caractère de séparabilité partielle. Il sera d'autant plus efficace que le nombre de variables en jeu est important. Les problèmes de gestion du trafic aérien font intervenir plusieurs agents (avions, secteurs, routes) plus ou moins liés entre eux par des contraintes (conflits potentiels, charges de coordination, croisements). Ils sont généralement partiellement séparables et l'opérateur de croisement adapté s'avère souvent indispensable lorsqu'un algorithme génétique est utilisé. Dans le prochain (et dernier) chapitre de cette partie, on donne un exemple d'utilisation d'algorithmes génétiques et plus particulièrement de l'opérateur de séparabilité partielle pour optimiser la circulation d'aéronefs sur une plate-forme aéroportuaire.

Chapitre 3

Exemple d'application : optimisation de la circulation des aéronefs sur un aéroport

3.1 Introduction

Le développement récent de nombreux hubs commerciaux est à l'origine d'une nouvelle forme de congestion sur les grandes plates-formes aéroportuaires : la plupart des mouvements ont tendance à être programmés aux mêmes heures «stratégiques».

Les nombreux retards occasionnés sont la source d'une incertitude croissante sur les heures de décollage et d'atterrissage. Les retards peuvent atteindre plusieurs dizaines de minutes aux heures de pointe, ce qui est extrêmement pénalisant pour l'ensemble des acteurs du transport aérien. Cette application présente une modélisation du trafic au roulage sur un aéroport. Elle est intégrée dans un simulateur arithmétique de trafic. Les études préliminaires ont été réalisées dans le cadre du stage de DRU de Brankica Pesic [Pes00]. L'application fait l'objet de la thèse de Jean-Baptiste Gotteland [Got04].

Différentes méthodes d'optimisation selon le délai qu'elles génèrent et le nombre de mouvements impliqués. Elles utilisent des algorithmes génétiques et des algorithmes de parcours de graphe pour trouver le meilleur chemin et/ou les meilleurs points d'attente pour chaque avion, en respectant les normes de séparation au sol, pistes comprises.

3.1.1 Modélisation

Le problème est de trouver un ensemble optimal de trajectoires admissibles pour le trafic au roulage.

Une trajectoire est définie par une heure de départ (ou d'arrivée), **un chemin** et des **points d'attente** sur ce chemin.

Le caractère optimal de l'ensemble des trajectoires peut avoir différentes définitions, et sera considéré globalement comme le minimum d'une fonction de coût précisée ci-dessous.

Les trajectoires sont admissibles si d'une part les chemins empruntés sont conformes avec les contraintes opérationnelles de l'aéroport (décrites au paragraphe 3.1.3), et si d'autre part les avions sont séparés (comme détaillé dans les parties suivantes).

3.1.2 Fonction de coût

La fonction de coût évaluant un ensemble de trajectoires admissibles peut faire intervenir différents facteurs, comme par exemple la durée et/ou la longueur de chaque trajectoire. Une attente peut être jugée plus ou moins coûteuse qu'une déviation. On peut aussi privilégier les départs soumis à un créneau de décollage en rendant leurs retards plus coûteux...

Dans la version actuelle, la fonction de coût f_c retenue prend en compte, pour chaque avion i ($1 \leq i \leq N$), son temps de roulage r_i et le délai d_i dû à sa déviation par rapport au plus court chemin :

$$f_c = \sum_{i=1}^{N} f_{c_i} \quad \text{avec} \quad f_{c_i} = r_i + d_i$$

Les déviations sont ainsi deux fois plus coûteuses que les attentes.

3.1.3 L'aéroport

Afin d'attribuer à chaque mouvement un ensemble de chemins admissibles, l'aéroport est modélisé par un graphe reliant ses parkings, ses taxiways et ses pistes.

Le coût pour aller d'un taxiway à un autre élément est le temps nécessaire pour parcourir ce taxiway. Ce temps est calculé en fonction du rayon de virage et des procédures spécifiques liées aux pistes (point d'arrêt obligatoire, sorties lentes, normales ou rapides...) et aux parkings (vitesses restreintes, attentes après *push-backs*...)

Certains taxiways sont des «sens-interdits » utilisables à titre plus ou moins exceptionnel. Leur coût est alors pondéré par un coefficient représentant la tolérance de passage par ce taxiway, conformément à la pratique opérationnelle.

Les différents chemins admissibles pour un mouvement peuvent alors être obtenus avec des algorithmes classiques de parcours de graphe.

L'algorithme du Dijkstra [AMO93] donne tous les meilleurs chemins et les coûts minimums correspondants (i.e. les temps de parcours minimaux) pour aller d'un nœud donné à tous les autres. A partir du résultat du Dijkstra, il est possible de trouver les k_0 meilleurs chemins pour aller d'un point à un autre par Enumération Récursive [JM99]. En réitérant m fois ces deux algorithmes (Dijkstra et Enumération Récursive), et en augmentant à chaque fois le coût des nœuds empruntés par les chemins trouvés, on peut obtenir jusqu'à mk_0 chemins «suffisamment» différents les uns des autres.

Chaque mouvement se voit ainsi attribuer k chemins possibles ($k \leq mk_0$) allant du parking à la piste demandée pour un départ, et de la piste d'atterrissage au parking pour une arrivée.

Les figures 3.1 et 3.2 représentent les graphes de Roissy et d'Orly et donnent un exemple du tracé des k meilleurs chemins allant d'un point à un autre.

3.1.4 Le trafic

Les plans de vol

Les mouvements sont décrits par leur plan de vol. Celui-ci fournit notamment l'heure de départ ou d'arrivée, le parking et la piste demandés, et le type d'avion. Le type d'avion permet d'appréhender la distance de décollage ou d'atterrissage, donc le(s) point(s) d'entrée ou de sortie de la piste envisageables. Il donne également la catégorie de turbulence de sillage (faible, moyenne ou élevée) utilisée pour les règles de séquencement de piste.

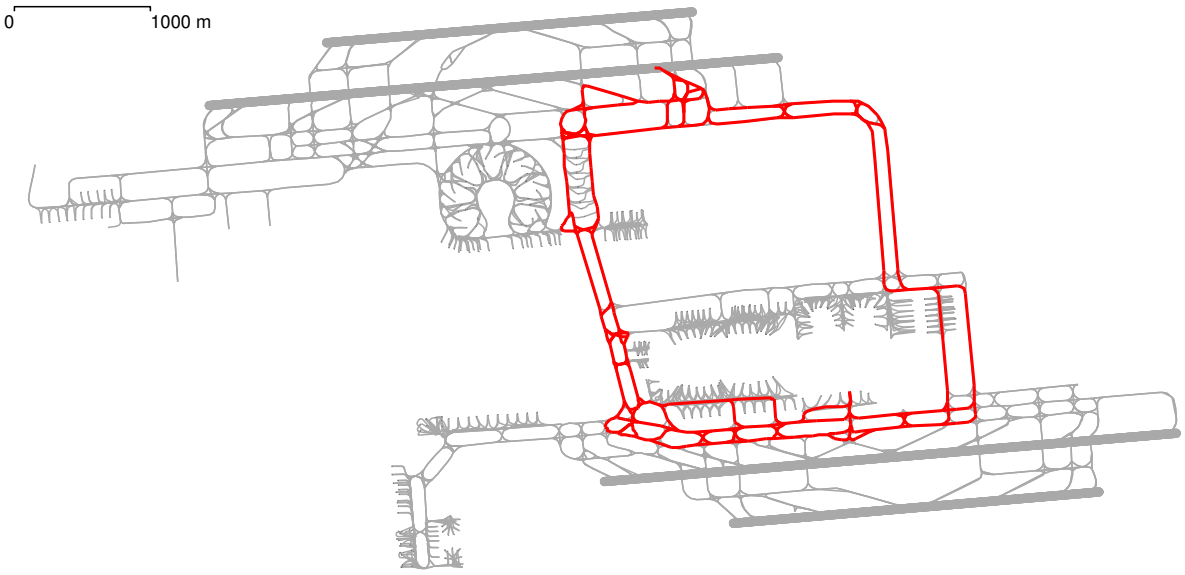

FIG. 3.1 – Graphe de Roissy

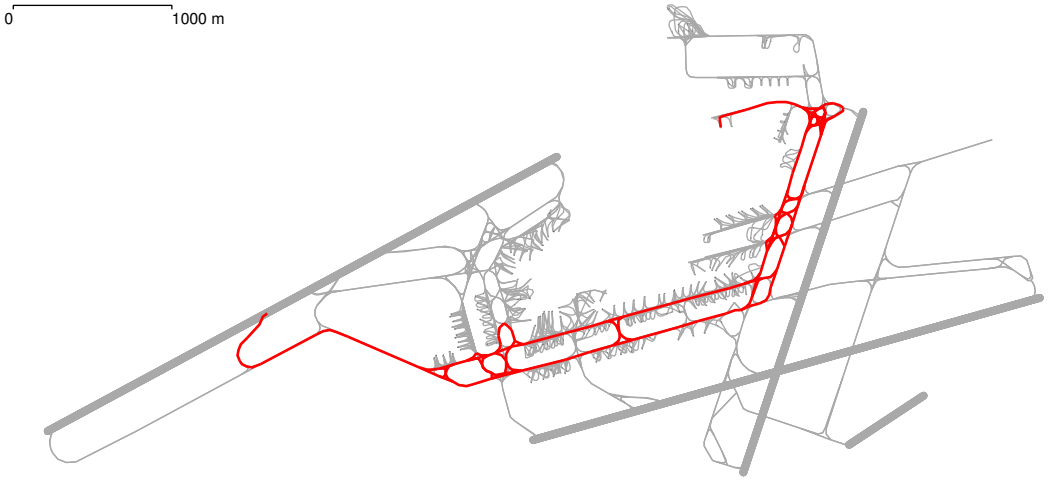

FIG. 3.2 – Graphe d'Orly

Séparation des avions

Les règles opérationnelles de séparation des avions sont les suivantes :
- la distance entre deux avions en mouvement doit toujours être supérieure à 60 mètres ;
- il ne peut y avoir qu'un seul décollage ou atterrissage à la fois sur chaque piste ;
- après un décollage, une séparation de 1, 2 ou 3 minutes (selon la catégorie de turbulence de sillage) est nécessaire avant le décollage ou l'atterrissage suivant ;
- un avion peut circuler dans l'aire de piste (90 mètres de part et d'autre de la piste) pendant un décollage ou un atterrissage à condition qu'il soit derrière l'avion concerné.

On dira qu'il y a conflit entre deux avions si leurs trajectoires prévues ne respectent pas ces règles.

Incertitudes sur la vitesse

Dans la simulation, les règles de séparation déterminent l'admissibilité des trajectoires, moyennant une incertitude relative sur la vitesse de déplacement des avions. Cette incertitude transforme la position d'un avion en un segment (et non un point). Les règles de séparation sont alors appliquées à tous les points du segment.

Deux particularités doivent cependant être prises en compte :
- Dans le cas où deux avions se suivent, le deuxième pilote est supposé adapter sa vitesse à celle du premier, et seules les positions principales (sans incertitudes) sont considérées.
- Lorsqu'un avion doit attendre, le point et l'heure de fin d'attente sont supposés être respectés, ce qui permet de réduire les positions d'incertitudes de l'avion.

3.2 Simulation

La simulation s'effectue par fenêtres glissantes sur toute une journée de trafic : à chaque pas de simulation t (toutes les Δ minutes), la prédiction de trafic est effectuée sur un horizon temporel T_w ($T_w > \Delta$). La situation est résolue sur cet horizon, et cette résolution donne la nouvelle situation Δ minutes plus tard.

L'horizon de prédiction (et de résolution) étant limité, des «effets d'horizon» néfastes peuvent apparaître :
- Deux avions peuvent être amenés l'un en face de l'autre, créant un blocage définitif pour les situations futures.
- Un avion peut se retrouver bloqué dans l'aire de piste alors qu'une arrivée (non prévue par les résolutions précédentes) se présente...

Une analyse des positions en fin d'horizon ($t + T_w$) est donc nécessaire pour assurer la viabilité d'une résolution : toute paire d'avions en position de face à face en fin d'horizon sera considérée comme conflictuelle, et aucun avion ne sera admis sur la piste à l'heure $t + \Delta$ s'il ne l'a pas libérée avant la fin de l'horizon.

Ces nouvelles règles de séparation viennent s'ajouter aux précédentes et seront en vigueur pour toutes les méthodes de résolution présentées dans les parties suivantes.

3.2.1 BB : Méthode 1 contre n

Dans cette stratégie, les mouvements prévus sur l'horizon sont classés et résolus l'un après l'autre : le problème est donc réduit à celui d'un seul avion devant éviter des avions dont la trajectoire a

été précédemment choisie. Ce problème peut être résolu par *Branch & Bound* [HT95] sur le graphe présenté ci-dessous.

Définition du graphe

Sur chacun des chemins possibles de l'avion, le problème peut être résolu rapidement par une recherche *meilleur en premier* dans l'arbre suivant :

- Un nœud de l'arbre est une position de l'avion à une heure donnée.
- La racine de l'arbre est la position initiale de l'avion au début de l'horizon de prédiction.
- Les feuilles (nœuds terminaux) sont constituées de feuilles solution (les positions non conflictuelles de l'avion en fin d'horizon ou à la fin du chemin) et de feuilles non solution : toute position conflictuelle de l'avion.
- Chaque nœud (non terminal) a deux fils : le premier correspond au cas où l'avion avance sur son chemin, le deuxième au cas où l'avion attend. Si le premier aboutit à une feuille solution, c'est la meilleure solution depuis ce nœud.

On peut obtenir la meilleure solution pour l'avion en itérant cette recherche *meilleur en premier* sur chacun de ses chemins.

Cependant, il est possible de connaître pour chaque nœud de l'arbre relatif à l'un des chemins, le retard pris par l'avion jusqu'à ce nœud. Ceci permet de borner la recherche par le minimum des retards pris par l'avion sur les chemins déjà explorés : si le retard dépasse cette borne, la recherche sur ce chemin sera abondonnée.

L'algorithme devient ainsi un *Branch & Bound* avec stratégie d'exploration *meilleur en premier*.

Classement des mouvements

Le classement des mouvements est un point crucial, puisque les derniers avions sont extrêmement pénalisés, la plupart des classements pouvant même s'avérer sans solution.

L'analyse du problème permet cependant de dégager deux principales contraintes que doit respecter le classement :

- Les avions prévus à l'atterrissage ne peuvent pas être retardés avant d'être sortis de la piste, et doivent donc disposer de la piste libre : ils doivent être classés avant les décollages sur cette piste.
- Les avions au décollage faisant la queue pour la piste ne peuvent être classés que dans l'ordre donné par la queue, c'est-à-dire en fonction de la distance restante avant la piste.

En dehors de ces contraintes, un classement «réaliste»des mouvements peut être obtenu par comparaison des heures d'activation du plan de vol.

3.2.2 GA et GA+BB : algorithmes génétiques

Deux stratégies de résolution sont développées et utilisent des algorithmes génétiques classiques [Gol89c, Mic92a].

La première stratégie recherche un chemin et éventuellement une attente par mouvement. La deuxième recherche un chemin et un ordre de classement par mouvement, en utilisant un algorithme BB (cf. 3.2.1) pour développer et évaluer les trajectoires correspondantes.

Codage des données

Dans la première stratégie, la trajectoire d'un avion est décrite par 3 paramètres : le numéro n du chemin suivi, la position p d'attente, et l'heure t de fin d'attente (si p est atteinte après l'heure t, l'avion n'attend pas). Les éléments de la population (ou chromosomes) sont donc constitués de $3N$ variables pour un problème à N avions.

Dans la deuxième stratégie, la trajectoire d'un avion est donnée par 2 paramètres : le numéro n du chemin suivi et l'ordre de classement o de l'avion. Les chromosomes sont donc constitués de $2N$ variables.

Fonction d'adaptation

Dans les deux stratégies, les trajectoires décrites par un chromosome peuvent ne pas être admissibles. La fonction d'adaptation (ou *fitness*) F à maximiser, comprise entre 0 et 1, doit assurer qu'un chromosome représentant une solution admissible est toujours mieux évalué qu'un chromosome décrivant une situation avec conflits.

Pour cela, la *fitness* d'une solution admissible sera toujours supérieure à $\frac{1}{2}$ et la *fitness* des éléments contenant des conflits inférieure à $\frac{1}{2}$.

Si n_c est le nombre de conflits et f_c la fonction de coût décrite au 3.1.2, la *fitness* sera donnée par :

$$\begin{aligned} &\text{si } n_c > 0, \qquad F = \tfrac{1}{1+n_c} \\ &\text{si } n_c = 0, \quad F = \tfrac{1}{2} + \tfrac{1}{1+f_c} \end{aligned}$$

Croisement et Mutation

Le caractère partiellement séparable du problème de résolution de conflits est à nouveau utilisé.

Une *fitness* locale F_i est calculée pour chaque avion i, en fonction du nombre de conflits n_{c_i} impliquant cet avion et de sa participation f_{c_i} à la fonction de coût (cf. 3.1.2) :

$$\text{si } n_{c_i} > 0,\ F_i = K n_{c_i} \quad \text{sinon, } F_i = f_{c_i}$$

(K étant une constante telle que $K \gg f_c$)

Sharing

Le problème est combinatoire et présente de nombreux optima locaux. Le sharing décrit dans Yin et Germay [YG93b] permet d'empêcher la population de s'homogénéiser trop rapidement sur les optima locaux.

Il nécessite l'introduction d'une distance entre deux éléments A et B de la population, cette distance étant utilisée pour dissocier différents groupes (*clusters*) dans la population. La distance définie est la suivante :

$$D(A,B) \quad = \quad \frac{\sum_{i=1}^{N} |l_{A_i} - l_{B_i}|}{N}$$

l_{A_i} (resp l_{B_i}) étant la longueur du chemin de l'avion i dans le chromosome A (resp B).

Critère d'arrêt

Le critère d'arrêt de l'algorithme génétique est défini par un nombre maximal absolu de 50 générations, et un nombre maximal de 20 générations sans conflit (i.e. à meilleur élément décrivant une solution admissible).

Résolutions par *Clusters*

Pour diminuer, autant que possible, la complexité du problème, une fermeture transitive est appliquée aux paires d'avions en conflit dans la situation initiale (sans attente ni déviation). Les ensembles (ou *clusters*) d'avions ainsi formés sont résolus séparément.

A chaque fois que la résolution séparée de deux *clusters* provoque des conflits entre leurs avions, les deux *clusters* sont réunis et une nouvelle optimisation est effectuée.

3.3 Résultats

3.3.1 Simulations

Les simulations sont effectuées avec les plans de vol réels d'une journée de forte activité (18/06/1999) de Roissy et d'Orly. Les résultats obtenus pour Roissy et Orly étant similaires, seuls ceux de Roissy sont présentés et commentés dans ce chapitre.

Les trois stratégies (BB, GA et GA+BB) sont comparées avec les paramètres suivants choisis empiriquement :

- Nombre de chemins par avion : $k = 30$
- Sens uniques : Oui
- Horizon : $T_w = 5mn$
- Pas de résolution : $\Delta = 2mn$
- Incertitude sur les vitesses : $\delta = 10\%$

3.3.2 Comparaison des stratégies

La figure 3.3 donne la corrélation entre le nombre de mouvements actifs et le délai généré pour chacune des stratégies de résolution.

Pour les situations à faible trafic (moins de 10 mouvements actifs), la meilleure méthode est la méthode GA : elle ne classe pas les avions et peut ainsi trouver des solutions plus proches de l'optimum global.

Lorsque le nombre de mouvements augmente, c'est la méthode GA+BB qui génére le moins de retards : ceci signifie que le classement des avions est beaucoup moins pénalisant à forte densité de trafic, et qu'à partir d'un certain nombre de mouvements, le problème devient trop complexe pour que la méthode GA s'approche suffisamment de l'optimum global.

La méthode BB obtient de moins bons résultats que GA+BB, car elle ne remet pas suffisamment en cause le classement initial des mouvements.

Le classement s'avère donc une méthode efficace pour la résolution de conflits impliquant un grand nombre de mouvements, à condition de revoir régulièrement ce classement pour "l'adapter" aux nouvelles situations.

La figure 3.4 donne le nombre de mouvements actifs à chaque pas de simulation. La méthode GA+BB parvient souvent à des situations moins denses que les autres méthodes, et ce surtout pendant les périodes de pointe.

Ceci fait apparaître un phénomène important concernant les aéroports : une bonne résolution de la situation réduit d'une part les délais dans l'immédiat, et amène d'autre part des meilleures situations futures (impliquant moins de mouvements).

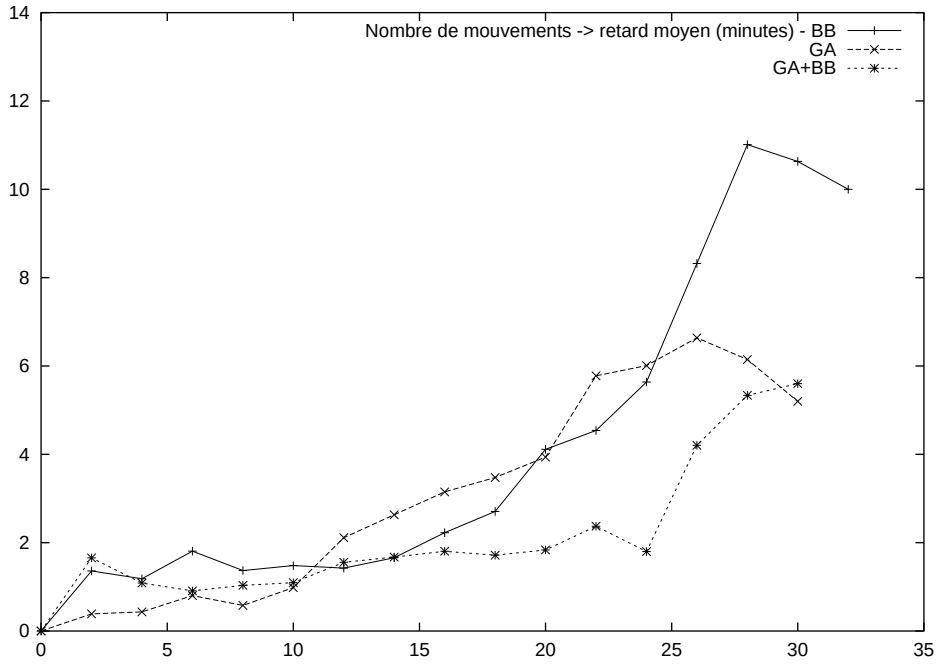

FIG. 3.3 – Retard moyen (en minutes) en fonction du nombre de mouvements

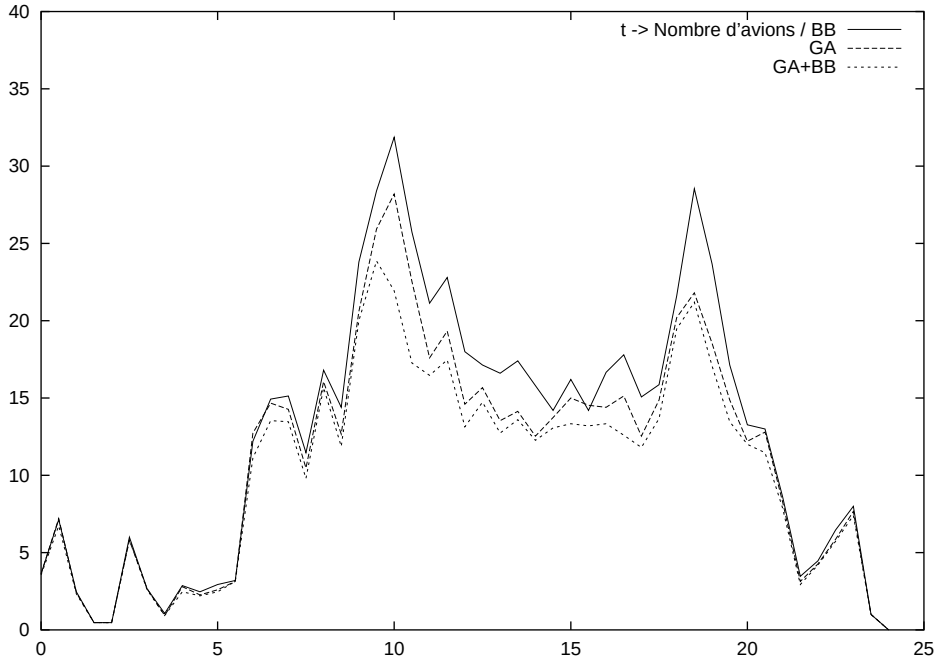

FIG. 3.4 – Nombre d'avions en mouvements en fonction de l'heure de la journée

3.3.3 Remarques

Ces premiers résultats font apparaître que le retard moyen du trafic au roulage sur des aéroports comme Roissy ou Orly peut être réduit de plusieurs minutes en fonction de la stratégie d'optimisation employée. Ceci montre l'intérêt potentiel du développement d'outils d'aide au contrôle au sol (affectation de chemins et de points d'attente).

Les algorithmes génétiques semblent bien adaptés pour traiter ce problème combinatoire car leurs solutions s'approchent plus facilement de l'optimum global, tandis que des algorithmes déterministes (méthode 1 contre n) se bornent à des optima locaux créant des retards plus importants.

On peut noter également que le modèle peut être amélioré facilement (nouvelles pistes à Roissy, incertitudes sur les vitesses, prise en compte des sens uniques...) sans modifier pour autant les algorithmes d'optimisation. Les simulations peuvent ainsi servir à évaluer la mise en place de nouvelles structures ou de nouvelles procédures sur un aéroport.

Beaucoup d'améliorations restent à apporter : par exemple, le critère d'optimisation devra prendre en compte les mouvements ayant un créneau de départ impératif (suite à la régulation européenne du trafic), et les contraintes des secteurs d'approche.

3.4 Conclusion

A travers cet exemple d'application, on a pu montrer que la combinaison d'un algorithme génétique et d'un algorithme de type *Branch & Bound* pouvait s'avérer plus efficace que l'une des deux approches prises séparément. Les algorithmes génétiques s'avèrent efficaces lorsque l'on est confronté à des problèmes combinatoires de grande taille, mais moins performants pour des optimisations locales. Les méthodes déterministes de type BB ne sont applicables que sur des problèmes de taille réduite sur lesquels elles garantissent l'optimalité des solutions trouvées. Cet exemple nous a également permis d'utiliser la séparabilité partielle du problème pour améliorer les performances de l'algorithme génétique.

Deuxième partie

La résolution des conflits en route

Chapitre 4

Le problème de résolution de conflits

Introduction

Historiquement, les avions volaient "à vue" et utilisaient les règles de l'air pour assurer leur séparation, et ce en respectant le principe "voir et être vu". L'amélioration des performances des avions et la densification du trafic ont conduit les Etats à organiser un système de gestion du trafic fondé sur une série de filtres, chaque filtre ayant des objectifs différents et gérant des espaces et des horizons temporels distincts. En Europe, on peut grossièrement distinguer cinq niveaux :

A long terme (plus de 6 mois), le trafic est organisé de façon macroscopique. Sont concernés par exemple les schémas d'orientation de trafic, les mesures du comité des horaires, ou encore les accords inter-centres et les accords avec les militaires, qui permettent aux civils d'utiliser leurs zones aériennes pour écouler les pointes de trafic du vendredi après-midi.

A plus court terme, on parle souvent de pré-régulation : elle consiste à organiser une journée de trafic, la veille ou l'avant-veille. A ce stade, on dispose d'une grosse partie des plans de vols, on connaît la capacité de contrôle que peut offrir chaque centre[1] en fonction de ses effectifs, le débit maximal d'avions pouvant pénétrer dans un secteur[2], appelé capacité du secteur. C'est le rôle de la CFMU[3].

Le jour même, des ajustements sont réalisés en fonction des derniers événements. Le trafic transatlantique, par exemple, peut être pris en compte à ce stade, les avions supplémentaires se voient affecter leurs routes et heures de décollage, on peut également ré-allouer des créneaux horaires non utilisés et tenir compte de la météo du jour. Ce rôle est en général joué par les FMP[4] dans chaque centre.

Le dernier filtre de la chaîne du contrôle aérien est le filtre tactique : il s'agit du contrôle à l'intérieur d'un secteur. Le temps moyen passé par un avion dans un secteur est de l'ordre d'une quinzaine de minutes. La visibilité du contrôleur est un peu supérieure puisqu'il dispose des plans de vol quelques minutes avant l'entrée de l'avion dans le secteur. Le contrôleur assure les tâches de surveillance, de résolution de conflit et de coordination avec les secteurs voisins. Il convient de préciser la définition d'un conflit aérien : deux avions sont dits en conflit lorsque la distance

[1] La France est divisée en cinq centres de contrôle (Paris, Reims, Brest, Bordeaux et Aix en Provence) qui gèrent chacun une partie de l'espace aérien.

[2] Chaque centre de contrôle gère entre 15 et 20 secteurs élémentaires qui peuvent être regroupés en fonction de la densité de trafic et des équipes de contrôleurs disponibles.

[3] «Central Flow Management Unit», située à Bruxelles.

[4] «Flow Management Position»

horizontale qui les sépare est inférieure à 5 milles nautiques[5] et leur différence d'altitude est inférieure à 1000 pieds[6]. Les méthodes de résolution de conflits appliquées par les contrôleurs aériens font appel avant tout à leur expérience. Lorsque plusieurs couples d'avions interagissent dans le même conflit, ils commencent par séparer les problèmes pour n'avoir que des conflits élémentaires à résoudre.

Le filtre d'urgence n'est censé intervenir que lorsque le système de contrôle est absent ou a été défaillant : pour le contrôleur, le *filet de sauvegarde* prédit la trajectoire de chaque avion avec un horizon temporel de quelques minutes à l'aide des positions radar passées et d'algorithmes de poursuite et déclenche une alarme en cas de conflit. Il ne propose pas de solution aux conflits détectés. A bord des avions, le TCAS[7] a pour rôle d'éviter une collision présumée. La prédiction temporelle est inférieure à la minute et varie entre 25 et 40 secondes. Il est alors trop tard pour que le contrôleur intervienne puisque l'on estime qu'il lui faut entre 1 et 2 minutes pour analyser une situation, trouver une solution et la communiquer aux avions. Actuellement, le TCAS détecte les avions environnants et donne un avis de résolution au pilote (pour le moment dans le plan vertical). Ce filtre doit résoudre les conflits non prévisibles comme, par exemple, un avion dépassant un niveau de vol donné par le contrôle, ou un accident technique qui dégraderait notablement les performances de l'avion.

Le problème de résolution de conflits aérien se situe au niveau du filtre tactique : connaissant les positions des avions à un instant donné et leurs positions futures (avec une précision donnée), quelles sont les manœuvres à donner à ces avions afin que les trajectoires ne génèrent aucun conflit et que le retard engendré soit minimal.

Un certain nombre de contraintes doivent être précisées.

- Un avion ne peut pas modifier sa vitesse (ou très faiblement), sauf dans sa phase de descente ;
- On ne peut pas considérer qu'un avion vole à vitesse constante, sauf éventuellement lorsqu'il est en croisière et qu'il n'y a pas de vent. De plus en montée et en descente, sa trajectoire n'est pas rectiligne. On ne peut donc pas en donner de description analytique. L'évaluation des positions futures d'un avion requiert l'utilisation d'un simulateur ;
- Les avions sont contraints en taux de virage, les pilotes préfèrent généralement manœuvrer latéralement que verticalement, sauf dans les phases de montée ou de descente ;
- Bien que les pilotes automatiques soient aujourd'hui largement plus performants que les pilotes humains (dans les situations normales de vol), il ne paraît pas réaliste pour l'instant d'envisager des trajectoires qui ne soient pas exécutables par un pilote humain.
- L'incertitude sur les taux de montée et de descente est très forte (entre 10% et 50% de la vitesse verticale). En croisière l'incertitude sur le vitesse est plus réduite (voisine de 5%). Latéralement l'incertitude ne croît pas avec le temps, de même qu'un avion en croisière tient en général bien son altitude.

La nécessité d'avoir recours à un simulateur pour calculer les positions futures des avions rend impossibles la recherche de solutions analytiques au problème de résolution de conflits aériens, ainsi que l'utilisation de méthodes d'optimisation classiques, ayant recours au gradient ou au hessien du critère à optimiser. Toutefois, la principale difficulté tient plus à la complexité du problème lui-même qu'aux contraintes citées précédemment.

[5] 1 mille nautique équivaut à 1852 mètres

[6] un pied équivaut à 30,48 cm.

[7] «Traffic alert and Collision Avoidance System» : système embarqué à bord de certains avions que les Etats-Unis ont rendu obligatoire pour tous les avions de plus de 30 passagers.

4.1 Complexité du problème de résolution de conflit

Si les avions sont aujourd'hui largement optimisés et automatisés, on peut s'étonner que les tâches de contrôle soient restées pour la plupart artisanales, faisant appel à l'expérience humaine plus qu'à la puissance de calcul d'un ordinateur. Ceci résulte en partie de la complexité du problème de résolution de conflits.

Définition 1. *Pour une fenêtre temporelle de prévision de trajectoire donnée, on appelle conflit potentiel entre deux avions tout conflit détecté pendant la durée de la prévision, et ce en tenant compte des incertitudes sur les trajectoires.*

La relation «est en conflit avec», ou «est en conflit potentiel avec», définit une relation d'équivalence. Les classes d'équivalence associées à cette relation seront appelées «clusters» d'avions en conflit ou tout simplement «clusters».

Si l'on se restreint au plan horizontal, dans le cas où l'on s'intéresse à un cluster comprenant n avions, on peut se retrouver en présence de $\frac{n\,(n-1)}{2}$ conflits potentiels. On peut montrer [Dur96] que l'ensemble des solutions admissibles contient $2^{\frac{n\,(n-1)}{2}}$ composantes connexes, ce qui suppose, si l'on veut utiliser une méthode d'optimisation locale (déformation continue de trajectoires) qu'il faut autant d'exécutions de l'algorithme de recherche. Ainsi, pour un cluster à 6 avions, cela représente 32768 composantes connexes. Dans la pratique, compte tenu des performances des avions, toutes les composantes connexes n'ont pas besoin d'être explorées. Néanmoins, la présence théorique d'autant d'ensembles disjoints et l'impossibilité de connaître a priori lequel contient la solution optimale rend le problème fortement combinatoire. En relaxant la contrainte de séparation, on reste tout de même en présence d'un problème d'optimisation globale comportant au moins autant d'optima locaux que de composantes connexes.

L'ajout de la dimension verticale ne permet pas de réduire le caractère combinatoire du problème dans la mesure où l'on ne propose pas à un avion une manœuvre simultanément dans les deux plans.

4.2 Approche centralisée, approche autonome

La résolution des conflits aériens est réalisée actuellement de façon centralisée. C'est le contrôleur, qui a une vision globale des problèmes à résoudre, qui donne aux avions les ordres de manœuvre.

Il est impossible de parler d'automatisation de la résolution de conflits sans parler du concept *Free-Flight*. Il est apparu outre-Atlantique il y a une dizaine d'années, sous la pression des compagnies aériennes. Elles étaient alors désireuses de s'affranchir des contraintes imposées par les routes aériennes et des coûts associés au contrôle. La vision américaine du *Free-Flight* est sensiblement différente de la vision européenne. Les américains ont moins de problèmes de capacité en route et le souci majeur des compagnies aériennes était de pouvoir réduire le temps de vol de leurs avions en utilisant des routes directes, de l'aéroport d'origine à l'aéroport de destination. Ils n'écartaient pas dans leur notion du Free-Flight la possibilité de conserver un contrôle de type centralisé : ce type de gestion est plutôt appelé en Europe *Free-Route*. En effet, les européens ont une approche un peu différente, car le trafic est limité par la capacité en route. Le Free-Flight a alors été envisagé dans le but de s'affranchir d'un contrôle centralisé, au profit d'un contrôle embarqué, tout en utilisant des routes directes, et ce afin d'alléger la charge de travail du contrôleur.

Le problème est que le *Free-Flight* a pris aujourd'hui un sens politique, et que l'on ne sait jamais bien exactement ce qu'il recouvre. Nous utiliserons par la suite deux notions orthogonales qui permettent de mieux définir les choses :

– Contrôle *centralisé* ou contrôle *autonome* : le contrôle centralisé est un mécanisme où un système au sol assure la sécurité de l'écoulement du trafic. Dans un système autonome, les avions doivent eux-mêmes assurer leur sécurité.
– Routes *directes* ou routes *standards* : dans un système utilisant des routes directes, les avions volent directement de leur origine à leur destination. Avec les routes standards, ils doivent suivre les routes aériennes existantes

4.3 Les projets d'automatisation existants

On distingue dans la littérature deux grands types d'approches du problème de résolution de conflits. Les approches opérationnelles, historiquement plus anciennes, généralement à l'initiative des autorités du contrôle aérien, tiennent généralement largement compte du contexte opérationnel, mais ne s'attaquent pas toujours au problème d'optimisation sous-jacent. Les approches théoriques, apparues plus récemment, s'attaquent efficacement à la complexité du problème, en faisant parfois des hypothèses irréalistes dans un contexte opérationnel. De rares approches abordent le problème combinatoire tout en essayant de respecter les contraintes opérationnelles.

4.3.1 La tentative américaine abandonnée, AERA

Les travaux de recherche de la MITRE sur AERA[8] [Cel90, SPSS83, NFC$^+$83, Nie89b, Nie89a, PA91] ont été financés par la FAA[9]. La première phase de ce projet, AERA 1 permettait de prévoir les trajectoires des avions en fonction des intentions des pilotes et de détecter d'éventuelles violations de la séparation standard ou de restriction de flux pour des plans de vol existants ou désignés par un contrôleur. C'était donc avant tout un outil d'aide à la décision, qui ne proposait pas de solution aux opérateurs.

AERA 2 [Cel90] proposait d'importantes nouvelles aides aux opérateurs, et notamment une liste de "résolutions recommandées par l'ordinateur" aux éventuels conflits élémentaires détectés par AERA 1. AERA 2 introduisait également des outils d'aide à la coordination entre contrôleurs[10]. Néanmoins, dans AERA 2, la responsabilité de séparer les avions relevait en dernier ressort du contrôleur.

Dans AERA 3 [NFC$^+$83, Nie89b, Nie89a, PA91], la responsabilité de séparer les avions était laissée à la machine. La structure d'AERA 3 était hiérarchique. Au niveau national, l'ATMS[11] restait en l'état. Il assurait une gestion des flux de trafic acceptable pour AERA 3.

AERA 3 se décomposait en trois niveaux hiérarchiques :
– L'ASF[12] séparait les paires d'avions.
– Le MOM[13] assurait le respect du contexte global par ASF (qui ne gérait que des paires d'avions séparées).
– L'AMPF[14] s'assurait que MOM pouvait opérer avec succès. En prévenant les trop fortes densités de trafic, il permettait d'éviter que les avions aient trop peu de possibilités de manœuvres.

[8] Automated En-Route Air Traffic Control
[9] Federal Aviation Administration
[10] Les contrôleurs surveillent et interviennent sur des secteurs de contrôle ; quand un avion passe d'un secteur à l'autre, il doit être libéré par le premier et être pris en compte par le suivant.
[11] Air Traffic Management System
[12] Automated Separation Function
[13] Maneuver Option Manager
[14] Airspace Manager Planning Functions

ASF et MOM étaient les deux niveaux dont l'automatisation complète était envisagée. Néanmoins, le seul niveau détaillé clairement d'AERA 3 était l'ASF, qui ne résolvait que des conflits à deux avions. Les objectifs de MOM étaient assez clairement définis. Par contre, son fonctionnement est toujours resté très flou et peu convaincant. La recherche d'optimalité globale n'apparaissait pas du tout dans AERA 3.

L'ASF était assurée par un algorithme appelé "Gentle-Strict" (GS) dont le but était de résoudre automatiquement des conflits de croisement horizontal entre deux avions en utilisant des manœuvres latérales d'offset[15]. GS prévoyait :

- une manœuvre d'offset latéral pour un des deux avions dit "gentle" selon un sens prédéterminé.
- une consigne de tenue de trajectoire pour l'autre avion dit "strict" dans le but d'éviter les déviations supérieures à un certain seuil de tolérance par rapport à sa trajectoire nominale.

GS ne résolvait un conflit que dans le plan horizontal et ne changeait jamais les vitesses des avions ou leurs niveaux de vol. GS agissait le plus tard possible afin d'éviter les manœuvres inutiles.

Le projet AERA n'a jamais abouti. Les algorithmes utilisés ou envisagés dans le MOM n'ont jamais été évoqués dans les rapports disponibles. Sur le plan de la modélisation, AERA3 peut garder un intérêt. Par contre, ce projet n'apporte pas de réponse au problème d'optimisation globale dès que le nombre d'avions dépasse 3.

4.3.2 ARC2000 et ses dérivés

Le projet d'automatisation complète du contrôle en route européen, ARC2000 [K$^+$89, FMT93] a connu de profonds changements depuis son origine. Néanmoins la modélisation en "tubes 4D" des trajectoires d'avions est restée et a permis de définir des outils d'aide à la décision innovants pour le contrôleur [MG94].

Si l'on est capable de prédire avec une très bonne précision la trajectoire d'un avion, on peut représenter celle-ci par une courbe dans $\mathbb{R}^4$, les trois premières variables représentant la position dans l'espace de l'avion au temps t, quatrième variable. Les imprécisions de mesures de positions et de tenues de trajectoires $4D$ transforment, dans un souci de réalisme, cette courbe en "tube" en lui donnant une certaine section. Pour séparer les trajectoires d'avions, il faut donc fabriquer un "tube" pour chaque avion de sorte que les différents tubes soient d'intersection vide. Pour cela, ARC2000 utilise un algorithme d'optimisation locale (de type "gradient") qui permet, étant donné n tubes d'intersection vide, de construire un tube $n+1$ d'intersection vide avec tous les précédents et minimisant le retard de l'avion $n+1$.

Le principe d'ARC2000 était donc originellement le suivant : le premier avion entrant dans le système se voyait affecter un tube optimal respectant son plan de vol. Dès qu'un nouvel avion se présentait, son tube optimal était calculé en considérant les tubes précédemment affectés comme des contraintes fixes. Autrement dit, un tube déjà affecté n'était pas remis en cause. L'optimalité globale n'était donc pas recherchée. Dans le cas où un avion ne respectait pas le tube qui lui était affecté, il devait négocier un nouveau tube respectant tous les autres tubes. La faiblesse de ce principe se caractérisait par son manque de robustesse. En effet, il est possible que le non respect d'un tube 4D soit lié à des événements météorologiques par exemple, et qu'il ne touche pas seulement un avion mais plusieurs. On pouvait alors craindre un phénomène chaotique qui remettait en cause le principe adopté.

Les hypothèses de départ d'ARC2000 le projetaient bien au-delà de l'an 2000. En effet, la négociation des tubes 4D supposait que les avions soient équipés de FMS-4D[16], ce qui n'était pas réaliste

[15] un offset est une mise en parallèle par rapport à la trajectoire initiale de l'avion

[16] Flight Management System en 4 dimensions

à court terme.

Le plan stratégique d'ARC2000 consistait à garantir des trajectoires sans conflit pour les 20 minutes à venir. Il fallait en effet trouver un juste milieu entre prévoir des trajectoires sans conflit très longtemps à l'avance, et ne pas tenir compte des conflits à l'avance et réajuster continuellement les trajectoires. Le compromis d'ARC2000 consistait à prévoir la trajectoire dans sa totalité avec le moins possible de conflits, tout en surveillant les conflits potentiels. 20 ou 30 minutes à l'avance, les conflits étaient éliminés. Avant cette période, seuls les conflits qui étaient fermement diagnostiqués pouvaient entraîner une réorganisation des trajectoires.

ARC2000 s'est ensuite tourné vers des hypothèses plus réalistes. La modélisation de la trajectoire totale sans conflit (de l'origine à la destination) semble avoir été abandonnée pour une gestion à 20 ou 30 minutes des groupes de conflits ou clusters. La recherche de solutions optimales a également évolué puisqu'au principe du "dernier arrivé, dernier servi", a succédé une série de règles permettant de classer les avions du plus prioritaire au moins prioritaire, et les manœuvres de la plus opportune à la moins opportune. On retiendra essentiellement que la recherche d'un optimum global a toujours été délaissée au profit d'un parcours d'arbre dans un ordre régi par des règles empiriques.

Le projet ARC2000 a été arrêté au milieu des années 90.

Il est à noter que les recherches d'Eurocontrol ont permis de produire des outils d'aide à la décision très innovants tels qu'HIPS[17][MG94], qui ne s'intéresse plus à l'optimisation automatique de trajectoire, mais donne au contrôleur des outils d'aide à la représentation des trajectoires et des conflits.

On notera enfin qu'ARC2000 a été testé sur du trafic réel. Comme pour AERA3, la modélisation du problème est intéressante, mais les algorithmes de résolution de clusters ne s'attaquent pas au problème d'optimisation global des trajectoires.

4.3.3 Le projet SAINTEX

Dans le projet SAINTEX[18] [AL92], trois approches d'automatisation du contrôle en route étaient abordées.

- Le scénario "Détection-Résolution" était un système orienté système expert. Le système essayait de reproduire le comportement du contrôleur. Les conflits étaient détectés en extrapolant les trajectoires d'avions 10 minutes dans le futur (6 minutes pour les avions évolutifs[19]). Aussitôt détecté, un conflit était classé suivant différents critères, tels que l'angle formé par les trajectoires, le rapport des vitesses, etc. Pour chaque classe de conflits, une manœuvre prédéfinie était appliquée. Le système expert ne pouvait résoudre qu'un conflit entre deux avions.
- Dans le scénario "4D", une trajectoire sans conflit était engendrée pour tout avion entrant dans le secteur. Une trajectoire était représentée par un ensemble de points et de contraintes verticales. Pour chaque avion, on construisait un tube 4D représentant sa trajectoire, compte tenu des incertitudes sur sa vitesse et sa position. Pour construire un tube admissible (sans conflit), SAINTEX faisait diverses tentatives partant de la trajectoire idéale (directe) vers des trajectoires de plus en plus pénalisantes, mais résolvant les conflits. Du temps que l'on consacrait à la recherche d'une trajectoire admissible découlait la qualité de la résolution. La trajectoire de l'avion était ensuite surveillée de manière à s'assurer que l'avion respectait bien la trajectoire qui lui avait été affectée. Ce scénario, purement algorithmique, ressemblait fortement à celui décrit dans ARC2000.

[17] Highly Interactive Problem Solver

[18] SAINTEX faisait référence au vol de nuit. Le projet était prévu pour être mis en œuvre avec de faibles densités de trafics, comme la nuit, par exemple.

[19] Un avion évolutif, contrairement à un avion dit "stable", est soit en montée, soit en descente.

– Dans le scénario hybride, les avions stables étaient gérés par le système Détection-Résolution et les autres par le système 4D.

Utiliser un système expert pour résoudre un conflit à deux avions ne se justifie pas. Le projet SAINTEX, par définition, s'intéressait à la gestion automatique d'espaces aériens peu saturés. Le problème de la résolution des clusters d'avions était évoqué, mais non résolu.

4.3.4 Le Projet FREER

Le projet FREER[20] est né en 1995 à Eurocontrol, Brétigny. Il a connu plusieurs évolutions.

Au départ [DHN97, DH97], on pouvait distinguer deux concepts : dans FREER-1, les avions étaient totalement autonomes dans des espaces à faible densité, comme par exemple l'espace au-dessus du niveau 390 ou la Méditerranée. Il n'y avait alors plus d'infrastructure au sol pour gérer le trafic. La prévision et la résolution de trafic se faisait avec une anticipation de 6 à 8 minutes. Les conflits ne devaient cependant pas impliquer plus de 4 avions, faute de quoi les algorithmes embarqués ne garantissaient plus la résolution des conflits. FREER-2 était censé, à une échéance de 15 à 20 minutes, assurer le respect de cette dernière contrainte. Il s'agissait d'un filtre pré-tactique géré depuis le sol, chargé de veiller à ce que l'espace *Free-Flight* ne soit pas saturé.

Les hypothèses de FREER-1 étaient les suivantes. L'espace aérien concerné est le FFA (*Free-Flight Airspace*) défini par l'EATMS[21] dans lequel on utilise des "règles de l'air étendues" ou EFR[22] [DHN$^+$96], qui sont une extension des règles de l'air utilisées en vol à vue dans les espaces aériens non contrôlés par exemple.

Nous ne donnerons pas de description exhaustive de ces règles de l'air étendues. Le lecteur intéressé se reportera aux références bibliographiques sur FREER. Les concepteurs de FREER ont dû se résoudre à compléter les règles de l'air de façon à pouvoir, d'une part prendre en compte toutes les configurations de conflit à deux avions, et d'autre part définir un ordre total sur l'ensemble des avions, dès lors que l'on s'intéresse à trois avions ou plus. Par exemple, si trois avions arrivent simultanément sur le même point, en suivant des routes Nord, 120 degrés et 240 degrés, la règle de priorité à droite ne permet pas de définir l'ordre de priorité pour ces trois avions. Un ordre basé sur le code transpondeur peut, par exemple, lever l'incertitude dans ce cas là.

Les modifications de trajectoires sont gérées par les avions qui respectent des règles de l'air et les procédures en vigueur. Les algorithmes embarqués proposés pour la définition des manœuvres d'évitement sont fondés sur HIPS[23], un outil dérivé d'ARC2000, qui permet de visualiser en temps réel l'évolution des NoGoZones (Zones de conflits) lorsque l'on fait évoluer les trajectoires des avions. HIPS suppose néanmoins que la trajectoire de l'avion à éviter est connue.

Le fonctionnement de FREER-2 dans ce contexte n'a jamais été précisément décrit.

Par la suite, FREER-2 a sensiblement évolué [DF98]. Il s'agissait de proposer un outil de délégation partielle de la résolution de conflits élémentaires (impliquant deux avions) dans différents espaces plus denses. Les avions devaient être équipés d'un système de type ASAS[24]. Plusieurs simulations ont été menées afin de savoir si un tel concept pouvait alléger la charge de travail des contrôleurs. Si les conclusions sont assez optimistes, la délégation reste très partielle, les concepts introduits peu novateurs et la gestion globale du trafic n'est pas vraiment changée.

[20] Free-Route Experimental Encounter Resolution
[21] European Air Traffic Management System
[22] Extended Flight Rules
[23] Highly Interactive Problem Solver
[24] Aiborne Separation Assurance Systems

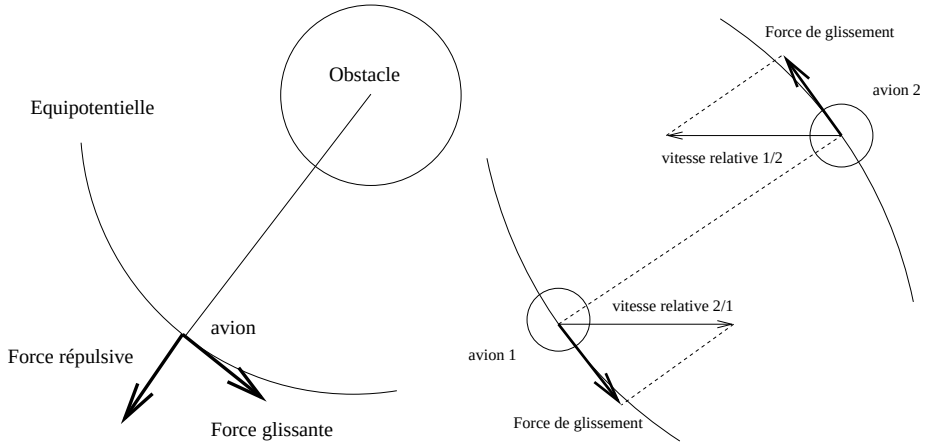

FIG. 4.1 – Force répulsive et force glissante, forces de glissement coordonnées.

Actuellement, des expérimentations sont en cours à Eurocontrol pour évaluer la faisabilité d'une délégation des tâches d'espacements aux équipages.

4.3.5 Les méthodes de forces répulsives.

Le projet ATLAS[25] est parmi les premiers projets à avoir envisagé l'hypothèse d'avions autonomes ou hybrides [DA93].

Plusieurs méthodes se sont appuyées sur cette hypothèse. La plus ancienne est celle étudiée par Karim Zeghal [Zeg93, Zeg94] dans sa thèse, dans laquelle il introduit la notion de coordination d'actions, grâce à différentes forces qui s'exercent sur les agents, dans notre cas, les avions.

Karim Zeghal définit ainsi trois types de forces qui devront agir, suivant l'urgence :

- Les forces attractives, qui permettent aux avions d'atteindre leur objectif (une balise ou leur destination finale par exemple).
- Les forces répulsives, qui permettent aux avions d'éviter un obstacle proche, donc dangereux. Cet obstacle peut être un avion ou une zone interdite.
- Les forces de glissement, qui permettent de contourner les obstacles. Les avions ont alors une action coordonnée (voir figure 4.1). Une force de glissement est définie de la manière suivante : si l'on observe l'équipotentielle, de danger passant par l'avion, une force de glissement est tangente à cette équipotentielle alors que la force répulsive est normale à cette équipotentielle. Il y a donc plusieurs forces de glissement possibles. Si l'on reste dans le plan horizontal, la figure 4.1 montre que l'on peut définir deux sens de glissement (à droite ou à gauche). Le sens optimal (il s'agit ici d'optimalité locale, pour l'avion concerné) est celui qui favorise le rapprochement vers l'objectif. Dans le cas d'un obstacle mobile, Karim Zeghal définit une action coordonnée d'évitement. Si deux avions arrivant au même point suivent des forces de glissement complémentaires, alors l'évitement peut se faire de façon très efficace. Dans le cas simple de deux avions (voir figure 4.1), on projette par exemple la vitesse relative sur l'équipotentielle de danger pour chaque avion, et on obtient ainsi deux forces de glissement coordonnées.

Il s'agit ensuite, pour limiter les changements brutaux de direction, de gérer l'intensité des différentes forces. En effet, les avions étant limités par leurs taux de virage, de montée et de descente,

[25] *Air Traffic Land and Airborne Systems*, étude de la CCE/DG XIII, étude de définition d'un système ATM/CNS unique en Europe

ils ne peuvent pas effectuer de manœuvre trop rapide. Les différentes forces s'exerçant sur les avions s'additionnent donc avec des coefficients variables, suivant l'imminence du danger.

Pour gérer les conflits à plus de deux avions, Karim Zeghal propose d'additionner les forces relatives à chaque avion. "Intuitivement, excepté quelques cas exotiques, cela devrait permettre d'obtenir des directions de forces cohérentes entre elles et par rapport aux obstacles". Les fondements de cette hypothèse ne reposent que sur l'étude de cas pratiques. Néanmoins, ces résultats pratiques semblent prometteurs pour des faibles densités.

La coordination d'actions grâce aux forces de glissement n'a pas pour objectif la recherche d'optimalité, mais vise, avant tout, une bonne efficacité, ainsi qu'une bonne robustesse des solutions.

L'étude des possibilités d'utilisation d'une théorie de la coordination d'actions pour les besoins de la navigation aérienne a commencé au CENA en 1994. Trois systèmes sont envisagés :

- Dans la mesure où le processus individuel ne nécessite que des informations locales à l'appareil, il peut fonctionner de façon autonome à partir d'une perception de son environnement. Karim Zeghal propose donc un système d'avion autonome "hybride"[26]. L'intérêt premier de ce système est sa robustesse. La panne d'un avion ne remet pas en cause la sécurité du système. Il apparaît une forme de redondance dans le système qui renforce sa robustesse. Ce système devrait gérer des conflits entre 4 et 10 minutes à l'avenir, ce qui suppose qu'un système supérieur continue à gérer la densité locale de trafic, afin d'éviter une saturation.
- Karim Zeghal propose également un système d'aide au contrôleur. Il suffit pour cela d'utiliser le processus individuel sous forme de processus centralisé et de simuler les trajectoires futures. Le résultat d'une simulation est alors un ensemble de trajectoires assurant l'évitement sur l'intervalle de temps considéré pour l'ensemble des appareils. Ceci pourrait, selon lui, avoir un grand intérêt pour planifier les trajectoires et faire des propositions au contrôleur.
  Les résolutions proposées par ces techniques étant des commandes continues, il semble peu probable que celles-ci puissent être mises en pratique dans le cas où le pilote et les contrôleurs sont maintenus dans le système. Il me semble que ce dernier point soit un obstacle majeur à l'utilisation de ces techniques.
- Enfin, on peut imaginer un système double "intermédiaire". Dans ce cas, certains avions sont autonomes et d'autres pas. Le contrôleur a alors une disponibilité supplémentaire, et la capacité de son secteur augmente.

La robustesse des solutions est un des atouts des travaux de Karim Zeghal. Il reste néanmoins de nombreux problèmes à résoudre. Dans la mesure où l'on souhaite continuer à confier à l'homme son rôle de pilote dans l'avion, il est indispensable de pouvoir lui transmettre des manœuvres exécutables et donc simples. Par exemple, on peut demander à un pilote de changer de cap pendant un certain temps, de prolonger une montée ou de modifier une heure de début de descente. On ne peut pas lui demander de modifier en permanence son cap, sa vitesse et son taux de montée ou de descente. Il serait donc nécessaire de revoir la modélisation des trajectoires pour les simplifier et les rendre accessibles aux pilotes. De plus, si l'on peut espérer prouver que ces techniques peuvent résoudre tous les conflits pour deux avions, la généralisation à n avions (qui consiste à additionner les forces générées par chaque avion) n'a été vérifiée que sur des exemples. Enfin, la modélisation choisie ne permet aucune recherche d'optimalité globale. Des approches similaires utilisant des champs de potentiel sont testées par le département d'aéronautique de Berkeley [GT00], mais elles ne permettent pas pour l'instant de résoudre des conflits à plus de trois avions, tout comme les méthodes neuronales testées dans la partie 6.1.

[26] L'équivalent du TCAS américain (*Traffic alert and Collision Avoidance System*) mais à plus longue échéance

4.3.6 Résolution par programmation linéaire en nombres entiers

Dans le cadre d'une résolution de conflits aériens dans le plan horizontal uniquement, et en admettant que les avions se déplacent à vitesse constante, Pallatino, Feron et Bicchi [PFB02] ont montré qu'il était possible de linéariser le problème dans le cadre de résolutions en vitesses ou de résolutions en cap, et ce moyennant l'ajoût d'un certain nombre de variables booléennes. Ainsi pour une résolution en vitesses, un conflit à n avions engendre un problème à $2\,n^2 - n$ variables et $4\frac{n(n-1)}{2} + 2\,n$ contraintes. Pour une résolution en caps, le problème engendre $11\frac{n(n-1)}{2} + n + 1$ variables et $35\frac{n(n-1)}{2} + 2\,n$ contraintes. L'utilisation de CPLEX [ILO99] permet aux auteurs de résoudre des situations de conflits à 15 avions, mais les hypothèses requises sur les profils de trajectoires sont assez contraignantes (avions stables, vitesses constantes, pas de gestion du retour sur trajectoire). Toutefois, ces travaux sont parmi les rares travaux existants prenant en compte l'aspect combinatoire du problème de résolution de conflit. Ils constituent, à ce titre, une des *trop rares* approches sérieuses du problème.

Chapitre 5

Approches centralisées

Introduction

Dans ce chapitre sont détaillées les différentes méthodes de résolution que j'ai explorées durant les 10 dernières années, utilisant une approche centralisée du problème de résolution de conflits, à savoir :

- les résultats obtenus durant ma thèse de doctorat grâce aux algorithmes génétiques [Dur96] ;
- une approche utilisant un AG combiné à un algorithme de programmation linéaire qui a fait l'objet du stage de DEA de Frédéric Médoni, co-encadré avec Jean-Marc Alliot [MDA94] ;
- une méthode de *branch and bound* par intervalles, reprise dans la thèse de Frédéric Médioni [Méd98] ;
- une méthode de programmation semi-définie, présentée par Eric Féron [FMF01], et reprise au cours du stage de DEA de Pierre Dodin [Dod99], co-encadré avec Jean-Marc Alliot.

Toutes ces méthodes seront comparées sur un même cas d'école : un conflit à 5 avions volant à la même vitesse dans le plan horizontal et convergeant vers le centre d'un demi-cercle.

5.1 Résolution de conflits par algorithmes génétiques

L'algorithme de résolution de conflits actuellement utilisé par le simulateur de trafic CATS/OPAS[1] met en œuvre un algorithme génétique développé au cours de ma thèse de Doctorat [Dur96]. La modélisation du problème, le codage des données, les différents opérateurs et les paramètres utilisés sont détaillés dans les prochains paragraphes.

5.1.1 Modélisation du problème

Le simulateur effectue toutes les δ minutes (en pratique $\delta = 2$ ou 3 mn) une détection de conflits par paire sur une fenêtre temporelle de T_w minutes (en pratique T_w se situe entre 8 et 12 mn). Les conflits par paires sont regroupés en *clusters* par fermeture transitive de la relation d'équivalence "est en conflit avec". Chaque *cluster* est ensuite résolu par un algorithme génétique qui propose au simulateur des manœuvres de résolution. Avant d'envoyer les ordres aux avions, le simulateur refait une détection de conflit afin de vérifier que les nouvelles trajectoires de deux avions n'appartenant pas aux mêmes *clusters* n'interfèrent pas. Dans un tel cas, les deux *clusters* sont regroupés en un seul et une nouvelle résolution est opérée.

[1] *Complete Air Traffic Solver* / Outil de Planification ATM et Simulations

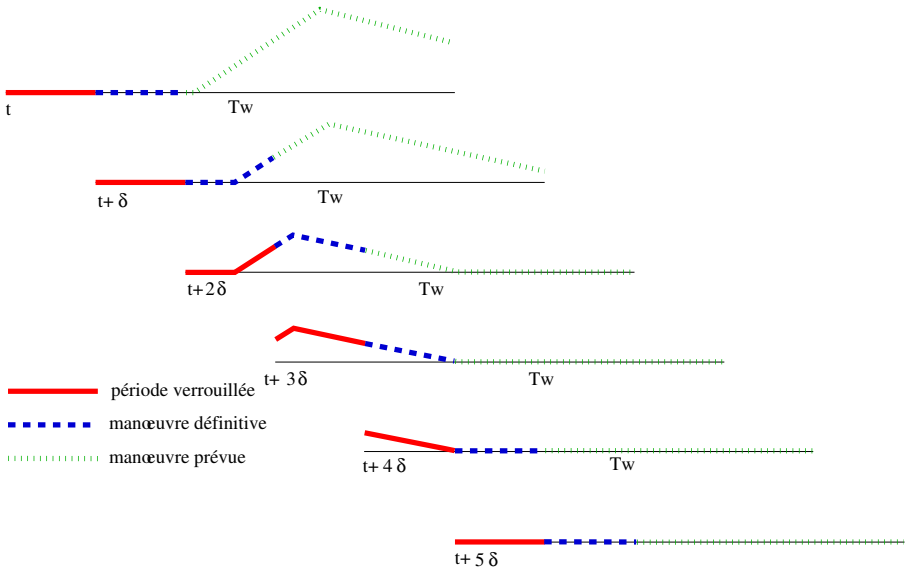

FIG. 5.1 – Résolution en temps réel.

La figure 5.1 montre comment est modélisé le problème en temps réel. Trois périodes sont distinguées dans l'horizon temporel. La première période d'une durée de δ minutes est la *période verrouillée*. Aucune modification de trajectoire ne peut être opérée pendant cette période. En effet, pendant le temps nécessaire à l'évaluation de la situation, la résolution des éventuels conflits et la transmission des ordres de manœuvre aux avions, les avions continuent de voler. Il n'est par conséquent pas possible de modifier leur trajectoire. La période suivante est appelée *période définitive* car les ordres de manœuvres donnés pendant cette période ne pourront pas être modifiés au cours de la prochaine optimisation. La dernière période est la *période de manœuvres prévues*. Ces manœuvres seront reconsidérées au cours de la prochaine optimisation du processus. En raison de l'incertitude, certains conflits peuvent disparaître, et les manœuvres qui les résolvaient être supprimées.

5.1.2 Modélisation des manœuvres

Le cas des routes directes est le plus simple. Le lecteur intéressé par l'adaptation aux routes standard se reportera à la thèse de Géraud Granger [Gra02]. Dans le plan horizontal, le solveur propose des changements de cap de 10, 20 ou 30 degrés à droite ou à gauche, débutant à un instant t_0 donné, et se terminant à t_1. A l'issue de la manœuvre, l'avion reprend un cap direct vers sa destination. Cette manœuvre est modélisée par trois variables (l'altération de cap, t_0 et t_1).

Dans le plan vertical, on distingue quatre phases dans un vol, comme le montre la figure 5.2. Pendant la phase de montée, celle-ci peut être interrompue à t_0 pour reprendre à t_1. En phase de croisière, l'avion peut se voir attribuer le niveau immédiatement inférieur à t_0 pour reprendre son niveau initial à t_1. Environ 50 milles nautiques[2] avant le début de descente, celle-ci peut être anticipée à t_0, un palier étant marqué à t_1 en attendant de rattraper le plan de descente. Aucune manœuvre dans le plan vertical n'est possible pendant la descente. Toutefois, il pourra être demandé à l'avion de réduire sa vitesse de 10%, 20% ou 30%.

Le modèle prévoit qu'il ne peut être donné qu'une manœuvre à la fois : un avion ne peut par

[2]Dans la suite du texte, le sigle NM est utilisé pour désigner un mille nautique (1852 m)

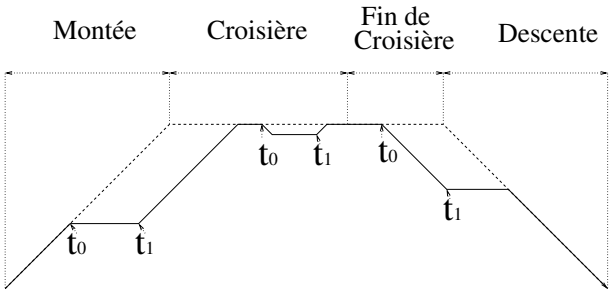

FIG. 5.2 – Manœuvres dans le plan vertical.

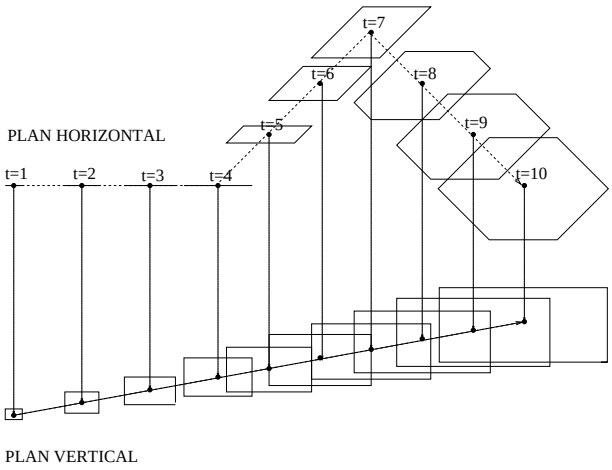

FIG. 5.3 – Modélisation des incertitudes.

exemple pas être mis en descente anticipée alors qu'il est déjà sous le coup d'une déviation dans le plan horizontal. De plus, une manœuvre commencée ne peut pas être remise en cause.

5.1.3 Modélisation des incertitudes

A l'instant initial (voir figure 5.3), l'avion est représenté par un point. Tant que l'avion ne change pas de direction, le point se transforme en segment dans la direction du vecteur vitesse de l'avion. L'avion occupe un point du segment. Un changement de direction dû à une manœuvre va transformer le segment en un parallélogramme. Le parallélogramme s'étire dans la direction du nouveau vecteur vitesse. Lors d'un nouveau changement de direction, le parallélogramme se transforme en hexagone qui suit la direction du nouveau vecteur vitesse pour s'étendre, et ainsi de suite... Pour tester un conflit, on mesure la distance entre deux convexes, chaque convexe (segment, parallélogramme, hexagone) délimitant les positions possibles de l'avion dans le plan horizontal.

Dans le plan vertical, on utilise un modèle cylindrique. Tous les avions ont une altitude maximale et minimale. Pour tester le respect de la norme de séparation verticale, on compare l'altitude la plus basse de l'avion le plus haut à l'altitude la plus haute de l'avion le plus bas.

5.1.4 La fonction d'évaluation

La fonction à optimiser pour chaque *cluster* tient compte de plusieurs critères différents :

– assurer toutes les séparations entre les avions ;

– minimiser les délais imposés aux avions ;
– minimiser le nombre de manœuvres nécessaires ainsi que le nombre d'avions déviés ;
– minimiser les temps de manœuvres pour que l'avion soit libre aussitôt que possible.

Une manœuvre étant modélisée par 3 variables (le sens de la manœuvre, t_0 et t_1), un *cluster* à n avions engendre un problème d'optimisation à $3\,n$ variables. Le modèle prévoit qu'il ne peut être donné plus d'une manœuvre à la fois et qu'une manœuvre commencée ne peut pas être remise en cause.

La taille de la population utilisée est proportionnelle à la taille du problème (10 éléments par avion avec un maximum de 200 éléments de population pour les problèmes impliquant plus de 20 avions).

Pour chaque configuration, une matrice F de taille $(n \times n)$ permet de stocker les informations suivantes :

– Le terme de la diagonale $F_{i,i}$ mesure l'allongement de la trajectoire de l'avion i. Il est nul si aucune manœuvre n'est donnée à l'avion i.
– Le terme $F_{i,j}$ avec $i < j$ mesure la violation de séparation entre l'avion i et l'avion j. Il est nul lorsque les deux avions ne sont pas en conflit.
– Le terme $F_{i,j}$ avec $i > j$ mesure l'efficacité de la résolution de conflit entre l'avion i et l'avion j.

La fonction d'évaluation proposée est de la forme :

$$\begin{aligned} \exists(i,j),\, i \neq j,\, F_{i,j} \neq 0 \;\Rightarrow\; F &= \frac{1}{2 + \sum_{i \neq j} F_{i,j}} \\ \forall(i,j),\, i \neq j,\, F_{i,j} = 0 \;\Rightarrow\; F &= \frac{1}{2} + \frac{1}{1 + \sum_{i} F_{i,i}} \end{aligned}$$

Elle garantit qu'une configuration sans conflit est toujours mieux évaluée qu'une configuration où un ou plusieurs conflits subsistent.

5.1.5 L'opérateur de croisement adapté

La «*fitness* locale» associée à chaque avion est définie de la façon suivante :

$$F_i = \sum_{j=1}^{n} (F_{i,j})$$

L'opérateur de croisement est appliqué sur 50% de la population.

5.1.6 L'opérateur de mutation

Un opérateur de mutation adapté est également utilisé. La figure 5.5 en donne la description. Un avion est tiré parmi ceux dont la fitness locale est supérieure à un seuil donné. Il peut par exemple s'agir des avions qui sont encore en conflit.

L'opérateur de croisement est appliqué sur 15% de la population.

5.1.7 Autres paramètres

La distance utilisée pour le *sharing* est la suivante : deux manœuvres sont considérées égales si elles sont toutes deux verticales ou horizontales et, dans ce dernier cas, si elles s'effectuent du

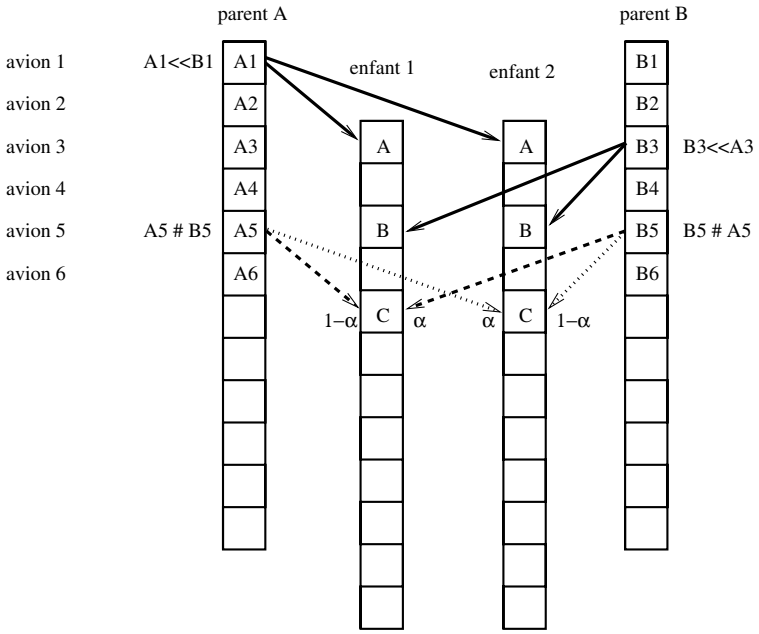

FIG. 5.4 – Un opérateur de croisement adapté

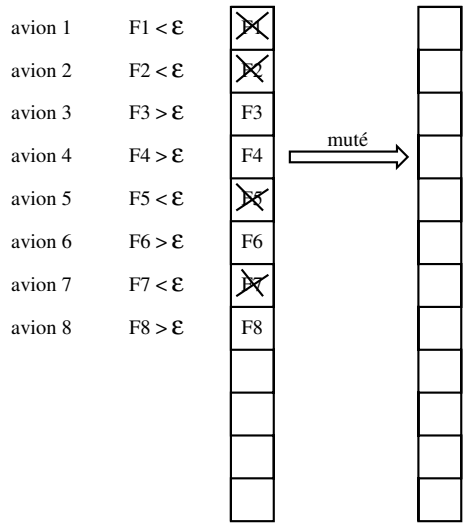

FIG. 5.5 – Un opérateur de mutation adapté

même côté. Pour mesurer la distance entre deux configurations, on compte le nombre de manœuvres différentes.

L'élitisme est utilisé : à chaque génération, le ou les meilleurs individus de la population sont protégés de façon à ne pas disparaître au cours d'un croisement ou d'une mutation.

Compte tenu des impératifs temporels imposés par une gestion en temps réel du trafic, le critère d'arrêt utilisé est très simple. Il consiste à interrompre l'optimisation au bout d'un certain nombre de générations (une vingtaine généralement). Toutefois ce nombre est augmenté tant que l'algorithme ne parvient pas à trouver de solution sans conflit (un maximum de 40 générations est toutefois fixé).

5.1.8 Prise en compte de l'effet horizon

La solveur n'a qu'une vision à court terme des trajectoires des avions. Avec une fonction de coût qui consiste à simplement limiter le retard engendré par une manœuvre, le solveur est parfois tenté de reporter un conflit au-delà de la fenêtre temporelle, sans pour autant le résoudre. Afin de contrer cet «effet horizon», on peut mesurer l'efficacité de résolution d'un conflit et modifier la fonction de coût de l'algorithme de résolution.

Pour toute paire d'avions impliqués dans un *cluster* :

- Si les avions ne sont pas en conflit, il n'y a pas lieu de pénaliser la fonction de coût.
- Dans le cas contraire, si les trajectoires entre les positions courantes des avions et leurs destinations se croisent, la fonction de coût est pénalisée lorsque les trajectoires entre les positions des avions en fin de fenêtre temporelle et leurs destinations se croisent encore. Dans ce cas, l'algorithme de résolution n'aura pas résolu le conflit, mais seulement repoussé le conflit au-delà de la fenêtre de résolution.

5.1.9 Conflit à 5 avions

Dans cet exemple, 5 avions initialement positionnés sur un demi-cercle convergent vers le centre du cercle à une vitesse de $400\,kts$[3] avec 3% d'erreur sur la vitesse horizontale. La situation est révisée toutes les 5 minutes. La séparation standard est fixée à 5 milles nautiques. Le nombre de générations est fixé à 20 et la taille de la population à 50.

La figure 5.6 illustre la résolution optimale. L'AG nécessite 750 appels à la fonction d'évaluation.

Les résultats représentés ne donnent que la meilleure solution de ce problème à 5 avions. L'algorithme génétique propose grâce au *sharing* d'autres solutions proches de la meilleure solution (notamment la solution symétrique de la figure 5.6).

5.1.10 Conflit à 20 avions

L'exemple précédent est repris dans les mêmes conditions avec une vingtaine d'avions positionnés sur un cercle.

La figure 5.7 illustre une solution obtenue par algorithmes génétiques qui résout tous les conflits.

Parmi les méthodes testées dans ce rapport, c'est la seule ayant permis de s'attaquer à un problème à 20 avions de façon globale.

[3] le sigle *kts* est utilisé pour désigner les noeuds (1 noeud = 1 mille nautique / heure = 1852 m/heure)

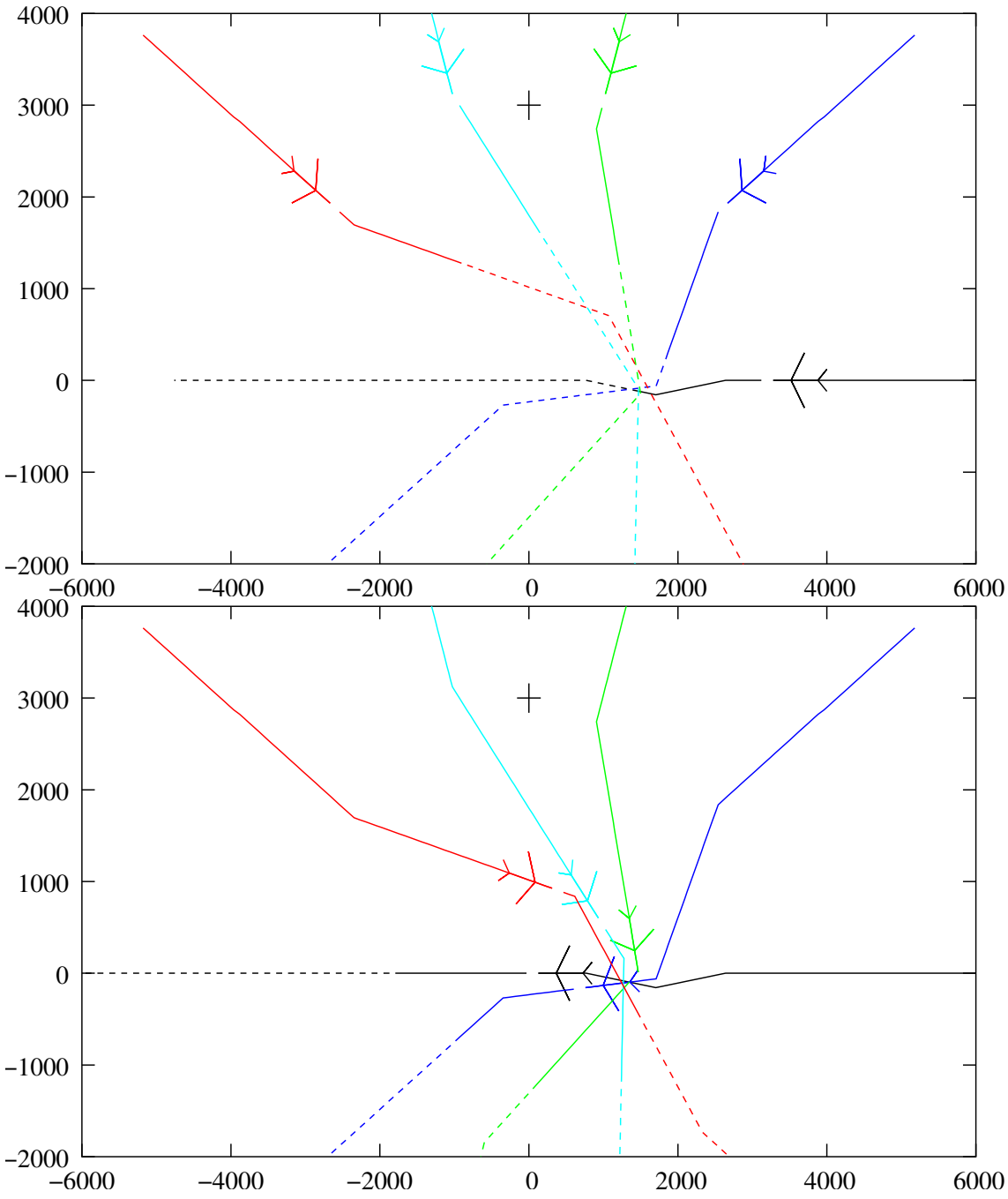

FIG. 5.6 – Résolution d'un conflit à 5 avions (l'unité de mesure est le soixantième de nautique).

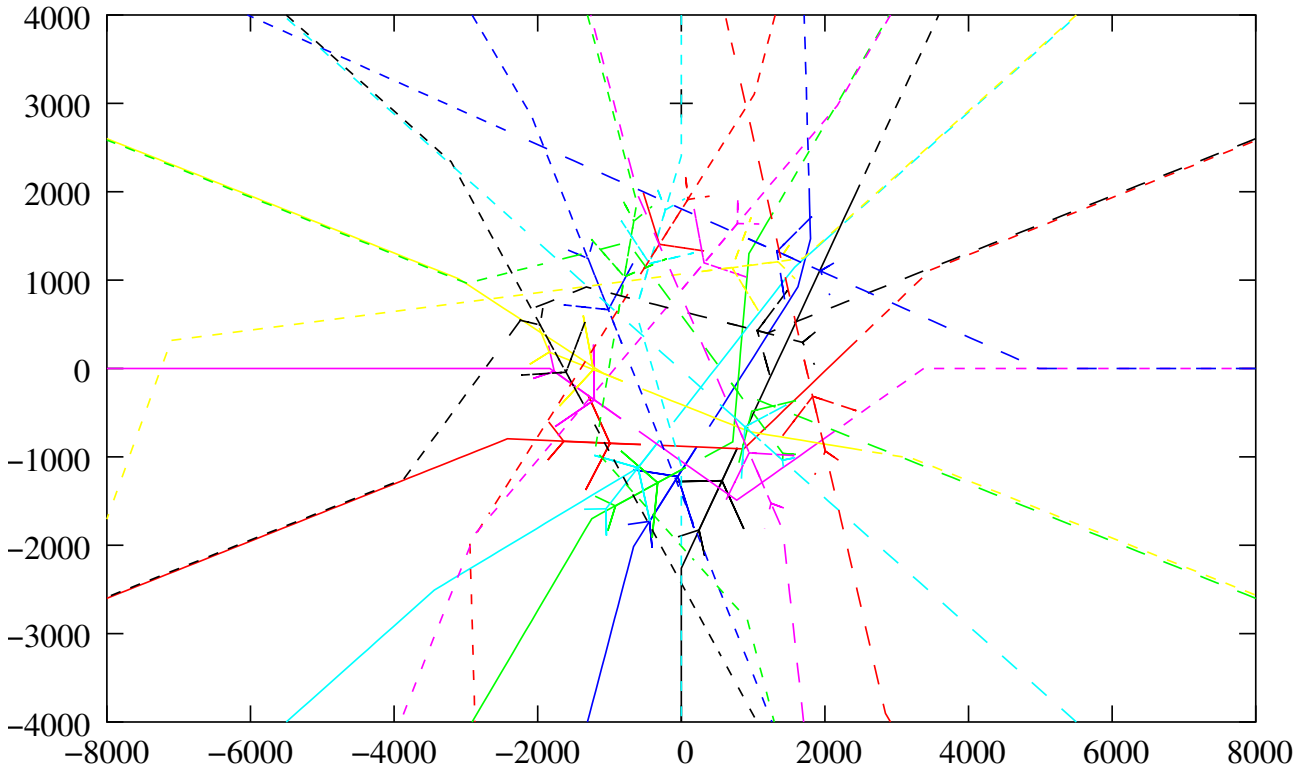

FIG. 5.7 – Résolution d'un conflit à 20 avions.

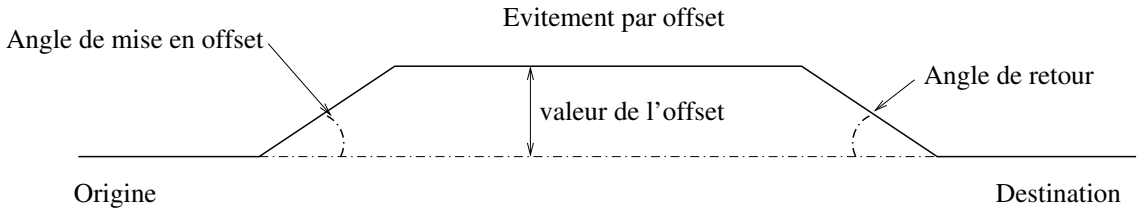

FIG. 5.8 – Méthode d'évitement par offset.

5.2 AG et programmation linéaire

5.2.1 Hypothèses

Les hypothèses retenues sont en fait plus restrictives que précédemment : on considère que les trajectoires des avions sont contenues dans un plan horizontal. Dans le cadre d'une application à des problèmes réels, ces hypothèses sont assez restrictives.

5.2.2 Choix de la modélisation

La modification de trajectoire consiste en une "mise en parallèle" ou "offset" de l'avion pendant la durée de l'évitement (voir figure 5.8).

L'angle de mise en offset comme l'angle de retour sur la trajectoire sont fixés. La durée de la seconde phase (temps pendant lequel la trajectoire de l'avion reste parallèle à sa trajectoire) n'a pas d'incidence directe sur l'allongement de la trajectoire. La modélisation de l'évitement *par offset* mène

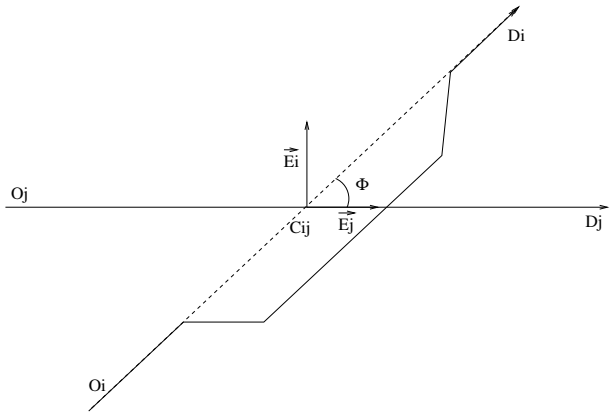

FIG. 5.9 – Conflit à deux avions, un seul avion étant dévié.

à un problème d'optimisation linéaire avec contraintes linéaires, au prix de simplifications supplémentaires que nous préciserons. Nous étendrons ce résultat au cas d'un conflit à n avions.

Condition de séparation

Soient deux avions, notés a_i et a_j, dont les positions à un instant t sont données par les couples $(x_i(t), y_i(t))$ et $(x_j(t), y_j(t))$. A l'instant t_o les avions a_i et a_j sont respectivement aux points O_i et O_j, et souhaitent atteindre à t_f les points D_i et D_j. On suppose qu'en l'absence de déviation les trajectoires des avions sont concourantes (en un point C). Les trajectoires d'évitement doivent alors satisfaire la contrainte suivante :

$$\forall t \in [t_o, t_f], \;\; (x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2 \geq d^2 \tag{5.1}$$

où d est la norme de séparation horizontale choisie.

Pour simplifier, on pose $t_o = 0$. On note $\phi_{i,j}$ l'angle formé par les trajectoires non déviées de a_i et a_j (voir figure 5.9), t_{ij}^i (resp. t_{ij}^j) l'instant auquel la trajectoire non déviée de l'avion a_i (resp. a_j) coupe celle de l'avion a_j (resp. a_i), et v_i et v_j les normes des vecteurs vitesse des avions (par hypothèse constantes sur $[t_o, t_f]$).

La contrainte est satisfaite si et seulement si :

$$v_i{}^2 v_j{}^2 \sin(\phi_{ij})^2 (t_{ij}^i - t_{ij}^j)^2 \geq d^2 (v_i{}^2 + v_j{}^2 - 2 v_i v_j \cos(\phi_{ij})) \tag{5.2}$$

Cette condition donne les deux conditions suivantes, selon l'ordre de passage des avions a_i et a_j au point de croisement C_{ij}, point d'intersection de leurs deux trajectoires :

– Si l'avion a_i passe derrière l'avion a_j, on obtient :

$$(t_{ij}^i - t_{ij}^j) v_i v_j \sin(\phi_{ij}) \geq d \sqrt{v_i{}^2 + v_j{}^2 - 2 v_i v_j \cos(\phi_{ij})} \tag{5.3}$$

– Si l'avion a_i passe devant l'avion a_j, on obtient :

$$(t_{ij}^j - t_{ij}^i) v_i v_j \sin(\phi_{ij}) \geq d \sqrt{v_i{}^2 + v_j{}^2 - 2 v_i v_j \cos(\phi_{ij})} \tag{5.4}$$

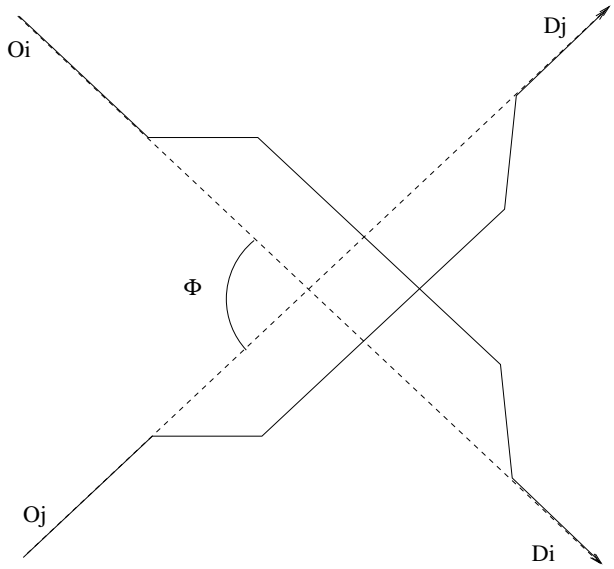

FIG. 5.10 – Conflit à deux avions, les deux offsets étant à l'extérieur.

La condition (5.2) est une condition nécessaire et suffisante de séparation des deux avions pour $t \in]-\infty, +\infty[$. Sur l'intervalle $[t_o, t_f]$ la condition de séparation (5.2) est suffisante, mais pas nécessaire. On peut réduire les conséquences de ce problème en s'assurant d'abord que le moment où les avions sont le plus proche l'un de l'autre est situé dans $[t_o, t_f]$, c'est-à-dire en n'appliquant cette méthode que dans le cas où un risque de conflit entre a_i et a_j a été repéré.

Effet des offsets

La mise en offset d'un avion (ou des deux avions a_i et a_j) a pour effet d'introduire un décalage d'une trajectoire par rapport à l'autre et d'influer ainsi sur l'instant auquel la trajectoire de chacun des deux avions coupe celle de l'autre.

Pour chacun des deux avions a_i et a_j, l'évitement par offset a trois effets sur les temps $t^i_{i,j}$ et $t^j_{i,j}$. Ces effets dépendent de l'angle de mise en offset, noté β, du sens de l'offset, et de sa valeur. Ainsi un offset de l'avion a_i d'une valeur d_i aura les effets suivants :

- t^i_{ij} est augmenté, quelque soit le sens de l'offset, du retard dû à la mise en offset, c'est-à-dire de $\frac{d_i \tan(\frac{\beta}{2})}{v_i}$;
- si la trajectoire de l'avion a_i est déviée vers l'extérieur de l'angle formé par les trajectoires des deux avions (a_i et a_j), t^i_{ij} est augmenté du retard dû au déplacement de l'avion a_i, c'est-à-dire de $\frac{d_i \cot(\phi_{ij})}{v_i}$. Si la trajectoire de a_i est déviée vers l'intérieur de cet angle, t^i_{ij} est diminué de cette valeur ;
- de même, si la trajectoire de l'avion a_i est déviée vers l'extérieur de l'angle formé par les trajectoires des deux avions (a_i et a_j), t^j_{ij} est aussi augmenté du retard dû au déplacement de la trajectoire de l'avion a_j, c'est-à-dire de $\frac{d_i}{v_j \sin(\phi_{ij})}$.

Ainsi, quand l'avion a_i passe derrière l'avion a_j, selon le sens de l'offset de chacun des deux avions a_i et a_j on obtient quatre conditions de séparation linéaires en d_i et en d_j : par exemple, si les

offsets des deux avions sont à l'extérieur (c'est le cas qui est représenté sur la figure 5.10) :

$$\begin{aligned}(t_{ij}^i + \frac{d_i \tan(\frac{\beta}{2})}{v_i} + \frac{d_i \cot(\phi_{ij})}{v_i} + \frac{d_j}{v_i \sin(\phi_{ij})} \\ -t_{ij}^j - \frac{d_j \tan(\frac{\beta}{2})}{v_j} - \frac{d_j \cot(\phi_{ij})}{v_j} - \frac{d_i}{v_j \sin(\phi_{ij})})v_i v_j \sin(\phi_{ij}) \\ -d\sqrt{{v_i}^2 + {v_j}^2 - 2v_i v_j \cos\phi_{ij}} \quad \geq \quad 0\end{aligned}$$

On obtient quatre autres équations linéaires dans le cas où l'avion a_i passe devant l'avion a_j, ce qui représente donc 8 équations linéaires par paire d'avions.

On a considéré jusqu'ici que les droites portant les trajectoires non déviées des avions a_i et a_j étaient sécantes (et, comme on l'a vu, que les trajectoires des avions se coupaient pour $t \in [t_o, t_f]$). Dans le cas où les trajectoires sont parallèles, on obtient aussi des conditions linéaires, avec la même combinatoire, la discussion sur l'ordre de passage des deux avions est alors remplacée par le choix entre "a_i passe à gauche de a_j" et "a_j passe à gauche de a_i".

Les conditions ci-dessus ne tiennent pas compte de la séparation des avions pendant les phases de mise en offset et de retour sur leurs trajectoires d'origine. Les conditions de séparation pendant ces phases ne sont, elles, pas linéaires, on n'en tient pas compte dans la suite de cet exposé.

Linéarité du problème

Une fois fixé le sens de l'offset pour chaque avion, et pour chaque paire d'avions, l'ordre de passage de ces avions l'un par rapport à l'autre, les contraintes du problème à résoudre sont alors les suivantes :

- Les contraintes de séparation des avions deux à deux, ce qui donne $\frac{n(n-1)}{2}$ contraintes.
- Les contraintes traduisant la positivité des valeurs des offsets ($\forall i \in [1..n],\ d_i \geq 0$).

ce qui représente pour n avions, $\frac{n(n+1)}{2}$ contraintes. (En fait, on pourra avoir n contraintes supplémentaires pour borner la valeur des offsets). La fonction à minimiser est la somme des retards pour les n avions. Ainsi, pour l'avion a_i, le retard entraîné par un offset de valeur d_i est égal à :

$$R(d_i, v_i) = 2\frac{d_i \tan(\frac{\beta}{2})}{v_i} \tag{5.5}$$

La somme des retards pour tous les avions est donc égale à :

$$S(d_1, \ldots, d_n) \quad = \quad 2\sum_{i=1}^{n} \frac{d_i \tan(\frac{\beta}{2})}{v_i} \tag{5.6}$$

Cette fonction est linéaire en d_i pour $1 \leq i \leq n$.

Ainsi, pour une configuration donnée, on doit résoudre un problème d'optimisation linéaire à n inconnues (les valeurs des offsets des n avions), et à $\frac{n(n+1)}{2}$ contraintes (ou à $\frac{n(n+2)}{2}$ contraintes, si on borne les valeurs des offsets). Ce problème peut se résoudre facilement, à l'aide par exemple d'un algorithme du type *simplexe*. Cependant on obtiendra alors au mieux un optimum local dépendant de la configuration initiale. On peut n'obtenir aucun résultat dans certains cas (contraintes trop fortes). Le problème combinatoire peut être résolu par un algorithme génétique, chargé de déterminer la configuration optimale, un algorithme de programmation linéaire du type *simplexe* servant à résoudre le problème linéaire pour chaque configuration du problème.

5.2.3 Mise en œuvre de l'algorithme génétique

Un algorithme génétique est utilisé pour parcourir l'ensemble des configurations du problème, chacune d'entre elles menant à un problème d'optimisation linéaire. Un élément de population représente donc une configuration du problème. Cet élément de population est d'autant mieux adapté qu'il permet, par l'intermédiaire d'un programme d'optimisation linéaire de conduire à une solution résolvant tous les conflits avec un retard minimal.

Codage des données du problème

On utilise un codage de données binaire. Pour un problème à n avions, les n premiers bits codent les sens des offsets de chaque avion. Les $\frac{n(n-1)}{2}$ bits suivants codent le sens de passage des avions pour chaque couple d'avions. Ainsi, chaque élément de population est constitué de $\frac{n(n+1)}{2}$ bits.

Génération aléatoire d'une population

Pour engendrer aléatoirement une population initiale de N éléments, on procède successivement à N tirages aléatoires d'un entier de $\frac{n(n+1)}{2}$ bits. L'expression binaire de chaque entier ainsi engendré représente une configuration du problème.

Croisement

Pour un chromosome de taille k, on utilise un croisement à k points.

Mutation

Un des bits de la séquence est modifié aléatoirement.

Distance utilisée pour le *sharing*

La distance entre deux éléments de la population doit traduire une notion de distance sur l'espace des données, à savoir une distance entre les configurations. Intuitivement, on peut considérer que deux configurations sont d'autant plus éloignées l'une de l'autre, qu'un nombre plus important d'avions ont des sens de déviation différents, ou qu'un plus grand nombre de couples d'avions correspondent à des ordres de passage différents.

Pour traduire cette idée, on peut retenir la *distance de Hamming*.

Fonction d'évaluation

Le critère d'évaluation d'une configuration est le retard minimum obtenu après résolution du problème linéaire.

Ainsi, pour l'évaluation d'un élément de population, le programmme d'optimisation linéaire est exécuté, avec les données de la configuration codée. La fonction *fitness* est alors une fonction décroissante du retard obtenu.

Certaines configurations mènent à des problèmes d'optimisation *infaisables*, c'est-à-dire pour lesquels les contraintes ne peuvent pas être toutes satisfaites simultanément. Prendre une *fitness* nulle pour les éléments de la population correspondant à de telles configurations n'est pas satisfaisant :

une configuration menant à un problème infaisable peut être mieux *adaptée* qu'une autre, en ce sens qu'elle peut être plus proche qu'elle d'une configuration menant à un problème faisable.

Ce point est délicat. L'idéal pour appliquer à ce problème un algorithme génétique serait de pouvoir estimer précisément l'adéquation des configurations menant à des problèmes infaisables. Une telle estimation est bien sûr très difficile. On peut cependant procéder de la manière suivante :

si une configuration mène à un problème d'optimisation infaisable, on supprime une contrainte, si les contraintes restantes ne peuvent toujours pas être satisfaites simultanément, on en supprime une seconde, et ainsi de suite, jusqu'à ce que les contraintes restantes soient simultanément satisfaites.

Cette méthode se base sur l'idée intuitive selon laquelle un problème d'optimisation infaisable est d'autant plus "proche" d'un problème faisable qu'il y a peu de contraintes à supprimer pour que les contraintes restantes puissent être toutes simultanément satisfaites. Ainsi, on peut calculer la *fitness* d'un élément x_j de la population correspondant à une configuration menant à un problème infaisable de la manière suivante :

$$fit(x_j) = f_S(S(x_j)) \tag{5.7}$$

où S_j est le nombre de contraintes qu'on a dû supprimer avant d'obtenir un problème faisable, en partant de la configuration correspondant à x_j, et f_S est une fonction décroissante de $\mathbf{N}$ dans $\mathbf{R}$. On peut envisager plusieurs façons de choisir les contraintes que l'on supprime et dans quel ordre. Le lecteur intéressé par ce problème se reportera au rapport de DEA de F. Médioni [MDA94].

5.2.4 Conflit à 5 avions

Dans le cas d'un conflit mettant en jeu 5 avions, il est encore possible de traiter par un programme d'optimisation linéaire de manière systématique et déterministe toutes les combinaisons possibles de sens de déviation et d'ordre de passage (il y en a 32 768), et de comparer tous les résultats obtenus, afin de déterminer la meilleure solution.

Nous considérons pour cela l'exemple suivant : les avions vont à la même vitesse (400 kts), ils sont au temps t_o régulièrement répartis sur un demi cercle de centre C (et de rayon 100 Nm), et leurs trajectoires se coupent en C (voir figure 5.11). La norme de séparation est de 7 Nm mais aucune incertitude n'est prise en compte sur la vitesse des avions.

On obtient alors deux solutions optimales équivalentes : soit les quatre premiers avions sont déviés vers la droite, tandis que le cinquième n'est pas dévié, et si $i < j$, a_i passe derrière a_j, soit le premier avion n'est pas dévié, et les quatre suivants sont déviés vers la gauche, et si $i < j$, a_i passe devant a_j. C'est cette dernière solution qui est représentée sur la figure 5.11.

L'algorithme utilise une population de 150 individus sur moins d'une centaine de générations pour trouver l'optimum (Test effectué 50 fois).

5.2.5 Conflit à 6 avions

La figure 5.12 illustre un cas de conflit à 6 avions similaire au cas précédent : les avions ont la même vitesse (400 kts), ils sont au temps t_o régulièrement répartis sur un demi-cercle de centre C (et de rayon 100 Nm), et leurs trajectoires se coupent en C (voir figure 5.12). Les meilleures solutions obtenues par cette méthode sont alors comparables aux solutions optimales pour le conflit à 5 avions de la section précédente. Comme pour l'exemple à 5 avions, le programme est lancé sur cet exemple à 6 avions 50 fois, en changeant à chaque fois l'initialisation des fonctions aléatoires. Sur les 50 essais, une des deux *meilleures solutions* est obtenue 39 fois seulement. Un parcours exhaustif de tous les cas de figure nécessiterait, 2 097 152 optimisations du programme linéaire.

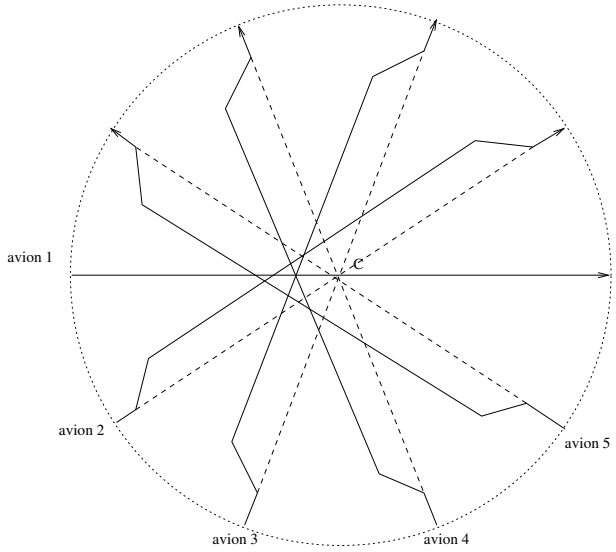

FIG. 5.11 – Trajectoires d'évitement optimales pour un conflit à 5 avions.

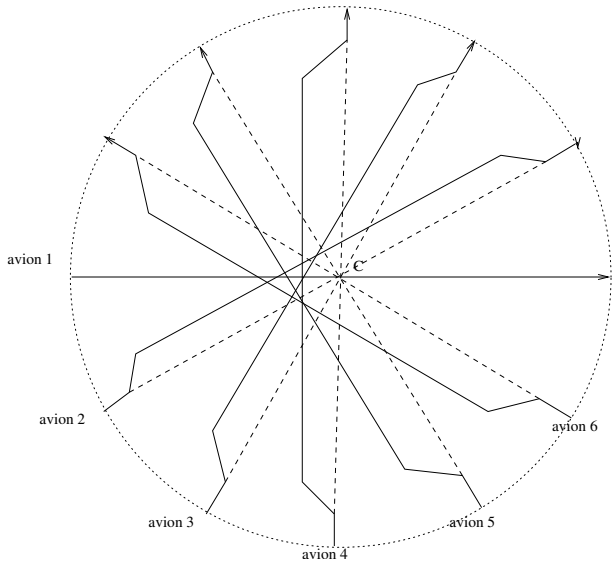

FIG. 5.12 – Meilleures trajectoires d'évitement obtenues pour un conflit à 6 avions.

Cette modélisation ne permet pas d'aborder des problèmes de très grande taille (une vingtaine d'avions) car le coût d'évaluation d'une configuration devient trop important mais elle a été abandonnée pour d'autres raisons :

- Cette modélisation ne permet pas de prendre en compte une prévision de trajectoire réaliste utilisant un modèle d'incertitude. De plus, dans les phases de montée et de descente, les trajectoires ne peuvent pas être modélisées par des droites.
- Il faut s'assurer qu'aucun conflit n'intervient pendant les phases de mise en offset ou de retour sur trajectoire, ce qui complique énormément le problème dans les zones denses.

5.3 *Branch and Bound* par intervalles

On peut calculer l'ensemble des positions possibles d'un avion à tout instant t donné lorsque les *instants de début et de fin de manœuvre* t_0 et t_1 prennent leurs valeurs à l'intérieur d'*intervalles de manœuvres*. 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.

Ceci permet d'appliquer au problème de l'évitement des algorithmes d'optimisation de type *Branch and Bound* utilisant des méthodes d'intervalles. Nous présentons deux approches (l'une globale, l'autre séquentielle) utilisant ces méthodes sur le problème de résolution de conflit.

5.3.1 L'analyse d'intervalles

Cette section a pour but d'introduire brièvement les rudiments de l'analyse d'intervalles nécessaires à la compréhension de ce paragraphe. 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.

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.

On appelle $I(\mathbb{R})$, 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 $\mathbb{R}$. On note x_{max} et x_{min} les bornes inférieures et supérieures de l'intervalle X de $I(\mathbb{R})$.

Définition 2 (Largeur d'un intervalle)**.** *La largeur $l(X)$ d'un intervalle $X = [x_{min}, x_{max}]$ est définie par $l(X) = x_{max} - x_{min}$.*

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

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

Définition 3 (Point milieu d'un intervalle)**.** *Le point milieu $l(X)$ d'un intervalle $X = [x_{min}, x_{max}]$ est :*

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

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

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

Définition 4 (Intervalle dégénéré)**.** *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 $\mathbb{R}^n$ $(x_1, .., x_n)$;

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

$$X_1 \; op \; X_2 \quad = \quad \{y = x_1 \; op \; x_2; \; x_1 \; \in X_1 \; et \; x_2 \; \in X_2\} \tag{5.8}$$

Cette définition s'applique directement pour l'addition, la soustraction et la multiplication, la division lorsque $0 \notin X_2$. Hanson [Han68] et Kahan [Kah68] ont introduit 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 autres, de définir la division par un intervalle contenant 0. On définit de même à partir de (5.8) 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 :

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

Définition 5 (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'est-à-dire 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 6 (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 :*

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

Définition 7 (Monotonie pour l'inclusion)**.** *Une fonction F de $I(\mathbb{R}^n)$ dans $I(\mathbb{R}^m)$ est dite monotone pour l'inclusion si :*

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

Théorème 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 $I(\mathbb{R}^n)$.*

La démonstration découle immédiatement des définitions 6 et 7.

A une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}^m$ ne correspond pas une unique fonction d'inclusion pour f sur $I(\mathbb{R}^n)$.

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

Définition 8 (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 α, ou d'ordre de convergence α 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 9 (Fonction d'inclusion K-lipschitzienne)**.** *Avec les mêmes notations que pour la définition précédente (8), 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].

5.3.2 *Branch and bound* par intervalles

Dans ce paragraphe, on rappelle le principe général d'un algorithme de type *branch and bound* basé sur des méthodes d'intervalles tel qu'il est présenté dans [RR95]. Décrivons tout d'abord 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 contrainte :

On cherche à minimiser une fonction de $D \in I(\mathbb{R}^n)$ dans $\mathbb{R}$).

L'algorithme utilise :

- Une séquence ordonnée d'intervalles (une file de priorité) notée $\mathcal{F}_{int}$, dans laquelle les intervalles sont ordonnés selon un critère qui pourra dépendre du problème à résoudre.
- Une fonction d'inclusion de f sur D, notée F. 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 atteinte sur l'intervalle X, notée $f_{est}(X)$.
  Ainsi, si $F(X) = [f_{min}(X), f_{max}(X)]$ est l'image de X par F, on a sur X :

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

- 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 intervalles $[a, c]$ et $[c, b]$, où c est le milieu de $[a, b]$. En dimension $n > 1$, on coupe en deux l'intervalle X_i de $X = X_1 \times .. \times X_n$ qui a la plus grande largeur.
- 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 insère a des conséquences importantes sur la rapidité de l'algorithme, puisque celui-ci traite ensuite systématiquement le premier élément de $\mathcal{F}_{int}$.
- Un critère selon lequel un intervalle X de $\mathcal{F}_{int}$ passe ou non dans une autre file, qui sera appelée *file-résultat* et notée $\mathcal{F}_{res}$. On peut, par exemple, s'arrêter lorsque la taille de l'intervalle X est

inférieure à une valeur donnée, et que la borne inférieure de $F(X)$ est inférieure à la plus petite valeur de la fonction f calculée par l'algorithme (le plus petit estimateur déjà rencontré par l'algorithme, qui est 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 pas le critère c_r, on coupe X en deux intervalles X_1 et X_2. Les quatre étapes suivantes seront menées pour $j = 1$ et $j = 2$.*
      i. *calcul d'un estimateur $f_{est}(X_j)$ et $F(X_j)$.*
      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 conservera d'autant moins d'intervalles inutiles dans $\mathcal{F}_{int}$ que les bornes inférieures des images intervalles par la fonction d'inclusion F seront proches de celles de leurs images par la fonction f. Il est donc essentiel de disposer de la meilleure fonction d'inclusion possible.

Le traitement des contraintes utilise la définition suivante :

Définition 10 (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'intervalle de départ D est indéterminé, et de taille supérieure à ϵ.

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 :

$$\bigcup_{X_i \in \mathcal{L}_s} X_i \subset D_f \subset \bigcup_{X_i \in \mathcal{L}_s} X_i \cup \bigcup_{X_j \in \mathcal{F}_{ind}} X_j \tag{5.9}$$

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 inférieure à un réel $\epsilon > 0$, ou que la somme des largeurs des éléments de $\mathcal{F}_{ind}$ est inférieure à un $\epsilon > 0$.

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 $\mathcal{F}_{ind}$ peuvent 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 caractérise, comme on le verra dans la suite, 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 contrainte 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.

5.3.3 Problème à résoudre

On se limite à nouveau à la résolution dans le plan horizontal, avec des avions volant à une vitesse constante sans incertitude.

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 prennent place à l'intérieur d'un intervalle de temps.

Modélisation des trajectoires d'évitement

La trajectoire d'évitement utilisée est illustrée par la figure 5.13 et 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 α 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 α 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 5.3.4). 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. L'avion 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.

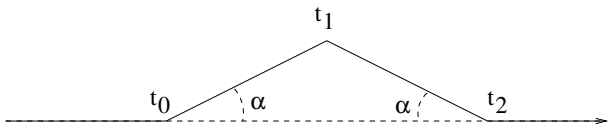

FIG. 5.13 – Modélisation d'une trajectoire d'évitement par point tournant.

Ainsi définie, la trajectoire d'évitement d'un avion, l'angle α 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 ($t_2 = 2\,t_1 - t_0$). On vérifiera que $t_1 \geq t_0$. Le cas limite $t_1 = t_0$ correspond à une trajectoire non déviée. t_0 et t_1 sont remplacés par des intervalles, l'angle α prend des valeurs discrètes, selon le sens de la déviation. Le choix du sens de déviation est intégré à l'algorithme d'optimisation.

Position de l'avion à un temps t

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. 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. On distingue quatre cas, correspondant aux quatre étapes de la trajectoire d'évitement :

1. $t \leq t_0$. On a alors :
$$\begin{cases} x(t) = vt \\ y(t) = 0 \end{cases} \tag{5.10}$$
L'avion n'a pas encore été dévié.

2. $t_0 \leq t \leq t_1$. On a alors :
$$\begin{cases} x(t) = v\,(t_0(1 - \cos\alpha) + t\cos\alpha) \\ y(t) = (t - t_0)\sin\alpha \end{cases} \tag{5.11}$$
L'avion a été dévié, et s'écarte de sa trajectoire d'origine, avec l'angle α.

3. $t_1 \leq t \leq t_2$. On a alors :
$$\begin{cases} x(t) = v\,(t_0(1 - \cos\alpha) + t\cos\alpha) \\ y(t) = (2t_1 - t_0 - t)\sin\alpha \end{cases} \tag{5.12}$$
L'avion revient sur sa trajectoire d'origine, avec un angle de retour égal à α.

4. $t \geq t_2$. On a alors :
$$\begin{cases} x(t) = v\,(t - 2(t1 - t_0)(1 - \cos\alpha)) \\ y(t) = 0 \end{cases} \tag{5.13}$$
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 :

$$r(t_0, t_1) = 2(t_1 - t_0)(1 - \cos\alpha) \tag{5.14}$$

5.3.4 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 $T_1 = [t_{1min}, t_{1max}]$). On peut calculer l'ensemble des positions possibles d'un avion à un instant t quand ses instants de manœuvres sont donnés par de tels intervalles. L'ensemble des positions possibles à un instant t pour l'avion est noté $P(t)$.

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

Pour savoir quelle équation utiliser, parmi les équations 5.10 à 5.13, 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 ne pas avoir quitté sa trajectoire d'origine si et seulement si :

$$t \leq t_{0max} \tag{5.15}$$

Deuxième étape : l'avion peut être en train de s'éloigner de sa trajectoire d'origine si et seulement si :

$$t_{0min} \leq t \leq t_{1max} \tag{5.16}$$

Troisième étape : l'avion peut être en train de revenir vers sa trajectoire d'origine si et seulement si

$$t_{1min} \leq t \leq 2t_{1max} - 2t_{0min} \tag{5.17}$$

Quatrième étape : l'avion peut être revenu sur sa trajectoire d'origine si et seulement si :

$$t \geq \max(t_{1min}, 2t_{1min} - t_{0max}) \tag{5.18}$$

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. On suppose que l'instant t est fixé, ainsi que les deux intervalles $T_0 = [t_{0min}, t_{0max}]$, et $T_1 = [t_{1min}, t_{1max}]$.

L'ensemble $P(t)$ est construit de la manière suivante :

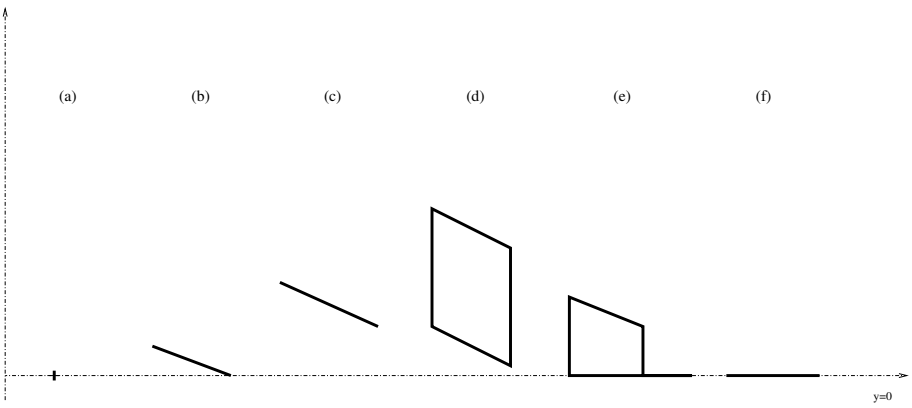

FIG. 5.14 – *Différentes formes possibles pour l'ensemble P(t).*

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 5.14.

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. C'est le cas (c) de la figure 5.14.
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). Le cas (d) de la figure 5.14 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 5.14.

Pour mesurer la distance entre les avions, on discrétise l'intervalle de temps sur lequel est menée l'optimisation, et on minimise, sur l'ensemble des pas de temps, les séparations minimales et maximales entre les positions possibles des avions correspondant aux intervalles de manœuvres à chaque pas de temps. On peut alors déterminer si la violation de la contrainte est certaine (dans ce cas, l'intervalle n'est pas admissible), si elle est impossible, (dans ce cas, l'intervalle est dit admissible, et la vérification ne sera plus à faire) ou si elle est indéterminée (il faudra alors de nouveau vérifier la contrainte après le prochain découpage de l'intervalle de manœuvres en deux).

Pour des intervalles de manœuvres T_0 et T_1, l'ensemble des retards possibles est un intervalle. On note

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

et

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

Pour un avion, l'ensemble des valeurs possibles de $r(t_0, t_1)$ est donné par l'intervalle $R(T_0, T_1)$:

$$R(T_0, T_1) = [r_{min}(T_0, T_1), r_{max}(T_0, T_1)] \tag{5.19}$$

Application de l'algorithme d'optimisation

Deux approches différentes du problème ont été testées, l'approche *globale* et l'approche *séquentielle*. Dans l'approche globale, les trajectoires de tous les avions impliqués dans un *cluster* sont optimisées simultanément. Dans le cadre de l'approche séquentielle, elles sont optimisées l'une après l'autre, chaque optimisation tenant compte des trajectoires précédemment optimisées.

Variables de l'optimisation.

Pour un problème à n avions, traité dans le cadre de l'approche globale, l'espace de recherche est :

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

On cherche 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 $t_0 \leq t_1$:

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

A chaque étape de partage de l'intervalle en deux, on vérifie que la contrainte $t^i_{0min} \leq t^i_{1max}$ est respectée pour tout i, faute de quoi l'intervalle est supprimé. De même, on doit vérifier que $t^i_{0min} \leq t^i_{1min}$ et $t^i_{0max} \leq t^i_{1max}$. Si tel n'est pas le cas, on réduit la taille des intervalles.

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 séparations 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 $t_r = \big((t^1_0, t^1_1, \alpha^1), .., (t^n_0, t^n_1, \alpha^n)\big) \in D$.

- Si t_r respecte les contraintes de séparation, la fonction f^n_{opt} traduit le retard entraîné par les trajectoires :

$$f^n_{opt}(t_r) = \sum_{i=1}^{n} t^i_1 - t^i_0 \tag{5.20}$$

- Si t_r ne respecte pas les contraintes de séparation,

$$f^n_{opt}(t_r) = G_r - \delta_{min}(t_r) \tag{5.21}$$

où δ_{min} est la distance minimale possible entre deux avions. En fonction des données du problème, on choisit G_r suffisamment grand pour que les valeurs de f^n_{opt} sur les points ne respectant pas les contraintes de séparation soient plus grandes que ses valeurs sur les points les respectant.

5.3.5 Déroulement de l'algorithme

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

Les intervalles de cette file sont successivement extraits, coupés en deux nouveaux intervalles et, pour chacun des deux intervalles ainsi produits, sont calculés un intervalle image et un estimateur.

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, soit insérés dans la liste f^{sat}_{res}, 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 séparation 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 $r_{min}T$).

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.

Elagage

L'approche globale mène à des temps de calcul très importants dès que le nombre d'avions dépasse deux. C'est pourquoi on utilise une technique d'élagage.

Avec l'élagage, 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 s'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 :

$$i_{min}(T) \leq est_0 - c_e \tag{5.22}$$

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 séparation et un point ne les respectant pas sera toujours supérieure à c_e.

5.3.6 Approche globale

La méthode décrite dans ce chapitre traite assez rapidement divers cas de conflit à deux avions. Les cas de conflit à trois avions mènent à des problèmes beaucoup plus grands. Supposons que nous cherchons, après avoir découpé l'ensemble sur lequel est menée la recherche d'intervalles de manœuvres pour les 3 avions ($E_{opt}^{3} = ([0, t_f] \times [0, t_f] \times \{\alpha, -\alpha\})^3$) en intervalles de manœuvres de largeur ϵ_{res}^{sat}, associés aux différents sens de déviations possibles pour chaque avion, le meilleur de ces intervalles par une recherche exhaustive.

Supposons que l'on ait $t_f = 800\ s$ et $\epsilon_{res}^{sat} = 10\ s$. On a donc $n_{int} = 80$ intervalles de taille ϵ_{res}^{sat} dans $[0, t_f]$. Pour chaque avion, pour un sens de déviation donné, compte tenu du fait que les temps de manœuvres t_0^i et t_1^i doivent respecter l'inégalité $t_0^i \leq t_1^i$, 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_{opt}^{3}, chacun des 3 avions pouvant prendre deux sens de déviation, il faudrait donc $N_{int} = (n_{int}(n_{int}+1))^3$ éléments de I_{opt}^{3}, c'est-à-dire, avec les valeurs prises pour ϵ_{res}^{sat} 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_{opt}^{n} de l'ordre de la dizaine de milliards.

Nous présentons ici la méthode globale sur un cas-test de conflit à trois 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 kts.

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_{res}^{sat} est ϵ_{res}^{sat} = 10 secondes ;
- La taille maximale des intervalles insérés dans f_{res}^{ind} est ϵ_{res}^{ind} = 5 secondes.
- La norme de séparation est $n_h = 5NM$.

- Le pas de temps utilisé pour le calcul des distances entre avions au cours de l'évitement est de 10 secondes.

La solution montrée sur la figure 5.15 a été obtenue après avoir calculé les distances 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 trois avions, à un retard de 9 secondes. Le cas-test de conflit présenté ici étant symétrique, il n'est pas surprenant que les trois trajectoires proposées pour les trois avions soient identiques, à une rotation près.

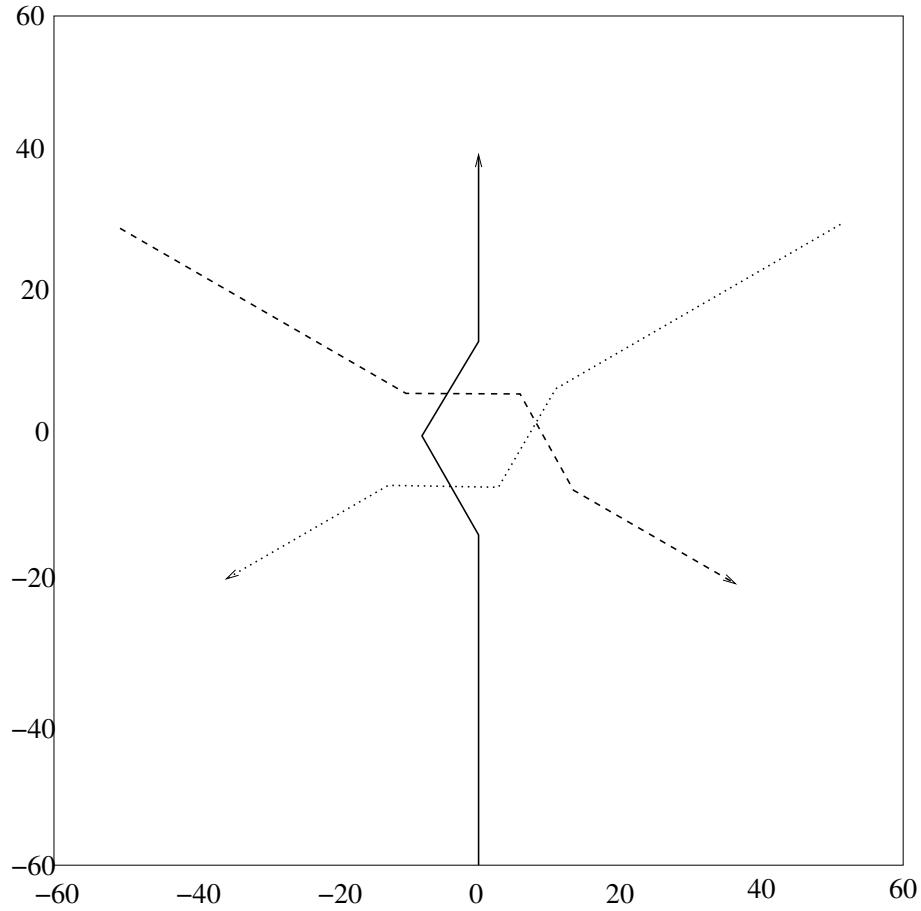

FIG. 5.15 – Conflit à trois avions, évitement par point tournant.

Ce résultat, très satisfaisant en terme de qualité, a cependant nécessité un temps de calcul très important, qui peut être réduit en utilisant la technique d'élagage décrite précédemment.

Les résultats présentés dans le tableau 5.1 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 trois avions, par la figure 5.15 : au temps $t = 0$, les avions sont régulièrement répartis sur un centre de rayon $28\ NM$ et volent à la vitesse de $420\ kts$, en convergeant vers le centre.

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, partant, le temps de calcul. Les détériorations observées en terme de qualité

TAB. 5.1 – *Différentes coupures d'élagage pour différents nombres d'avions*

Nombre d'avions	Coupure d'élagage	Retard moyen	Nombre d'éléments de I^n_{opt} traités
3	30 s	22 s	200
3	15 s	20 s	13 000
3	0 s	16 s	500 000
4	45 s	42 s	350
4	30 s	31 s	45 000
5	75 s	66 s	160
5	45 s	39 s	116 000
6	120 s	76 s	90
6	90 s	58 s	500
6	60 s	49 s	160 000
7	120 s	102 s	1 600
7	90 s	88 s	3 900

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.

5.3.7 Approche séquentielle

Comme on l'a vu précédemment, l'approche globale atteint très rapidement ses limites. 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 engendre, pour chaque avion tour à tour, en suivant cet ordre de priorité, une trajectoire d'évitement permettant à l'avion de rester séparé des avions qui le précèdent dans l'ordre de priorité.

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 engendrent leurs trajectoires.

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

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 ϵ^{sat}_{res} = 10 secondes.
- La taille maximale des intervalles insérés dans f^{ind}_{res} est ϵ^{ind}_{res} = 5 secondes.
- La norme de séparation est $n_h = 5NM$.
- 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 $n_h = 5,52NM$.

Nous avons traité ce cas de conflit test avec la méthode séquentielle, pour les 120 combinaisons possibles d'ordres 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 obtient donc en fait, sur les 120 combinaisons, 12 solutions différentes, les autres se déduisant de celles-ci par rotations et symétries.

Sur les 120 ordres de priorité possibles, 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. A 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 5.17 montre les trajectoires obtenues avec l'ordre de priorité $(3, 1, 4, 5, 2)$, pour lequel le dernier avion ne peut pas passer, et 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.

Les ordres de priorité permettant à chacun des cinq avions d'adopter une trajectoire sans conflit mènent cependant à des solutions très différentes en terme de qualité : le retard moyen sur les cinq 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 5.18 montre la solution ayant un retard minimal (ordre $(1, 2, 3, 4, 5)$).

5.3.8 Conclusion

Les méthodes de *branch and bound* par intervalles ne semblent pas très bien adaptées au problème de résolution de conflits pour plusieurs raisons :

- Elles supposent de simplifier le modèle de prévision de trajectoires afin de pouvoir en donner une expression analytique. Dans le cadre d'une résolution dans le plan horizontal, cela peut être envisageable sur un horizon temporel réduit. Si l'on veut tenir compte des profils de montée et de descente des avions, cela ne paraît pas réaliste.
- Les méthodes de *branch and bound* par intervalles ne permettent pas de résoudre des problèmes de grande taille et contraignent l'utilisateur à envisager une approche séquentielle, sous-

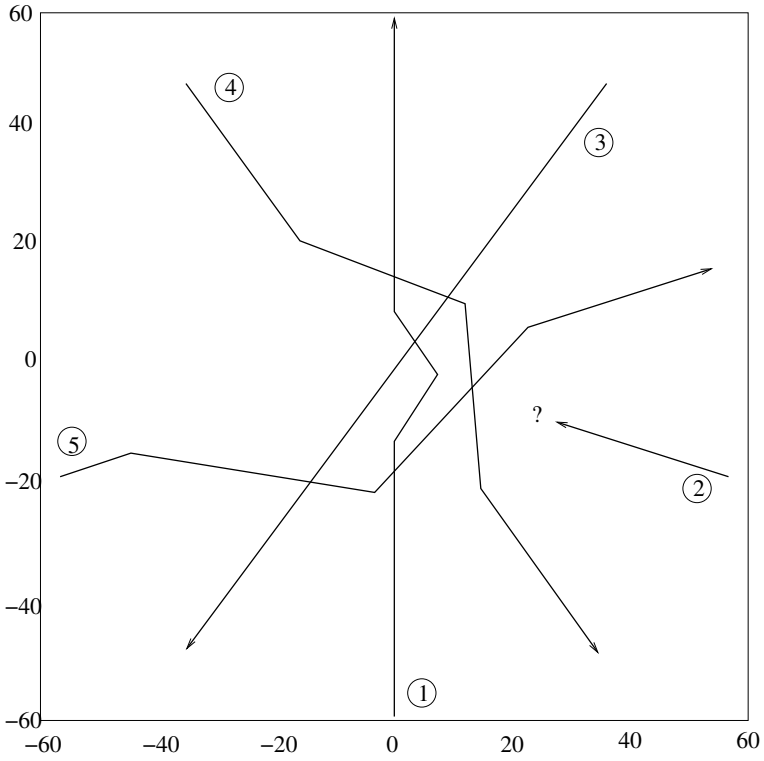

FIG. 5.16 – Ordre de priorité $(3, 1, 4, 5, 2)$ (l'unité est le mille nautique).

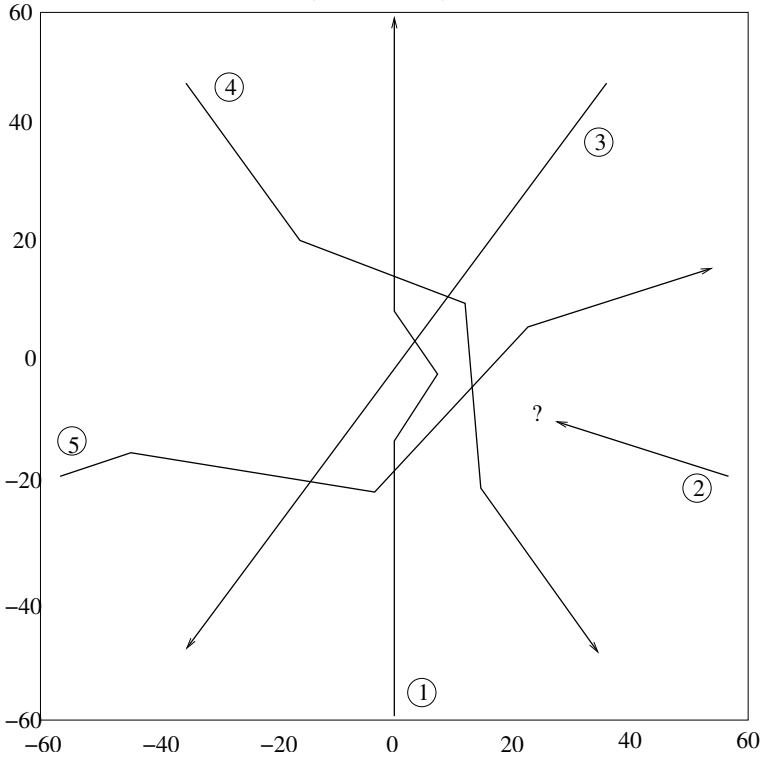

FIG. 5.17 – Ordre de priorité $(3, 1, 4, 2, 5)$.

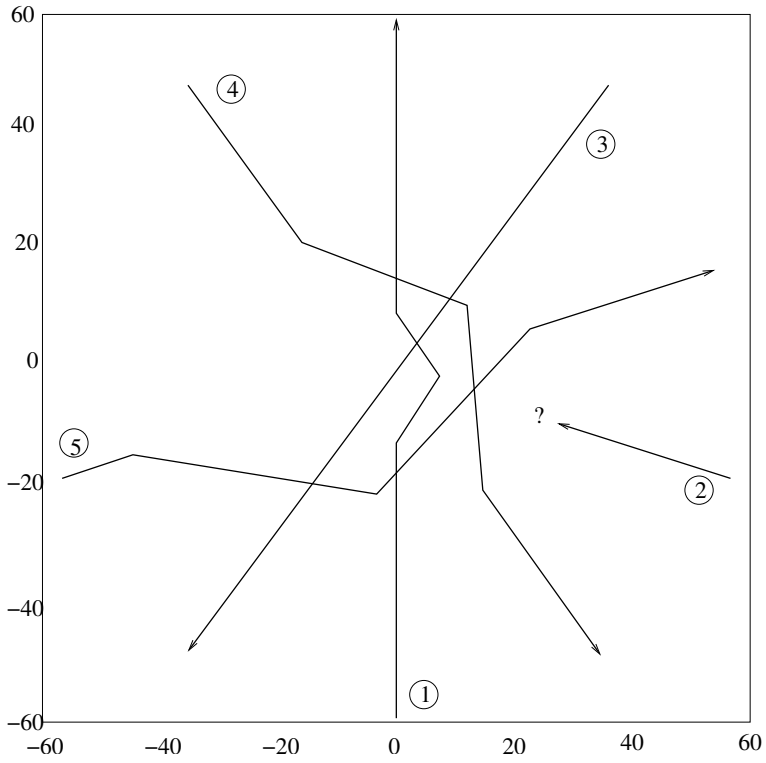

FIG. 5.18 – Ordre de priorité $(1, 2, 3, 4, 5)$.

optimale, et dont la qualité du résultat dépend largement de la stratégie d'ordonnancement des avions.

5.4 Programmation semi-définie (SDP)

On présente, dans cette partie, une méthode proposée par Eric Féron, E. Frazzoli et Z.-H. Mao [FMF01] permettant de s'affranchir du problème combinatoire en effectuant une relaxation convexe des contraintes de séparation. A partir de la solution obtenue, on peut espérer déterminer des règles de dépassement entre les avions. L'utilisation d'une méthode d'optimisation convexe sous contraintes convexes permet alors de déterminer des manœuvres de résolution. Cette méthode a été testée numériquement au cours du stage de DEA de Pierre Dodin [Dod99].

On présente, dans un premier temps, la modélisation utilisée, puis les méthodes d'optimisation convexe intervenant dans la résolution du problème (BFGS, EMM[4]), et ensuite une méthode de relaxation de problèmes quadratiques par SDP). Enfin, l'application à notre problème de résolution de conflits est détaillée.

5.4.1 Le modèle

Considérons n avions dans un plan horizontal libres de tout conflit à $t = 0$. Soient p_{k0} et v_{k0} respectivement la position et la vitesse de l'avion k à $t = 0$. Pour tout vecteur x (position ou vitesse),

[4]Méthode des Multiplicateurs Exponentiels

on notera $x_{ij} = x_i - x_j$. La position relative des avions i et j est $V_{ij0}t + p_{ij0}$. Et, par minimisation de carrés, la distance de séparation minimum entre deux avions est

$$d_{ij} = \left\{ \begin{array}{cc} \frac{\|p_{ij0} \wedge v_{ij0}\|}{\|v_{ij0}\|} & \text{si } \langle v_{ij0}, p_{ij0} \rangle \leq 0 \\ \|p_{ij0}\| & \text{sinon} \end{array} \right\}$$

Le modèle choisi consiste, pour effectuer la résolution, à ajouter une commande u_i à la vitesse initiale v_{i0}, à $t = 0$. Ainsi il peut a priori aussi bien s'agir d'un changement de cap que d'un changement de vitesse, la première option est préférable.

Contraintes de séparation

La contrainte de séparation entre deux avions peut s'exprimer ainsi :
$d_{ij} \geq ds$ peut s'écrire :

$$< p_{ij0}, v_{ij0} > + \sqrt{\|p_{ij0}\|^2 - ds^2} \|v_{ij0}\| \geq 0$$

Cette contrainte peut être vue comme la réunion de deux demi-plans définis par les deux contraintes linéaires $(v_{ij0} + u_{ij})^T n_{ij1} \geq 0$ et $(v_{ij0} + u_{ij})^T n_{ij2} \geq 0$, où n_{ij1} et n_{ij2} sont représentés sur la figure 5.19.

Cette contrainte doit être rendue quadratique pour pouvoir utiliser la relaxation lagrangienne par SDP. Pour ce, $n(n-1)/2$ variables d'écart $w_{i,j}$ sont introduites pour chaque paire d'avions. On peut alors réécrire la contrainte de séparation à l'aide des deux contraintes suivantes :

$$\begin{aligned} < p_{ij0}, v_{ij0} + u_{ij} > + \sqrt{\|p_{ij0}\|^2 - ds^2} w_{i,j} \geq 0 \\ < v_{ij0} + u_{ij}, v_{ij0} + u_{ij} > \geq w_{i,j}^2 \end{aligned}$$

Contraintes de manœuvres

Les avions peuvent subir un changement important de cap lors de la résolution, mais le changement de vitesse doit être réduit. Ce point rend le problème très difficile. En effet, la contrainte de vitesses constantes s'exprime :

$$\|v_i + u_i\|^2 = \|v_i\|^2$$

Pour gérer une telle contrainte, on peut, dans un premier temps, relâcher l'égalité. La contrainte devient alors :

$$(\frac{1}{1+k})^2 \|v_i\|^2 \leq \|v_i + u_i\|^2 \leq (1+k)^2 \|v_i\|^2$$

avec k petit (par exemple 10%).

La couronne ainsi obtenue n'est pas convexe. Le plus grand domaine convexe que l'on peut extraire de la couronne peut être obtenu en considérant l'intersection d'un demi-plan tangent au cercle intérieur avec la couronne (voir figure 5.20).

Le domaine s'exprime :

$$\|v_{i0} + u_i\| \leq (1+k) * \|v_{i0}\| \tag{5.23}$$

$$< v_{i0} + u_i, v_{i0} > \geq \frac{1}{1+k} * \|v_{i0}\| \tag{5.24}$$

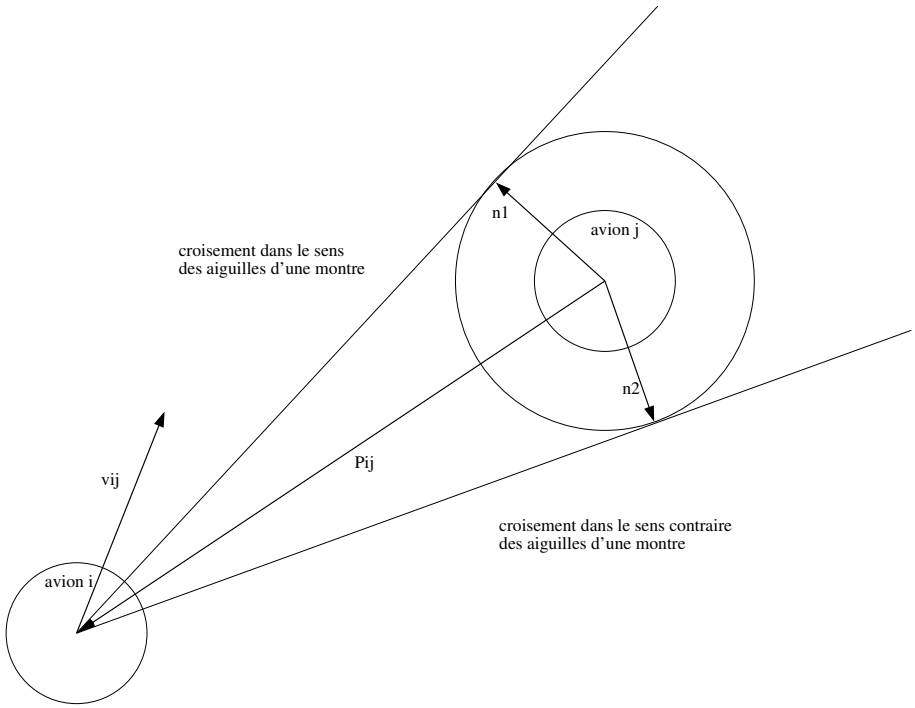

FIG. 5.19 – Contraintes de séparation.

Diminuer la valeur de k permet de durcir la contrainte de vitesse, mais limite également la modification de cap.

Pour éviter ce problème, on peut effectuer plusieurs optimisations successives, en modifiant à chaque fois le demi-plan, tout en diminuant k (voir paragraphe 5.4.5).

Le critère d'optimisation

Le critère le plus simple (voir [FMF01]) consiste à minimiser :

$$\sum_{i=1}^{n} \|u_i\|^2$$

En fait, nous minimiserons le critère normalisé :

$$\sum_{i=1}^{n} \frac{\|u_i\|^2}{\|v_{i0}\|^2} \tag{5.25}$$

D'autres critères peuvent être utilisés, comme par exemple

$$\sum_{i=1}^{n} \frac{\|v_{i0} + u_i\|^2}{\|v_{i0}\|^2} \tag{5.26}$$

Ces deux critères conduisent au même minimum ($u = 0$) lorsqu'il n'y a pas de contrainte de séparation et que k tend vers 0. Ils ont par ailleurs tous deux tendance à minimiser les angles de manœuvre des avions lorsque la contrainte est saturée. Cependant, on observera que SDP se comporte très différemment avec ces deux critères, en fonction du conflit traité. Sur certains conflits, le premier critère

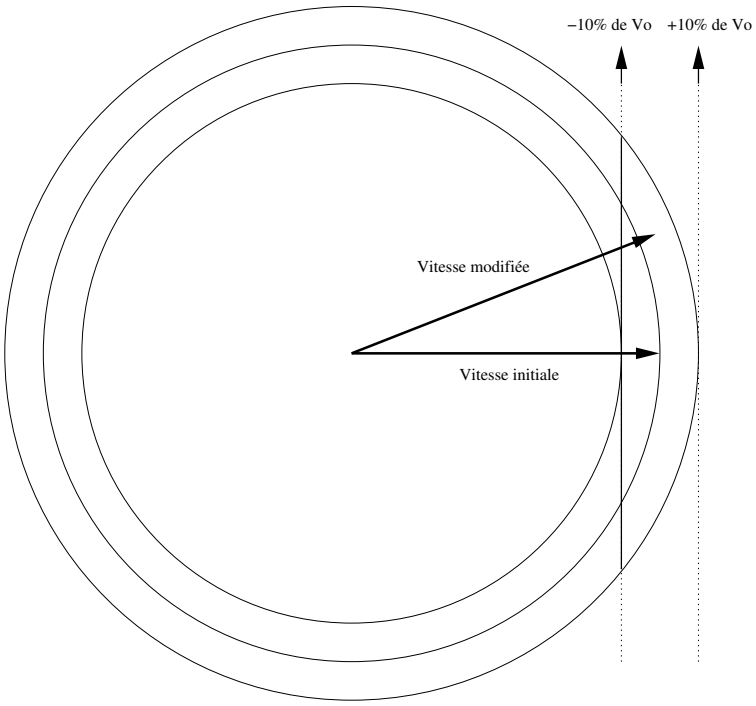

FIG. 5.20 – Contraintes de manœuvre.

s'est montré très efficace et le deuxième beaucoup moins ; sur d'autres, on observe le phénomène inverse. Il est difficile de donner une explication à ce phénomène.

Réduction à un problème d'optimisation convexe

Le domaine de réalisabilité induit par les contraintes de séparation est une réunion de demi-plans, rendant le problème combinatoire. Une fois choisi, pour chaque paire d'avions, le sens de dépassement, ou l'un des deux demi-plans, le problème à résoudre devient convexe.

Si l'on se réfère à la figure 5.19, le sens de dépassement correspond au signe de :

$$p_{ij0} \wedge (v_{ij0} + u_{ij}) \tag{5.27}$$

On choisit le sens de dépassement de la façon suivante : on résout le problème quadratique original après relaxation lagrangienne. On obtient un vecteur

$$(u^1_1, u^2_1, u^1_2, \cdots, w^1_{1,2}, w^2_{1,2}, w^1_{1,3}, \cdots, w^1_{(n-1),n}, w^2_{(n-1),n})$$

Si le signe du produit vectoriel (5.27) est positif, les avions vont s'éviter dans le sens *Inverse des Aiguilles d'une Montre* (IAM), dans le cas contraire, les avions vont s'éviter dans le sens des *Aiguilles d'une Montre* (AM).

En pratique, pour n avions, le sens de résolution pour chaque paire d'avions est donné par $\frac{n\,(n-1)}{2}$ valeurs, qui peuvent être définies par une matrice symétrique contenant des valeurs binaires (0 ou 1). Par la suite, la valeur 0 correspondra au sens AM et 1 correspondra au sens IAM. Les valeurs de la diagonale n'ont pas de sens.

Le problème quadratique original destiné à SDP s'écrit de la façon suivante :

$$\min \sum_{i=1}^{n} \frac{\|u_i\|^2}{\|v_{i0}\|^2} \tag{5.28}$$

$$\begin{aligned} \text{sous les contraintes } \|v_{i0}+u_i\| &\leq (1.1)*\|v_{i0}\| && (5.29)\\ < v_{i0}+u_i, v_{i0} > &\geq (1/1.1)*\|v_{iO}\| && (5.30)\\ < p_{ij0}, v_{ij0}+u_{ij} > +\sqrt{\|p_{ij0}\|^2 - ds^2}w_{i,j} &\geq 0 && (5.31)\\ < v_{ij0}+u_{ij}, v_{ij0}+u_{ij} > &\geq w_{i,j}^2 && (5.32) \end{aligned}$$

La solution approchée de ce problème quadratique (voir paragraphe 5.4.4) fournit une commande de laquelle on extrait la matrice de décision, grâce au signe des produits vectoriels (5.27) pour chaque paire d'avions.

On peut réécrire les équations des demi-plans réalisables, à savoir :

$$(v_{ij0}+u_{ij})^T n_{ij1} \geq 0 \quad et \quad (v_{ij0}+u_{ij})^T n_{ij2} \geq 0$$

sous la forme :

$$\mu_{AM_{ij}}^T (v_{ij0}+u_{ij}) \geq 0$$
$$\mu_{IAM_{ij}}^T (v_{ij0}+u_{ij}) \geq 0$$

avec

$$\mu_{IAM_{ij}}^T = \Gamma_{ij} p_{ij0}, \mu_{AM_{ij}}^T = \Gamma_{ij}^{-1} p_{ij0}$$

et

$$\Gamma_{ij} = \begin{bmatrix} \sqrt{\frac{ds^2}{\|p_{ij0}\|^2}} & -\sqrt{1-\frac{ds^2}{\|p_{ij0}\|^2}} \\ \sqrt{1-\frac{ds^2}{\|p_{ij0}\|^2}} & \sqrt{\frac{ds^2}{\|p_{ij0}\|^2}} \end{bmatrix}$$

Avec ce formalisme, en notant u^* la commande fournie par SDP, la contrainte μ_{ij}^* du problème convexe sera choisie de la façon suivante :

$$\mu_{ij}^* = \left\{ \begin{array}{ll} \mu_{AM_{ij}} & \text{si } (\mu_{AM_{ij}} - \mu_{IAM_{ij}})^T (u_{ij}^* + v_{ij0}) \geq 0 \\ \mu_{IAM_{ij}} & \text{sinon} \end{array} \right\}$$

Le problème convexe est donc le suivant :

$$\begin{aligned} \min \sum_{i=1}^{n} \frac{\|u_i\|^2}{\|v_{i0}\|^2} & \\ \text{sous les contraintes } \|v_{i0}+u_i\| &\leq (1.1)*\|v_{i0}\| \\ < v_{i0}+u_i, v_{i0} > &\geq (1/1.1)*\|v_{iO}\| \\ {\mu_{ij}^*}^T (v_{ij0}+u_{ij}) &\geq 0 \end{aligned}$$

5.4.2 Algorithmes d'optimisation convexe

Présentation

Il existe de nombreux algorithmes permettant de trouver le minimum d'une fonction convexe f de classe C^2 sur R^n.

On peut gérer le problème des contraintes de diverses manières. Considérons par exemple le cas suivant :

$$\min \quad f(x) \tag{5.33}$$
$$\text{sous la contrainte} \quad x \geq 0 \tag{5.34}$$

La contrainte définit un espace convexe.

Soit la fonction :

$$g : x \quad \rightarrow \quad f(x) + I_{\mathbf{R}^+}(x) \tag{5.35}$$

où $I_{\mathbf{R}^+}$ est la fonction indicatrice de $\mathbf{R}^+$ qui vaut 0 sur $\mathbf{R}^+$ et $+\infty$ ailleurs.

Alors :

$$\min_R g \quad = \quad \min_{R^+} f$$

On s'affranchit grâce à g de la contrainte de positivité de x. Toutefois, g n'est plus de classe C^2. Pour conserver les bonnes propriétés de f, on peut alors utiliser à la place de $I_{\mathbf{R}^+}$, une fonction de classe C^2 qui l'approche. Au fur et à mesure que l'on approche de la solution, on peut modifier cette fonction de sorte qu'elle approche de mieux en mieux $I_{\mathbf{R}^+}$.

Ce principe est utilisé dans l'algorithme EPM[5] et dans la méthode EMM (voir annexe B.2.1).

En fait, tout problème de minimisation convexe sous contraintes convexes est équivalent à un problème de minimisation convexe sous contraintes de positivité (pour démontrer ce résultat, il suffit de s'intéresser au problème dual du problème convexe initial).

Ainsi, une fois le sens de résolution choisi pour chaque paire d'avions (autrement dit, une fois la matrice de décision fixée), on se retrouve confronté à un problème de minimisation convexe sous contraintes convexes entrant dans le cadre de ce paragraphe.

La méthode la plus naturelle pour calculer le minimum d'une fonction convexe de classe C^2 est la méthode de Newton. Très efficace sur des problèmes de petite dimension, elle s'avère catastrophique dès que le nombre de variables devient important. En effet, le calcul de l'inverse du Hessien est nécessaire à chaque déplacement du point courant. Il existe des méthodes qui évitent ce problème. Nous verrons successivement une méthode de type Newton modifié, qui n'effectue qu'une fois le calcul de l'inverse du Hessien, puis la méthode de gradient conjugué, qui calcule une approximation du Hessien inverse préservant sa symétrie, puis enfin BFGS (Broyden-Fletcher-Goldfarb-Shanno, voir par exemple [Pol97]) algorithme qui préserve en plus la définie positivité du Hessien (voir annexe B.1.3).

[5] Méthode Proximale Entropique

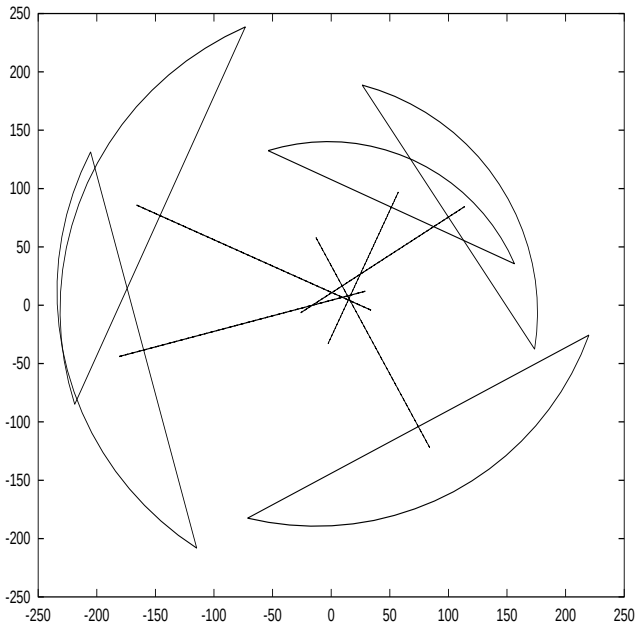

FIG. 5.21 – Les différents vecteurs vitesse des 5 avions et les contraintes de manœuvre.

5.4.3 Application à la résolution de conflits

Une fois choisi le sens de résolution des différentes paires d'avions, le problème prend l'expression suivante :

$$\begin{aligned} \min \sum_{i=1}^{n} \frac{\|u_i\|^2}{\|v_{i0}\|^2} & \\ \text{sous les contraintes } \|v_{i0}+u_i\| &\leq (1.1) * \|v_{i0}\| \\ < v_{i0}+u_i, v_{i0} > &\geq (1/1.1) * \|v_{iO}\| \\ \mu_{ij}^{*\,T}(v_{ij0}+u_{ij}) &\geq 0 \end{aligned}$$

On considère un exemple de 5 avions dont les positions initiales sont réparties sur un cercle et les vitesses initiales dirigées vers le centre. Toutefois, afin de casser la symétrie du problème, les positions et vitesses initiales sont légèrement modifiées de façon aléatoire. La figure 5.21 représente les différents vecteurs vitesse, ainsi que les contraintes de manœuvre.

Soit la matrice de décision :

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Paramètres de BFGS **(choisis empiriquement)**

- Pas : 0.001
- Précision : 1.e-7
- Coefficient de la matrice identité : 0.1

Paramètres de EMM

- Vecteur primal initial : Vecteur nul
- Vecteur dual initial : Vecteur $(1, \cdots, 1)$
- λ_k : $1000 + k$
- Précision : 1.e-7

L'algorithme atteint le critère d'arrêt après trois itérations de EMM. Le problème ne pose pas de difficulté réelle, mais la solution trouvée est totalement dépendante de la matrice de décision choisie.

5.4.4 Méthode de relaxation de problèmes quadratiques par SDP

Programmation semi-définie

Considérons le problème de minimisation d'une fonction linéaire sous contrainte d'inégalité matricielle :

$$\min_{F(x)\geq 0} c^T x$$

où

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i$$

et $F(x) \geq 0$ signifie "F(x) est une matrice semi-définie positive".

Le problème dual associé est, au sens du produit scalaire défini sur les matrices symétriques :

$$< A, B >= \mathbf{Tr}(A^T B)$$

s'écrit :

$$\begin{aligned} \max \quad & -\mathbf{Tr} F_0 Z \\ \text{sous les contraintes} \quad & \mathbf{Tr} F_i Z \quad = c_i, i = 1, \cdots, m \end{aligned}$$

Nous utilisons par la suite la programmation semi-définie dans un cadre très particulier, la relaxation lagrangienne de problèmes quadratiques.

Relaxation des problèmes quadratiques

Le problème de base du modèle de résolution de conflit est un problème quadratique. La difficulté du problème de résolution de conflits est due à la contrainte de séparation, qui induit des contraintes non convexes.

Sa résolution directe est donc impossible.

Un problème quadratique standard peut s'écrire :

$$\begin{aligned} \min \quad & f_0(x) && (5.36)\\ \text{sous la contrainte} \quad & f_i(x) \quad \leq 0, i = 1, \cdots, L, && (5.37) \end{aligned}$$

où chaque f_i est définie par :

$$f_i(x) = x^T A_i x + 2b_i^T x + c_i$$

La relaxation lagrangienne du problème quadratique standard s'écrit :

$$\begin{array}{c} \max t \\ \text{sous les contraintes} \quad \left[\begin{array}{cc} A_0 & b_O \\ b_0^T & c_0 - t \end{array} \right] + \tau_1 \left[\begin{array}{cc} A_1 & b_1 \\ b_1^T & c_1 \end{array} \right] + \cdots + \tau_L \left[\begin{array}{cc} A_L & b_L \\ b_L^T & c_L \end{array} \right] \geq 0 \\ \tau_1, \tau_2, \cdots, \tau_L \geq 0 \end{array}$$

On peut vérifier que ce problème semi-défini donne une borne inférieure au problème quadratique. Pour cela, on réécrit les f_i sous la forme :

$$f_i(x) = \left[\begin{array}{c} x \\ 1 \end{array} \right]^T \left[\begin{array}{cc} A_i & b_i \\ b_i^T & c_i \end{array} \right] \left[\begin{array}{c} x \\ 1 \end{array} \right]$$

Si $t, \tau_1, \cdots, \tau_L$ vérifient les contraintes du problème semi-défini, alors :

$$\begin{aligned} 0 \quad \leq \quad & \left[\begin{array}{c} x \\ 1 \end{array} \right]^T \left\{ \left[\begin{array}{cc} A_0 & b_O \\ b_0^T & c_0 - t \end{array} \right] + \tau_1 \left[\begin{array}{cc} A_1 & b_1 \\ b_1^T & c_1 \end{array} \right] + \cdots + \tau_L \left[\begin{array}{cc} A_L & b_L \\ b_L^T & c_L \end{array} \right] \right\} \left[\begin{array}{c} x \\ 1 \end{array} \right] \\ = \quad & f_0(x) - t + \tau_1 f_1(x) + \cdots + \tau_L f_L(x) \\ \leq \quad & f_0(x) - t \end{aligned}$$

cqfd.

On peut maintenant s'intéresser au problème dual de ce problème. Calculons le dual du programme précédent :

$$\begin{aligned} \min \quad & \mathbf{Tr} X A_0 + 2b_0^T x + c_0 \\ \text{sous la contrainte} \quad & \mathbf{Tr} X A_i + 2b_i^T x + c_i \quad \leq 0, i = 1, \cdots, L, \\ & \left[\begin{array}{cc} X & x \\ x^T & 1 \end{array} \right] \geq 0 \end{aligned}$$

Ce programme semi-défini est une relaxation du programme quadratique original, qui peut en effet être écrit :

$$\begin{aligned} \min \quad & \mathbf{Tr} X A_0 + 2b_0^T x + c_0 \\ \text{sous la contrainte} \quad & \mathbf{Tr} X A_i + 2b_i^T x + c_i \quad \leq 0, i = 1, \cdots, L, \\ & \left[\begin{array}{cc} X & x \\ x^T & 1 \end{array} \right] = 0 \end{aligned}$$

Pour montrer ce résultat, notons tout d'abord que :

$$\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} = 0$$

est équivalent à :

$$X = xx^T$$

On peut ensuite écrire, comme les f_i sont des scalaires :

$$f_i(x) = x^T A_i x + 2b_i^T x + c_i = \mathbf{Tr}(x^T A_i x + 2b_i^T x + c_i) = \mathbf{Tr}(xx^T A_i) + 2b_i^T x + c_i$$

et en posant :

$$xx^T = X$$

il reste :

$$f_i(x) = \mathbf{Tr} X A_i + 2b_i^T x + c_i$$

ce que nous voulions prouver.

Il reste, pour être complet, à exprimer le problème relatif à la relaxation par un problème SDP classique. Un moyen simple est d'intégrer les contraintes de positivité des multiplicateurs dans la matrice primale. En effet, dans un programme SDP, plusieurs inégalités matricielles (ici une inégalité matricielle et des inégalités scalaires) peuvent être combinées en une seule inégalité, en ajoutant autant de lignes et de colonnes qu'il y a d'inégalités scalaires. Malheureusement, ce procédé est inefficace numériquement, car il augmente la taille de la matrice primale, rendant plus ardus tous les traitements qu'elle va devoir subir dans l'algorithme Primal-Dual : transformation de Cholesky, minimisation de carrés ... [VB96].

Des programmes directement dédiés aux relaxations SDP existent, comme *SDPpack*. Nous n'avons pas utilisé de tels programmes.

Toutefois, si l'on se fie à l'indicateur d'écart de dualité, c'est-à-dire l'écart mesuré par le programme entre la valeur primale et la valeur duale, on peut assurer que la programme a été correctement résolu dans tous les cas que nous avons eu à traiter.

Tout le travail consiste à transformer le problème quadratique sous une forme lisible par SDP. Typiquement, si on veut transformer la i-ème contrainte :

$$f_i(x) = x^T A_i x + 2b_i^T x + c_i$$

Il faut fournir à SDPA la matrice :

$$\begin{pmatrix} A_i & b_i & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ b_i^T & c_i & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Le 1 dans la matrice exprime la contrainte de positivité du i-ème multiplicateur. Il se trouve donc en position $(n+1+i, n+1+i)$, où n est la dimension de la matrice A.

Calculons la dimension de la matrice totale (augmentée des contraintes scalaires). On peut ranger les variables de la façon suivante :

$$(u_1^1, u_1^2, u_2^1, \cdots, w_{1,2}^1, w_{1,2}^2, w_{1,3}^1, \cdots, w_{(n-1),n}^1, w_{(n-1),n}^2)$$

Donc, pour p avions, la dimension du vecteur, et donc de la matrice A, est :

$$2 * (p + (p(p-1))/2) = p(p+1)$$

Donc la taille totale de la matrice (matrice creuse) traitée par SDPA est

$$(p(p+1)) + 1 + k$$

où k est le nombre total de contraintes du programme quadratique. En se référant à notre problème de résolution de conflits, on obtient :

$$k = 2p + p(p-1) = p(p+1)$$

Finalement SDPA traite des matrices de taille :

$$2\,p\,(p+1) + 1$$

Amélioration de la solution : génération aléatoire d'un bruit autour de la solution SDP

La relaxation lagrangienne du problème initial ne fournit pas nécessairement la solution optimale du problème (sauf dans le cas où l'on n'a qu'une seule contrainte). La solution fournie peut néanmoins être un bon point de départ pour la recherche de la solution optimale. Karger [KMS98] propose de chercher la solution optimale dans un voisinage de la solution fournie par SDP, et ce par bruitage ou "randomisation".

On notera $\widehat{x}$ la solution SDP et $\widetilde{x}$ la variable bruitée (dans un voisinage de $\widehat{x}$).

La matrice $X - \widehat{x}\widehat{x}^T$ est un indicateur du degré de relaxation du problème, il est donc naturel de l'interpréter comme une matrice de covariance de la variable bruitée $\widetilde{x}$.

On veut donc que :

$$\mathbf{E}(X - \widetilde{x}\widetilde{x}^T) = 0$$

ou encore :

$$\mathbf{E}(X_{ij} - \widetilde{x_i}\widetilde{x_j}) = 0$$

Considérer $X - \widehat{x}\widehat{x}^T$ comme matrice de covariance revient à écrire :

$$\begin{aligned} X_{ij} - \widehat{x_i}\widehat{x_j} &= \mathbf{E}((\widetilde{x_i} - \mathbf{E}(\widetilde{x_i}))(\widetilde{x_j} - \mathbf{E}(\widetilde{x_j}))) && (5.38) \\ &= \mathbf{E}(\widetilde{x_i}\widetilde{x_j}) - \mathbf{E}(\widetilde{x_i})\mathbf{E}(\widetilde{x_j}) && (5.39) \end{aligned}$$

Comme on veut :

$$X_{ij} = \mathbf{E}(\widetilde{x_i}\widetilde{x_j})$$

Il suffit de prendre :

$$\widehat{x_i} = \mathbf{E}(\widetilde{x_i})$$

Afin de pouvoir bruiter indépendamment chaque variable, il faut diagonaliser la matrice de covariance. On peut alors utiliser un bruit gaussien sur chaque variable.

L'algorithme **QR** permet de diagonaliser simplement la matrice de covariance. Dans la version décrite dans ce paragraphe, la convergence de la matrice de départ vers une matrice diagonale est assurée théoriquement, car nous traitons des matrices semi-définies positives. Mais, pour éviter un trop grand nombre d'itérations, il faut utiliser certains raffinements, comme la décomposition de Hessenberg et Le Shift [GL89].

L'algorithme **QR** est basé sur la décomposition **QR** d'une matrice A $(m \times n)$ en

$$A = QR$$

où Q est une matrice $(m \times m)$ du groupe orthogonal et R est une matrice $(m \times n)$ triangulaire supérieure.

Une fois la décomposition QR effectuée, on peut appliquer l'algorithme QR proprement dit. On cherche à diagonaliser A :

Algorithme QR

- $T_o = A$
- Pour $k = 1, 2, \cdots$
  $T_{k-1} = U_k R_k$ (décomposition QR)
  $T_k = R_k U_k$
- fin

Comme $T_k = R_k U_k = U_k^T (U_k R_k) U_k = U_k^T T_{k-1} U_k$, par récurrence, on obtient :

$$T_k = (U_0 U_1 \cdots U_k)^T A (U_0 U_1 \cdots U_k)$$

Et donc chaque T_k est semblable à A.

5.4.5 Application au problème de résolution de conflits.

Principe général de résolution

Le principe général de résolution est le suivant :

1. On calcule la relaxation lagrangienne du problème quadratique défini par les relations 5.28-5.32 par programmation semi-définie.
2. On diagonalise la matrice de covariance en utilisant l'algorithme QR (paragraphe 5.4.4).
3. On engendre aléatoirement des points dans un voisinage de la solution SDP : à chaque génération, on calcule la solution du programme convexe relatif à sa matrice de décision.
4. On continue le processus jusqu'à ce qu'on estime, à l'aide par exemple de la borne SDP, avoir obtenu une "bonne solution".

Détaillons le déroulement sur l'exemple à 5 avions traité dans la partie 5.4.3.

On exécute tout d'abord SDPA. La lecture dans la matrice duale donne la commande, le calcul par QR fournit l'amplitude du bruit pour chaque variable. Pour ce problème, seules deux matrices sont obtenues après bruitage, autrement dit, la *randomisation* n'a produit qu'une seule nouvelle matrice. L'heuristique 5.27 permet de calculer les matrices de décision correspondantes :

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \text{ et } \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

On peut désormais effectuer une optimisation convexe sur chacune de ces matrices, et choisir celle qui induit la plus faible valeur du critère à minimiser.

La deuxième matrice est retenue en fonction du critère 5.25 :

$$\sum_{i=1}^{n} \frac{\|u_i\|^2}{\|v_{i0}\|^2}$$

ou le critère 5.26 :

$$\sum_{i=1}^{n} \frac{\|v_{i0} + u_i\|^2}{\|v_{i0}\|^2}$$

Solution à vitesse constante

Nous présentons dans cette section une méthode simple pour chercher une solution à vitesse constante au problème de résolution de conflits. On utilise pour cela une méthode itérative qui définit, à chaque itération, un domaine convexe autour de la solution trouvée à l'itération précédente, la taille du domaine convergeant vers 0 (Dans la pratique, on peut par exemple diviser la taille du domaine par deux à chaque itération, en partant de $k = 10\%$).

La figure 5.22 montre les deux premières itérations pour un avion.

Les figures 5.23 et 5.24 donnent la première, deuxième, troisième et neuvième itération du processus.

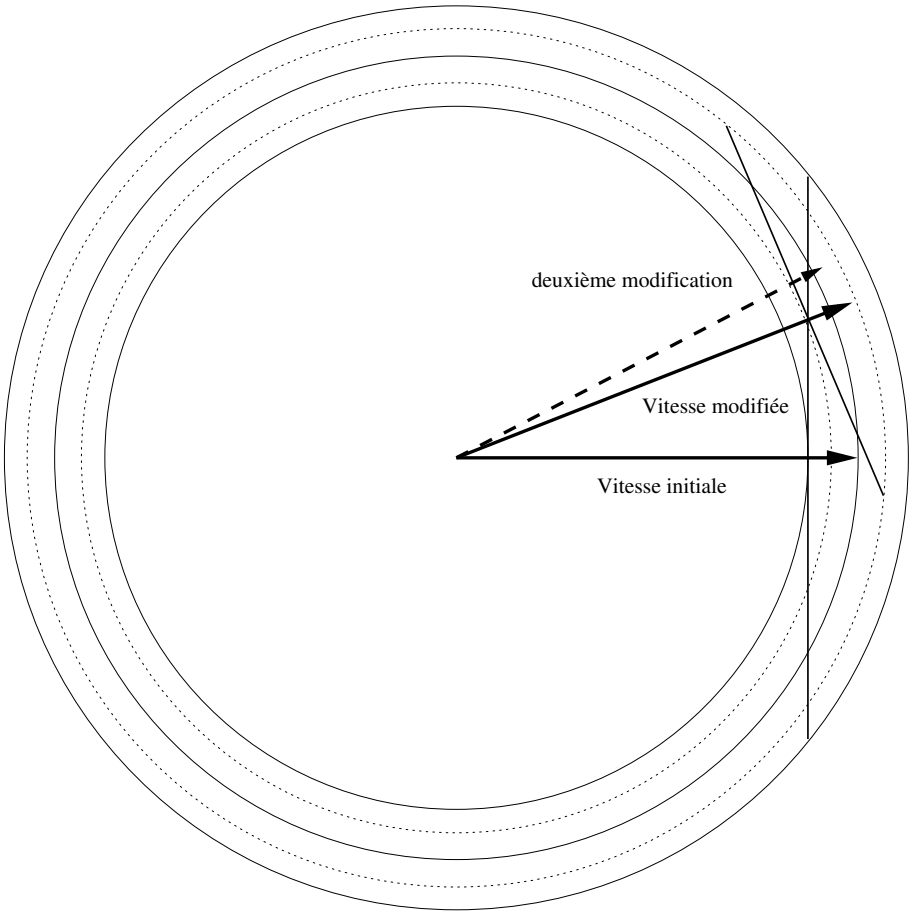

FIG. 5.22 – Calcul de la nouvelle contrainte à partir de l'ancienne, les deux cercles (le plus interne et le plus externe) représentent une marge de 10%, les deux cercles en pointillé représentent une marge de 5%

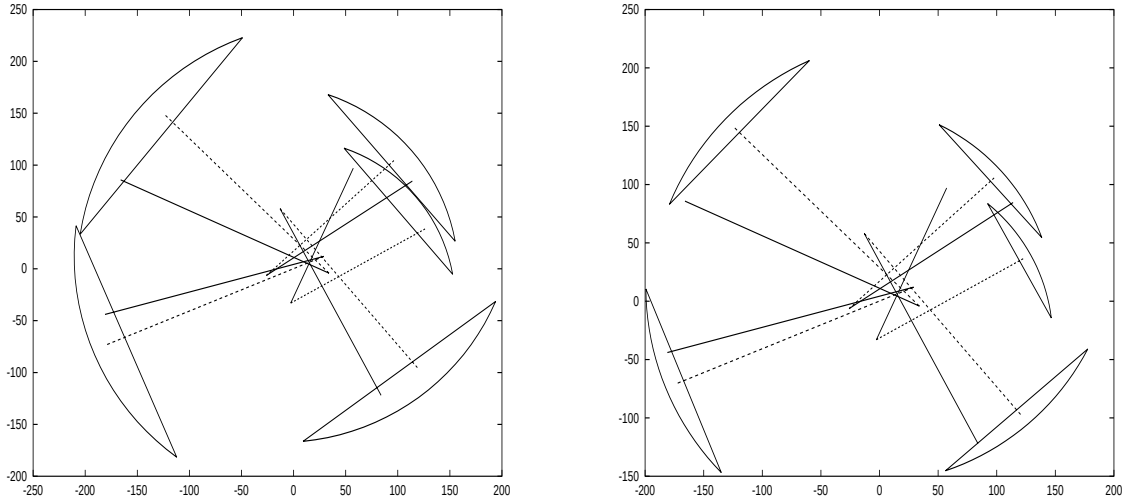

FIG. 5.23 – Itération 1, taille du domaine : $k = 5\%$ - Itération 2, taille du domaine : $k = 2,5\%$

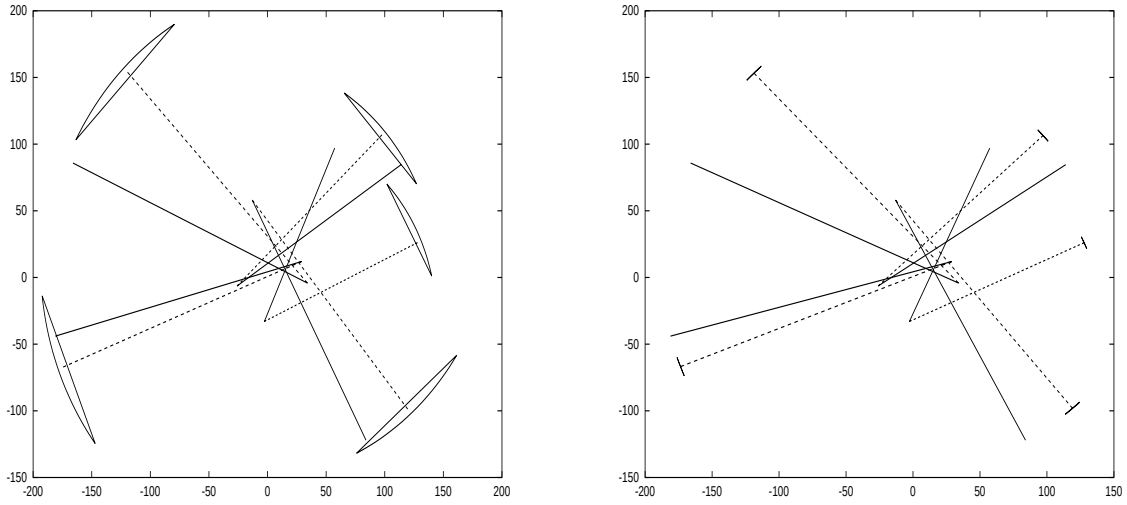

FIG. 5.24 – Itération 3, taille du domaine : $k = 1,75\%$ - Itération 9, taille du domaine : $k = 0.001\%$

Conclusion

Les statistiques proposées dans ce paragraphe prouvent l'efficacité de la combinaison entre la relaxation SDP et le bruitage gaussien pour la résolution des programmes quadratiques. De plus, la méthode des multiplicateurs exponentiels, méthode récente d'optimisation convexe, permet de gérer les contraintes du problème une fois la matrice de décision choisie.

Toutefois, la simplification des contraintes de manœuvres, rendue inévitable pour l'application d'une méthode de relaxation, a conduit à un modèle très éloigné de la réalité. En effet :

- dans la réalité, la marge de manœuvre sur la norme de la vitesse des avions est très réduite, il est d'ailleurs difficile de connaître avec précision la vitesse future de l'avion. Or, dans notre application, nous avons pu observer un exemple qui montre que, dans certains cas, l'algorithme conduit à une résolution *en vitesse* du conflit sans pouvoir respecter la contrainte de vitesses constantes ;
- tous les avions doivent changer de cap en même temps, ce qui est inapplicable en pratique. Généralement, on attend que le conflit devienne certain avant de le résoudre. Une phase de surveillance du conflit permet dans certains cas d'éviter toute manœuvre ;
- Le modèle utilisé suppose que la vitesse des avions est constante, ce qui n'est vrai que pour une faible partie du trafic aérien (un quart du trafic en France par exemple). Ceci rend difficile la prise en compte d'avions en montée ou en descente. En effet, pour tenir compte des variations de vitesse verticales de tels avions non placés dans le plan horizontal, une résolution à vitesse variable est indispensable.

Finalement, on peut dire que l'algorithme de résolution implémenté est efficace et intéressant d'un point de vue théorique, mais il paraît bien difficile d'éviter les difficultés décrites précédemment pour une application dans des conditions réelles.

5.5 Conclusion

Les approches présentées dans ce chapitre sont très différentes en terme de modélisation. L'utilisation des algorithmes génétiques permet de ne pas faire d'hypothèse contraignante sur les trajectoires des avions. Les trois autres modélisations sont plus restrictives. L'approche combinant un AG et un algorithme de programmation linéaire requiert l'utilisation d'*offsets* et ne peut prendre en compte d'incertitude sur les vitesses des avions. La méthode mettant un oeuvre un *branch and bound* par intervalles suppose que les avions aient des vitesses constantes, ce qui ne permet pas de prendre en compte les avions en montée et en descente. La méthode de programmation semi-définie suppose également que les avions volent à vitesse constante sur des trajectoires rectilignes. En terme d'efficacité, seuls les algorithmes génétiques, grâce à l'utilisation de la séparabilité partielle, permette de résoudre des conflits de grande taille impliquant plus de 20 avions.

Chapitre 6

Approches autonomes ou embarquées de résolution de conflits

Introduction

Dans ce chapitre sont détaillées les différentes méthodes de résolution utilisant une approche autonome ou distribuée du problème de résolution de conflits.

On dira qu'un système d'évitement est *embarqué* (ou éventuellement *distribué*) lorsque la trajectoire d'évitement de chaque avion est engendrée individuellement (dans le cadre d'une application d'une telle méthode, elle serait le plus souvent engendrée à bord de l'avion lui-même, d'où le terme *embarqué*).

On distinguera à ce titre deux grandes approches, selon que les avions manœuvrent de façon simultanée ou séquentielle :

- parmi les approches réactives, où les avions réagissent simultanément, on abordera une méthode à base de réseaux de neurones, détaillée dans la thèse de Frédéric Médioni [Méd98].
- parmi les approches séquentielles, on détaillera une méthode d'allocation de jetons combinée avec un algorithme A^* permettant de gérer les conflits dans un contexte *Free-Flight*. Cette approche a fait l'objet du DEA de Géraud Granger [Gra98] et de Nicolas Archambault [Arc03], co-encadrés avec Jean-Marc Alliot.

6.1 Méthodes neuronales

L'évitement réactif consiste 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 impliqués dans un conflit potentiel avec lui. Aucun ordonnancement n'est requis entre les avions qui doivent réagir simultanément.

En limitant les actions d'évitement à des modifications de cap, on peut envisager un système d'évitement réactif modifiant le cap d'un l'avion, à partir de grandeurs caractérisant sa situation. Les *grandeurs* en question décrivent la situation de cet avion, mais doivent aussi contenir des informations sur les avions avec lesquels il risque d'entrer en conflit. Elles nécessitent bien entendu un échange d'informations entre avions, 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, 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). Puisqu'aucune fonction rapidement calculable répondant au critère précédent (engendrer des trajectoires sans conflit de longueur minimale) n'est connue, on cherche à obtenir une fonction l'approchant ayant une bonne capacité de généralisation sur tous les conflits possibles.

Les réseaux de neurones, qui sont des approximateurs universels [HSW89], semblent préférables à d'autres approximateurs, tels que les polynomes par exemple, parce qu'ils permettent d'obtenir une meilleure généralisation.

Le but est le suivant : 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 utilisés sont des réseaux multi-couches totalement connectés à une couche cachée (voir 6.1.1). Ils sont utilisés pour résoudre des conflits à deux ou trois avions. Les entrées du réseau sont données dans la section 6.1.2 (conflits à deux avions).

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 6.1.1). 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. 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 est calculée à partir de l'évaluation de la qualité du réseau sur un ensemble de configurations de conflits. C'est cet ensemble qui constituera la *base d'apprentissage*.

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 réseaux, jouent un rôle fondamental dans l'obtention de cette capacité de généralisation.

Deux manières de mener l'évaluation des réseaux sont testées : 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 6.1.3 et comparées dans les sections 6.1.4 (conflits à deux avions) et 6.1.6 (conflits à trois avions).

6.1.1 Les réseaux de neurones

Cette section présente brièvement les réseaux de neurones. Pour plus de détails, on peut 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]. Très schématiquement, 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 considérer, en simplifiant, que le noyau transmet un influx nerveux à l'axone si la somme pondérée des influx lui parvenant des différentes dendrites (les poids pouvant être négatifs) est supérieure à un certain seuil. 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*).

Le modèle mathématique du neurone, donné par McCulloch et Pitts, utilisait une fonction seuil f_θ de $\mathbb{R}$ dans $\{0,1\}$, définie à partir d'un seuil $\theta \in \mathbb{R}$:

$$\begin{cases} f_\theta(x) = 1 & \text{si } x > \theta, \\ f_\theta(x) = 0 & \text{sinon.} \end{cases} \tag{6.1}$$

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 θ qui caractérisent le neurone, la sortie étant donnée par :

$$O = f_\theta(\sum_{i=1}^{n_e} W_i I_i) \tag{6.2}$$

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 peut prendre des valeurs réelles dans l'intervalle [0,1]. La fonction seuil est remplacée par une *fonction d'activation* de $\mathbb{R}$ dans $[0,1]$. Généralement, on utilise une fonction de type *sigmoïde* :

$$f_\theta(x) = \frac{1}{1 + e^{-(x-\theta)}} \tag{6.3}$$

Les réseaux de neurones utilisés sont des réseaux multi-couches, cas particuliers de réseaux à sens unique :

Définition 11 (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 12 (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 entrées 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 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 couples 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.

Dans le cas présent, l'apprentissage des réseaux de neurones est réalisé au moyen d'un algorithme génétique. Les individus de la population codent les poids des réseaux de neurones. La fonction d'adaptation utilisée 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.

6.1.2 Génération de trajectoires d'évitement

Hypothèses

On se place à nouveau dans un contexte de deux avions volant à vitesse constante dans un plan horizontal. On notera n_h la séparation standard et l'évitement est recherché grâce à des modifications de caps des avions. On suppose qu'à chaque pas de temps de discrétisation, les avions connaissent leur position, vitesse, cap et destination, ainsi que les position, vitesse, cap et la destination de l'autre avion.

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 6.1.1). Il utilise 8 entrées décrites dans la section 6.1.2 et renvoie une sortie réelle. Les poids du réseau de neurones sont modifiés au cours de l'apprentissage par un algorithme génétique. La fonction d'activation est une sigmoïde, dont l'expression est donnée par l'équation 6.3. 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 θ de la fonction d'activation.

La couche cachée des réseaux que nous utilisons comporte 25 neurones (nombre déterminé empiriquement après plusieurs essais). En comptant les poids correspondant aux seuils de fonctions sigmoïde utilisées par les neurones, on obtient un total de 251 poids.

Le temps est discrétisé, on note l_t la longueur d'un pas de temps, N_t le nombre de pas de temps. Les trajectoires des deux avions sont engendrées au moyen de réseaux de neurones identiques strictement. Il n'y a pas d'autre coordination entre avions que la transmission réciproque des position, vitesse, cap et destination de chaque avion. Le réseau de neurones n'est utilisé que si un conflit est détecté.

Entrées et sortie du réseau

Il est difficile de déterminer *a priori* quelles informations seront les plus utiles aux réseaux de neurones. Les entrées présentées ici sont celles grâce auxquelles ont été obtenus les meilleurs résultats sans que l'on puisse affirmer qu'un autre jeu d'entrées serait forcément moins efficace. Dans la suite, l' *avion courant* sera celui pour lequel le réseau de neurones fournit une modification de cap (il sera noté a_1), et l'*intrus* sera l'autre avion (noté a_2) impliqué dans le conflit. La vitesse de l'avion a_i est notée v_i, et son cap à un instant t est noté $c_i(t)$.

Les entrées 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 6.1) :

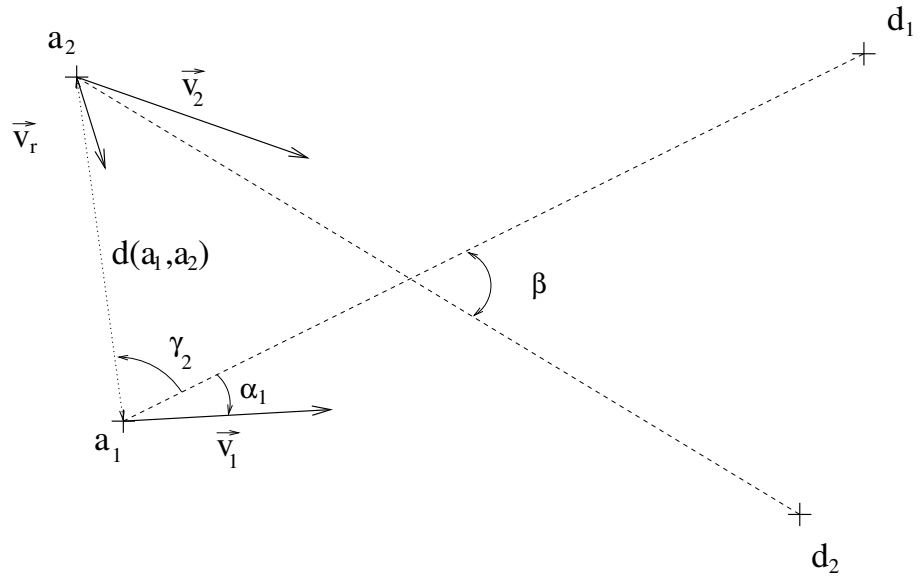

FIG. 6.1 – Entrées du réseau de neurones.

- L'angle, noté α_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 γ_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 β entre les routes directes des avions de leurs positions courantes à leurs destinations.

Ces 6 données sont traduites par 8 entrées. Les angles α_1 et γ_2 ne sont pas donnés directement au réseau de neurones ; chacun d'eux est traduit par deux entrées : son sinus et son cosinus. Le fait d'utiliser directement les valeurs des angles introduit une discontinuité ne correspondant pas à la réalité. L'angle β 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 degrés.

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.

Le taux de virage maximal t_v de l'avion sera le taux standard de trois degrés par secondes ($360°$ en deux minutes).

Le cap courant c_i de l'avion a_i est modifié à l'instant $t = t_k$ d'une valeur δ_{ck} : $c_1(t_{k+1}) = c_1(t_k) + \delta_{ck}$ où

$$\delta_{ck} = (s_k - 0,5)t_v l_t \tag{6.4}$$

s_k est la sortie du réseau de neurones obtenue avec les entrées correspondant à l'instant $t = t_k$.

Configurations de conflit

Pour générer des configurations de conflits, on définit une distance d'alerte d_a en dessous de laquelle le réseau de neurones intervient. Il faut ensuite, pour que la configuration soit admissible, que les positions initiales et finales des avions soient sans conflit et qu'il existe un instant t dans l'horizon temporel fixé pour lequel les deux avions sont en conflit.

Le lecteur intéressé trouvera dans la thèse de F. Médioni [Méd98] une méthode de génération aléatoire de configurations de conflits à deux avions.

6.1.3 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.

Chacune de ces évaluations sera réalisée au moyen d'une *base d'apprentissage*.

Définition 13 (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 créées 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. La base d'apprentissage renouvelée a été envisagée 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.

On note N_a le nombre de configurations de conflit contenues dans la base d'apprentissage. 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). 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 créées au moyen du réseau de neurones, par rapport aux trajectoires originelles des avions.

On définit un entier noté $N_v(\mathcal{R}, \mathcal{C}_i)$, d'abord initialisé à 0, qui compte le nombre de pas de temps sur lesquels les contraintes de séparation sont violées, et un réel, noté $r(\mathcal{R}, \mathcal{C}_i)$, qui correspond à la somme des retards des deux avions.

On calcule les valeurs de $(\mathcal{R}, \mathcal{C}_i)$ en procédant de la manière suivante :

1. *A $t = 0$, les avions sont placés aux positions initiales données par la configuration $\mathcal{C}_i$, avec les caps et vitesses correspondants.* $N_v = 0$ *et* $r = 0$
2. *De $k = 0$ à $k = N_t$, 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 de 1.

(c) Pour chaque avion a_j, on calcule les entrées du réseau de neurones correspondant à la situation (voir section 6.1.2), 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).

3. *Si les avions peuvent rejoindre leurs destinations sans conflit, on calcule $r(\mathcal{R}, \mathcal{C}_i)$ 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 (valant N_t).*

On définit $N_{vt}(\mathcal{R})$, le nombre sur toute la base d'apprentissage, de pas de temps sur lesquels les avions n'ont pas été séparés :

$$N_{vt}(\mathcal{R}) = \sum_{i=1}^{N_a} N_v(\mathcal{R}, \mathcal{C}_i) \tag{6.5}$$

Et $r_t(\mathcal{R})$, la moyenne quadratique des retards des avions :

$$r_t(\mathcal{R}) = \sqrt{\frac{\sum_{i=1}^{N_a} r(\mathcal{R}, \mathcal{C}_i)^2}{N_a}} \tag{6.6}$$

Pour une base d'apprentissage fixe, la fonction d'adaptation $f_a(\mathcal{R})$ est définie de deux manières différentes, selon que $\mathcal{R}$ engendre ou non des trajectoires sans conflit pour toutes les configurations de la base d'apprentissage :

Si $N_{vt}(\mathcal{R}) \neq 0$:

$$f_a(\mathcal{R}) = \frac{1000}{1 + N_{vt}(\mathcal{R})} \tag{6.7}$$

Si $N_{vt}(\mathcal{R}) = 0$:

$$f_a(\mathcal{R}) = 1000 + \frac{1000}{1 + r_t(\mathcal{R})} \tag{6.8}$$

L'intérêt d'utiliser, dans le calcul de f_a, la moyenne quadratique 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.

Les réseaux obtenus avec une base d'apprentissage fixe ne donnent pas de très bons résultats sur certains conflits *non appris*.

Le principe de la base d'apprentissage renouvelée est d'utiliser, à chaque génération de la population de réseaux de neurones, de nouvelles configurations de conflit, à chaque fois créées 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 ; de *bons réseaux* de neurones sur une ou plusieurs générations, disparaissaient de la population, pour avoir obtenu de mauvais 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 est le suivant : à une génération donnée, un nouveau réseau $\mathcal{R}_{j_1}$ a une meilleure fonction d'adaptation sur les N_a configurations de conflit actuellement présentes dans la base d'apprentissage qu'un réseau $\mathcal{R}_{j_0}$ apparu plus tôt dans la population. Cependant le réseau $\mathcal{R}_{j_0}$ peut avoir obtenu de bons résultats sur de nombreuses générations. 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 ce dernier plus *fiable*.

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 α_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 :

$$f_r^{(k)}(\mathcal{R}) = \alpha.f_r^{(k-1)}(\mathcal{R}) + (1-\alpha).f_a(\mathcal{R}) \tag{6.9}$$

Cependant la relaxation utilisée telle quelle ne correspond pas parfaitement à cette *fiabilité* que nous recherchons pour les réseaux obtenus.

Un calcul de probabilité permet de se rapprocher de cette notion de fiabilité. Ce calcul a été initialement utilisé par Jean-Marc Alliot pour optimiser un programme de jeu (Othello) [AD95]. On veut en fait estimer la probabilité d'un réseau à résoudre tous les conflits à partir d'un certain nombre d'observations.

Si on suppose que la probabilité de résoudre un conflit est p, la probabilité de résoudre m conflits sur n est donnée par :

$$P(p,m,n) = \left(\begin{array}{c} n \\ m \end{array} \right) p^m (1-p)^{(n-m)} \tag{6.10}$$

On cherche maintenant une estimation de la probabilité p, à partir du nombre de situations résolues (m), sur le nombre de situations jouées (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 $p_f \in [0,1]$:

$$\int_{p_{m,n,p_f}}^{1} P(p,m,n)dp = \frac{p_f}{n+1} \tag{6.11}$$

En fixant p_f à 0,95, on 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.

Le nombre de configurations de conflit présentes dans la base d'apprentissage est 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 terme de retard. Si ce nombre est trop faible, il est difficile d'obtenir un réseau fiable. Pour des conflits à deux avions, c'est avec $N_a = 50$ que les meilleurs résultats ont été obtenus.

On définit, pour chaque réseau de neurones, 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 6.11. 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 6.11, on ne s'intéresse qu'aux cas $m = n = n_s(\mathcal{R})$. Empiriquement, on a choisi $p_f = 0,95$, et on calcule préalablement à

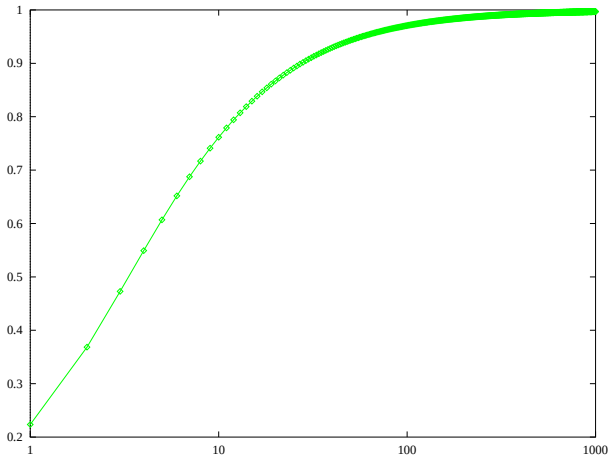

FIG. 6.2 – Valeur de la pondération en fonction du nombre de parties gagnées ($p_f = 0.95$).

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. On a calculé ces valeurs pour n_s allant jusqu'à 1 000 (voir figure 6.2).

$s(\mathcal{R})$ est utilisé dans le calcul de la fonction d'adaptation du réseau $\mathcal{R}_j$. 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 de ces bases d'apprentissage est utilisée. Ainsi, le nombre de violations $N_{vt}(\mathcal{R})$ et la moyenne quadratique des retards $r_t(\mathcal{R})$ sont remplacés par $N'_{vt}(\mathcal{R})$ et $r'_t(\mathcal{R})$ calculés 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.

Si $N'_{vt}(\mathcal{R}) \neq 0$:

$$f_a(\mathcal{R}) = \frac{1000}{1 + (\mathcal{R})} \tag{6.12}$$

Si $N'_{vt}(\mathcal{R}) = 0$:

$$f_a(\mathcal{R}) = 1000 + s(\mathcal{R}).\frac{1000}{1 + r'_t(\mathcal{R})} \tag{6.13}$$

6.1.4 Résultats obtenus sur des conflits à deux avions

Paramètres de l'algorithme génétique

Les chromosomes contiennent 251 réels correspondant aux 251 poids des réseaux. Le croisement utilisé est le croisement barycentrique classique, et la mutation consiste à ajouter un bruit gaussien à certains poids.

Les paramètres de l'algorithme génétique utilisé sont :

- 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.
- Nombre de configurations de conflit dans la base d'apprentissage : 50.
- L'algorithme utilise l'élitisme.
- La sélection est une sélection par le rang.
- L'algorithme utilise le *sharing* avec *clustering* adaptatif, avec un taux de *sharing* de 0,8.

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 quelconques mais fixes, comprenant 50 configurations de conflits créées aléatoirement. On obtient très vite 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). Le réseau obtenu est ensuite testé sur 10 000 configurations créées aléatoirement. Du point de vue du respect des contraintes de séparation, le réseau de neurones obtenu ne généralise pas parfaitement, mais on peut considérer que les résultats obtenus sont corrects : il y a violation des contraintes dans moins de 4 pour cent des configurations. De plus, comme le montre le tableau 6.1, une grande proportion des violations de contraintes mènent en fait à des distances minimales entre avions proches de la norme de séparation.

TAB. 6.1 – Configurations (sur 10 000) pour lesquelles le conflit n'a pas été résolu.

Distance minimale	Nombre d'occurrences
de $3,75$ à 4 NM	152
de $3,5$ à $3,75$ NM	80
de 3 à $3,5$ NM	115
de 2 à 3 NM	41
de 1 à 2 NM	0
de 0 à 1 NM	0
total	388

Le tableau 6.2 donne la répartitions des retards pour les configurations résolues. Le retard moyen sur ces configurations est de 5 secondes.

Apprentissage sur une base renouvelée

Avec une base d'apprentissage renouvelée, on constate tout d'abord que la généralisation est beaucoup plus efficace : le réseau obtenu fournit des trajectoires sans conflit pour les 10 000 configurations sur lesquelles il a été testé. Les résultats obtenus, en terme de retards, sont donnés par le tableau 6.3.

Les retards entraînés sont légèrement plus importants qu'avec une base d'apprentissage fixe. Il semble qu'une meilleure capacité de généralisation des réseaux, en terme de respect des contraintes de séparation, est obtenue au prix d'une certaine *prudence* préjudiciable à l'efficacité en terme de retards.

TAB. 6.2 – Base fixe. Distribution des retards sur les configurations résolues.

Retards	Nombre d'occurrences
de 0 à 5 s	5 581
de 5 à 10 s	2 221
de 10 à 20 s	1 737
de 20 à 30 s	63
de 30 à 60 s	10
retard moyen	$5,1$ s
total	9 612

TAB. 6.3 – Base renouvelée. Distribution des retards sur les configurations résolues.

Retards	Nombre de configurations
de 0 à 5 s	4 771
de 5 à 10 s	2 060
de 10 à 20 s	2 666
de 20 à 30 s	353
de 30 s à 1 mn	132
de 1 à 2 mn	18
retard moyen	$7,5$ s
total	10 000

6.1.5 Résolution de conflits à trois avions

La génération de configurations de conflits à trois avions (ou clusters à trois avions) est beaucoup plus complexe qu'à deux avions. Le lecteur intéressé pourra se reporter à la thèse de Frédéric Médioni [Méd98] pour obtenir une présentation détaillée d'une approche possible.

Pour chaque avion, les deux autres avions impliqués dans le conflit seront appelés intrus. Un avion sera considéré par la suite comme un *intrus menaçant* si en suivant un cap direct vers sa destination, il entre en conflit avec l'avion courant (en supposant que ce dernier prenne également un cap direct vers sa destination).

Deux approches ont été envisagées pour adapter le problème à trois avions. La première consiste à utiliser un réseau de neurones *à un intrus* identique à celui utilisé pour les conflits à deux avions.

Deux stratégies sont alors possibles.

- La première stratégie consiste alors à considérer en entrée du réseau uniquement l'intrus menaçant le plus proche.
- La deuxième stratégie consiste à considérer en entrée du réseau l'intrus le plus proche, qu'il soit menaçant ou non.

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, ce qui paraît inévitable si l'on ne tient compte que d'un intrus.

La deuxième approche consiste à envisager un réseau prenant en entrée des paramètres décrivant la situation de l'avion courant par rapport aux deux intrus. Le réseau de neurones décrit dans la section 6.1.2 comportait, en plus d'une entrée fixée à 1.0, 8 entrées. 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 utilise donc 14 entrées.

6.1.6 Résultats obtenus sur les conflits à trois avions

Les trois méthodes décrites précédemment ont été testées en réalisant l'apprentissage successivement au moyen d'une base d'apprentissage fixe (de 100 conflits) et renouvelée (de 25 conflits). Les autres paramètres de l'algorithme génétique sont les mêmes que dans le cadre des conflits à deux avions.

Avec une base d'apprentissage fixe, les réseaux obtenus ont des capacités de généralisation très médiocres : sur $10\,000$ configurations de conflit créées aléatoirement, plusieurs centaines mènent à des violations des contraintes de séparation.

Contrairement à ce que l'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 totalement satisfaisants en terme de capacité de généralisation des réseaux (tableau 6.4).

En terme de retards, le tableau 6.5 montre que les retards engendrés peuvent devenir très importants.

6.1.7 Conclusion

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, permet, dans le cas de la résolution de conflit à deux avions, d'obtenir de bons résultats en terme de capacité de généralisation des réseaux de neurones, ce qui est très important.

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

TAB. 6.4 – Base d'apprentissage renouvelée : violations des contraintes de séparation ; nombre de configurations (sur 10 000) non résolues.

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

TAB. 6.5 – Base d'apprentissage renouvelée : retards engendrés sur les configurations résolues.

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

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 créées aléatoirement, mais parfois au prix d'un retard très important.

Les difficultés rencontrées sur des conflits à trois avions n'ont pas permis d'envisager une extension à 5 avions ou plus.

6.2 Méthode d'allocation de jetons et A^*

Il s'agit dans cette partie de détailler une méthode qui permet de coordonner un ordonnancement des avions dans un conflit avec une méthode de jetons, chaque avion choisissant ensuite sa trajectoire en évitant les conflits avec les plus prioritaires que lui.

Les premières études lancées sur ce thème ont été initiées au cours du DEA de Géraud Granger [Gra98].

On suppose que chaque avion peut communiquer à tout autre avion dans un rayon donné (dans la pratique, 90 milles nautiques) quelques informations supplémentaires. Les avions volent en route directe de l'origine à la destination. Les manœuvres de résolution proposées aux avions sont simples. Elles se limitent au plan horizontal : l'avion altère son cap de 10, 20 ou 30 degrés à droite ou à gauche de sa trajectoire à t_0 et reprend un cap vers sa destination à t_1.

Comme pour une approche centralisée, on fixe une fenêtre de détection et résolution F_a (de l'ordre de 5 minutes) et un temps de "rafraichissement" Δ (de l'ordre de la minute en pratique).

Chaque avion fait toutes les Δ minutes une détection de conflits, et prévoit une nouvelle trajectoire pour les F_a prochaines minutes. Lors de la recherche d'une nouvelle trajectoire, celle-ci ne peut pas être modifiée pendant Δ minutes. Si la modification intervient entre Δ 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 sera reconsidérée à l'étape suivante.

Si l'on suppose que les avions les plus rapides ont des vitesses voisines de 500 nœuds et si l'on prend le cas le plus défavorable du conflit en face à face, deux avions peuvent, en 5 minutes, se rapprocher de $\frac{1000\times 5}{60}$ milles nautiques soit environ 84 milles nautiques. Avec une séparation standard de 5 milles nautiques, un rayon de détection de 90 milles nautiques garantit que les avions ne peuvent pas être en conflit dans la fenêtre F_a.

Plus généralement, si V_{max} est la vitesse maximale possible pour les avions, si N_h est la norme horizontale et R_d le rayon de détection, on doit avoir :

$$R_d \quad \geq \quad 2\,V_{max} \times F_a + N_h$$

Le principe général de la méthode est le suivant : on définit d'abord un ordre de priorité entre les avions. Les avions les plus prioritaires ne modifieront pas leur trajectoire, puis les avions de priorité immédiatement inférieures construisent leur trajectoire en prenant pour contrainte la trajectoire des avions déjà présents, et ainsi de suite jusqu'à ce que tous les avions aient défini leurs trajectoires.

6.2.1 Ordonnancement des avions

Principe général

La coordination est séquentielle et basée sur un ordre de priorité entre les avions. Les avions qui ne sont en conflit avec aucun autre avion sont toujours prioritaires. Leurs trajectoires ne sont donc jamais modifiées et les avions voisins doivent en tenir compte pour optimiser leur trajectoire.

Les règles de l'air pour le vol à vue sont un exemple de règle de priorité qui pourrait être choisie pour des conflits à deux avions. Néanmoins, ces règles de priorité ne permettent pas de définir une relation d'ordre entre les avions. Par exemple, la transitivité n'est pas assurée par le principe de priorité à droite.

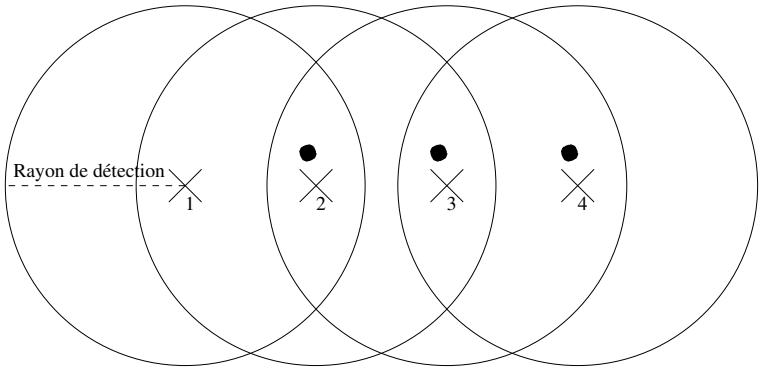

FIG. 6.3 – Visibilité entre 4 avions.

On peut, par exemple, définir une ordre total (certes simpliste) en utilisant les numéros CAUTRA[1] des avions (c'est d'ailleurs un ordre du même type qui est utilisé par le système d'anticollision TCAS[2]).

Ordonnancement et information partielle

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.

Considérons l'exemple de la figure 6.3. 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.

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 avion de savoir s'il doit trouver une trajectoire optimale car il est devenu le plus prioritaire, ou attendre que d'autres avions plus prioritaires que lui optimisent 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 : si un avion n'a pas de jeton, il peut déterminer une trajectoire optimale respectant les contraintes de séparation avec tous les avions de son rayon de détection n'ayant pas de jeton eux-mêmes en ne tenant pas compte des avions ayant des jetons. Une fois cette opération réalisée, il reprend tous les jetons qu'il a donnés (la consistance et la complétude de la méthode seront discutées dans la section 6.2.1). 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 qui n'ont pas de jeton. L'avion 2 ayant un jeton, l'avion 1 n'en

[1] Contrôle AUtomatisé du TRafic Aérien

[2] *Traffic alert and Collision Avoidance System*. Au détail près que c'est le code transpondeur qui est utilisé pour le TCAS.

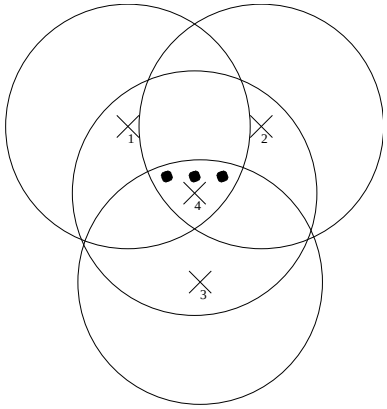

FIG. 6.4 – Répartition des jetons entre 4 avions.

tient pas compte et ne modifiera donc pas sa trajectoire. L'avion 1 récupère donc le jeton qu'il a donné à l'avion 2. C'est ensuite l'avion 2 qui doit optimiser sa trajectoire en tenant compte des avions de son rayon de détection n'ayant pas de jeton, à savoir, l'avion 1. L'avion 2 reprend ensuite son jeton à l'avion 3. 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 qu'a chaque avion n'indique pas le temps d'attente de chaque avion. Dans l'exemple de la figure 6.4, seul l'avion 4 doit attendre. Les avions 1, 2 et 3 optimisent simultanément leurs trajectoires et reprennent chacun leur jeton. L'avion 4 doit ensuite éviter les avions 1, 2 et 3.

Exemple détaillé

Ici (figure 6.5), 8 avions appartiennent au même *cluster*. L'avion $A7$ voit tout le cluster. L'avion $A8$ ne voit que les avions $A5$, $A6$ et $A7$. L'avion $A4$ voit les avions $A1$, $A2$ et $A7$. L'avion $A6$ voit les avions $A2$, $A3$, $A7$ et $A8$. L'avion $A2$ voit les avions $A1$, $A3$, $A4$, $A6$ et $A7$. L'avion $A3$ voit les avions $A2$, $A6$ et $A7$. L'avion $A1$ voit les avions $A2$, $A4$ et $A7$. L'avion $A5$ voit les avions $A7$ et $A8$. Les avions en conflit sont les paires $A1 - A2$, $A1 - A4$, $A2 - A3$, $A2 - A7$, $A3 - A6$, $A5 - A7$ et $A5 - A8$. L'avion le plus prioritaire est l'avion $A1$, le moins prioritaire est l'avion $A8$ ($A1 > A2 > A3 > A4 > A5 > A6 > A7 > A8$).

Dans l'exemple présenté, le tableau 6.6 donne la répartition des jetons au cours des 6 étapes successives que requiert la résolution. A l'étape 1, les avions $A1$ et $A5$ déterminent une trajectoire sans conflit en ne tenant compte d'aucun autre avion (en effet tous les avions de leur zone de détection ont déjà des jetons). $A1$ $A2$, $A4$ et $A7$ décrémentent alors leur nombre de jetons car $A1$ a résolu. $A7$ et $A8$ font de même car $A5$ a résolu. A l'étape 2, l'avion $A2$ n'a plus de jeton et modifie sa trajectoire pour résoudre le conflit avec l'avion $A1$ (0 jeton). $A3$, $A4$, $A6$, et $A7$ décrémentent alors leur nombre de jetons. A l'étape 3, l'avion $A3$ modifie sa trajectoire pour résoudre le conflit avec $A2$ (0 jeton), l'avion $A4$ modifie sa trajectoire pour résoudre le conflit avec $A1$ (0 jeton) en évitant l'avion $A2$ (0 jeton). $A6$ décrémente alors son total de jetons de 1 ($A3$) et $A7$ le décrémente de 2 ($A4$ et $A3$). A l'étape 4, l'avion $A6$ (0 jeton) modifie sa trajectoire pour résoudre le conflit avec $A3$ en évitant $A2$. $A7$ et $A8$ décrémentent leur total de 1. A l'étape 5, $A7$ résout les conflits avec $A2$ et $A5$ en évitant $A1$, $A3$, $A4$ et $A6$. $A8$ décrémente son total de 1. Enfin, à l'étape 6, $A8$ résout le conflit avec $A5$ en évitant $A6$ et $A7$.

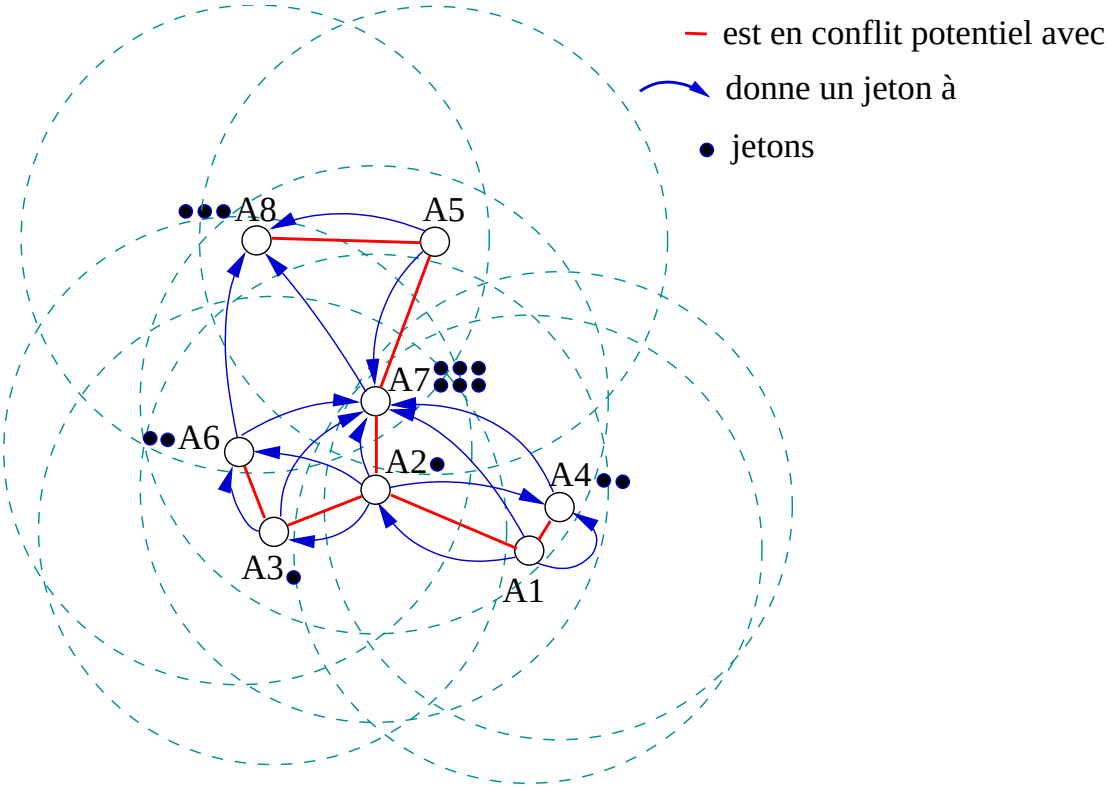

FIG. 6.5 – Exemple de cluster.

	Etape 1	2	3	4	5	6
A7	6	4	3	1	0	
A8	4	3	3	2	1	0
A4	2	1	0			
A6	2	2	1	0		
A2	1	0				
A3	1	1	0			
A1	0					
A5	0					

TAB. 6.6 – évolution de la distribution des jetons

Ce processus de distribution de jetons permet donc aux avions de se coordonner, tout en n'ayant qu'une vue partielle du *cluster*.

Complétude et consistance de la méthode

La technique de distribution - résolution - récupération de jetons décrite 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 le résoudre simultanément.

En effet, si l'ordre défini sur les avions est un ordre total, toute paire d'avions 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 choisi sa trajectoire, il reprend ses jetons. L'avion le plus prioritaire parmi les avions restants ne peut alors plus avoir de jeton et peut donc modifier sa trajectoire. Ce processus se répète jusqu'à ce que tous les avions aient repris leurs jetons. Lorsque 2 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 optimisé sa trajectoire. Deux avions en conflit ne peuvent donc pas se trouver sans jeton simultanément.

Remarque :

Le principe de distribution de jetons permet de donner une représentation mentale simple du fonctionnement de l'algorithme de résolution. Dans la pratique, aucune communication bilatérale entre avions n'est nécessaire. En effet, il suffit que chaque avion émette sa position, son numéro CAUTRA et son type d'appareil, ses éventuelles manœuvres en cours et sa destination. Ceci pourrait se faire par l'intermédiaire de l'*ADS-Broadcast*[3] par exemple. A partir de ces informations, et en supposant que les avions ont des horloges synchrones, tous les avions peuvent savoir combien de jetons ils ont reçus et distribués. A chaque étape de la résolution séquentielle, les avions manœuvrant émettent leurs nouvelles manœuvres. Chaque avion sait alors s'il doit ou non rendre un jeton et passer à l'étape de résolution.

Amélioration de l'ordre de priorité

L'utilisation d'un ordre aléatoire, comme par exemple l'utilisation du numéro CAUTRA de l'avion, n'est pas très efficace et peut être améliorée en tenant compte d'autres critères, tels que par exemple la manœuvrabilité des avions en conflit. L'ordre de priorité entre les avions est un élément tout à fait déterminant. Le lecteur intéressé par la recherche d'une bonne heuristique d'ordonnancement se reportera au mémoire de DEA de Nicolas Archambault [Arc03]. Il ressort de ce travail que l'ordonnancement des avions est très important et peut très rapidement limiter l'efficacité du système.

6.2.2 Résolution d'un conflit à l'aide d'un algorithme A *

Il faut trouver, pour un avion, une trajectoire libre de conflit en considérant n autres trajectoires comme contraintes. L'espace des solutions est décrit par trois variables, t_0, t_1 et α. Pour rechercher et évaluer dans cet espace de solutions, la meilleure trajectoire, on peut utiliser un algorithme de type A^*, tel que décrit par Thomas SCHIEX et Jean-Marc ALLIOT dans [AS92].

[3] *Automatic Dependent Surveillance Broadcast*

L'arbre de recherche

L'utilisation d'un algorithme de parcours d'arbre, comme l'algorithme A^*, pour la recherche d'une trajectoire optimisée, 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, elle contient la description des trajectoires des avions contraintes ainsi que le cap, la position et la destination de l'avion central (nous désignons par *avion central* l'avion qui doit établir sa nouvelle trajectoire).
- Les feuilles terminales de l'arbre correspondent aux états dans lesquels on a terminé les 5 minutes de simulation ou aux états dans lesquels l'avion central a atteint sa destination.
- Chaque branche de l'arbre représente une trajectoire de l'avion central. Toutes les trajectoires possibles peuvent être représentées par un chemin entre la racine de l'arbre et une des feuilles, mais l'heuristique permet de ne pas parcourir l'intégralité de l'arbre.
- Le coût d'un chemin sur l'arbre doit correspondre à la fonction que l'on cherche à minimiser. Dans notre cas, c'est la longueur de la trajectoire.

L'évitement

L'avion central doit établir sa trajectoire parmi d'autres avions qui ont déjà optimisé leur trajectoire ou qui, n'ayant pas de conflit, n'ont pas de résolution à effectuer. Tous ces avions sont dans le rayon de détection de l'avion central. Celui-ci n'aura qu'à conserver les normes de séparation avec les positions des avions voisins. L'algorithme doit trouver des trajectoires libres de conflits, tout en minimisant la longueur de la déviation. La plus courte de ces trajectoires est bien évidemment celle qui permet de rejoindre directement la destination, mais elle n'est pas nécessairement libre de conflit.

Développement d'un nœud de l'arbre

Ce paragraphe présente les différentes étapes d'une résolution de type point tournant. L'avion peut passer par 4 états différents. Durant le premier état (E_0), l'avion suit sa trajectoire prévue jusqu'au temps t_0. Pendant le deuxième état (E_1), à partir de t_0, l'avion est dévié de sa trajectoire d'origine, vers la droite ou vers la gauche d'un cap de plus ou moins 10, 20 ou 30 degrés. Ce cap est conservé jusqu'au temps t_1. Lors du troisième état (E_2), au temps t_1, l'avion prend le cap qui lui permet de rejoindre directement sa position finale. Enfin, dans le dernier état (E_3), l'avion a atteint sa position finale.

La nouvelle trajectoire de l'avion est donc entièrement déterminée par t_0, t_1 et le cap pris en déviation.

Ces différents états ne peuvent pas s'enchaîner de manière quelconque. Considérons les possibilités de déroulement d'une trajectoire d'évitement :

Lorsqu'un avion est encore dans l'état E_0, il a, à tous les pas de temps, le choix entre 7 actions possibles :

1. Poursuivre sa route en conservant son cap d'origine, en restant dans la première étape de l'évitement (E_0) ;
2. Modifier sa trajectoire en opérant un changement de cap de 10, 20 ou 30 degrés à droite ou 10, 20, ou 30 degrés à gauche, et ainsi passer dans le deuxième état de l'évitement (E_1) ;

Dans le cas où l'avion a atteint sa destination, il passe directement dans l'état (E_3).
Lorsque l'avion est dans l'état E_1, il a le choix entre 2 actions possibles :

1. Poursuivre sa route en conservant son cap modifié, en restant dans l'état (E_1) ;

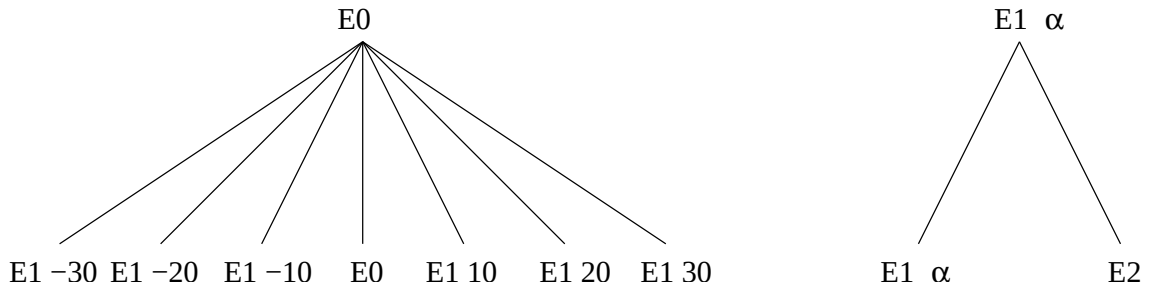

FIG. 6.6 – Développement des nœuds d'un arbre.

2. Opérer un virage pour suivre un cap direct vers sa position finale, il passe alors dans l'état (E_2).

Lorsque l'avion est dans l'état E_2, il n'a plus de choix possible. S'il a atteint sa destination, il passe automatiquement dans l'état E_3.

Les temps t_0 et t_1 prennent des valeurs discrètes (multiples du pas de temps). Les trajectoires d'évitement ainsi définies sont représentées par les chemins d'un arbre.

La fonction de coût calcule la distance, pour l'avion central uniquement, d'un nœud au nœud suivant. Elle tient aussi compte des séparations, et si celles-ci ne sont pas respectées, elle renvoie une valeur très importante, qui stoppe l'évaluation de la branche dans cette direction.

L'heuristique doit donner une estimation du coût du nœud courant à un état terminal. L'idéal est d'avoir une heuristique parfaite, ce qui n'est pas possible (il faudrait connaître par avance ce que l'on cherche à calculer). On peut, en revanche, choisir une heuristique minorante qui garantit que l'algorithme A^* trouvera la solution optimale. La fonction qui minore le coût entre le nœud courant et l'état terminal est la longueur qui sépare ces deux points. Le coût réel est nécessairement supérieur ou égal à cette distance, la ligne droite étant le plus court chemin entre 2 points.

L'efficacité de croisement

Le critère d'optimisation décrit précédemment ne prend pas en compte le problème lié à l'effet horizon. En effet, dans certaines situations de croisement, l'algorithme A^* propose une manœuvre qui repousse le conflit au-delà de la fenêtre de prévision temporelle, sans le résoudre. C'est ce que l'on appelle l'"effet horizon".

Afin de contrer cet effet, lors de la recherche de résolution, on vérifie si les avions en conflit sont dans une situation de croisement en début et en fin de fenêtre. Pour ce, on a besoin de connaître la destination finale des avions.

La fonction de coût est alors modifiée : si les avions suivent des trajectoires qui se croisent et qu'en fin de fenêtre ils ne se sont toujours pas croisés, une pénalité est ajoutée. Ainsi, une trajectoire qui permet de réaliser le croisement plus tôt sera préférée à une trajectoire qui le repousse.

En général, beaucoup plus de chemins sont explorés, engendrant de nombreux retours "en arrière" dans l'arbre. La profondeur de l'arbre dépend de la durée de la prévision. Pour une prévision de 5 minutes et un pas de discrétisation de 15 secondes, la profondeur est de 16.

6.2.3 Résultats sur 5 avions

On reprend l'exemple classique du conflit à 5 avions convergeant vers le centre d'un cercle. La figure 6.7 montre le résultat obtenu pour les ordres de priorité $(5 > 3 > 1 > 4 > 2)$ et $(1 > 2 > 3 > 4 > 5)$. Les résultats sont équivalents en terme de retards sur ces deux exemples. La figure 6.8

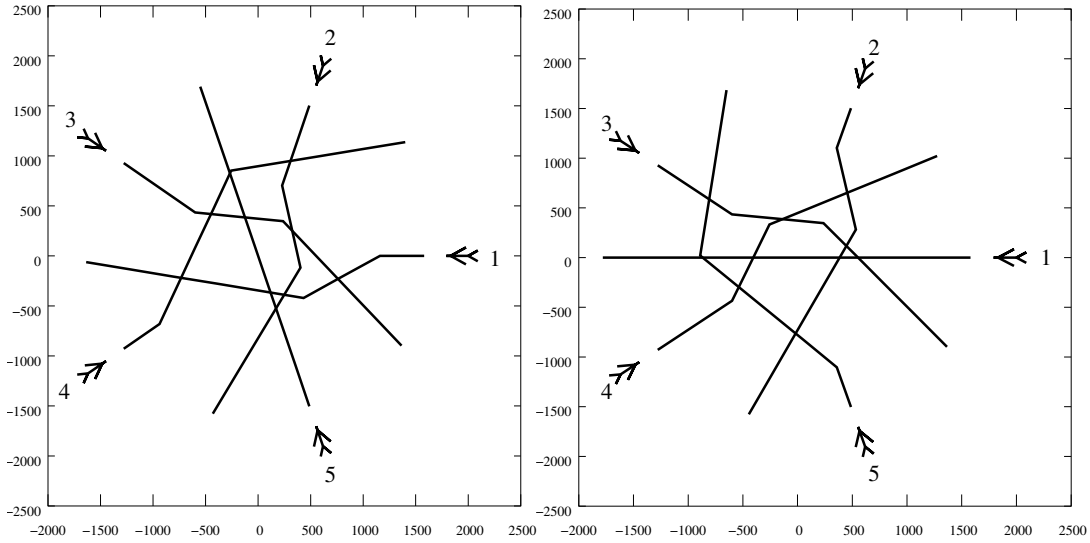

FIG. 6.7 – Résolution avec l'ordonnancement $(5 > 3 > 1 > 4 > 2)$ (gauche) et $(1 > 2 > 3 > 4 > 5)$ (droite).

montre un exemple d'échec de résolution pour un conflit à trois avions seulement. N'importe quel autre ordonnancement (non symétrique) permet de résoudre le conflit.

On entrevoit dans ces exemples la faiblesse de ce type d'approche, qui peut s'avérer efficace dans des espaces de faible densité, à condition d'avoir une bonne stratégie d'ordonnancement, mais ne peut concurrencer une méthode globale.

6.3 Conclusion

Les approches embarquées ou autonômes, que l'on a pu explorer dans ce chapitre, se sont montrées nettement moins efficaces que les approches centralisées pour résoudre des conflits de grande taille. La modélisation par un réseau de neurones ne semble pas pouvoir être facilement étendue à trois avions. La modélisation par allocation de jetons est cependant très intéressante car elle assure la coordination des manœuvres de résolution et pourrait s'avérer efficace dans des espaces de faible densité. Le choix de l'ordre de priorité reste toutefois un problème ouvert.

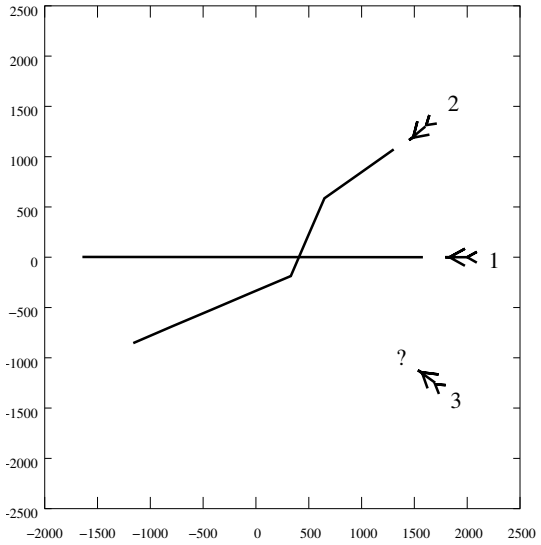

FIG. 6.8 – Echec de la résolution avec l'ordonnancement (1>2>3).

Conclusion générale

La gestion du trafic aérien est source de problèmes de grande taille, à variables mixtes, généralement très combinatoires et difficiles à modéliser tant les contraintes liées au contexte sont difficiles à appréhender. De plus, les problèmes rencontrés sont très liés entre eux. Il est par exemple difficile de séparer les problèmes de circulation sur une plate-forme aéroportuaire des problèmes de trafic en approche, tous deux dépendant de la gestion des créneaux de décollage. De même les problèmes d'organisation du réseau de routes aériennes a une forte incidence sur la sectorisation et la résolution de conflits, ….

Pour toutes ces raisons, la phase d'identification des problèmes et de modélisation est sans doute la tâche la plus difficile pour le chercheur. Il s'agit avant tout de respecter les contraintes opérationnelles essentielles, sans se noyer dans tous les détails, afin de pouvoir identifier des problèmes de taille raisonnable sur lesquels on va pouvoir appliquer des méthodes d'optimisation. Il faut en outre garder en permanence à l'esprit les contraintes initialement laissées de côté et veiller à ce qu'on puisse les réintégrer ultérieurement, une fois les méthodes de résolution identifiées. Dans ce contexte, on a pu observer, au travers des différentes approches du problème de résolution de conflits, que les différents algorithmes choisis imposent des modélisations qui ne permettent pas toujours la réintégration des contraintes opérationnelles. Il faut donc tenir compte non seulement de la performance de l'algorithme sur la modélisation choisie, mais également de sa capacité à intégrer les contraintes opérationnelles.

Une fois un problème identifié, on peut résumer l'approche de la façon suivante :

- modélisation mathématique du problème
- calcul de sa complexité
- recherche d'algorithmes adaptés
- validation des résultats obtenus, sur un plan expérimental par la simulation, sur un plan scientifique par la publication.

La validation des résultats obtenus sur le plan expérimental est une tâche très délicate et coûteuse en temps, la qualité des données dont nous disposons devant être vérifiée avec précaution. Le traitement des données brutes est souvent fastidieux.

La recherche d'algorithmes adaptés nous a entraînés dans divers domaines d'optimisation. Si le laboratoire s'est largement spécialisé dans les algorithmes génétiques, ou plus généralement évolutionnaires, nous tâchons de rechercher les méthodes d'optimisation les plus efficaces. Cela nous a conduits à aborder des méthodes aussi diverses que des méthodes d'intervalles, de réseaux de neurones, de parcours de graphe, ou de programmation semi-définie. L'hybridation de différentes méthodes s'avère également parfois très efficace. La grande difficulté est alors d'acquérir la compétence nécessaire dans l'emploi des différents algorithmes, afin de les mettre en œuvre de façon efficace, rendant ainsi les comparaisons pertinentes.

En pratique, on observe aujourd'hui que les algorithmes génétiques se sont révélés les plus performants pour la résolution de conflits aériens. D'une part, ils permettent de s'attaquer à de gros *clusters*, et d'autre part, ils permettent de conserver des profils de trajectoires réalistes. On ne peut néanmoins

envisager d'application opérationnelle concrète de tels algorithmes. Le principe d'allocation de jetons associé à une résolution *1 contre n*, bien que nettement moins performant, semble plus facilement intégrable dans des espaces de faible densité.

Estimer la portée des travaux réalisés est très difficile. Les retombées en matières d'études statistiques pour le Centre d'Etudes de la Navigation Aérienne sont certes nombreuses, mais elles exploitent peu les algorithmes trouvés. Si l'on prend l'exemple de l'optimisation de la circulation des aéronefs sur un aéroport, le travail de J.B. Gotteland en terme de modélisation est assez poussé, mais on reste encore loin d'une possibilité de mise en application pratique.

On se concentre désormais au LOG sur deux thèmes de recherche : le premier est l'organisation de l'espace aérien, en terme de réseau de routes, d'affectation de flux et de regroupement de secteurs. Nous poursuivons actuellement en thèse le travail de DEA de C.E. Bichot dont le sujet porte sur l'optimisation des regroupements de secteurs au niveau européen. Sur le plan algorithmique, ce sujet nous permet d'aborder plusieurs métaheuristiques parmi lesquelles le recuit simulé, les colonies de fourmis, utilisés seuls ou en hybridation avec des méthodes locales. Le second thème de recherche porte sur la résolution de conflits en vitesse. L'incertitude sur les vitesses des avions, dans le plan horizontal, et surtout dans le plan vertical, laisse une marge de manoeuvre pour résoudre des conflits sans modifier les trajectoires des avions. Actuellement, N. Archambault poursuit son travail de thèse sur le sujet. Ce travail nécessite une collaboration importante avec les avionneurs afin de déterminer les possibilités offertes par les FMS[4] des avions. Les nouveaux moyens de communication (*Data-Link*) et de navigation (GPS) disponibles ou prévus ouvrent en effet de nouvelles perspectives en matière de contrôle aérien. En outre, les efforts devraient porter sur l'amélioration de la prévision de trajectoire. Tous les travaux menés au sein du laboratoire depuis dix ans ont montré l'importance de la réduction des incertitudes sur la prévision de trajectoires.

Il me paraît essentiel de poursuivre des activités de recherche au sein de la Direction Générale de l'Aviation Civile, car c'est la seule façon d'appréhender l'environnement lié à la gestion du trafic aérien et d'accéder aux données opérationnelles. L'accès à ces données est indispensable pour vérifier par la simulation l'efficacité des méthodes développées. La survie d'une équipe de recherche ainsi intégrée est toutefois difficile, tant les attentes du monde de la recherche et du monde opérationnel sont différentes. Si la gestion du trafic aérien séduit généralement les chercheurs en tant que domaine d'application, initier le monde opérationnel à certaines démarches scientifiques est parfois plus long. En ce sens, il semble essentiel de poursuivre les efforts entrepris pour développer des équipes de recherche scientifique au sein des organismes de la navigation aérienne.

[4] *Flight Management Systsem*

Annexe A

Convergence théorique des AGs

Dans cette annexe sont présentées dans un premier temps la théorie des schémas de Goldberg [Gol89a], dont l'intérêt scientifique a été très largement mis en cause, mais qui présente à mon sens un intérêt "historique", et dans un deuxième temps une modélisation par chaînes de Markov, résultat des travaux de Cerf [Cer94]. On peut regretter que le travail de Cerf n'ait pas d'utilité pratique pour programmer un algorithme génétique.

A.1 Théorie des schémas

Historiquement, les algorithmes génétiques binaires sont les plus anciens et ont donc été les plus étudiés sur le plan théorique. Goldberg [Gol89a] a largement développé la théorie des schémas résumée dans [AS92] et résumée dans cette annexe.

A.1.1 Définitions fondamentales

Définition 14 (Séquence)**.** *On appelle séquence A de longueur $l(A)$ une suite $A = a_1 a_2 \dots a_l$ avec $\forall i \in [1, l], a_i \in V = \{0, 1\}$.*

En codage binaire, les chromosomes sont des séquences.

Définition 15 (Schéma)**.** *On appelle schéma H de longueur l une suite $H = a_1 a_2 \dots a_l$ 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.

Définition 16 (Instance)**.** *Une séquence $A = a_1 \dots a_l$ est une instance d'un schéma $H = b_1 \dots b_l$ si pour tout i tel que $b_i \neq *$ on a $a_i = b_i$.*

Ainsi, $H = 010 * 0101$ est un schéma et les séquences 01000101 et 01010101 sont des instances de H (ce sont même les seules).

Définition 17 (Position fixe, position libre)**.** *Soit un schéma H. On dit que i est une position fixe de H si $a_i = 1$ ou $a_i = 0$. On dit que i est une position libre de H si $a_i = *$.*

Définition 18 (Ordre d'un schéma)**.** *On appelle ordre du schéma H le nombre de positions fixes de H. On note $o(H)$ l'ordre de H.*

Par exemple, le schéma $H = 01**10*1$ a pour ordre $o(H) = 5$, le schéma $H' = *****101$ a pour ordre $o(H') = 3$. On remarquera qu'un schéma H de longueur $l(H)$ et d'ordre $o(H)$ admet $2^{(l(H)-o(H))}$ instances différentes.

Définition 19 (Longueur fondamentale)**.** *On appelle longueur fondamentale du schéma H la distance séparant la première position fixe de H de la dernière position fixe de H. On note cette longueur fondamentale $\delta(H)$.*

Ainsi, le schéma $H = 1**01*** a pour longueur fondamentale $\delta(H) = 5 - 1 = 4$, le schéma $H' = 1******1$ a pour longueur fondamentale $\delta(H') = 8 - 1 = 7$, et pour le schéma $H'' = **1*****$ nous avons $\delta(H'') = 3 - 3 = 0$.

Définition 20 (Adaptation d'une séquence)**.** *On appelle adaptation d'une séquence A une valeur positive que nous noterons $f(A)$.*

f est la fonction objectif ou *fitness* du problème à résoudre.

Définition 21 (Adaptation d'un schéma)**.** *On appelle adaptation d'un schéma H la valeur*

$$f(H) = \frac{\sum_{i=1}^{2^{(l(H)-o(H))}} f(A_i)}{2^{(l(H)-o(H))}}$$

où les A_i décrivent l'ensemble des instances de H, c'est-à-dire la moyenne des adaptations de ses instances.

A.1.2 Effets de la reproduction

Soit un ensemble $S = \{A_1, \ldots, A_i, \ldots, A_n\}$ de n séquences de bits tirées aléatoirement. Durant la reproduction, chaque séquence A_i est reproduite avec une probabilité :

$$p_i = \frac{f(A_i)}{\sum_{i=1}^{n} f(A_i)}$$

Supposons qu'il y ait à l'instant t un nombre $m(H,t)$ de séquences représentant le schéma H dans la population S. A l'instant $t+1$, statistiquement, ce nombre vaut :

$$m(H, t+1) = m(H,t).n.\frac{f(H)}{\sum_{i=1}^{n} f(A_i)}$$

Posons

$$\overline{f_t} = \frac{\sum_{i=1}^{n} f(A_i)}{n}$$

$\overline{f_t}$ représente la moyenne de l'adaptation des séquences à l'instant t. La formule précédente devient :

$$m(H, t+1) = m(H,t)\frac{f(H)}{\overline{f_t}}$$

Posons $c_t(H) = \frac{f(H)}{\overline{f_t}} - 1$. On obtient alors :

$$m(H, t+1) = (1 + c_t(H))m(H,t)$$

Il est donc clair qu'un schéma, dont l'adaptation est au-dessus de la moyenne, voit son nombre de représentants augmenter, suivant une progression qui est de type géométrique si nous faisons l'approximation que $c_t(H)$ est constant dans le temps. Alors :

$$m(H,t) = (1 + c(H))^t.m(H,0)$$

Si la reproduction seule était en jeu, les schémas forts élimineraient très rapidement les schémas faibles.

A.1.3 Effets des croisements

Nous allons nous intéresser à la probabilité de survie $p_s(H)$ d'un schéma H lors d'une opération de croisement. Par exemple, soit le schéma $H = **10*1**$. Supposons qu'une instance de ce schéma soit croisée avec une autre instance. Quelle est la probabilité pour que la séquence résultante soit encore une instance de H ? Il est impossible de répondre exactement à la question, tout au plus peut-on donner une borne inférieure de cette valeur. Il est clair que H ne sera pas détruit si le site de croisement qui est tiré au sort est inférieur à 3 (avant le premier 1) ou s'il est supérieur à 6 (après le dernier 1)[1].

On voit donc immédiatement qu'une borne inférieure de la probabilité de détruire un schéma H est $\delta(H)/(l-1)$. Donc la probabilité de survie dans un croisement est $1 - \delta(H)/(l-1)$. Si, d'autre part, on ne croise qu'une fraction p_c de séquences dans une population donnée, la probabilité de survie est donnée par :

$$p_s \geq 1 - p_c\frac{\delta(H)}{l-1}$$

De ce résultat et des résultats précédents découle la loi d'évolution d'une population[2] :

$$m(H,t+1) \geq m(H,t)(1 + c_t(H))(1 - p_c\frac{\delta(H)}{l-1})$$

A.1.4 Effets des mutations

Soit p_m la probabilité de mutation d'un bit dans une séquence. Dans un schéma H, seules les positions fixes peuvent être détruites. La probabilité de survie d'un bit étant $1 - p_m$, la probabilité de survie d'un schéma H contenant $o(H)$ positions fixes est $(1 - p_m)^{o(H)}$. La probabilité de mutation étant supposée petite devant 1, un développement limité au premier ordre donne une probabilité de survie égale à $1 - o(H)p_m$.

L'équation finale s'écrit donc :

$$m(H,t+1) \geq m(H,t)(1 + c_t(H))(1 - p_c\frac{\delta(H)}{l-1} - o(H)p_m)$$

A.1.5 Conclusion sur le codage binaire

Des calculs précédents découlent deux résultats :

– les schémas dont la longueur fondamentale est petite sont plus favorisés que les autres, lors de la génération d'une nouvelle population.

[1] Dans tous les autres cas, H peut être (ou ne pas être) détruit.

[2] Les croisements et les mutations se font sur la population reproduite, et non sur la population initiale.

– les schémas dont l'ordre est petit sont plus favorisés que les autres, lors de la génération d'une nouvelle population.

Ces résultats conditionnent la qualité du codage des données. En effet, les schémas qui codent les données "intéressantes" pour le problème considéré doivent avoir un ordre et une longueur fondamentales faibles[3], alors que les données "sans intérêt" doivent être codées par des schémas qui ont un ordre et une longueur fondamentale élevés.

Les résultats présentés ci-dessus ne sont qu'une introduction à la théorie des schémas, il existe de nombreux approfondissements. On trouvera dans les références suivantes les principaux modèles théoriques sur la théorie des schémas et ses extensions :[Gol89c, Vos91, BM93, Gol89b, BG87, CS88, BM94, DM92, SGE91]

A.2 Modélisation par chaînes de Markov

Cette dernière approche est la plus satisfaisante tant sur le plan mathématique, que sur celui de la modélisation, les différents opérateurs étant présentés comme "perturbant" un processus Markovien représentant la population à chaque étape. Ici encore il apparaît que seul l'opérateur de mutation est important, le croisement pouvant être totalement absent.

Les principaux résultats asymptotiques portant directement sur les algorithmes génétiques ont été développés par R. Cerf [Cer94] sur la base des travaux de Catoni [Cat90] et de Trouvé [Tro93]. Ces travaux se fondent sur la théorie des petites perturbations aléatoires d'un processus dynamique de type Markovien. Plus particulièrement, la théorie de Freidlin et Wentzell [FW83] constitue la pierre d'angle de ces études. Nous donnons ici quelques uns des résultats de la dynamique des algorithmes génétiques développés par Cerf. Nous les présentons simplifiés et dans un cadre restreint, laissant le lecteur intéressé se reporter à la difficile lecture des références citées.

Nous travaillerons sur la base d'un codage binaire, P représentant le nombre de bits utilisés pour le codage. La fonction d'évaluation, f, est définie sur l'espace $E = \{0,1\}^P$ à valeurs dans $\mathbb{R}^+$. Le problème est donc de localiser l'ensemble des maxima globaux de f , ou, à défaut, de trouver rapidement et efficacement des régions de l'espace où se situent ces maxima.

Comme nous l'avons vu, l'algorithme génétique est un algorithme stochastique itératif qui opère sur des ensembles de points, et qui est bâti à l'aide de trois opérateurs : mutation, croisement et sélection, que nous présentons plus formellement à présent.

A.2.1 Description formelle rapide d'un algorithme génétique

Soit N la taille (fixe) de la population, notons X_k la population de la génération k : il s'agit d'une matrice $X_k = (X_k^1, X_k^2, .. X_k^N)$ de E^N dont les N éléments sont des chaînes de bits (chromosomes) de taille P. Le passage de la génération k à la génération $k+1$, c'est-à-dire de X_k à X_{k+1} se décompose en trois étapes :

$$X_k \overset{Mutation}{\longrightarrow} Y_k \overset{Croisement}{\longrightarrow} Z_k \overset{Sélection}{\longrightarrow} X_{k+1}$$

Chacune de ces étapes peut être modélisée formellement.

– Mutation $X_k \longrightarrow Y_k$:

[3]Les données intéressantes sont bien entendu les données qui sont proches de la solution optimale. Un bon codage des données implique donc d'avoir une *idée* de la forme de l'optimum.

L'opérateur considéré ici est le suivant : pour chaque composante de chaque élément X_k^i, une variable de Bernouilli de paramètre P_m est tirée indépendamment et, suivant le résultat, l'élément binaire examiné est changé ou non. (0 est changé en 1 et 1 en 0).
La probabilité P_m de mutation doit être préalablement choisie et est généralement faible.
Comme nous le verrons par la suite, cet opérateur joue un rôle clé dans la convergence de l'algorithme génétique.

– Croisement $Y_k \longrightarrow Z_k$:
L'opérateur étudié ici est l'opérateur à un point (*slicing crossover*). Ici encore, la probabilité de croisement P_c est fixée initialement. Pour construire la population Z_k, $N/2$ couples sont formés à partir de la population Y_k (par exemple en appariant les individus consécutifs de Y_k, ou bien en choisissant au hasard et uniformément des individus dans Y_k). Pour chaque couple, une variable de Bernoulli de paramètre P_c est tirée pour décider si le croisement a lieu. Si c'est le cas, un site de coupure est tiré au hasard, et les segments finaux des deux chromosomes sont échangés.
Une nouvelle paire d'individus est ainsi obtenue (identique à l'ancienne s'il n'y a pas eu de croisement) et est stockée dans la population Z_k. En général, le paramètre P_c est choisi grand.
Remarquons que les opérateurs de mutation et de croisement ne font pas intervenir la fonction f, ce sont des opérateurs stochastiques d'exploration. C'est le troisième et dernier opérateur, la sélection, qui guide la population vers les valeurs élevées de la fonction f.

– Sélection $Z_k \longrightarrow X_{k+1}$
Les N individus de la population X_{k+1} sont obtenus après sélection des individus de Z_k. On conserve ainsi les "meilleurs" individus de Z_k, indépendamment à l'aide d'une distribution de probabilité qui favorise les individus de Z_k les mieux adaptés.
Le choix le plus fréquent est l'unique distribution telle que la probabilité d'un individu soit proportionnelle à son adaptation, i.e la probabilité de sélection de l'individu Z_k^i est :

$$P_i = P(Z_k^i) = \frac{f(Z_k^i)}{\sum_{j=1}^{N} f(Z_k^j)}$$

En tirant les individus dans la population Z_k conformément aux probabilités P_i, on constitue la nouvelle génération X_{k+1}.

A.2.2 Modélisation

La présentation rapide des opérateurs nous permet de modéliser la suite des $(X_k)_{k\in\mathbb{N}}$ en une chaîne de Markov, d'espace d'états $E = \left(\{0,1\}^P\right)^N$. L'algorithme génétique ne doit donc pas être interprété comme une procédure d'optimisation mais plutôt comme une marche aléatoire dans l'espace d'état, attirée vers les fortes valeurs de f .

La propriété première de cette formalisation est que la loi de X_k est déterminée de manière unique par :

– la loi de la génération initiale X_0
– le mécanisme de transition de X_k à X_{k+1}, mécanisme scindé en trois étapes détaillées précédemment.

Ce mécanisme de transition possède toutefois des propriétés essentielles qui font l'intérêt et la puissance de cette formalisation (voir [Cer94]) :

– Il est homogène (c'est-à-dire indépendant de la génération k considérée)

– Il est irréductible, la probabilité de joindre deux points quelconques de l'espace d'état, en un nombre fini de générations est non nulle soit :

$$\forall x, y \in E \qquad \exists r \in \mathbb{N} \qquad P\left[X_{k+r} = y \mid X_k = x\right] > 0$$

Le mécanisme permet donc d'explorer tout point de l'espace d'état, avec une probabilité non nulle.

– Il est apériodique, cette hypothèse n'est cependant pas fondamentale.

Ces propriétés permettent de conclure à l'ergodicité de la chaîne de Markov, et à l'existence d'un processus limite.

Théorème 2. *Une chaîne de Markov homogène irréductible apériodique d'espace d'états fini est ergodique et possède une unique mesure de probabilité stationnaire ou invariante.*

Cette mesure stationnaire correspond à la loi régissant l'équilibre du processus, elle est définie , pour tout y, comme :

$$\mu(y) = \lim_{k \to \infty} P\left[X_k = y \mid X_0 = x\right]$$

Nous savons également que tout élément de l'espace d'état est de probabilité non nulle pour cette mesure.

Toutefois, si ce résultat nous permet de savoir qu'il existe une dynamique de fond de l'algorithme génétique, il nous reste à en déterminer les propriétés, l'influence des opérateurs (et des paramètres associés) qui jouent un grand rôle dans le processus.

Pour cela nous introduisons les notations suivantes :

Si $x = (x_1, ..., x_N)$ est un élément de E^N et i un point de E, nous noterons :

$$\begin{aligned} \widehat{f}(x) &= \widehat{f}(x_1, ..., x_N) = \max\left\{f(x_i) : 1 \leq i \leq N\right\} \\ \widehat{x} &= \left\{x_k \in \arg\max f(x)\right\} \\ [x] &= \left\{x_k \ : \ 1 \leq k \leq N\right\} \end{aligned}$$

De manière générale, les lettres z, y, z, u, v.. désignent des populations, i.e. des éléments de E^N, et les lettres i, j des points de E.

Processus de fond (X_k^∞)

C'est à partir de ce processus de fond qu'est reconstitué l'algorithme génétique, en étudiant ses perturbations aléatoires par les différents opérateurs. Il est défini comme processus limite, lorsque les perturbations ont disparu. C'est également une chaîne de Markov sur E^N dont le mécanisme de transition est très simple, puisque correspondant à la situation limite suivante :

Les N composantes de X_{k+1}^∞ sont choisies indépendamment et suivant la loi uniforme sur l'ensemble $\widehat{X_k^\infty}$.

– Les individus dont l'adaptation n'est pas maximale en k, sont éliminés et n'apparaissent pas dans la génération $k+1$;

– Les individus dont l'adaptation est maximale ont des chances de survie égales.

Cette chaîne est tout d'abord piégée dans l'ensemble S des populations ayant la même adaptation (ou ensemble des populations d'*équi-adaptation*),

$$S = \left\{x = (x_1, ..., x_N) \in E^N \quad : \ f(x_1) = f(x_2) = \cdots = f(x_N)\right\}$$

Cette population représente les *attracteurs* de la chaîne (voir A.2.3 plus loin), puis elle est absorbée par une population uniforme, de sorte que :

$$\forall x \in E^N \qquad P\left[\exists x_i \in \widehat{x} \quad \exists K \quad \forall k \geq K \quad X_k^\infty = x_i \mid X_0^\infty = x_{ini}\right] = 1$$

Lorsque la population est devenue uniforme et en l'absence ici de perturbations, il ne se passe plus rien.

Ceci peut également se traduire en définissant les populations uniformes comme les *états absorbants* de la chaîne X_k^∞. Nous allons maintenant étudier la situation où ce processus est perturbé.

Processus perturbé (X_k^l)

La modélisation proposée par Cerf, part du processus de fond (X_k^∞), décrit ci-dessus, qui est perturbé aléatoirement, les perturbations sont indicées par le paramètre l. La chaîne de Markov (X_k^∞) devient donc une suite de chaînes de Markov (X_k^l), dont le mécanisme de transition est donné par la succession des transformations engendrées par les opérateurs.

$$X_k^l \overset{Mutation}{\longrightarrow} U_k^l \overset{Croisement}{\longrightarrow} V_k^l \overset{Sélection}{\longrightarrow} X_{k+1}^l$$

Il nous faut pour cela modéliser précisément les opérateurs.

– Mutations $X_k^l \longrightarrow U_k^l$:
Les mutations sont définies comme de petites perturbations aléatoires, indépendante des individus, de la population X_k^l. Il est assez naturel d'introduire la probabilité $p_l(i,j)$ de transition[4] de mutation entre les points i et j de E, comme un noyau Markovien p_l.
Trivialement on a :

$$\forall i \in E \qquad \sum_{j \in E} p_l(i,j) = 1$$

Sur la chaîne X_k^l, la probabilité de transition entre les points x et u de E^N est :

$$P\left[U_k^l = u \mid X_k^l = x\right] = p_l(x_1,u_1) \cdot p_l(x_2,u_2) \cdots p_l(x_N,u_N)$$

Plus précisément, et afin d'analyser la dynamique de (X_k^l) lorsque l tend vers l'infini, nous reportons ici les hypothèses sur le mode et la vitesse de convergence des probabilités de transition. Pour cela nous supposons l'existence d'un noyau irréductible, α, sur E, i.e. : $\forall\ i,j \in E$, $\exists\ i_o,\ i_1, \cdots,\ i_r$ (c'est-à-dire un chemin dans E) tel que $i_0 = i$ et $i_r = j$ tel que :

$$\prod_{0 \leq s \leq r-1} \alpha(i_k, i_{k+1}) > 0$$

L'hypothèse d'irréductibilité du noyau α est essentielle, elle assure que tout point de l'espace est potentiellement atteignable.
La vitesse de convergence du noyau p_l, est caractérisée par le réel positif $\mathbf{a}$, tel que p_l admette le développement suivant :

$$\forall i,j \in E \qquad \forall s \qquad p_l(i,j) = \begin{cases} \alpha(i,j) \cdot l^{-\mathbf{a}} + o(l^{-s}) & si\ i \neq j \\ 1 - \alpha(i,j) \cdot l^{-\mathbf{a}} + o(l^{-s}) & si\ i = j \end{cases} \tag{A.1}$$

[4] C'est la probabilité $P_l(i,j)$ pour un point i de E de se transformer par mutation en un point j de E

La condition de positivité de $\mathbf{a}$ nous permet de faire disparaître les perturbations, lorsque l tend vers l'infini.

$$\forall i,j \in E \qquad \lim_{l\to\infty} p_l(i,j) = \delta(i,j) = \begin{cases} 0 & si\ i \neq j \\ 1 & si\ i = j \end{cases} \tag{A.2}$$

– Croisement $U_k^l \longrightarrow V_k^l$: Ici encore l'opérateur est modélisé comme effectuant de petites perturbations aléatoires sur des couples de la population U_k^l. Ces couples sont ici formés par les éléments successifs de la population, les transitions sont gérées par le noyau Markovien q_l sur $E \times E$, cette fois, de sorte que :

$$P\left[V_k^l = v \mid U_k^l = u\right] = q_l\left((u_1,u_2)\cdot(u_3,u_4)\cdots(u_{N-1},u_N)\right)$$

Pour ce noyau q_l nous supposerons l'existence d'un noyau irréductible β sur $E \times E$, la vitesse de convergence est alors paramétrée par le réel positif $\mathbf{b}$ tel que :

$$\forall\,(i_1,j_1) \in E\times E \qquad \forall\,(i_2,j_2) \in E\times E \qquad \forall s$$

$$q_l\left((i_1,j_1),(i_2,j_2)\right) = \begin{cases} \beta\left((i_1,j_1),(i_2,j_2)\right)\cdot l^{-\mathbf{b}} + o(l^{-s}) \\ si\ \ (i_1,j_1) \neq (i_2,j_2) \\ \\ 1-\beta\left((i_1,j_1),(i_2,j_2)\right)\cdot l^{-\mathbf{b}} + o(l^{-s}) \\ si\ \ (i_1,j_1) = (i_2,j_2) \end{cases} \tag{A.3}$$

L'évanouissement asymptotique des croisements est également imposée par la positivité de $\mathbf{b}$:

$$\forall i_1,\ i_2,\ j_1,\ j_2 \in E \qquad \lim_{l\to\infty} q_l\left((i_1,i_2)(j_1,j_2)\right) = \delta(i_1,i_2)\cdot\delta(j_1,j_2) \tag{A.4}$$

– Sélection $V_k^l \longrightarrow X_{k+1}^l$: C'est l'opérateur le plus compliqué et également le plus important puisqu'il permet la convergence vers les optima de f.
Il est modélisé à l'aide d'une fonction de sélection F_l dont Cerf nous donne une définition précise, pouvant être résumée par :

$$\begin{aligned} F_l \quad : \quad \{1,\cdots,N\}\times(\mathbb{R}^+)^N \quad &\longrightarrow [0,1] \\ (i,\ f_1,f_2,\cdots f_N) \qquad &\longrightarrow F_l\left(i,\ f_1,f_2,\cdots,f_N\right) \end{aligned}$$

telle que :

1. $F\left(\cdot,f_1,f_2,\cdots f_N\right)$ est une probabilité sur $\{1,\cdots,N\}$
2. Cette probabilité est indépendante de l'indexation des $f_1,f_2,\cdots f_N$ (on peut permuter les f_i)
3. La probabilité favorise les éléments i associés à des valeurs élevées (i.e.)

$$Si \quad f_1 \geq f_2 \geq \cdots \geq f_N \qquad Alors$$

$$F_l\left(1,\ f_1,f_2,\cdots,f_N\right) \geq F_l\left(2,\ f_1,f_2,\cdots,f_N\right) \geq \cdots \geq F_l\left(N,\ f_1,f_2,\cdots,f_N\right)$$

Cet outil nous permet d'écrire la probabilité de transition correspondant à la dernière étape.

$$P\left[X_{k+1}^l = x \mid V_k^l = v\right] = \prod_{r=1}^{N} \Upsilon_l(x_r, v_r)$$

Ceci signifie que la probabilité de transition est le produit des probabilités sur chacune des N composantes de E.
La probabilité Υ_l entre deux composantes (x_r, v_r) est donnée par :

$$\Upsilon_l(x_r, v_r) = \sum_{k\,:\,v_k = x_k} F_l\left(k,\ f(v_1), f(v_2), \cdots, f(v_N)\right)$$

De même que pour les autres opérateurs, la fonction de sélection doit être choisie et sa vitesse de convergence caractérisée :

$$F_l\left(i,\ f_1, f_2, \cdots, f_N\right) = \frac{\exp\left(\mathbf{c} \cdot f_i \cdot \ln(l)\right)}{\sum_{r=1}^{N} \exp\left(\mathbf{c} \cdot f_r \cdot \ln(l)\right)} \tag{A.5}$$

Ce choix correspond bien à une probabilité de sélection avantageant les fortes adaptations au détriment des faibles, le réel positif c indexant cette fonction.
Le mécanisme de sélection opérant sur le processus de fond (X_k^∞), correspond à la fonction de sélection F_∞ définie par :

$$F_\infty\left(k, f(x_1), f(x_2), \cdots, f(x_N)\right) = \frac{\mathbf{1}_{\widehat{x}}(x_k)}{card(\widehat{x})}$$

c'est-à-dire, la loi uniforme sur l'ensemble $\widehat{x} = \{x_k \in \arg\max f(x)\}$
La suite $(F_l)_{l\in\mathbb{N}}$ des fonctions de sélection tend vers cette loi uniforme

$$\begin{array}{l} \forall x \in E^N \qquad \forall k \\ \\ \lim_{l\to\infty}\ F_l\left(k, f(x_1), f(x_2), \cdots, f(x_N)\right) = F_\infty\left(k, f(x_1), f(x_2), \cdots, f(x_N)\right) \end{array} \tag{A.6}$$

Les conditions A.2, A.4, et A.6 nous permettent d'assurer que le mécanisme de transition de la chaîne (X_k^l) converge vers celui du processus de fond (X_k^∞) :

$$\forall y, z \in E^N \qquad \lim_{l\to\infty} P\left[X_{k+1}^l = z \mid X_k^l = y\right] = P\left[X_{k+1}^\infty = z \mid X_k^\infty = y\right]$$

C'est également en ce sens que l'on interprète la chaîne (X_k^l) comme une perturbation de la chaîne (X_k^∞).

Les vitesses de convergence intervenant dans chacun des opérateurs jouent un rôle important. La formulation proposée en (A.1), (A.3) et (A.5), permet un ajustement équitable de ces vitesses (elles sont logarithmiquement du même ordre) de sorte qu'aucun opérateur ne domine les autres dans la dynamique. Lorsque l tend vers l'infini, les conditions (A.2), (A.4), et (A.6) nous permettent d'assurer que le mécanisme de transition de la chaîne (X_k^l) converge vers celui du processus de fond (X_k^∞), et on a :

$$\forall y, z \in E^N \qquad \lim_{l\to\infty} P\left[X_{k+1}^l = z \mid X_k^l = y\right] = P\left[X_{k+1}^\infty = z \mid X_k^\infty = y\right]$$

La chaîne (X_k^l) se comporte alors comme le ferait (X_k^∞). La théorie de Freidlin-Wentzell nous donne les outils pour simplifier l'étude de ces processus à temps continu.

A.2.3 Application de la théorie de Freidlin et Wentzell

Principe

Soit le système différentiel de $\mathbb{R}^N$ satisfaisant les équations déterministes :

$$\begin{cases} dx_t = b(x_t)\ dt \\ x_0 = x_{ini} \end{cases} \tag{A.7}$$

Sous de "bonnes" hypothèses, il existe une solution (trajectoire) unique, $x(t)$ à l'équation (A.7) et à la condition initiale associée. l'une des préoccupation immédiates est de savoir si cette solution va, ou non, tendre vers un équilibre (qui n'est pas forcément unique). Et, si oui, quel en est l'ensemble de stabilité. L'équilibre est défini comme une fonction constante x^* telle que $x^* = \lim_{t\to\infty} x_t$, et l'ensemble de stabilité comme l'ensemble $K(\ x^*)$ des points de départ qui mènent à cet équilibre[5]. On peut élargir ces notions d'équilibre et de stabilité par celles, très proches, d'attracteur et de bassin d'attraction.

Un attracteur du système est le voisinage compact K_i d'un point visité une infinité de fois, et le bassin d'attraction l'ensemble des points de départ qui mènent à cet attracteur. Nous supposerons que $\mathbb{R}^d$ possède un nombre fini d'attracteurs $K_1, \cdots, K_r$.

La théorie de Freidlin-Wentzell étudie l'évolution du système A.7 lorsqu'il subit des perturbations Browniennes, d'intensité ε. Le système déterministe A.7 devient alors un système différentiel stochastique.

$$\begin{cases} dX_t^\varepsilon = b(X_t^\varepsilon)\ dt + \varepsilon d\omega_t \\ X_0^\varepsilon = x_{ini} \end{cases} \tag{A.8}$$

Le processus $(X_t^\varepsilon)_{t\in\mathbb{R}^+}$ est maintenant un processus stochastique perturbé par le mouvement brownien $(\omega_t)_{t\in\mathbb{R}^+}$ et dépendant de ε. La situation change alors puisque les perturbations browniennes permettent au processus de s'échapper de n'importe quel bassin d'attraction, et en fait le processus les visite tous.

De plus, le processus est ergodique et admet une unique mesure de probabilité invariante, i.e.

$$\forall \mathcal{B}\ borelien\ \in \mathbb{R}^d \qquad \lim_{t\to\infty} P\left[X_t^\varepsilon \in \mathcal{B} \mid X_0^\varepsilon = x_{ini}\right] = \mu^\varepsilon(\mathcal{B})$$

existe et $\mu^\varepsilon(\mathcal{B})$ est la probabilité de présence du processus dans le Borélien $\mathcal{B}$, lorsque le système a atteint son état d'équilibre. Cette probabilité μ^ε est invariante avec le point de départ x_{ini}.

Lorsque les perturbations cessent, le processus se comporte comme dans A.7 et reste presque sûrement au voisinage $V(K_1 \cup \cdots \cup K_r)$ des attracteurs, tandis que la probabilité de présence dans n'importe quel Borélien $\mathcal{A}$ disjoint de $K_1 \cup \cdots \cup K_r$ disparaît.

$$\lim_{\varepsilon\to 0} \mu^\varepsilon\left(V(K_1 \cup \cdots \cup K_r)\right) = 1$$

$$\lim_{\varepsilon\to 0} \mu^\varepsilon\left(\mathcal{A}\right) = 0$$

[5] L'ensemble de stabilité de l'équilibre x^* est :

$$K(x^*) = \left\{ x_{ini} \in \mathbb{R}^N,\ t.q.\ pour\ x_t\ solution\ de\ A.7\ ;\ \lim_{t\to\infty}\ x_t = x^* \right\}$$

Pour chaque équilibre on définit ainsi son ensemble de stabilité. Cet équilibre est stable s'il contient un voisinage de l'équilibre, et instable s'il existe des points de départ infiniment proches de l'équilibre qui ne mènent pas à celui-ci.

Le résultat principal de Freidlin et Wentzell repose sur l'équivalence du processus $(X_t^\varepsilon)_{t\in\mathbb{R}^+}$ à temps continu et espace d'état $\mathbb{R}^d$ et du processus $(Z_n^\varepsilon)_{n\in\mathbb{N}}$ à temps discret et espace d'état fini $\{1,\cdots .r\}$ décrivant les visites au $n$ième attracteur.

La construction précise de $(Z_n^\varepsilon)_{n\in\mathbb{N}}$, n'est pas reportée ici mais nous en donnons un aperçu afin de mieux comprendre ce dernier processus.

– Si x_{ini} est "proche" de l'attracteur K_h alors $Z_o^\varepsilon = h \in \{1,\cdots .r\}$
– puis le processus, sous l'influence de (ω_t), est attiré par K_s et $Z_1^\varepsilon = s$
– etc..

La chaîne de Markov[6] ainsi créée a pour espace d'états $\{1,\cdots .r\}$, est irréductible, et possède une unique mesure de probabilité invariante ν^ε.

Théorème 3. *L'étude du comportement asymptotique de la mesure μ^ε est "équivalente" à l'étude du comportement asymptotique de la mesure ν^ε*

Nous passons sous silence l'étude des probabilités de transition $P\left[Z_n^\varepsilon = i \mid Z_n^\varepsilon = j\right]$ de la chaîne $(Z_n^\varepsilon)_{n\in\mathbb{N}}$ qui s'écrivent comme des intégrales sur l'ensemble des fonctions ϕ qui lient les attracteurs K_i et K_j, laissant le lecteur intéressé se reporter à la lecture de Freidlin et Wentzell, ou de Cerf.

Notons toutefois que ces probabilités de transition s'écrivent :

$$P\left[Z_n^\varepsilon = i \mid Z_n^\varepsilon = j\right] \underset{\ln}{\sim} \exp -\frac{V(i,j)}{\varepsilon^2}$$

où $V(i,j) = \inf\{V(\phi), \phi(\cdot) \text{ continue } [0.1] \longmapsto \mathbb{R}^d, \phi(0) \in K_i, \phi(T) \in K_j\}$

et $V(\phi) = \int_0^1 \left|\phi(t) - \dot{b}(\phi(t))\right|^2 dt$ est une constante associée à ϕ et caractérisant sa vitesse de convergence.

La quantité $V(i,j)$ ou *coût de communication,* mesure le coût de passage de l'attracteur K_i à l'attracteur K_j.

Les intensités de transition de la chaîne $(Z_n^\varepsilon)_{n\in\mathbb{N}}$, nous ouvrent la voie pour déterminer la mesure invariante ν^ε.

Mesure invariante ν^ε

Les outils qui permettent de déterminer cette mesure invariante ont été développés, une nouvelle fois, par Freidlin et Wentzell, nous aurons besoin de certains d'entre eux.

Définition 22. *Soit i un élément de $\{1,...,r\}$. Un $i-$graphe sur $\{1,...,r\}$ est un graphe g sur $\{1,...,r\}$ possédant les propriétés suivantes :*

– *$\forall j \neq i$, le graphe g contient un arc unique issu de j*
– *Il existe un chemin dans g qui mène de j à i*
– *g ne contient pas d'arc issu de i*

Il s'agit donc d'un graphe sans cycles formé de chemins qui aboutissent en i. On note $G(i)$ l'ensemble des $i-$graphes sur $\{1,...,r\}$.

Définition 23. *La fonction d'énergie virtuelle W est la fonction de $\{1,...,r\}$ dans $\mathbb{R}^+$ définie par :*

$$\forall i \in \{1,...,r\} \qquad W(i) = \min_{g\in G(i)} \sum_{(\alpha\to\beta)\in g} V(\alpha,\beta)$$

[6] La nature Markovienne de ω_t, nous permet de montrer qu'il s'agit bien là d'une chaîne de Markov.

A cette fonction est associé l'ensemble W^* des minima globaux de W.

Finalement, la mesure invariante ν^ε est caractérisée par :

$$\forall i \in \{1, ..., r\} \qquad \nu^\varepsilon(i) \underset{\ln}{\sim} \exp - \frac{W(i) - W(W^*)}{\varepsilon^2}$$

où $W(W^*) = \min \{W(i) \ : \ i \in \{1, ..., r\}\}$.

Le comportement asymptotique de ν^ε (et par la même occasion de μ^ε) est donc connu : la mesure ν^ε se concentre sur les attracteurs dont l'indice est dans W^* et décroît vers zéro à la vitesse $\exp - \frac{Cste}{\varepsilon^2}$ pour les autres attracteurs. Il existe donc un sous-ensemble de W^* de l'ensemble des attracteurs sur lequel se concentre la mesure invariante du processus.

$$\lim_{\varepsilon \to 0} \lim_{t \to \infty} P \left[X_t^\varepsilon \in V \left(\bigcup_{i \in W^*} K_i \right) \mid X_0^\varepsilon = x_{ini} \right] = 1$$

La dynamique du processus peut donc être caractérisée.

Dynamique du processus

Dans sa thèse, Cerf nous donne une très claire interprétation de la hiérarchie des cycles qui caractérisent la dynamique du processus. Supposons que le processus soit initialement dans le bassin d'attraction de K_1. Il quitte K_1 au bout d'un temps fini. Parmi toutes les trajectoires de sortie, il en existe une plus "probable" que les autres, qui l'amène vers un nouvel attracteur ; par exemple K_2 puis, bientôt K_3. L'ensemble des attracteurs étant par hypothèse fini, le processus finit par revisiter un attracteur formant un cycle d'ordre 1 sur lequel le processus tourne longtemps, très longtemps. Englobons maintenant ces trois attracteurs dans une boîte. Comme toujours, les perturbations browniennes finissent par pousser le processus hors de cette boîte, et ici encore, il existe une trajectoire de sortie canonique qui fait tomber le processus dans un nouveau bassin d'attraction, ou plus généralement, dans un autre cycle d'ordre 1.

Les cycles d'ordre 1 sont aussi en nombre fini, et le processus finit par revisiter un cycle d'ordre 1 : un cycle d'ordre 2 est alors formé, dans lequel le processus reste piégé très longtemps. En continuant de la sorte, il est possible de construire toute une hiérarchie de cycles qui épuise l'ensemble des attracteurs et fournit une image très précise de la dynamique asymptotique du processus. A chaque transition entre cycles est associée une constante qui caractérise la difficulté de la transition.

Enfin, lorsque ε décroît avec le temps ($\varepsilon = \varepsilon(t)$ est une fonction de t qui tend en décroissant vers 0), nous obtenons un processus de Markov inhomogène (le mécanisme de transition dépend du temps).

- Si $\varepsilon(t)$ décroît très lentement, de sorte qu'à chaque instant la loi de X_t soit proche de l'état d'équilibre associé au niveau de perturbation $\varepsilon(t)$, la situation ne change pas fondamentalement. La loi limite correspond à la limite de la suite des lois d'équilibre.
- Si au contraire $\varepsilon(t)$ décroît très rapidement, le processus risque de rester piégé dans certains sous-ensembles d'attracteurs : plus précisément, dans la hiérarchie des cycles, certaines transitions ne pourront être effectuées qu'un nombre fini de fois, alors que d'autres, plus "faciles", seront réalisées une infinité de fois avec la probabilité 1. La loi limite dépend alors fortement du point de départ.

La hiérarchie des cycles permet ainsi de décrire les dynamiques possibles de (X_t) en fonction de la vitesse de décroissance de $\varepsilon(t)$.

Résultats de convergence

Lorsque l croît vers l'infini, les perturbations affectant le processus (X_k^l) diminuent de sorte que cette chaîne se comporte, presque sûrement, comme la chaîne (X_k^∞). Plus précisément, nous savons que les attracteurs de la chaîne (X_k^∞) sont les populations d'équi-adaptation S et les populations uniformes (attracteurs stables). La chaîne (X_k^l) va donc être attirée par ses attracteurs, en commençant par l'ensemble S.

La théorie de Freidlin et Wentzell nous permet de reporter cette étude sur celle de la chaîne des (Z_k^l) des visites successives de (X_k^l) à l'ensemble S. Nous poserons donc $Z_k^l = X_{T_k}^l$ où T_k est l'instant de la $k$ième visite de (X_k^l) dans S.

Les probabilités de transition de la chaîne (Z_k^l) sont estimées à l'aide des opérateurs définis en A.2.2 et selon le schéma développé ci-dessus. Les fonctions de coût de communication $V(i,j)$ et d'énergie virtuelle W sont définies et estimées.

Nous savons alors que la suite des mesures stationnaires de la chaîne (X_k^l) se concentre sur l'ensemble W^* des minima de W :

$$\forall x_{ini} \in E^N \qquad \lim_{l\to\infty} \lim_{k\to\infty} P\left[X_k^l \in W^* \mid X_0^l = x_{ini}\right] = 1$$

L'un des principaux résultats indique qu'il existe une taille de la population de (X_k^l), (taille critique) telle que les maxima de f soient atteints asymptotiquement avec la probabilité 1.

Taille critique

Supposons fixés l'espace d'état E, la fonction d'adaptation f, les noyaux de transition de mutation α et de croisement β, ainsi que les constantes positives gérant les trois opérateurs $\mathbf{a}$, $\mathbf{b}$, et $\mathbf{c}$.

Théorème 4. *(Cerf 1993)*

Il existe une valeur critique N^*, telle que lorsque la taille de la population de l'algorithme génétique dépasse N^*, l'ensemble f^* des maxima globaux de f, contient l'ensemble W^*.

Cette taille critique N^* , dépend fortement de l'espace d'état E, de la fonction d'adaptation f, des noyaux de transition de mutation α et de croisement β, ainsi que des paramètres $\mathbf{a}$, $\mathbf{b}$, et $\mathbf{c}$.

Une borne grossière, mais lisible de N^* est :

$$N^* \leq \frac{\mathbf{a}R + \mathbf{c}(R-1)\Delta}{\min(\mathbf{a},\ \frac{\mathbf{b}}{2},\ \mathbf{c}\delta)}$$

où :

- R est le nombre minimal de transition permettant de joindre deux points arbitraires de E par mutation
- Δ et δ sont des paramètres d'échelle :
  - $\Delta = \max\{|f(i) - f(j)| : i,j \in E\}$ paramètre mesurant les écarts maximaux de f
  - $\delta = \min\{|f(i) - f(j)| : i,j \in E,\ f(i) \neq f(j)\}$ mesurant les écarts minimaux de f.

Il est intéressant de relever que le résultat est obtenu sans faire intervenir l'opérateur de croisement, qui n'est donc pas indispensable. L'exploration par mutation et la sélection suffisent à assurer la convergence vers f^* ([Zhi91]).

Ce premier résultat nous indique que dès que $N \geq N^*$, la suite des mesures stationnaires de la chaîne (X_k^l) se concentre asymptotiquement sur f^* lorsque l tend vers l'infini. On peut dans une

étape suivante faire évoluer l , et donc l'intensité des perturbations, en fonction de la génération. Nous obtenons alors une chaîne de Markov inhomogène $(X_k^{l(k)})$ dont le mécanisme de transition dépend alors de la génération k.

Vitesse de convergence

Le principal problème est de savoir si cette chaîne inhomogène peut avoir un comportement proche de celui de la chaîne homogène (X_k^l), et si oui, sous quelles conditions. La vitesse de croissance de $l(k)$ vers l'infini, est bien évidemment, au centre du débat.

- Si $l(k)$ croît "lentement", alors la loi de X_k sera proche de la loi stationnaire $\mu^{\varepsilon(l(n))}$ de niveau de perturbation $\varepsilon(l(n))$ associé à $l(n)$.
- Si $l(k)$ croît "rapidement", alors X_k risque de rester piégé dans des bassins d'attraction ne correspondant pas aux maxima de f, l'intensité des perturbations devenant trop faible pour pouvoir s'en échapper.

La vitesse recherchée se situe entre ces deux extrêmes, permettant à X_k de s'échapper des "mauvais" bassins d'attraction (ne correspondant pas à des maxima de f) et de rester piégé dans le "bon" (celui des points de f^*).

La vitesse de convergence de la suite $l(k)$ est caractérisée par l'*exposant de convergence*[7], λ.

Définition 24. *L'exposant de convergence λ de la suite $l(k)$ est l'unique réel λ tel que :*

$$\sum_{k\in\mathbb{N}} l(k)^{-\theta} \quad \begin{array}{ll} \rightarrow & converge\ pour\ \theta > \lambda \\ \rightarrow & diverge\ pour\ \theta < \lambda \end{array}$$

Deux conditions pour la colonisation de f^* sont également données par Cerf, l'une nécessaire, l'autre suffisante.

Théorème 5. *Condition nécessaire pour la colonisation de f^**

Pour que :

$$\forall x_{ini} \in E^N \qquad P\left[\exists K \quad \forall k \geq K \quad [X_{T_k}] \subset f^* \mid X_0 = x_{ini}\right] = 1$$

c'est-à-dire, pour que la chaîne $Z_k = X_{T_k}$ des visites successives des attracteurs soit piégée dans f^* après un nombre fini K de transitions, il est nécessaire que l'exposant de convergence λ de la suite $l(k)$ appartienne à l'intervalle $]\phi, \psi[$.

Les constantes ϕ et ψ sont des caractéristiques du problème l'intervalle $]\phi, \psi[$ est alors non vide pour N assez grand.

Théorème 6. *Condition suffisante pour la colonisation de f^**

Il existe deux constantes η et ρ telles que si l'exposant de convergence λ de la suite $l(k)$ appartient à l'intervalle $]\eta, \rho[$, alors :

$$\forall x_{ini} \in E^N \qquad P\left[\exists K \quad \forall k \geq K \quad [X_{T_k}] \subset f^* \ , \ \widehat{X_k} \subset f^* \mid X_0 = x_{ini}\right] = 1$$

ce qui signifie qu'après un nombre fini de transitions, nous avons presque sûrement la situation suivante :

- *la chaîne X_{T_k} est piégée dans f^*,*
- *la population X_k contient toujours un ou des individus appartenant à f^*.*

[7] Egalement appelé *rayon de convergence.*

En guise de conclusion

D'autres résultats existent, tant dans le travail de Cerf, que dans la littérature citée dans cette annexe. Ils demandent cependant un investissement supplémentaire dans la compréhension des outils développés. Le but de cette annexe était de convaincre le lecteur que l'étude théorique des algorithmes génétiques donne (déjà) de substantiels résultats. De nombreuses interrogations demeurent cependant concernant les relations réelles entre les différents paramètres caractérisant l'algorithme génétique et les choix pratiques de ceux-ci. Dans ce domaine, la pratique devance encore la théorie, même si les mécanismes commencent à être plus clairs. Il reste également à étudier les nombreux raffinements (tels que le *scaling*, le *sharing*, le *clustering*, l'élitisme) indispensables dans la pratique. Je citerais pour conclure les *No Free Lunch Theorems* largement détaillés par Wolpert [WM97], qui formalisent le fait qu'il est impossible de trouver une modélisation ou une représentation qui s'adapte à tous les problèmes. En d'autres termes, pour chaque problème pour lequel on trouve un codage, des opérateurs de croisement et de mutation efficaces on peut fabriquer un problème qui rendra inneficace ce codage et ces opérateurs.

Annexe B

Rappels d'optimisation convexe

Les rappels d'optimisation convexe présentés dans ce chapitre permettent de préciser les algorithmes utilisés dans le paragraphe 5.4. On aborde dans un premier temps l'optimisation convexe sans contrainte avec l'algorithme de Newton modifié, la méthode du gradient conjugué et BFGS. Dans un deuxième temps, on aborde l'optimisation convexe sous contraintes d'inégalités avec la méthode proximale entropique et les méthodes de multiplicateurs.

B.1 Optimisation convexe sans contrainte

On notera dans la suite

$$\begin{aligned} g_k &= \nabla f(x_k)^T \\ y_k &= g_{k+1} - g_k \\ s_k &= x_{k+1} - x_k \\ F(x) &= \text{ le Hessien de } f \text{ en } x \end{aligned}$$

Pour minimiser les calculs, on peut utiliser le même Hessien en x_0 pour chaque nouveau point, au lieu de le recalculer. On posera

$$d_k = -[F(x_0)]^{-1} g_k$$

B.1.1 Algorithme de Newton modifié

1. Recherche linéaire de descente : chercher α_k tel que

$$f(x_k + \alpha_k d_k) \leq f(x_k)$$

2. Déplacement du point courant :

$$x_{k+1} = x_k - \alpha_k [F(x_0)]^{-1} \nabla f(x_k)^T$$

Cet algorithme ne sera efficace que si le Hessien ne change pas trop (les dérivées troisièmes sont petites), faute de quoi, le résultat risque tout simplement d'être faux. Ceci explique l'intérêt de trouver une bonne approximation de l'inverse du Hessien à chaque déplacement du point courant.

B.1.2 Méthode de gradient conjugué

On notera F_k l'approximation du Hessien et H_k l'approximation de son inverse. La formule de Taylor à l'ordre 1, appliquée au gradient de f, donne l'approximation suivante :

$$g_{k+1} - g_k \approx F(x_k)p_k$$

Les approximations du hessien F_k doivent vérifier :

$$H_{k+1}y_k = s_k \quad ou \quad y_k = F_{k+1}s_k$$

ainsi que la propriété : F_k symétrique $\Rightarrow F_{k+1}$ symétrique.
Pour cela on peut utiliser une correction de rang 1, c'est-à-dire :

$$F_{k+1} = F_k + z_k z_k^T \tag{B.1}$$

On en déduit immédiatement :

$$z_k < z_k, s_k >= y_k - F_k s_k \quad puis \quad < z_z, s_k >^2 =< y_k - F_k s_k, s_k >$$

et finalement :

$$F_{k+1} = F_k + \frac{1}{< y_k - F_k s_k, s_k >}(y_k - F_k s_k)(y_k - F_k s_k)^T \tag{B.2}$$

C'est une mise à jour de type gradient conjugué.

B.1.3 BFGS

Si l'on applique le même principe au rang 2, en développant la formule B.2 et en supprimant les termes non symétriques, on obtient :

$$F_{k+1} = F_k + \frac{1}{y_k^T s_k} y_k y_k^T - \frac{1}{s_k^T F_k s_k}(F_k s_k)(F_k s_k)^T$$

Pour pouvoir en déduire le Hessien inverse, on a besoin de la proposition suivante :

Proposition B.1.1. *Soit une matrice A non singulière, et $a, b \in R^n$ tels que $1 + < b, A^{-1}a > \neq 0$ alors*

$$[A + ab^T]^{-1} \quad = \quad A^{-1} - \frac{1}{1 + < b, A^{-1}a >}(A^{-1}a)(b^T A^{-1})$$

Par deux applications de cette formule, on déduit

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$$

où

$$\rho_k = \frac{1}{y_k^T s_k}, V_k = I - \rho_k y_k s_k^T$$

Vérifions que la propriété $y_k s_k^T > 0$ permet de transmettre la définie positivité de H_k à H_{k+1}. Soit $\sqrt{H_k}$ le facteur de Cholesky de H_k, alors :

$$x^T H_{k+1} x = \|\sqrt{H_k} V_k x\|^2 + \rho_k| < s_k, x >|^2 / y_k s_k^T$$

ce qui permet de vérifier la propriété.

L'algorithme BFGS peut être décrit de la façon suivante :

$$x_{k+1} = x_k - \lambda_k H_k g_k$$

où λ_k est un pas de descente, g_k est le gradient de la fonction à minimiser en x_k. L'approximation inverse du Hessien est calculée grâce à la formule :

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$$

où

$$\rho_k = 1/y_k^T s_k, V_k = I - \rho_k y_k s_k^T$$

et

$$s_k = x_{k+1} - x_k, y_k = g_{k+1} - g_k$$

En pratique, nous utilisons comme hessien inverse de départ H_0 la matrice identité multipliée par un coefficient compris entre 0 et 1.

Il existe plusieurs façons de calculer le pas de descente, la *Golden Section Search* s'est avérée très efficace pour notre problème.

L'algorithme suivant permet de trouver le minimum d'une fonction $\phi(.) : [a, b] \rightarrow R$ supposée différentiable et unimodale, c'est-à-dire possèdant un unique mimimum local.

Algorithme *Golden Section*

F désigne le nombre d'or : $\frac{1+\sqrt{5}}{2}$ et ϵ la précision recherchée.

- Poser $i = 0$, $[a_i, b_i] = [a, b]$, $l_i = b_i - a_i$
- Tant que $l_i > \epsilon$:

1. Poser :

$$\begin{aligned} a_i^{'} &= a_i + l_i(1-F), \\ b_i^{'} &= b_i - l_i(1-F), \end{aligned}$$

2. Si $\phi(b_i^{'}) \leq min\{\phi(a_i^{'}), \phi(b_i)\}$, alors poser :

$$\begin{aligned} a_{i+1} &= a_i^{'} \\ b_{i+1} &= b_i \end{aligned}$$

sinon poser :

$$\begin{aligned} a_{i+1} &= a_i \\ b_{i+1} &= b_i^{'} \end{aligned}$$

3. i :=i+1

Algorithme BFGS

- poser k=0
- Tant que $\|g_k\| > \epsilon$:
  1. Calculer :

$$\lambda_k = \text{argmin}_{\lambda \geq 0} f(x_k - \lambda H_k g_k)$$

  2. Calculer :

$$\begin{aligned} x_{k+1} &= x_k - \lambda_k H_k g_k \\ s_k &= x_{k+1} - x_k, y_k = g_{k+1} - g_k \\ \rho_k &= 1/y_k^T s_k, V_k = I - \rho_k y_k s_k^T \\ H_{k+1} &= V_k^T H_k V_k + \rho_k s_k s_k^T \end{aligned}$$

  3. i :=i+1 aller en **1.**

L'algorithme de recherche linéaire de descente a été modifié de façon à prendre en compte une fonction ϕ unimodale ou multimodale sur $[a, b]$. L'algorithme modifié recherche un minimum de ϕ qui peut être au delà de b.

Algorithme Recherche linéaire(a,b)

- Poser $a_0 = a$, $b_0 = b$, $i = 0$, $\alpha = 0.1$;
- Tant que $b_i - a_i > \epsilon$:
  1. Poser :

$$\begin{aligned} a_i^{'} &= a_i(l) + b_i(1-l) \\ b_i^{'} &= a_i(1-l) + b_i(l) \end{aligned}$$

2.

Si $\phi(a_i^{'}) < \phi(b_i^{'})$ alors $b_{i+1} := b_i^{'}$
Sinon si $\phi(b_i^{'}) \leq \phi(b_i)$ alors $a_{i+1} := a_i^{'}$
Sinon Recherche linéaire$(a_i, b_i + \alpha\,(b_i - a_i))$

3. $i := i + 1$

L'algorithme précédent est récursif et permet d'augmenter la taille de l'intervalle de recherche linéaire lorsque la borne b_i donne une valeur inférieure aux autres points observés. Dans la mesure où la fonction que l'on veut minimiser est coercive, le critère d'arrêt est nécessairement atteint. Dans la pratique nous prendrons $a = 0$ et $b =$ pas de BFGS. Cet algorithme nous permet de démarrer avec un intervalle de recherche dont la taille est suffisamment petite pour espérer trouver rapidement le minimum, tout en se réservant le droit d'augmenter la taille de l'intervalle.

B.2 Opti convexe sous contraintes d'inégalités

B.2.1 Méthode proximale entropique EPM

On s'intéresse désormais au problème suivant :

$$\min_{x \geq 0} f(x) \tag{B.3}$$

Pour pouvoir utiliser BFGS ou une autre méthode précédemment décrite sur ce problème, on peut intégrer la contrainte dans la fonction à optimiser, tout en conservant les propriétés de différentiabilité de la fonction modifiée. Cette idée résume le principe de EPM.

Algorithme EPM

1. Choisir $x_0 \in R^p_{++} := \{x \in R^p \ : x_j > 0, j = 1, \cdots, p\}$
2. Générer la suite x^k définie par :

$$x^k = \text{argmin}_{x \in R^p}\{f(x) + \lambda_k^{-1} d_\phi(x, x^{k-1})\},$$

où λ_k est une suite de nombre positifs et

$$d_\phi(x, y) := \sum_{j=1}^{p} y_j \phi(y_j^{-1} x_j)$$

est la distance "entropique" basée sur une fonction strictement convexe
$\phi : R \to (-\infty, +\infty]$ dont l'exemple fondamental est :

$$\phi_1(t) = tlog(t) - t + 1$$

3. Arrêter l'algorithme dès que $\|x_{k+1} - x_k\| < \epsilon$.

Nous supposerons dans la suite que la fonction dérivée de ϕ vérifie l'encadrement :

$$1 - (1/t) \leq \phi^{'}(t) \leq log(t) \; \forall t > 0$$

Notons $\sigma_n := \sum_{k=1}^{n} \lambda_k$, le théorème suivant donne l'estimation de la vitesse de convergence de la méthode EPM

Théorème B.2.1. *l'algorithme* EPM *a pour estimation de vitesse de convergence :*

$$f(x^n) - f(x) \leq \sigma_n^{-1}[H(x, x^0) - H(x, x^n)]$$

B.2.2 Méthodes de multiplicateurs

On s'intéresse désormais au problème plus général :

$$\inf\{f(x) \; : g(x) \leq 0, x \in R^n\} \tag{B.4}$$

où $f : R^n \to R$ et $g : R^n \to R^m$ sont convexes. Nous utilisons la notation $g(x) = (g_1(x), \cdots, g_m(x))^T$.
On peut se ramener au problème précédent de la façon suivante.
Le lagrangien associé à B.4 est

$$L(x,p) = \begin{cases} f(x) + \sum_{i=1}^{m} p_i g_i(x) & sip \geq 0 \\ -\infty & sinon \end{cases}$$

B.4 est équivalent à :

$$\inf_x \sup_{p^* \geq 0} L(x, p^*)$$

Le problème dual est défini par

$$\sup_{p^* \geq 0} \inf_x L(x, p^*)$$

On peut donc réécrire le dual B.5 en utilisant la fonctionnelle définie par $c(p) = \inf_x L(x,p)$ comme :

$$\max\{c(p) \; : p \geq 0\} \tag{B.5}$$

D'après les propriétés des $\inf$ et des $\sup$ nous obtenons :

$$\sup_{p^* \geq 0} \inf_x L(x, p^*) \leq \inf_x \sup_{p^* \geq 0} L(x, p^*)$$

autrement dit, la solution du problème dual, est inférieure à la solution du problème initial.

Sous certaines hypothèses, on peut avoir l'égalité le lecteur intéressé se reportera à [Roc70]. La méthode des multiplicateurs consiste à appliquer EPM sur le problème dual qui consiste exactement en une optimisation sur $\mathbf{R}^+$.

Pour décrire la méthode des multiplicateurs, nous avons besoin de la définition suivante.

Définition B.2.1. *Soit $f :\rightarrow \overline{R}$ une fonction numérique. Sa fonction conjuguée (au sens de Fenchel-Moreau) est la fonction $f^* : E^{'} \rightarrow \overline{R}$ définie par :*

$$f^*(x^*) = \sup_{x\in E}[< x^*, x > -f(x)]$$

La méthode de multiplicateurs relative à d_ϕ est : soit une suite de nombres positifs λ_k et un point de départ initial $p^0 \in R^m_{++}$, on créée la suite x^k, p^k par l'algorithme :

Algorithme de multiplicateurs

1. Choisir un point de départ $p^0 \in R^m_{++}$
2. Génération de la suite primale :

$$x^{k+1} \in \operatorname{argmin}_{x\in R^n} \left\{ f(x) + \lambda_k \sum_{i=1}^{m} p_i^k \phi^*(\lambda_k^{-1} g_i(x)) \right\} \tag{B.6}$$

3. Génération de la suite duale

$$p_i^{k+1} = p_i^k (\phi^*)^{'}(\lambda_k^{-1} g_i(x^{k+1})) = p_i^k (\phi^{'})^{-1}(\lambda_k^{-1} g_i(x^{k+1})),\ i = 1, \cdots m \tag{B.7}$$

Le théorème B.2.2 permet de prouver que la méthode des multiplicateurs n'est rien d'autre que EPM appliqué au problème dual.

$$p^0 \in R^m_{++} \tag{B.8}$$

$$p^{k+1} = \operatorname{argmax}_{p\geq 0}\{c(p) - \lambda_k d_\phi(p, p^k)\} \tag{B.9}$$

Ce théorème est une adaptation aux ϕ-divergences du théorème de Fenchel-Rockaffellar. La fonction conjuguée de d_ϕ avec la deuxième variable fixée est :

$$d^*_\phi(y, z) = \sum_{j=1}^{n} z_j \phi^*(y_j)$$

Théorème B.2.2. *Soient $h, d_\phi : E \rightarrow R \cup +\infty$, deux fonctions convexes. On suppose qu'il existe $x_0^* \in E^*$ tel que pour tout $z \in R^n_{++}$*

$$H \left\{ \begin{array}{l} h^*(x_0^*) < +\infty\ d^*_\phi(x_0^*, z) < +\infty \\ h^* \textit{ est continue en } x_0^* \end{array} \right\}$$

alors

$$\min_{x\in E}(h(x) + d_\phi(x, z)) = \sup_{x^*\in E'} (-h^*(-x^*) - d^*_\phi(-x^*, z))$$

De plus,les minima sont atteints aux points x et y satisfaisant

$$x \in \mathbf{R}^n_{++}, y = \nabla x \textit{ et } y_j + \phi^{'}(\frac{x_j}{z_j}) = 0 \tag{B.10}$$

On ne démontrera pas ce théorème, se reporter à [IST94], théorème 5.1, "A Moreau-like theorem with ϕ-divergences". Une conséquence directe du théorème B.2.2 est la proposition B.2.1, qui nous donne l'équivalence des suites B.6 - B.7 et B.8 - B.9.

Proposition B.2.1. *La suite x^k, p^k produite par B.6 - B.7 est équivalente à celle produite par B.8 - B.9.*

Le théorème B.2.1 assure la convergence du schéma B.8 - B.9. Il resterait à prouver que la solution est bien primale-réalisable, le lecteur intéressé se reportera à [IST94].

Nous choisirons $\phi = \phi_1$. La fonction conjuguée de ϕ_1 est :

$$\phi^*(s) = e^s - 1$$

Algorithme EMM

1. Choisir un point de départ $p^0 \in R^m_{++}$
2. Génération de la suite primale :

$$x_{k+1} \in \operatorname{argmin}_{x \in R}\{f(x) + \lambda_k \sum_{i=1}^{m} p_i^k e^{(\lambda_k^{-1} g_i(x))}\}$$

3. Génération de la suite duale

$$p_i^{k+1} = p_i^{k+1} e^{(\lambda_k^{-1} g_i(x_{k+1}))}$$

Bibliographie

[AD95] J.M. Alliot and N. Durand. A genetic algorithm to improve an othello program. *Lecture Notes in Computer Science, Artificial Evolution, Springer Verlag*, 1063, 1995.

[AL92] L. Angerand and H. LeJeannic. Bilan du projet SAINTEX. Technical report, CENA, 1992. CENA/R92009.

[AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows, Theory, Algorithms and Applications*. Prentice Hall, 1993.

[Arc03] N. Archambault. Optimisation du solveur de conflits du simulateur de trafic cats. Master's thesis, Paris VI, 2003.

[AS92] J.M. Alliot and T. Schiex. *Intelligence Artificielle et Informatique Théorique*. Cepadues, 1992. ISBN : 2-85428-324-4.

[BD93] A. Bertoni and M. Dorigo. Implicit parallelism in genetic algorithms. *Artificial Intelligence*, 61(2) :307–314, 1993.

[BG87] C.L. Bridges and D.E. Goldberg. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In *Proceedings of the Second International Conference on Genetic Algorithm*. ICGA, 1987.

[BG91] C.L. Bridges and D.E. Goldberg. An analysis of multipoint crossover. In *Proceedings of the Foundation Of Genetic Algorithms*. FOGA, 1991.

[BM93] T.N. Bui and B.R. Moon. Hyperplane synthesis for genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithm*. ICGA, 1993.

[BM94] T.N. Bui and B.R. Moon. Analysing hyperplane synthesis in genetic algorithms using clustered schemata. Technical Report Cse-94-026, Dep. of Comp. Sci. and Engineering, Penn. State University, March 1994.

[Cat90] O. Catoni. *Large deviations for Annealing*. PhD thesis, Université de Paris XI, 1990.

[Cel90] J.C. Celio. Controller perspective of AERA2. Technical report, MITRE, February 1990. MP-88W00015.

[Cer94] R. Cerf. *Une Théorie Asymptotique des Algorithmes Génétiques*. PhD thesis, Université Montpellier II (France), 1994.

[CJ91] R.J. Collins and D.R. Jefferson. Selection in massively parallel genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[CMMR87] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal unctions of continuous variables with the "simulated annealing" algorithm. In *Proceedings of the ACM Transaction and Mathematical Software*. ACM, 1987.

[CS88] R.A. Caruana and J.D. Schaffer. Representation and hidden bias : Gray versus binary coding for genetic algorithms. In *Proceedings of the Fifth International Conference on Machine Learning*, 1988.

[CW96] A.L. Corcoran and R.L. Wainright. Reducing disrupton of superior building blocks in genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.

[DA93] N. Durand and J.M. Alliot. Existing algorithms for collision avoidance, and what the future might hold. Technical report, CENA, 1993.

[Daw89] Richard Dawkins. *L'horloger aveugle*. Robert Laffont, 1989.

[DF98] V.N. Duong and P. Faure. On the applicability of the free-flight mode in european airspace. In *Procceedings of the 2nd USA/Europe Seminar*, 1998.

[DH97] V.N. Duong and E. Hoffman. Conflict resolution advisory in autonomous aircraft operations. In *Procceedings of the 16th IEEE Digital Avionics System Conference*, 1997.

[DHN+96] V.N. Duong, E. Hoffman, J.P. Nicolaon, L. Floc'hic, and A. Bossu. Extended flight rules to apply to the resolution of encounters in autonomous airborne separation. Technical report, Eurocontrol, 1996.

[DHN97] V.N. Duong, E. Hoffman, and J.P. Nicolaon. Initial results of investigation into autonomous aircraft concept (freer-1). In *Procceedings of the 1rst USA/Europe Seminar*, 1997.

[DM92] D. Dasgupta and D.R. McGregor. A structured genetic algorithm. Technical Report IKBS-8-92, Dep. of Computer Science. University og Strathclyde, Clasgow. UK, 1992.

[Dod99] P. Dodin. Résolution de conflits via la programmation semie-définie. Master's thesis, Paris VI, 1999.

[DPST03] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Métaheuristiques pour l'Optimisation Difficile*. Eyrolles, 2003. ISBN : 2-212-11368-4.

[Dur96] N. Durand. *Optimisation de Trajectoires pour la Résolution de Conflits en Route.* PhD thesis, ENSEEIHT, Institut National Polytechnique de Toulouse, 1996.

[FMF01] E. Frazzoli, Z.H. Mao, and E. Feron. Aircraft conflict resolution via semidefinite programming. *AIAA Journal of Guidance, Control and Dynamics*, 2001.

[FMT93] X. Fron, B. Maudry, and J.C. Tumelin. Arc 2000 : Automatic radar control. Technical report, Eurocontrol, 1993.

[FOW66] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley and sons. NY, 1966.

[FW83] M.I Freidlin and A.D Wentzell. *Random Perturbations of Dynamical Systems*. Springer-verlag, New-York, 1983.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. Pvm 3 user's guide and reference manual. Technical report, Oak Ridge National Laboratory, 1994.

[GL89] G.H. Golub and C.F. Van Loan. *Matrix computations*. The Johns Hopkins University Press, second edition, 1989.

[Gol89a] D. Goldberg. *Genetic Algorithms*. Addison Wesley, 1989. ISBN : 0-201-15767-5.

[Gol89b] D.E Goldberg. Genetic algorithms and walsh functions. part 1 and 2. *Complex Systems*, 3 :129–171, 1989.

[Gol89c] D.E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading MA Addison Wesley, 1989.

[Gol91] D.E Goldberg. Real-coded genetic algorithms, virtual alphabets and blocking. *Complex Systems*, 5 :139–167, 1991.

[Got04] J.B. Gotteland. *Optimisation du trafic au sol sur les grands aéroports*. PhD thesis, INPT, 2004.

[Gra98] G. Granger. Résolution de conflits embarquée dans les espaces de faibles densité. Master's thesis, INPT, 1998.

[Gra02] G. Granger. *Détection et résolution de conflits aériens : modélisation et analyse*. PhD thesis, Ecole Polytechnique, 2002.

[GT82] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In *Nonlinear Optimization 1981*. M. J. D. Powell, 1982.

[GT00] R. Gosh and C. Tomlin. Maneuver design for multiple aircraft conflict resolution. In *American Control Conference*, 2000.

[Han68] R.J. Hanson. Interval arithmetic as a closed arithmetic system on a computer. Technical report, Jet Propulsion Laboratory, 1968.

[Han92] E. Hansen. *Global Optimization using Interval Analysis*. Number 165 in Monographs and textbooks in pure and applied mathematics. Marcel Dekker, inc, New York, 1992.

[HKP91] J. Hertz, A. Krogh, and R Palmer. *Introduction to the theory of neural computation*. Addison Wesley, 1991.

[HN93] J. Horn and N. Nafpliotis. Multiobjective optimization using the nitched pareto genetic algorithm. Illigal Report 93005, University of Illinois at Urbana, 1993.

[Hol62] John Holland. Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machinery*, 3, 1962.

[Hol75] J.H Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan press, 1975.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, pages 359–366, 1989.

[HT95] R. Horst and H. Tuy. *Global Optimization, Deterministic Approaches*. Springler, 1995.

[ILO99] ILOG. *ILOG Cplex User-s Guide*, 1999.

[IR92a] L. Ingber and B. Rosen. Genetic algorithm and very fast simulated reannealing : a comparison. *Mathematical and Computer Modeling*, 16(1) :87–100, 1992.

[IR92b] L. Ingber and B. Rosen. Genetic algorithms and very fast simulated re-annealing. *Mathematical Computer Modeling*, 16(11) :87–100, 1992.

[IST94] A. N. Iusem, B. Svaiter, and M. Teboulle. Entropy-like proximal methods in convex programming. *Math. Oper. Res.*, 19 :790–814, 1994.

[JM99] V. M. Jimenez and A. Marzal. Computing the k shortest paths : A new algorithm and an experimental comparison. In Jeffrey Scott Vitter and Christos D. Zaroliagis, editors, *Proc. 3rd Worksh. Algorithm Engineering*, number 1668 in Lecture Notes in Computer Science, pages 15–29. Springer-Verlag, 1999.

[K+89] F. Krella et al. Arc 2000 scenario (version 4.3). Technical report, Eurocontrol, April 1989.

[Kah68] W.M. Kahan. A more complete interval arithmeticc. Lecture notes for a summer course, University of Michigan, 1968.

[KMS98] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2) :246–265, 1998.

[LeC87] Y. LeCun. *Modèles connexionnistes de l'apprentissage*. PhD thesis, Université Pierre et Marie Curie, Paris VI, 1987.

[MDA94] F. Medioni, N. Durand, and J.M. Alliot. Algorithmes génétiques et programmation linéaire appliqués a la résolution de conflits aériens. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 1994.

[MG92] S.W. Mahfoud and D.E. Goldberg. Parallel recombinative simulated annealing : A genetic algorithm. Illigal report 92002, University of Illinois, Urbana, IL 61801-2996, April 1992.

[MG94] C. Meckiff and P. Gibbs. PHARE : Highly interactive problem solver. Technical report, Eurocontrol, 1994.

[Mic92a] Z. Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs*. Springer-verlag, 1992.

[Mic92b] Z. Michalewicz. *Genetic algorithms+data structures=evolution programs*. Springer-Verlag, 1992. ISBN : 0-387-55387-.

[MJ91] Z. Michalewicz and C.Z. Janikov. Handling constraints in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithm*. ICGA, 1991.

[Moo66] R.E. Moore. *Interval Analysis*. Prentice Hall series in Automated Computation. Prentice Hall International, INC., London, 1966.

[MP43] W.S. McCulloch and W.A. Pitts. A logical calculus immanent in nervous activit. *Bulletin of mathematics and biophysics*, 5, 1943.

[Muh89] H. Muhlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[Méd98] F. Médioni. *Méthodes d'optimisation pour l'évitement aérien : systèmes centralisés, systèmes embarqués*. PhD thesis, Ecole Polytechnique, 1998.

[NFC+83] W.P. Niedringhaus, I. Frolow, J.C. Corbin, A.H. Gisch, N.J. Taber, and F.H. Leiber. Automated En Route Air Traffic Control Algorithmic Specifications : Flight Plan Conflict Probe. Technical report, FAA, 1983. DOT/FAA/ES-83/6.

[Nie89a] W.P. Niedringhaus. Automated planning function for AERA3 : Manoeuver Option Manager. Technical report, FAA, 1989. DOT/FAA/DS-89/21.

[Nie89b] W.P. Niedringhaus. A mathematical formulation for planning automated aircraft separation for AERA3. Technical report, FAA, 1989. DOT/FAA/DS-89/20.

[PA91] T.S. Perry and J.A. Adam. Improving the world's largest, most advanced system ! *IEEE Spectrum*, February 1991.

[Pes00] B. Pesic. Optimisation de la circulation des aéronefs au sol sur la plate-forme de roissy. Master's thesis, INPT, 2000.

[PFB02] L. Pallottino, E. Feron, and A. Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems*, 3(1) :3–11, 2002.

[PLG87] C. Pettey, M. Leuze, and J. Grefenstette. A parallel genetic algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, 1987.

[Pol97] E. Polak. *Optimization : Algorithms and consistent approximations*. Springer-Verlag, 1997.

[Roc70] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[RR84] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions*. Ellis Horwood, Chichester, 1984.

[RR95] H. Ratschek and J. Rokne. Interval methods. In Reiner Horst and Panos M. Pardalos, editors, *Handbook of Global Optimization*, volume 2 of *Nonconvex Optimization and Its Application*. Kluwer Academic Publishers, Dordrecht (NL), 1995.

[RSS95] C. Ravise, M. Sebag, and M. Schœnauer. Optimisation guidée par l'induction. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 1995.

[SGE91] R.E. Smith, D.E. Goldberg, and J.A. Earickson. *SGA-C : A C-language implementation of a Simple Genetic Algorithm*, May 1991. TCGA report No. 91002.

[SPF93] R.E. Smith, A.S. Perelson, and S. Forrest. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2) :127–149, 1993.

[SPSS83] E.M. Schuster, F.R. Petroski, R.K. Sciambi, and M. MC Stokrp. AERA 2 functional design and performance description. Technical report, MITRE, September 1983. MtR-83W136.

[SR94] M. Schoenauer and E. Ronald. Neuro-genetic truck backer-upper controller. In *Proceedings of WCCI 94, IEEE conference on evolutionary computation*. IEEE press, 1994.

[Tro93] A. Trouvé. *Parallélisation massive du recuit simulé*. PhD thesis, Université de Paris XI, 1993.

[VB96] A.L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38 :49–95, 1996.

[Vos91] M.D. Vose. Generalizing the notion of schema in genetic algorithms. *Artificial Intelligence*, 50 :385–396, 1991.

[WM97] D.H. Wolpert and W.G. Maready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1 :67–82, 1997.

[Wri91] A.H. Wright. Genetic algorithms for real parameter optimization. In *Proceeding of the Foundation Of Genetic Algorithms*. FOGA, 1991.

[YG93a] X. Yin and N. Germay. 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.

[YG93b] X. Yin and N. Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In C.R. Reeves R.F.Albrecht and N.C. Steele, editors, *In proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference, Insbruck Austria*. Springer-Verlag, 1993.

[Zeg93] K. Zeghal. Champs de forces symétriques : La logique d'un système anticollision coordonné. Technical report, ONERA, 1993.

[Zeg94] K. Zeghal. *Vers une théorie de la coordination d'actions, application à la navigation aérienne*. PhD thesis, Universite Paris VI, 1994.

[Zhi91] A.A. Zhigljavsky. *Theory of Global Random Search*. Kluwer Academic Plubishers, 1991.

Publications :

Chapitres de livres :

[] N. Durand and J.M. Alliot. Chapitre : *Algorithmes évolutionnaires*. Livre : *Intelligence Artificielle et Informatique Théorique*. Cepadues, (2 éd) :415–441, 2002. ISBN : 2-85428-578-6.

[] N. Durand and J.B. Gotteland. Chapitre : *Algorithmes génétiques appliqués à la gestion du trafic aérien*. Livre : *Métaheuristiques pour l'Optimisation Difficile*. Eyrolles, 267–294, 2003. ISBN : 2-212-11368-4.

Articles publiés dans une revue internationale ave comité de lecture :

[] J.M. Alliot and N. Durand. A genetic algorithm to improve an othello program. *Lecture Notes in Computer Science, Artificial Evolution, Springer Verlag*, 1063, 307–319, 1995.

[] F. Médioni, N. Durand, and J.M. Alliot. Air trafic conflicts resolution by genetic algorithms. *Lecture Notes in Computer Science, Artificial Evolution, Springer Verlag*, 1063, 370–383, 1995.

[] N. Durand, J.M. Alliot, and O. Chansou. Optimal resolution of en route conflicts. *ATC Quarterly*, 3(3) :139–161, 1996.

[] N. Durand, J.M. Alliot, and G. Granger. Faces : a free flight autonomous and coordinated embarked solver. *ATC Quarterly*, 8(2) :109–130, 2000.

[] N. Durand, J.M. Alliot, and F. Medioni. Neural nets trained by genetic algorithms for collision avoidance. *Applied Intelligence*, 13(3) : 205–213, 2000.

Articles publiés dans une revue nationale avec comité de lecture :

[] N. Durand and J.M. Alliot. Résolution de conflits de trafic aérien par algorithmes génétiques. *Nouvelle revue Aéronautique Astronautique*, 6 :27–35, 1996.

[] N. Durand, J.M. Alliot, and G. Granger. Peut-on supprimer le contrôle au sol ? *La recherche*, 57–61, avril 1999.

Communications à des colloques internationaux avec comité de lecture et proceedings :

[] N. Durand, N. Alech, J.M. Alliot, and Marc Schoenauer. Genetic algorithms for optimal air traffic conflict resolution. In *Proceedings of the Second Singapore Conference on Intelligent Systems*. SPICIS, 11 pages, November 1994.

[] N. Durand, J.M. Alliot, and J. Noailles. Collision avoidance using neural networks learned by genetic algorithms. In *Ninth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Fukuoka*, 8 pages, June 1996.

[] N. Durand, J.M. Alliot, and J. Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 11 pages, October 1996.

[] JM Alliot, JF Bosc, N Durand, and L Maugis. Cats : A complete air traffic simulator. *16th DASC*, 8 pages, October 1997.

[] J.M. Alliot, J.F. Bosc, N.Durand, and L.Maugis. An Experimental Study of ATM Capacity. In *1 ST U.S.A/EUROPE ATM R & D Seminar*, 27 pages, June 1997.

[] N. Durand and J.M. Alliot. Optimal resolution of en route conflicts. In *Procceedings of the 1rst USA/Europe ATM R& D Seminar*, 12 pages, June 1997.

[] N. Durand and J.M. Alliot. Genetic crossover operator for partially separable functions. In *Genetic Programming*, 8 pages, July 1998.

[] G. Granger, N. Durand, and J.M Alliot. FACES : a Free flight Autonomous and Coordinated Embarked Solver. In *2nd USA/Europe ATM R & D Seminar*, 10 pages, December 1998.

[] B. Bartolomé, N. Durand, J.M. Alliot, and M.L. Boucheret. Parallel turbo codes optimization. In *COST Spring'99 Workshop*, Neuchatel, Switzerland, 8 pages, May 1999.

[] N. Durand and J.M. Alliot. A combined Nelder-Mead simplex and genetic algorithm. In *Proceedings of GECCO'99*, 1 page, July 1999.

[] N. Durand, J.M. Alliot, and B. Bartolomé. Turbo codes optimization using genetic algorithms. In *1999 Congress on Evolutionary Computation*, Washington D.C., USA, 7 pages, July 1999.

[] B. Bartolomé, N. Durand, J.M. Alliot, and M.L. Boucheret. Turbo codes interleaver optimization. In *International Symposium on Turbo Codes*, Brest, 8 pages, September 2000.

[] B. Pesic and N. Durand. Aircraft ground traffic optimisation using a genetic algorithm. In *Proceedings of GECCO'01*, 8 pages, July 2001.

[] G. Granger, N. Durand, and J.M. Alliot. Optimal resolution of en route conflicts. In *4th ATM R & D Seminar*, 8 pages, December 2001.

[] G. Granger, N. Durand, and J.M. Alliot. Token allocation strategy for free-flight conflict solving. In *IJCAI'01*, 6 pages, August 2001.

[] J.B. Gotteland, E. Page, N. Durand, and J.M. Alliot. Aircraft ground traffic optimization. In *4th ATM R and D Seminar*, 8 pages, December 2001.

[] N. Durand, J.M. Alliot, and G. Granger. A statistical analysis of the influence of vertical and ground speed errors on conflict probe. In *Procceedings of the 4th USA/Europe R and D Seminar*, 8 pages, December 2001.

[] J.B. Gotteland and N. Durand. Genetic algorithms applied to airport ground traffic optimization. In *5th ATM R and D Seminar*, 8 pages, June 2003.

[] J.B. Gotteland and N. Durand. Genetic algorithms applied to airport ground traffic optimization. In *IEEE Conference on Evolutionnary Computation*, Canberra, 8 pages, December 2003. IEEE.

[] G. Granger and N. Durand. A traffic complexity approach through cluster analysis. In *5th ATM R and D Seminar*, 8 pages, December 2003.

[] D. Giannazza, N. Durand, and N. Archambault. Allocating 3d trajectories to air traffic flows using a* and genetic algorithms. In *Proceeding of the International Conference on Computational Intelligence for Modelling, Control and Automation*, 12 pages, July 2004.

Communications à des colloques nationaux avec comité de lecture et proceedings :

[] F. Medioni, N. Durand, and J.M. Alliot. Algorithmes génétiques et programmation linéaire appliqués a la résolution de conflits aériens. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 7 pages, septembre 1994.

[] N. Durand, J.M. Alliot, and J. Noailles. Algorithmes genetiques : un croisement pour les problemes partiellement separables. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 7 pages, septembre 1994.