N° d'ordre : 2341

THÈSE

présentée

pour obtenir le titre de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

École doctorale : Informatique et Télécommunications
Spécialité : Programmation et Systèmes

Par : Thomas Rivière

Optimisation de graphes sous contrainte géométrique : création d'un réseau de routes aériennes pour un contrôle Sector-Less

Soutenue le 24/05/2006, devant le jury composé de :

Pr.	Vu Duong	Université de Saïgon / Eurocontrol	(Président du jury)
Pr.	Joseph Noailles	ENSEEIHT	(Directeur de thèse)
Dr.	Philippe Baptiste	École Polytechnique	(Rapporteur)
Pr.	Vojin Tosic	Université de Belgrade	(Rapporteur)
Dr.	Pascal Brisset	ÉNAC	(Membre du jury)
Mr.	Patrick Dujardin	DSNA/DTI/SDER	(Membre du jury)

Remerciements

Tout a commencé un jour de projet où il s'est adressé à mon binôme et à moi-même en ces termes : *Ça ne vous dirait pas de faire un* DEA *?* Je ne suis pas encore tout à fait certain que cette phrase s'adressait également à moi... Après avoir tenté de se débarrasser de moi en m'envoyant de l'autre coté de la Manche, il a fallu qu'il se résigne, je suis revenu, sans même avoir commencé un doctorat. C'est donc tout d'abord à Pascal Brisset que vont mes remerciements, lui qui fut à l'origine de cette thèse et qui m'a encadré pendant toutes ces années.

Ensuite c'est à ma (future) femme que va ma plus grande gratitude, elle qui a non seulement accepté de quitter son pays pour que je puisse faire cette thèse, mais qui a en plus accepté sans faillir de relire le moindre de mes articles sans jamais avouer qu'elle ne comprenait rien à mes tournures de phrases.

Merci aussi à mon jury et à mes rapporteurs d'avoir accepté la tâche ingrate de relire et de juger mon travail, surtout une veille de week-end prolongé.

Je remercie également tout ceux qui ont contribué de près ou de loin à l'achèvement de ma thèse. Merci à Joseph Noailles mon directeur de thèse, pour sa gentillesse et ses superbes discours. Merci à Jean-Marc Alliot et Nicolas Durand de m'avoir accueilli à bras ouverts au Log alors que eux aussi pensaient être débarrassés de moi. Mon unique regret est qu'au moment où j'écris ses mots, la maison de Nicolas n'est toujours pas finie. Je n'aurai donc pas la fin d'une discussion qui nous aura tenu en haleine pendant plusieurs années. Merci à Vu Duong d'avoir bien voulu subventionner mes travaux ; si la thèse nourrit l'esprit, Eurocontrol nourrit le corps. Merci aux membres du Log et du Leea d'avoir su créer une ambiance si agréable que ce fut un réel plaisir de venir travailler le matin. Mention spéciale à Nicolas Barnier toujours prêt à me sortir d'un mauvais pas lorsque des végétariens débarquaient à la maison, à Jean-Baptiste Gotteland pour ses conseils en évitement de murs qu'il devrait mettre à profit, à Charles-Edmond Bichot même si comme le dit Kévin, on attend toujours ses mèls et à Kévin Guittet sans qui je n'aurais jamais gagné Roland-Garros et dont la capacité à faire des pizzas m'étonnera toujours. Enfin un grand merci à Nicolas Archambault, ingénieur système, grand spécialiste des tactiques souterraines, maître incontesté du loto-foot (même en jouant Marseille gagnant, c'est dire), pourfendeur de `sed` et accessoirement futur docteur qui fut pendant toutes ses années le compagnon de travail idéal, surtout pour martyriser les élèves.

Je passe enfin le flambeau à Cyril Allignol qui n'a pas encore la dextérité des thésards mais est déjà bien plus méchant que moi avec les élèves. Il fera donc une bonne thèse.

Résumé

Alors que le système de gestion du trafic aérien arrive à saturation, de nouveaux concepts sont étudiés afin de trouver une alternative. Cette thèse s'applique à vérifier la validité d'un de ces nouveaux concepts, le concept *Sector-Less* qui envisage un contrôle des aéronefs par flux, c'est-à-dire depuis leur aéroport de départ jusqu'à leur aéroport d'arrivée, par opposition au contrôle actuel effectué par zone géographique. La validation est faite à travers la construction et la validation d'un réseau de routes aériennes adapté à ce concept.

La définition de ce nouveau réseau est donnée uniquement en fonction des contraintes imposées par le concept *Sector-Less*, sans utiliser de base préexistante. Ce réseau initial de routes est optimisé par deux méta-heuristiques différentes, recuit simulé et algorithme génétique. Le processus d'optimisation vise à minimiser la longueur de la trajectoire pour chaque flux d'aéronefs. Il intègre en particulier des algorithmes dynamiques de calcul des plus courts chemins dans un graphe. Pour prendre en compte les capacités de navigation des aéronefs, deux techniques ont été utilisées pour intégrer au processus d'optimisation des contraintes géométriques. La première technique consiste à exprimer les contraintes géométriques directement à travers la structure du graphe et à utiliser les algorithmes classiques pour calculer les plus courts chemins. La seconde est basée sur un nouvel algorithme permettant, sans altérer le graphe, de calculer une approximation des plus courts chemins.

Le réseau de routes ainsi construit est testé du point de vue gestion du trafic aérien en simulant, à l'aide d'un simulateur de trafic en temps discret, le trafic dans l'espace aérien européen sur une journée complète. L'évaluation estime en particulier la charge de travail des contrôleurs aériens. L'analyse des résultats montre que le concept *Sector-Less*, tel qu'il est actuellement proposé, ne peut pas être mis en œuvre.

Abstract

Taking into consideration the fact that the current air traffic management system is becoming saturated, new concepts are being developed in order to find an alternative. The aim of this thesis is to verify the validity of one of these new concepts, the *Sector-Less* concept, which envisages the control of aircraft from their departure to their destination as opposed to the current control which is done geographically. The process of validation is performed by constructing and validating an air route network adapted to this concept.

The definition of this new network is designed from scratch, according to constraints imposed by the *Sector-Less* concept, without using any preexisting network. This initial route network is optimised using two different meta-heuristic, a simulated annealing algorithm and a genetic algorithm. The optimisation process tends to minimise the length of the trajectory of each flow. It integrates a graph dynamic shortest paths algorithm. Taking into account the aircraft navigation capacities, two techniques have been used in order to integrate geometric constraints with the optimisation process. The first technique consists of expressing geometric constraints directly through the graph structure and to use classic

dynamic shortest paths algorithms. The second one is based on a new algorithm allowing, without altering the graph, to calculate an approximation of the shortest path.

The route network constructed in this way is tested from an air traffic management point of view in simulating, using a discreet time air traffic simulator, European air traffic in the course of one day. The evaluation estimates in particular the workload of the air traffic controllers. The analysis of the results show that the *Sector-Less* concept, as it is currently proposed, cannot be put into effect.

Table des matières

Table des figures

Liste des algorithmes

Introduction

Malgré les différentes crises qui ont secoué le monde du transport aérien (attentats, SRAS) et celles qui sont toujours d'actualité (augmentation du cours du pétrole), l'espace aérien européen accueille toujours plus d'aéronefs et les prévisions de trafic annoncent encore et toujours que le domaine est en expansion puisqu'il est prévu pour les six prochaines années une croissance annuelle moyenne de 3,7% [Marsh 05].

Pour permettre d'absorber toujours plus de trafic dans le respect des règles de sécurité, les organismes de gestion et de contrôle du trafic aérien ont mit en place, au fil des ans, une certains nombre de mesures permettant également de maintenir un niveau de sécurité élevé : recrutement de personnels supplémentaires, restructurations de l'espace aérien, aides informatiques au contrôle accompagnées de nouvelles méthodes de travail afin d'augmenter la productivité des contrôleurs, etc. Néanmoins, les différents acteurs du transport aérien s'accordent aujourd'hui pour dire que le système actuel a atteint ses limites, que les retards existent et ne sont pas résorbés et qu'il faut faire évoluer le système.

Dans cette optique, différents concepts permettant de renouveler la gestion du trafic aérien ont été explorés :

- décharger à des degrés divers le contrôleur aérien de certaines tâches en les automatisant, par exemple en assistant le contrôleur dans sa tâche et lui désignant les aéronefs en conflit ;
- automatiser totalement le système et laisser les aéronefs négocier entre eux afin de s'éviter mutuellement ;
- optimiser la structure et la gestion de l'espace aérien européen ainsi que l'organisation des flux qui le traversent ;
- imaginer de nouvelles méthodes de contrôle du trafic aérien.

Le travail présenté dans cette thèse est centré sur ce dernier point et sur l'étude d'un nouveau concept nommé *Sector-Less*, imaginé par une équipe de l'agence Eurocontrol [Duong 01]. Afin de répondre aux problèmes posés par l'augmentation du trafic aérien, ce nouveau concept redéfinit radicalement la manière dont la gestion et le contrôle du trafic aérien sont réalisés. Dans ce concept, le rôle du contrôleur est totalement différent de celui qu'il a aujourd'hui. Au lieu d'avoir deux contrôleurs gérant un nombre limité d'aéronefs dans une zone géographique déterminée, le concept *Sector-Less* prévoit qu'un contrôleur est responsable d'un certain nombre d'aéronefs depuis leur aéroport de départ jusqu'à leur destination.

Plan du document

Ce document a pour but d'étudier la viabilité de ce nouveau concept. Ce dernier n'étant défini que dans ses grandes lignes par ses concepteurs, son évaluation passe tout d'abord par la définition et la création d'éléments pratiques et notamment d'un réseau de routes aériennes.

Ce document se divise en deux parties. La première partie présente les principes généraux du contrôle aérien, les différents concepts existants qui visent à faciliter la gestion d'un nombre toujours croissant d'aéronefs et plus particulièrement le concept qui nous intéresse, et enfin la modélisation de celui-ci.

La seconde partie présente la contribution apportée par cette thèse. Partant d'un réseau initial de routes adapté au concept *Sector-Less*, celui-ci est optimisé de diverses manières afin de fournir aux avions le chemin le plus court possible puis évalué du point de vue de la gestion du trafic aérien.

Le chapitre 4 expose les notions propres au problème du calcul des plus courts chemins dans un graphe ainsi que de la version dynamique de ce problème, c'est-à-dire le maintien de ces plus courts chemins lorsque le graphe est modifié. Ce chapitre présente également un nouvel algorithme permettant de résoudre ce problème.

Les chapitres 5 et 6 présentent deux types d'algorithmes choisis pour l'optimisation du réseau initial de routes, les algorithmes génétiques et le recuit simulé, ainsi que la manière dont ceux-ci ont été utilisés.

Le chapitre 7 présente les résultats de l'évaluation du concept *Sector-Less* du point de vue de la gestion du trafic aérien ainsi que ceux de l'évaluation du nouvel algorithme de maintien des plus courts chemins. Il montre également comme ce dernier peut être utilisé dans le calcul approché d'un plus court chemin sous contrainte géométrique.

Le travail effectué durant la thèse a donné lieu à trois communications lors de conférences internationales [Rivière 04a, Rivière 04b, Rivière 05b] et de deux communications aux séminaires Eurocontrol [Rivière 03, Rivière 05a].

Première partie

Contexte de l'étude et modélisation

Chapitre 1

Le système de gestion du trafic aérien aujourd'hui

Avant l'année 2002, le nombre de vols contrôlés connaissait une augmentation régulière et ininterrompue. Malgré les nombreuses crises qui ont suivi (attentats du 11 septembre, guerre en Irak, SRAS, etc.) et une baisse conjoncturelle, ce nombre recommence à augmenter pour, après avoir retrouvé le niveau de 2002, le dépasser largement. Les prévisions à long terme [Marsh 05] montrent que cette augmentation devrait probablement continuer[1].

Dans ce contexte, le système de gestion du trafic aérien s'est lentement structuré pour permettre de gérer un nombre d'avions toujours plus grand par l'aménagement de l'espace aérien ainsi que par l'adaptation des méthodes de contrôle.

1.1 Structure de l'espace aérien

L'aménagement de l'espace aérien est réalisé à travers deux grands modules : la construction d'un réseau de routes aériennes adapté ainsi que le découpage de l'espace en secteurs.

1.1.1 Réseau de route

Il existe deux modes de vol différents lorsque l'on souhaite effectuer un trajet en avion :

Vol VFR (pour *Visual Flight Rules*) : Les aéronefs souhaitant évoluer en VFR doivent assurer eux-mêmes leur séparation les uns par rapport aux autres par des moyens visuels. En particulier, les vols VFR ne peuvent pas « percer la couche », c'est-à-dire passer au travers d'une couche de nuages. Ils doivent être équipés d'un moyen de radiocommunication pour entrer dans les zones contrôlées.

Vol IFR (pour *Instruments Flight Rules*) : Les aéronefs souhaitant évoluer en IFR doivent être équipés de matériels de radionavigation et de vol sans visibilité. Ils

[1] Cette thèse à été démarrée bien avant et écrite au début de l'importante augmentation que le prix du pétrole a connu fin 2005. Il n'a donc été tenu compte ni de son impact dans les prévisions présentées ici, ni dans l'établissement du concept ATM que cette thèse étudie.

doivent également déposer un *plan de vol* (nous verrons la signification de ce terme plus loin) et doivent obtenir une autorisation de décollage ou de survol.

Historiquement, un aéronef voulant aller d'un aéroport à un autre suivait des balises de radionavigation existant physiquement au sol. Le pilote se repérait grâce aux moyens de radionavigation embarqués (VOR, TACAN, etc.), chaque balise au sol émettant un signal sur une fréquence qui lui est propre. Cet ensemble de points de report (ou *waypoints*) forme ce qu'on appelle une *route aérienne*. Pour des raisons de visibilité, ces balises étaient souvent placées en fonction du relief ce qui donnait aux routes un aspect parfois tortueux.

L'amélioration des moyens de communication embarqués a permis, au fil du temps, le développement d'un second type de points de report. Ceux-ci sont alors parfaitement fictifs et ne correspondent à aucun équipement physique au sol mais sont parfaitement identifiables par les nouveaux systèmes de gestion de vol de type *Flight Management System* [2] (ou FMS). Ils calculent leur position par triangulation à l'aide des balises au sol mais utilisent également d'autres moyens de positionnement tel que le GPS. Cette possibilité de pouvoir définir des points de report quelconques a permis de rendre les routes aériennes mieux adaptées aux besoins du trafic aérien. Cependant il n'est pas possible de définir une route à travers n'importe quelle zone de l'espace aérien. Il faut respecter certaines contraintes d'environnement (surtout à cause du bruit) mais également éviter certaines zones d'espaces réservées, notamment aux militaires.

Le réseau de routes ainsi défini par les balises au sol et par les points de report fictifs est aujourd'hui relativement rigide, les libertés apportées par les points fictifs étant presque toutes utilisées. On s'aperçoit alors que pour aller d'un point à un autre il n'existe bien souvent qu'une seule route possible. Considérons par exemple le survol de la Turquie. La figure 1.1 représente les routes utilisables pour un aéronef, les points nommés étant des points de report. On constate alors que ce réseau de routes ne laisse pas beaucoup de choix aux pilotes. Les schémas de routage ainsi définis sont employés par les pilotes afin d'établir leur *plan de vol*. Il s'agit d'un document que chaque aéronef volant suivant les règles IFR doit remettre aux autorités chargées de la gestion de l'espace aérien (pour les vols VFR, il est possible mais non obligatoire de déposer un plan de vol, le vol se faisant à vue). Il contient les informations fondamentales suivantes :

- l'heure de départ ;
- le niveau de vol[3] demandé pour la croisière ;
- la route prévue.

[2] Le Flight Management System est le calculateur principal des aéronefs modernes. Il centralise l'ensemble des informations sur l'aéronef, assure la navigation des appareils en fonction des moyens de repérage disponibles, permet d'optimiser vitesses et niveaux de croisière en fonction de paramètres météorologiques et économiques, et peut envoyer au pilote automatique et aux régulations moteurs toutes les informations nécessaires à l'exécution du plan de vol.

[3] Le niveau de vol correspond à l'altitude de l'avion et est exprimé en centaine de pieds. Ainsi, un aéronef volant au niveau 300 est à une altitude de 30000 ft, c'est-à-dire environ 9150 m. Il est possible de demander une succession de niveaux de vol. En effet la consommation de carburant allégeant l'aéronef, il est alors possible de monter plus haut et plus l'aéronef est haut, moins la consommation est importante.

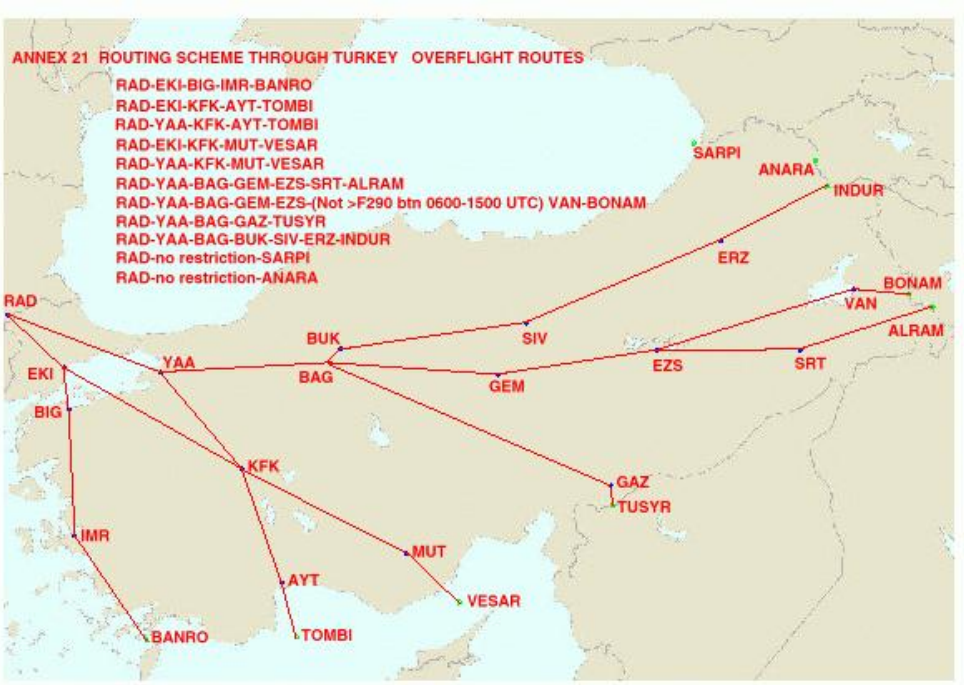

FIG. 1.1 – Réseau de routes aériennes pour le survol de la Turquie

Ces informations sont utilisées par les autorités de régulation aérienne pour assurer une bonne gestion des flux de trafic et informer les contrôleurs aériens en charge des secteurs de contrôle de l'arrivée des aéronefs traversant ces derniers (voir section 1.1.2). Les pilotes (dans le cas de l'aviation civile commerciale la décision vient en général des compagnies) peuvent librement choisir la route qu'ils veulent suivre.

1.1.2 Sectorisation de l'espace

L'espace aérien est divisé en *volumes d'espace* qui ont été classés par l'Organisation de l'Aviation Civile Internationale (OACI) en 7 catégories (nommées de A à G) selon les services rendus (service de contrôle, service d'information de vol et service d'alerte) et le type de vols pouvant traverser ces espaces [SIA 03]. La classe A, par exemple, est réservée uniquement aux aéronefs en régime de vol IFR. Elle est strictement interdite aux avions en VFR. Les services rendus dans cette classe d'espace par l'organisme de contrôle est l'espacement entre aéronefs IFR. Dans un espace de classe E, on peut trouver à la fois des vols IFR contrôlés et des vols VFR non contrôlés (pour lesquels le contact radio avec l'organisme de contrôle n'est pas obligatoire).

L'espace contrôlé, quelle que soit sa classe, est découpé en tranches d'espaces volumiques, appelées *secteurs de contrôle*. La figure 1.2 montre la projection en deux dimensions du découpage de l'espace supérieur aérien français. Chaque secteur est géré par une équipe de contrôleurs (en général deux) chargés d'assurer la sécurité du trafic à l'intérieur

de leur volume d'espace.

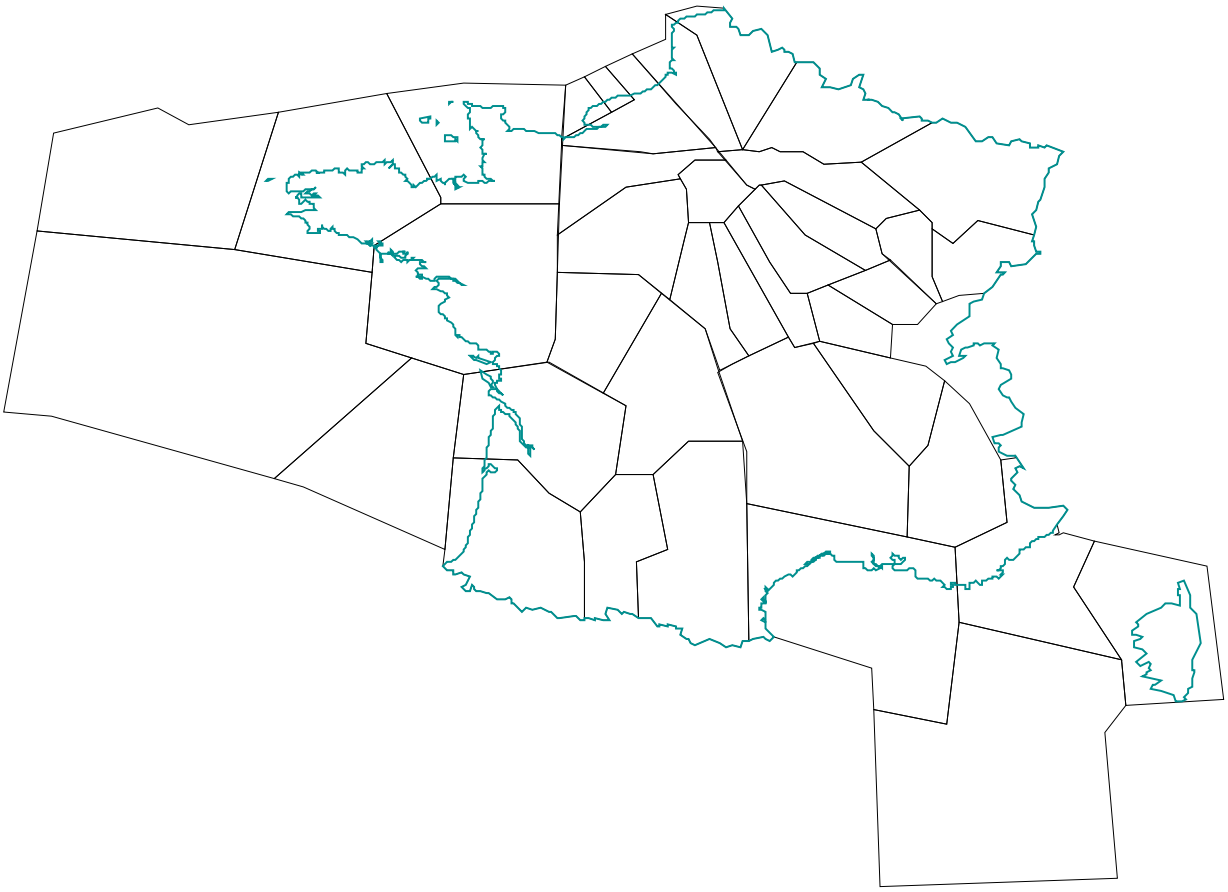

FIG. 1.2 – Projection 2D du découpage de l'espace supérieur aérien français en secteurs de contrôle

1.2 Contrôle aérien et régulation

1.2.1 Rôle des contrôleurs

La définition officielle du contrôle du trafic aérien est la suivante :

> *Le but premier du contrôle du trafic aérien est d'assurer la sécurité du trafic et donc d'éviter les abordages entre aéronefs opérant dans le système, puis d'optimiser les flux de trafic.*

Le contrôle aérien peut être divisé en trois catégories :

Le contrôle au sol : il correspond au contrôle des aéronefs sur les plates-formes aéroportuaires (roulage, etc). Nous n'en parlerons pas ici (pour plus d'informations, le lecteur pourra se référer à [Gotteland 04]).

Le contrôle d'approche : il correspond au contrôle autour des plates-formes aéroportuaires. Le contrôle d'approche fait appel à des techniques spécifiques et se décompose en deux type de problèmes : assurer la sûreté des aéronefs en évitant les collisions et insérer un aéronef dans une file pour permettre le séquencement de la file en vue de leurs atterrissages. En raison de la vitesse plus réduite des aéronefs, les contrôleurs d'approche peuvent employer des techniques de contrôle plus souples. Ils peuvent aussi mettre les aéronefs en attente dans des hippodromes. Un hippodrome est une zone de l'espace aérien où l'on « empile » les aéronefs en les faisant voler à un niveau de vol constant sur une trajectoire fermée ayant la forme d'un hippodrome. L'aéronef en bas de l'hippodrome est dirigé vers l'aéroport lorsque la piste est prête à le recevoir, libérant ainsi un niveau pour l'aéronef immédiatement au dessus, etc. Les règles de contrôle sont spécifiques et les trajectoires d'approche fortement liées à leur environnement (par exemple dans le cas de mise en place d'un plan de réduction des nuisances sonores).

Le contrôle en route : il correspond au contrôle hors zones d'approche. C'est celui qui nous intéressera directement et dont nous allons détailler maintenant le mode opératoire.

Le contrôle en route est effectué par des contrôleurs regroupés en centres de contrôle. On distingue 5 centres de contrôle en route en France (Aix-en-Provence, Bordeaux, Brest, Paris et Reims), 36 en Europe, chaque centre pouvant gérer une ou plusieurs *zones de qualifications* [4] (la France en compte 7). La localisation géographique de ces centres n'a pas de signification particulière par rapport au zones qu'ils contrôlent (même si en France ces derniers sont plus ou moins situés au centre de leur zone d'action) mais correspond plutôt aux différentes politiques d'aménagement du territoire. Il existe encore peu de centres communs à plusieurs pays. Le centre suisse contrôle deux secteurs frontaliers avec la France, ceci grâce à un accord entre les deux pays. Il existe également deux centres « européens », c'est-à-dire gérés non plus par les pays dont ils contrôlent l'espace mais directement par l'agence Eurocontrol, agence regroupant divers pays européens dont le but est de développer et coordonner les différentes politiques de gestion de trafic aérien.

Les contrôleurs travaillent généralement par paire : un contrôleur *organique* et un contrôleur *tactique*. Chaque paire travaille devant une station de contrôle. Un exemple représentant une position occupée (ou *armée*) par deux contrôleurs est présenté figure 1.3. La personne au premier plan étant le contrôleur organique, la seconde, à l'arrière, le contrôleur tactique.

Le contrôleur organique est en charge de la gestion du trafic aérien à moyen terme. Pour cela il dispose de *bandes de progression* ou, plus couramment, *strips* (originellement des bandelettes de papiers, parfois remplacées par un affichage électronique).

Les strips contiennent toutes les informations fondamentales concernant chaque vol passant dans le secteur dont il a la charge (un strip par vol). Par exemple, le strip présenté

[4] Un contrôleur qualifié sur une zone géographique donnée ne peut pas contrôler sur une autre zone. Ainsi au sein d'un même centre de contrôle, un contrôleur ne peut pas travailler sur n'importe quelle position.

FIG. 1.3 – Une position de contrôle

figure 1.4 renseigne le contrôleur sur :

- L'identifiant du vol (AFR1715) et le type d'aéronef (A321),
- le niveau de vol d'entrée (310),
- le niveau de vol demandé (350),
- les aéroports de départ (LIMC, Milan) et d'arrivée (LFPG, Roissy-Charles de Gaulle),
- la fréquence radio de contact (132.1),
- et enfin la route prévue dans le plan de vol et l'heure de passage prévue sur chaque point de report (par exemple DIJ à 16h11, TRO à 16h23).

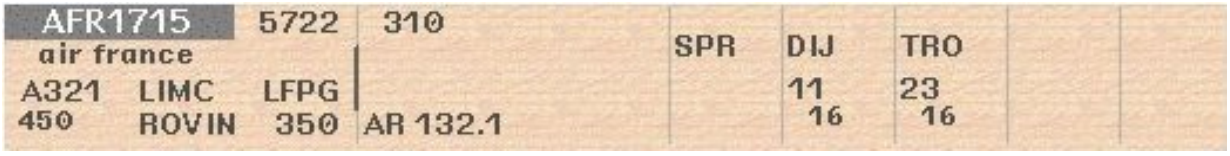

FIG. 1.4 – Un strip électronique

Les informations contenues dans les strips sont transmises au contrôleur tactique une dizaine de minutes avant l'entrée de l'aéronef dans le secteur dont il a la charge. Avant cela le contrôleur organique est chargé de détecter tout conflit qui pourrait être induit par la future entrée d'un aéronef dans le secteur qu'il contrôle. Le contrôleur organique doit également assurer la liaison avec le contrôleur du secteur précédent et celui du secteur suivant afin d'assurer une bonne transmission des aéronefs. Cette procédure, appelée *co-ordination*, est une des tâches majeures du contrôle. Il assure aussi la gestion globale du

flux dans le secteur et doit s'assurer que la tâche que devra effectuer le second contrôleur ne sera pas trop lourde.

Le contrôleur tactique a, quant à lui, la responsabilité à court terme du trafic et notamment la séparation des aéronefs entre eux. Pour cela, il doit maintenir en permanence une distance minimale entre les aéronefs. Cette séparation est d'environ 5 Nm (1 Nm = 1819 m) dans le plan horizontal ou 1000 ft (1 ft = 30 cm) dans le plan vertical. Lorsque ces deux normes sont simultanément violées, on dit qu'il y a perte de séparation ou **conflit**. Pour faciliter le travail du contrôleur, les aéronefs volent, comme nous l'avons vu, sur des routes aériennes et lorsqu'ils sont stables, c'est-à-dire lorsque les aéronefs ne sont ni en montée, ni en descente, ils se fixent sur un niveau de vol « entier ». Ainsi un aéronef volera par exemple à 31000 ft (niveau de vol FL310) mais pas à 31700 ft. Les niveaux de vol standards sont ainsi séparés de 1000 ft (FL290, FL300, FL310). Pour maintenir ces différentes séparations, le contrôleur tactique dispose d'une image radar lui montrant la direction de déplacement des aéronefs, leur vitesse respective, leur niveau de vol, etc. Un exemple est présenté figure 1.5. Celle-ci ne montre qu'une partie de l'image complète que peut avoir un contrôleur. La partie gris clair correspond au secteur dont les contrôleurs ont la charge, la partie gris foncé permettant de voir les aéronefs entrer et sortir du secteur dont ils ont la charge. Il existe encore très peu d'aide à la détection et à la résolution de conflit. Le seul système automatique actif de détection aujourd'hui, du point de vue du contrôleur aérien, est le filet de sauvegarde, un dispositif qui se déclenche lorsque le risque de collision est imminent.

Pour assurer la séparation, le contrôleur tactique dispose de deux catégories de manœuvres : les manœuvres en niveau et les manœuvres en cap. Les manœuvres en niveau se divisent elles-mêmes en trois catégories. Le contrôleur peut demander au pilote de :

- stabiliser son aéronef en montée ou en descente avant de l'autoriser à poursuivre vers son niveau initialement requis ;
- anticiper la descente lorsqu'il est proche de sa destination ;
- descendre [5] d'un niveau de vol lorsqu'il est stable en croisière (cette dernière manœuvre n'étant pas du tout populaire auprès des pilotes).

Les manœuvres en cap consistent simplement à modifier le cap d'un aéronef à gauche ou à droite, puis à lui faire reprendre sa trajectoire en lui communiquant un nouveau cap vers une balise de son plan de vol.

1.2.2 Régulation du trafic aérien

Nous avons vu précédemment que ce sont les pilotes qui choisissent leur plan de vol. Que se passe-t-il alors si tous les aéronefs veulent passer au même endroit au même moment ? Géré par des opérateurs humains, un secteur a une capacité limitée par la quantité d'aéronefs que peut gérer un couple de contrôleurs, capacité très inférieure à la capacité intrinsèque du secteur offert par son volume. Chaque secteur possède ainsi une capacité ho-

[5] Il est également possible de demander à un aéronef de monter d'un niveau mais ceci dépend des performances avions et n'est pas toujours possible.

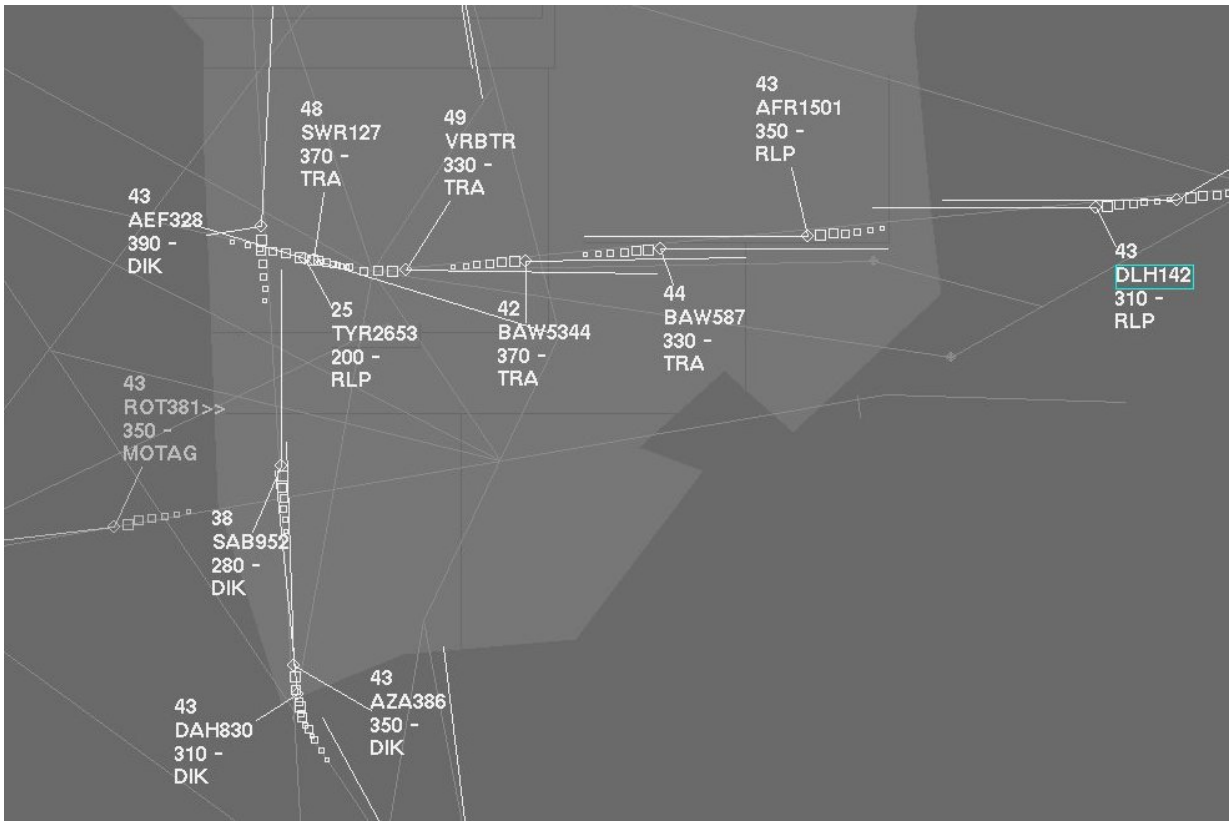

FIG. 1.5 – Exemple d'image radar

raire dépendant du type de trafic (en route, en approche), de l'emplacement géographique du secteur, de sa taille, etc. Si cette capacité est dépassée, le contrôleur n'est plus à même de gérer tous les aéronefs et donc de garantir la sécurité des vols. Des méthodes ont ainsi été mises en place pour éviter de telles situations. C'est la gestion des flux de trafic (Air Traffic Flow Management ou ATFM).

La majorité des vols dépassant le cadre purement national (environ 15000 vols par jours sur les 25000 circulant en Europe lors d'une journée moyenne en 2005), la régulation du trafic doit se faire au niveau européen plutôt que national. En effet pour un vol venant du nord et atterrissant à Roissy-Charles de Gaulle, il est parfois nécessaire d'appliquer des mesures de régulation alors que l'aéronef est toujours pris en charge pas le contrôle belge ou britannique. En Europe c'est l'agence Eurocontrol qui est en charge de fournir les services ATFM à travers la CFMU (Central Flow Management Unit). La figure 1.6 présente (en gris foncé) les pays européens couverts par la CFMU.

FIG. 1.6 – Pays collaborant au travers de la CFMU

Pour décoller d'un aéroport, la CFMU attribue aux aéronefs un créneau de décollage, c'est-à-dire que ceux-ci doivent impérativement partir dans un laps de temps compris entre 5 minutes avant et jusqu'à 10 minutes après l'heure officielle de décollage. Si un aéronef manque son créneau, il ne peut, en règle générale, décoller et doit théoriquement en redemander un autre. En fonction des demandes de régulation, la CFMU choisit de modifier l'heure de décollage de certains vols. Elle impose donc une heure de départ différente de celle prévue à l'origine (Si un aéronef doit être décalé, le délai appliqué est donné à la compagnie aérienne 24 heures à l'avance).

L'algorithme, nommé CASA (pour Computer Assisted Slot Allocation [CFMU 00]), utilisé par la CFMU est très simple. Pour chaque zone régulée, il génère une liste de créneaux de passage. Ainsi, si un secteur d'une capacité horaire de 30 aéronefs par heure se trouve régulé, CASA va générer une liste de créneaux de passage espacés de deux minutes

chacun (60 minutes divisées par la capacité) et cela pour toute la durée de la régulation.

La liste ainsi générée va se remplir au fur et à mesure que les compagnies aériennes soumettent leurs plans de vol à la CFMU. Si un aéronef « veut » passer dans un créneau déjà alloué à un autre vol c'est l'aéronef qui aurait dû passer le premier en l'absence de régulation qui obtient le créneau. L'autre aéronef est reclassé dans le créneau suivant. Si ce créneau est occupé alors le processus se répète. Il peut donc se produire une réaction en chaîne. Un nouveau vol peut décaler de proche en proche de nombreux autres.

Si un vol traverse plusieurs zones régulées il se voit imposer le retard le plus important. Il sera donc accepté avec ce retard dans toutes les zones. Le mécanisme employé ici par la CFMU n'est pas très clair. Illustrons ce qui se passe sur un exemple simple. Soit un aéronef qui traverse deux zones régulées. La première lui donne un retard de 10 minutes et la seconde un retard de 25 minutes. L'aéronef partira avec 25 minutes de retard. Mais le premier secteur peut-il prendre l'aéronef sans problème avec 25 minutes de retard au lieu de 10 ? Il n'existe pas de réponse officielle à ce problème.

Face aux problèmes d'incertitude sur les heures de passage et sur la capacité réelle des secteurs, la CFMU apporte une réponse pragmatique et statistique. L'ensemble des acteurs du transport aérien respecte les décisions prises et dans les faits les résultats sont acceptables. Toutefois, les fondements théoriques pour justifier son fonctionnement sont quasiment inexistants.

1.3 Saturation de l'espace aérien

Tous les principaux acteurs s'accordent aujourd'hui pour dire que le système actuel se rapproche de ses limites, du point de vue capacitif.

Jusqu'à présent, lorsque la régulation peinait à résoudre les problèmes de capacité ou lorsque ceux-ci étaient trop fréquents dans un secteur donné, la solution consistait à diviser le secteur en deux secteurs contrôlés chacun par une équipe de deux contrôleurs. Or le système de division a désormais atteint ses limites. En effet, comme on l'a vu dans la section 1.2.1, le contrôleur organique est responsable de la coordination avec les contrôleurs des secteurs adjacents, la communication s'effectuant le plus souvent par téléphone. Lorsque les secteurs deviennent trop petits, le contrôleur organique passe la majeure partie de son temps à se coordonner avec ses homologues et ne peux donc plus assurer pleinement son activité de détection de conflits potentiels. De plus un contrôleur a besoin d'une certaine quantité d'espace afin de pouvoir changer le cap de certains aéronefs et de résoudre les différents conflits. Un secteur trop petit peut donc engendrer des problèmes de résolution de conflits. Il existe donc une taille critique pour un secteur. En Europe, on considère que la quasi totalité des secteurs ne peuvent plus être subdivisés.

Un second facteur qui influe sur la capacité est l'obligation faite aux aéronefs de suivre le réseau de routes. En effet, s'il facilite le contrôle, le regroupement des aéronefs sur les mêmes routes renforce le phénomène de congestion, notamment aux points de croisement entre différentes routes. Une seconde solution pour absorber toujours plus de trafic a été la mise en place des RVSM (pour Reduced Vertical Separation Minima qui a consisté à réduire

les norme de séparation verticale passant de 2000 à 1000 ft) et également la réduction, dans certaines zones, de la distance de sécurité horizontale entre deux aéronefs, ceci ayant pu être fait grâce à la précision toujours croissante des appareils à bord des aéronefs mais également des radars au sol. Mais encore une fois, on peut penser que l'on se rapproche de la limite de réduction possible.

Or, d'après de récentes études menées par l'agence Eurocontrol [Marsh 05][6], le trafic aérien va encore augmenter dans les prochaines années. Malgré les épiphénomènes tels que la guerre en Irak ou le SRAS, le trafic aérien continue de croître et la demande de capacité va donc aller en augmentant puisqu'il est prévu pour les six prochaines années une croissance annuelle moyenne de 3,7% (voir figure 1.7).

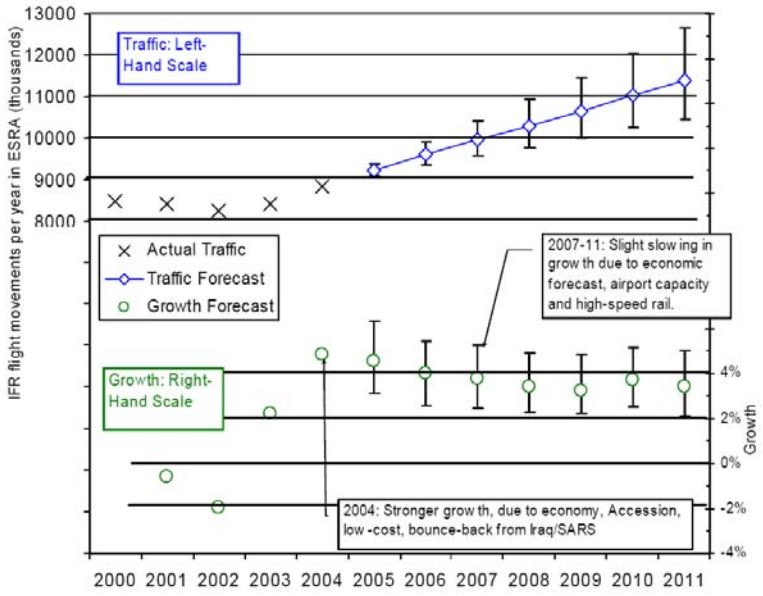

FIG. 1.7 – Prévision de croissance du trafic aérien entre 2005 et 2011 [Marsh 05]

1.4 Conclusion sur le système actuel

Le coût des retards aériens est évalué entre 7 et 11 milliards d'euros par an par l'institut du transport aérien dont 60% est imputable au contrôle aérien selon [ITA 00]. Il parait donc essentiel de les résorber.

Comme on l'a vu, le système de gestion du trafic aérien met en jeu de nombreux processus et de nombreux acteurs qui n'ont pas tous les mêmes intérêts : les pilotes veulent utiliser le chemin le plus court possible, les contrôleurs doivent assurer la sécurité sans être maîtres du plan de vol initial, etc. Les investissements, pour son amélioration, sont

[6]Cette étude ne tient pas compte de l'augmentation du prix du pétrole qui commençait au moment où celle-ci a été réalisée.

considérables et toute nouvelle fonctionnalité se chiffre en centaines de millions d'euros. De plus, il faut noter que par le passé, le système a toujours été capable de s'adapter pour faire face à l'augmentation du trafic aérien sans bouleversement majeur.

Mais aujourd'hui une autre raison pousse le système vers le changement et elle est politique. La commission européenne souhaite ardemment une unification du système afin de faire disparaître les monopoles nationaux. Ceci a provoqué l'émergence de nouveaux concepts ATM, de la simple aide au contrôle jusqu'à l'automatisation de ce dernier, chacun apportant une nouvelle vision de ce que devrait être le contrôle aérien du futur. Il faut alors pouvoir évaluer ces nouveaux concepts et c'est dans ce cadre que cette thèse s'inscrit.

Chapitre 2

Différents concepts de gestion du trafic aérien

La nécessité d'augmenter la capacité du système ATM a ainsi ouvert la voie à une multitude de nouveaux concepts. Ceux-ci peuvent être regroupés en deux grands courants de pensée : ceux qui visent à améliorer le système actuel tout en conservant les opérateurs humains dans leur rôle et ceux qui visent une réforme complète du système, soit en supprimant, à terme, l'opérateur humain, soit en lui affectant une tâche radicalement différente de celle qu'il pratique aujourd'hui. Après avoir détaillé quelques exemples de systèmes novateurs section 2.1 et 2.2, nous présenterons section 2.3 le concept *Sector-Less* qui est le cadre de travail de cette thèse.

2.1 Améliorer le système en place

Un des principaux projets, en France, d'amélioration de la gestion du trafic aérien est dénommé Erato [era 05]. Il regroupe un ensemble d'outils d'aide au contrôle en route qui offrent deux fonctions informatiques principales, le *filtrage* et l'*agenda*.

Le filtrage permet, à la demande du contrôleur, de marquer visuellement les vols étant potentiellement en conflit. À cette fonction est associé un processus de surveillance qui vérifie en permanence que le comportement des aéronefs est conforme aux prévisions faites par le système.

L'agenda propose une organisation du trafic sous forme de problèmes identifiés à partir des routes définies dans le plan de vol. Reprenant des fonctions remplies par les *strips*, il permet la mise en évidence par le contrôleur des conflits à venir, la planification et la gestion des tâches. Il offre un support visuel à la mémorisation du trafic et favorise la coopération entre le contrôleur organique et le contrôleur tactique. La présentation temporelle des tâches permet aux deux contrôleurs de planifier leur charge de travail en fonction de leurs priorités.

Si Erato est un projet bien avancé, puisqu'à l'heure actuelle il est en phase d'expérimentation en centre de contrôle en route, d'autres améliorations sont pour l'instant en phase

d'études. Une des dernières idées en date pour améliorer le système actuel a été présentée par Jacques Villiers. D'après [Villiers 04], le contrôleur travaille dans un zone d'incertitude, c'est-à-dire que la détection de conflits repose sur une « impression » plus que sur un calcul précis. Ainsi une très faible variation de vitesse des aéronefs, non perceptible par le contrôleur, peut suffire à supprimer un conflit latent. Nommé « contrôle subliminal », [Villiers 04] suggère de résoudre certains conflits (par exemple ceux entre deux aéronefs uniquement) de manière transparente pour le contrôleur, c'est-à-dire sans lui « dire ». Ainsi celui-ci aurait plus de temps pour gérer les « vrais » problèmes. Cette idée qui n'en est qu'à ses débuts est en court d'études pour évaluer sa faisabilité au travers d'un projet dénommé Erasmus.

Si les différentes solutions d'amélioration d'un système supposé avoir atteint ses limites peuvent paraître une bonne idée sur le cours et moyen terme, il n'en reste pas moins nécessaire d'imaginer le système du futur, soit par la suppression de l'opérateur humain, soit par la création d'une autre manière de concevoir le contrôle aérien.

2.2 Automatiser totalement le système

À la vue des améliorations, sûrement performantes mais forcément limitées que l'on peut amener au système, il faut bien alors se poser la question de la suppression du contrôle au sol [Durand 99] et de la possibilité de laisser aux aéronefs le soin de s'éviter mutuellement. Différents concepts, regroupés sous l'appellation relativement floue de *Free Flight*, ont été proposés.

Les premiers à s'intéresser à ce concept sont les américains. Celui-ci, baptisé *Free route* [MIT 95, RTCA 95] permettait aux aéronefs de suivre des routes choisies, donc non imposées par un quelconque réseau de routes, dans des espaces de faible densité. La version européenne du *Free Flight* a été proposée par l'agence Eurocontrol à travers le projet FREER [1] [Duong 96, Duong 97a, Duong 97b, Duong 98]

FREER se divisait en deux parties :

- FREER-1 devait permettre aux aéronefs d'être totalement autonomes dans des espaces de faible densité (au dessus des océans par exemple). Il n'y avait plus alors d'infrastructure au sol pour gérer le trafic. Les concepteurs ont complété les règles de l'air de façon à pouvoir, d'une part, prendre en compte toutes les configurations de conflit à deux aéronefs et, d'autre part, définir un ordre total sur l'ensemble des aéronefs dès lors qu'on s'intéresse à trois aéronefs ou plus. Cependant les règles de résolution des conflits impliquant plus de quatre aéronefs n'ont jamais été précisément décrit.
- FREER-2 devait ainsi assurer l'absence de conflit à plus de trois aéronefs en agissant tel un filtre en amont et depuis le sol pour veiller à ce que l'espace Free Flight ne soit pas saturé.

Afin d'améliorer ce système et notamment de permettre, par les aéronefs eux-mêmes, la

[1] Free-Route Experimental Encounter Resolution

résolution des conflits impliquant plus de trois appareils, différentes études ont été menées et notamment au LOG. Granger [Granger 01a, Granger 01b, Granger 02] montre qu'il est possible de traiter la totalité des conflits sur une journée de trafic européen, que les aéronefs volent en route directe ou en suivant le réseau de routes existant, certains conflits mettant alors en cause jusqu'à une vingtaine d'aéronefs.

Des concepts équivalents au Free Flight ont été proposés [Alliot 03] mais tous posent un même problème qui est la question fondamentale de l'automatisation : l'absence de pilote à bord. Le train à grande vitesse sans conducteur serait parfaitement possible sans la résistance psychologique des passagers face au concept de « machine sans pilote à son bord ». Notons d'ailleurs que, dans le cas du TGV, le rôle du conducteur consiste à mettre ses mains sur un volant qui n'en est pas un et à surveiller que tout se passe bien. Il est désormais possible d'avoir des métros sans pilote ce qui tend à montrer que le passager est prêt à accepter un certain degré d'automatisation. Néanmoins, l'aéronef sans pilote est-il envisageable ?

Un autre problème induit par ces nouveaux concepts est la transition entre le système actuel et un nouveau système. En effet il parait difficile de changer tout un système, qui de plus transcende les frontières, du jour au lendemain. Ainsi, le problème de mise en application de tous ces nouveaux concepts vient également de leur absence de processus de transition.

2.3 Le concept *Sector-Less*

Alors que les concepts d'automatisation tels que FREER supposent un transfert des techniques d'évitement du sol vers les airs, le concept *Sector-Less* propose la refonte du système tout en conservant la répartition des rôles actuels, c'est-à-dire en laissant la responsabilité de la séparation des aéronefs aux contrôleurs au sol. Ce concept a été présenté pour la première fois lors du $4^{ème}$ séminaire américano-européen de recherche et développement en ATM en 2001 [Duong 01]. L'essence même du concept *Sector-Less* est le contrôle des aéronefs non plus dans des volumes d'espaces prédéterminés mais de bout en bout par le ou les mêmes contrôleurs.

2.3.1 Rôle des contrôleurs

Dans le cas d'un contrôle *Sector-Less*, un contrôleur est responsable de la sécurité d'un nombre limité d'aéronefs, depuis leur aéroport de départ jusqu'à leur aéroport d'arrivée. En fait, le contrôle dans les zones terminales ou TMA étant bien spécifique, on considérera qu'un contrôleur est en charge d'un ou plusieurs aéronefs depuis leur sortie de TMA jusqu'à leur entrée dans la TMA d'arrivée. Ceci n'étant pas sans rappeler le contrôle aérien militaire [Rivière 02]. Pour gérer « ses » aéronefs, un contrôleur dispose alors des mêmes outils qu'à l'heure actuelle pour résoudre un conflit, c'est-à-dire qu'il peut leur donner des manœuvres en cap ou en niveau.

Pour faciliter leur travail, les contrôleurs se voient attribuer des vols effectuant le même

trajet (ou des trajets similaires) et/ou volant dans la même zone géographique. Même en réalisant une répartition « intelligente » des vols entre contrôleur, il est impossible d'éviter que deux aéronefs contrôlés par deux personnes différentes ne se croisent. Si aucune procédure, autre que la négociation verbale entre les deux personnes concernées, n'a été mise en place par [Duong 01] lors de la définition du concept *Sector-Less*, un aménagement du réseau de routes et de nouvelles règles adaptées à ce concept doit permettre d'éviter la trop grande fréquence de ce genre de problèmes.

2.3.2 Aménagement de l'espace aérien

Afin de simplifier la création d'un réseau de route, [Duong 01] divise le problème en deux sous problèmes :

- un réseau principal ou *Trunk Route Network* (TRN) couvrant la totalité de l'Europe ;
- un réseau secondaire ou *Secondary Route Network* (SRN) permettant de relier le réseau principal aux différents aéroports.

Les deux grandes règles d'aménagement de l'espace aérien associées au réseau principal sont celles gérant la séparation des flux allant dans des directions opposées et celles gérant les croisements entre flux orthogonaux. Comme présenté figure 2.2, le système de croisement fonctionne plus ou moins comme un rond-point, tout en utilisant la troisième dimension, l'altitude.

Séparation des flux parallèles

La figure 2.1 présente un croisement de routes orthogonales. Chaque route est doublée afin d'éviter que les aéronefs allant en sens opposés ne se retrouvent face à face. À l'heure actuelle, les aéronefs dans ce cas sont séparés en niveau. Dans le cadre d'un contrôle *Sector-Less*, ils peuvent alors utiliser les mêmes niveaux. En plus de la séparation des flux en opposition, chaque direction est constituée de trois voies : une voie principale et deux secondaires (ou *offsets*), respectivement de chaque coté de la première et pouvant être utilisées pour des dépassements.

Si les aéronefs volant dans la même direction ou dans des directions opposées peuvent voler au même niveau car ils sont séparés horizontalement, il faut également séparer les aéronefs volant dans des directions orthogonales. La solution retenue par [Duong 01] est alors de séparer ces aéronefs en niveau. Deux routes se croisant doivent donc occuper des niveaux de vol différents. On peut imaginer par exemple que les flux est-ouest occupent les niveaux de vol pairs, les flux nord-sud le niveaux de vol impairs. Ainsi le nombre de croisements entre aéronefs allant dans des directions différentes doit être fortement réduit. Il reste donc à gérer les changements de directions pour un aéronef, c'est-à-dire, le passage d'une route à l'autre.

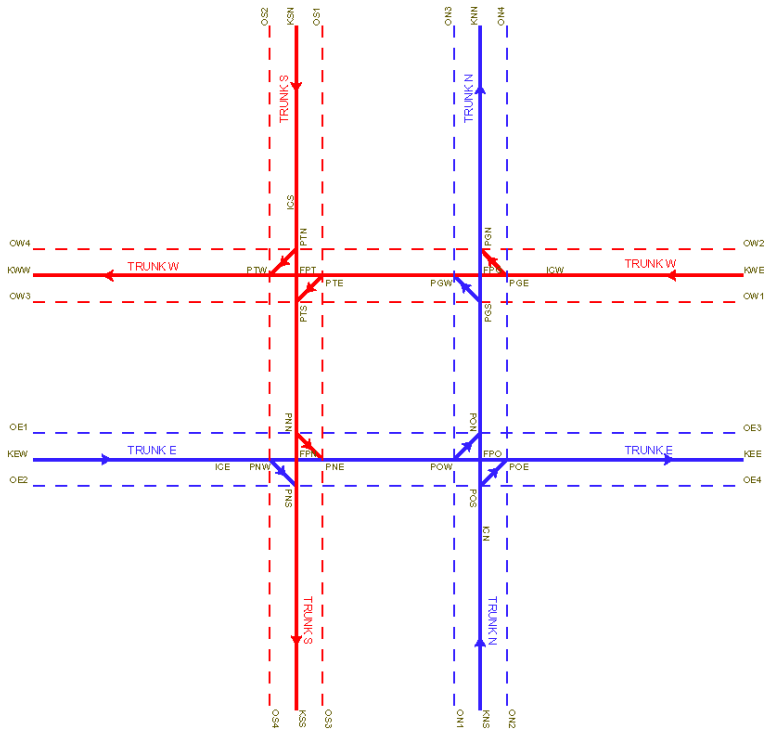

FIG. 2.1 – Séparation de flux dans Sector-Less : un croisement de 2 flux orthogonaux, 2 routes parallèles chacune ayant 1 voie principale (trait plein) et 2 voies secondaires (trait pointillé) utilisées pour les dépassements

Croisements de routes

Afin de rendre les croisements de routes compatibles avec les performances des aéronefs, la seconde règle du concept *Sector-Less* définit la manière dont les aéronefs peuvent changer de direction, ces changements s'effectuant à des endroits prédéfinis. La figure 2.2 schématise un de ces points de croisement prédéfinis et ne représente que deux niveaux de vol. Si les routes parallèles sont représentées, les *offsets* ne le sont pas mais leur intégration est relativement simple. En fait, les points de croisement fonctionnent comme un rond-point mais en trois dimensions.

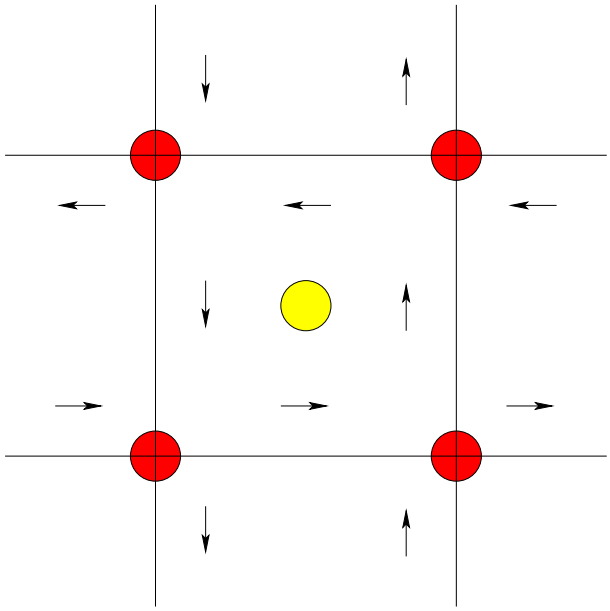

FIG. 2.2 – Schématisation d'un point de croisement dans le cas d'un contrôle *Sector-Less*

Comme précisé dans la section précédente, deux routes se croisant doivent être à des niveaux de vol différents. Ainsi un aéronef souhaitant changer de direction doit obligatoirement, dans le concept *Sector-Less*, monter ou descendre d'un niveau. Les règles de croisement sont les suivantes (voir figure 2.3) :

- Tout aéronef passant à travers un point de croisement sans changer de direction doit rester stable en niveau de vol.
- Tout aéronef tournant à droite ne doit pas traverser le « carrefour » mais change de niveau (que ce soit en montant ou descendant) pour rejoindre le niveau de vol de sa nouvelle voie.
- Tout aéronef tournant à gauche doit d'abord franchir le point de croisement pour ensuite changer de niveau et rejoindre sa nouvelle voie.

Ces deux règles sont censées faciliter le travail des contrôleurs et permettre une réduction importante du nombre de conflits à traiter. Dans ce concept, c'est au contrôleur d'assurer la bonne insertion des aéronefs dans leur nouvelle voie.

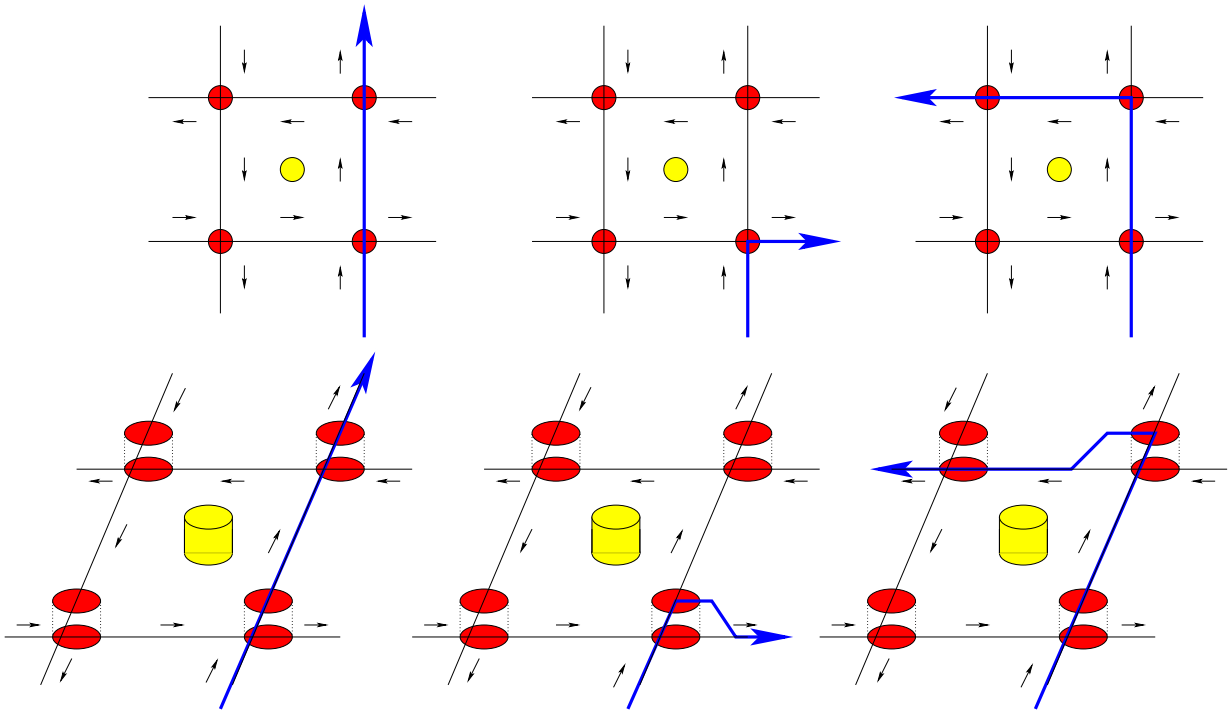

FIG. 2.3 – Les trois règles permettant d'utiliser un point de croisement dans le cas d'un contrôle *Sector-Less*

SuperSecteur : la transition

Un changement aussi radical dans la façon de concevoir la gestion du trafic aérien ne pourra, bien entendu, devenir réalité du jour au lendemain. Les auteurs du concept *Sector-Less* ont également pensé à la transition. C'est le projet SuperSecteur [Gawinowski 03a, Gawinowski 03b].

SuperSecteur est un concept à mi-chemin entre le système actuel, basé sur la sectorisation de l'espace et le concept *Sector-Less*. Il consiste à diviser l'espace aérien en « grands » secteurs pouvant faire la taille d'un pays. Dans chacun de ces secteurs, un contrôle de type *Sector-Less* est mis en place, chaque secteur contenant son réseau de routes principales et son réseau secondaire. La création du TRN devant se faire de façon cohérente pour pouvoir relier les différents secteurs entre eux. La viabilité de ce concept est en cours de test au centre expérimental d'Eurocontrol.

2.4 Conclusion

Il existe donc plusieurs visions, à plus ou moins long terme, du futur de la gestion du trafic aérien. Parmi ces visions, le concept *Sector-Less* fait partie des plus innovantes par sa volonté de garder l'opérateur humain dans la boucle de décision tout en remodelant totalement sa manière de travailler.

Les bases du concept ayant été posées, il faut désormais s'attacher à la génération d'un espace aérien adapté et donc, d'un réseau de routes permettant la meilleure utilisation possible de celui-ci. Il faudra également déterminer si ce concept est viable.

Chapitre 3

Modélisation du problème

Tel qu'il a été spécifié précédemment, le réseau de routes aériennes se divise en deux sous-réseaux : un réseau principal et un réseau secondaire. Le réseau secondaire est celui qui doit permettre de relier les aéroports au réseau principal. Il n'y a donc aucune raison pour que les différentes routes de celui-ci soient toutes reliées entre elles. Le réseau secondaire est ainsi très fortement corrélé au réseau principal. De plus chaque aéroport possède ses propres règles d'approche. Il est donc difficile d'automatiser la création de celui-ci. Ainsi dans la suite, nous ne nous intéressons qu'au réseau principal de routes.

Ce chapitre s'intéresse à la modélisation du problème, c'est-à-dire présente les choix effectués lors de la création du réseau initial (section 3.2), détaille section 3.3 le critère d'optimisation retenu afin d'obtenir le meilleur réseau de routes possible et enfin présente les choix d'algorithmes effectués pour l'optimisation de celui-ci.

3.1 Création d'un réseau de routes

Pour des raisons de faisabilité, la création d'un réseau de route a été, dans cette thèse, séparée en deux sous-problèmes. Le premier est la création effective du réseau de routes ainsi que son optimisation pour satisfaire certains critères tout en respectant les règles d'aménagement de l'espace aérien promues par le concept *Sector-Less*. Le second problème est l'utilisation effective de ce réseau par les aéronefs et la manière dont les contrôleurs aériens vont gérer le trafic sur celui-ci.

Même si l'étape de création du réseau de routes doit tenir compte de la manière dont ce dernier va être utilisé, c'est-à-dire que le résultat de l'optimisation doit être « utilisable en réalité », il parait difficile de se servir de la « facilité d'utilisation » comme critère d'optimisation. Le processus d'optimisation requiert un critère calculable. Or la gestion d'un réseau de routes fait appel à des notions en partie subjectives et difficilement transposables sous forme d'équations telles que la dureté de résolution d'un conflit qui varie d'un contrôleur à un autre ou encore le nombre de conflits pouvant être résolus par un contrôleur en un temps donné.

Le processus de création et d'optimisation (chapitres 4, 5 et 6) va donc optimiser un

réseau initial de routes selon un critère mesurable (voir section 3.3) puis l'adéquation entre le concept et le réseau ainsi généré est ensuite évaluée (chapitre 7)

3.2 Choix d'un réseau initial

Les seules spécifications données par [Duong 01] concernant le réseau de routes sont celles présentées dans le chapitre précédent. C'est une contrainte forte qui ne doit pas être modifiée. Ceci permet néanmoins une grande liberté de choix, la seule contrainte à respecter utile dans notre cas étant la topologie des points de croisements, c'est-à-dire qu'un point de croisement est la jonction de quatre routes, chacune étant la composition d'un ensemble d'*offsets* allant dans des directions opposées.

L'idée a donc été de partir d'une feuille blanche et non du réseau de routes déjà existant. La raison principale de ce choix est que le réseau actuel s'est développé au fil des ans de manière à améliorer la sécurité des vols tout en permettant un contrôle plus aisé et plus efficace en terme de capacité. Ainsi la conception du réseau existant est fortement orientée par le type de contrôle qui y est effectué à l'heure actuelle ainsi que par son évolution et celle des moyens techniques au fil des ans. De plus il ne respecte absolument pas la topologie des points de croisement. D'autres études pour d'autres concepts ont déjà testé l'adaptabilité du réseau de routes existant, notamment [Gianazza 04] au sein du LOG.

Ainsi, les choix de départ effectués pour le réseau de routes principales ont été totalement arbitraires. Puisqu'une optimisation allait être faite par la suite, il n'a pas semblé utile de créer un « bon » réseau initial mais plutôt un réseau simple et peu dense, caractéristiques souhaitées par les concepteurs de *Sector-Less*. Celui-ci a donc deux caractéristiques principales :

- une taille fixe de 4000 kilomètres de coté, afin de recouvrir la quasi-totalité de l'Europe ;
- un espacement régulier entre deux points de croisement de 240 kilomètres, ce qui fait un total de 256 points de croisement.

Le réseau présenté figure 3.1 ne montre pas les *offsets* propres à chaque route. Chaque axe nord-sud (respectivement est-ouest) vertical (respectivement horizontal) étant composé d'une route montante (respectivement allant de droite à gauche) et d'une route descendante (respectivement allant de gauche à droite), chacune étant composée de plusieurs voies. Les points de croisement formant le cadre extérieur du réseau sont, de même que ceux internes au réseau, la résultante du croisement de quatre routes, les routes « extérieures » n'étant pas représentées sur cette figure.

Étant donné que le réseau secondaire n'est pas défini, nous considérons dans la suite de cette thèse que les aéronefs volent en utilisant la route directe entre leur aéroport de départ (respectivement d'arrivée) et un premier (respectivement dernier) point de croisement, point d'entrée (respectivement de sortie) du réseau principal. Il en est de même pour les aéronefs en provenance ou à destination d'une zone extra-européenne.

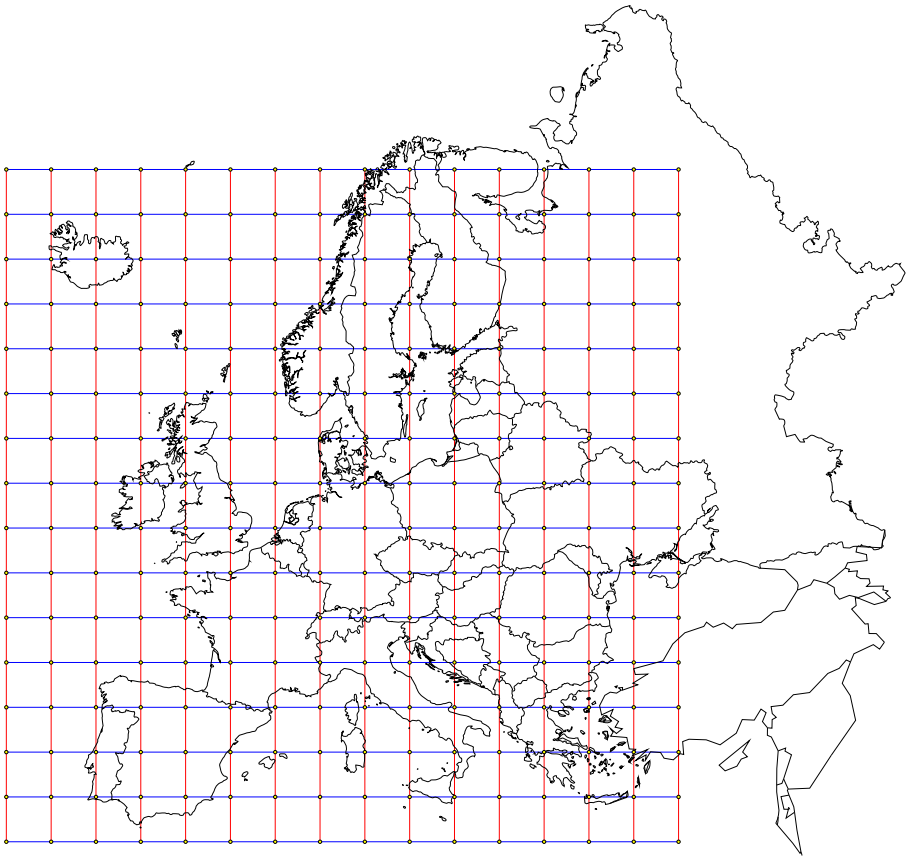

FIG. 3.1 – Réseau initial de routes principales choisi arbitrairement

3.3 Critère d'optimisation

Le but du concept défini par [Duong 01] est de proposer un aménagement simplifié de l'espace aérien tout en conservant un réseau de routes, ce qui peut impliquer une distance à parcourir (donc une durée de vol) plus importante qu'à l'heure actuelle. En contrepartie, le concept est censé pouvoir traiter un plus grand nombre de vols et réduire les retards. S'il n'est pas certain que les compagnies aériennes acceptent ce compromis, il est néanmoins certain que l'augmentation de la durée de vol ne doit pas être trop importante, surtout lorsque le prix du carburant augmente. Ainsi le critère d'optimisation va tenter de réduire cet allongement des trajectoires et pourquoi pas, permettre la création d'un réseau meilleur que celui utilisé actuellement. Le critère d'optimisation a alors été choisi de manière arbitraire.

Critère d'optimisation : Allongement global moyen des trajectoires [1] par comparaison avec la longueur des trajectoires directes.

Le choix d'une trajectoire est alors le plus court chemin passant par le réseau de routes principal entre une paire composée d'un aéroport d'origine et d'un aéroport de destination, les deux pouvant se situer hors de la zone européenne couverte par le réseau. Pour chaque extrémité, les points de croisement les plus proches (si l'aéroport concerné se trouve en Europe, ils sont au nombre de quatre, s'il est à l'extérieur, il y en a deux voire un seul) sont facilement calculable. Connaissant les plus courts chemins entre tous les points de croisement pris deux à deux (voir chapitre 4), il est alors aisé de calculer la trajectoire la plus courte. On remarquera alors que pour une paire origine-destination donnée, une unique trajectoire est prise en compte lors du calcul du critère, à savoir la plus courte.

Une fois le calcul de distance entre chaque paire d'aéroports effectué, l'évaluation des nouvelles trajectoires se fait par comparaison avec la route directe, c'est-à-dire la trajectoire la plus courte entre deux aéroports [2] (donc sans passer par un quelconque réseau de routes), ce qui, probablement, correspondrait à la solution optimale du point de vue des compagnies aériennes mais n'est pas envisageable dans le cadre d'un contrôle *Sector-Less*. Chaque différence est pondérée par le nombre d'aéronefs empruntant chacune des trajectoires durant une journée[3].

La valeur de ce critère, appliqué au réseau initial de routes, est 31.4%, ce qui veut dire qu'en moyenne un aéronef devra parcourir 31.4% de distance en plus que s'il volait en route directe.

[1] L'allongement des trajectoires utilisé dans le calcul du critère est celui de la projection des trajectoires au sol. On ne tient alors pas compte, dans le calcul du critère, des changements de niveaux.

[2] On ne tient pas compte ici des procédures spécifiques aux différents aéroports qui pourraient intervenir, celles-ci devant être totalement réécrites une fois un nouveau réseau de routes principales choisi afin de s'adapter a ce dernier.

[3] C'est la journée du 21 juin 2002 qui a été choisie arbitrairement car elle correspond, avec plus de 28542 vols et 10738 paires origine-destination distinctes à l'une des journées les plus chargées de l'année.

3.4 Optimisation du réseau initial

3.4.1 Justification du choix des techniques d'optimisation

Le choix d'une technique d'optimisation se fait tout d'abord en fonction de la nature du problème et également de sa difficulté. Certains problèmes peuvent être résolus par des méthodes déterministes, d'autres sont trop difficiles. Les problèmes d'optimisation difficile sont notamment les problèmes fortement combinatoires, les problèmes NP-difficiles ou encore ceux pour lesquels il n'existe pas de méthode de résolution exacte connue. Pour résoudre ceux-ci, un ensemble de méthodes, les *méta-heuristiques*, a été développé. Ces dernières sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents sans nécessiter de changements profonds dans l'algorithme employé. On trouve, parmi les méta-heuristiques, les algorithmes de recuit simulé, les algorithmes de colonies de fourmis ou encore les algorithmes génétiques. Ces méthodes ont en commun une approche stochastique, c'est-à-dire un parcours partiellement aléatoire de l'espace des solutions. Ce parcours peut alors être orienté vers la solution optimale (ou du moins vers une solution proche de l'optimum) par différents mécanismes, souvent inspirés par le monde réel que ce soit par la physique (recuit simulé), par la biologie de l'évolution (algorithmes génétiques) ou encore par l'observation du comportement du monde animal (algorithmes de colonies de fourmis).

S'il existe des résultats théoriques pour certaines méta-heuristiques, notamment sur la convergence théorique du recuit simulé (une présentation générale des résultats est faite par [Dréo 03]) et sur la convergence asymptotique des algorithmes génétiques [Cerf 91], ceux-ci sont peu exploitables en pratique lors du choix de la méthode à utiliser et/ou du réglage des paramètres de ces algorithmes.

Si parfois la nature du problème peut influencer l'utilisation de certaines méta-heuristiques plutôt que d'autres (difficulté d'écrire un opérateur de croisement qui ait du sens pour un algorithme génétique par exemple), il est parfois difficile de savoir à l'avance quelle sera la meilleure méta-heuristique.

L'optimisation du réseau de routes telle qu'elle est abordée dans cette thèse va donc faire appel aux méta-heuristiques pour les raisons suivantes :

- l'absence de formulation analytique de ce problème rend impossible sa comparaison avec d'autres problèmes connus et ainsi d'envisager l'utilisation de méthode classique ;
- le problème est non linéaire ;
- le problème est continu mais n'est pas réellement combinatoire. L'utilisation de méthodes de recherche complète telle que la programmation par contraintes parait peu évidente ici, en partie par la difficulté de décrire les variables du problème, les contraintes les reliant, etc.

Dans le cadre de cette thèse, il a donc été choisi, parmi les méta-heuristiques connues, d'utiliser un algorithme génétique (chapitre 5), puis un algorithme de recuit simulé (chapitre 6). La comparaison des résultats donnés pas ces deux méta-heuristiques permettra d'évaluer la pertinence de l'utilisation d'une méthode plutôt que l'autre.

3.4.2 Optimisation pas à pas

Au vue de la difficulté du problème, l'optimisation du réseau initial se fait de manière itérative en utilisant un algorithme génétique ou un recuit simulé. L'idée consiste donc à déplacer certains points de croisement afin de générer, à partir d'un réseau donné, un second, plus ou moins différent, et cela afin de raccourcir les trajectoires des différents vols. Ainsi on dira que l'optimisation se fait par déformation du réseau initial de routes.

Néanmoins le nouveau réseau ainsi généré se doit de respecter la topologie propre au concept de *Sector-Less*. Il faut donc que :

- la topologie ne change pas, c'est-à-dire que les croisements restent des sommets de graphe possédant exactement quatre arcs, rentrant et sortant. On ne peut donc ni enlever ni ajouter ni sommet ni arc ;
- les axes ne se croisent pas hors des points de croisement afin de restreindre les conflits aux endroits prédéfinis (on ne peut donc pas, par exemple, inverser la position de deux sommets).

De plus, deux points de croisement ne peuvent pas se trouver à une distance inférieure à 100 km. Cette distance, choisie arbitrairement, doit permettre d'éviter le chevauchement de deux points de croisement mais également faire en sorte qu'un aéronef passe un minimum de temps sur les axes avant d'entrer à nouveau dans un point de croisement.

3.4.3 Conservation de la topologie

Il faut s'assurer que l'effet que la modification du réseau initial de route va avoir sur l'aménagement de l'espace aérien reste compatible avec les règles promues par le concept *Sector-Less*. Si le processus de déformation n'a d'impact ni sur la séparation des flux, ni sur l'allocation des niveaux de vol pour chaque route, la forme des points de croisement doit en revanche être adaptée. La figure 3.2 présente ainsi un croisement modifié afin de correspondre au nouveau réseau de routes. Ses dimensions ainsi que la position relative des quatre balises le délimitant par rapport au centre de celui-ci ont évolué. Le seul paramètre qui reste fixe est la distance séparant les routes allant dans des directions opposées.

3.4.4 Limitations

Si le processus d'optimisation va permettre de fournir aux aéronefs la route la plus courte possible entre deux aéroports tout en essayant, grâce au concept *Sector-Less*, de minimiser les retards, il faut néanmoins s'assurer que les nouvelles routes sont « utilisables » par les aéronefs. La figure 3.3 présente un réseau de routes déformé sur lequel la trajectoire la plus courte pour aller de Moscou à Madrid impose aux aéronefs d'effectuer des angles de virages non réalisables en réalité, si ce n'est du point de vue performance, du moins du point de vue utilisation « normale » d'une aéronef.

Il a donc été imposé une limitation des angles de virage des trajectoires. Au vue de ce qui est fait à l'heure actuelle, on imposera aux routes de ne pas contenir d'angles inférieurs à 90 degrés.

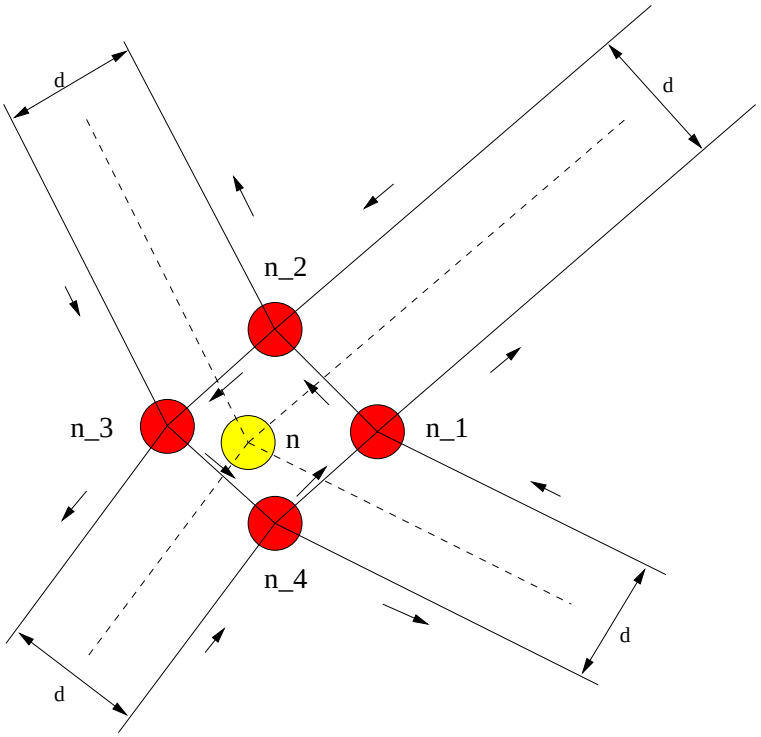

FIG. 3.2 – Adaptation de la structure des points de croisement à la déformation du réseau de routes

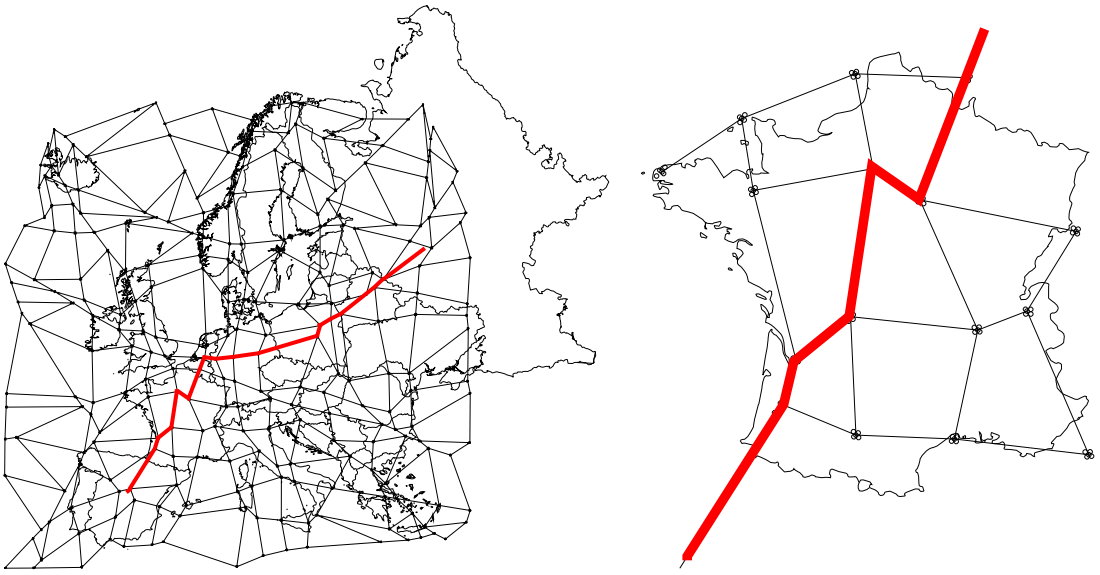

FIG. 3.3 – Exemple de trajectoire non utilisable entre Paris et Moscou

Deuxième partie

Résolution et simulation

Chapitre 4

Plus courts chemins dans un graphe et calcul de trajectoire

Le calcul du critère présenté chapitre 3 fait appel au calcul de tous les plus courts chemins entre tous les points du graphe pris deux à deux. L'utilisation de méta-heuristiques va rendre ce calcul fréquent. Dans le cas d'un algorithme génétique (chapitre 5), il faut évaluer la valeur du critère pour chaque individu à chaque génération et dans le cas du recuit simulé (chapitre 6), il faut évaluer la valeur du critère sur un réseau de routes à chaque itération.

Ce chapitre présente, dans un premier temps, section 4.1 quelques définitions et théorèmes utiles à la compréhension du problème du calcul des plus courts chemins dans un graphe puis, section 4.2, quelques algorithmes simples servant à effectuer ce calcul. Par la suite, section 4.3, les meilleurs algorithmes connus pour « maintenir » ces chemins après un changement dans le graphe sont décrits, c'est-à-dire les algorithmes permettant de ne pas recalculer tous les plus courts chemins mais uniquement ceux concernés par une modification du graphe. Enfin ce chapitre présente différentes solutions envisagées pour créer un nouvel algorithme de maintien des plus courts chemins, tout d'abord par l'utilisation des invariants issue de la programmation par contraintes (section 4.4.1) puis, section 4.4.2, par l'écriture d'un algorithme *ad-hoc*.

4.1 Plus courts chemins et graphes orientés

Dans cette section nous nous intéressons aux plus courts chemins dans un graphe orienté entre une source donnée et tous les autres sommets du graphe. Ce problème est désigné comme le *single-source shortest path problem*. La section 4.1.1 présente les définitions et propriétés des plus courts chemins, la section 4.1.2 justifie la limitation du problème aux graphes n'ayant pas d'arc de poids négatif et la section 4.2 présente les meilleurs algorithmes connus pour effectuer ce calcul.

4.1.1 Définitions et propriétés

Cette section présente quelques définitions sur les graphes orientés ainsi que la notion de plus court chemin dans un graphe [Diestel 00, Demetrescu 01].

Définition 4.1 *Un* **chemin** *est un graphe $G = (V, E)$ non nul de la forme :*

$$V = \{v_0, v_1, ..., v_k\} \qquad E = \{(v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k)\}$$

où les v_i sont tous distincts. Les sommets v_0 et v_k sont reliés par G et sont appelés **extrémités**. *On notera $v_0 \rightsquigarrow v_k$ ou $\langle v_0, ..., v_k \rangle$ le chemin de v_0 à v_k.*

Définition 4.2 *Soit $G = (V, E, w)$ un graphe orienté où V est l'ensemble des sommets du graphe, E l'ensemble des arcs et $w : E \to \mathbb{R}$ une fonction qui à tout arc (u, v) du graphe associe un nombre réel $w(u, v)$ appelé le* **poids** *de l'arc.*

Grâce à cette définition nous pouvons introduire le concept de longueur d'un chemin comme la somme des poids des arcs qui le composent :

Définition 4.3 *Soit $G = (V, E, w)$ un graphe orienté et $\pi = \langle v_0, ..., v_k \rangle$ un chemin dans G. La* **longueur** *de π est :*

$$l(\pi) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Il est alors possible de définir la notion de plus court chemin :

Définition 4.4 *Soit $G = (V, E, w)$ un graphe orienté. Un* **plus court chemin** *entre $u \in V$ et $v \in V$ est un chemin $\pi^* : u \rightsquigarrow v$ vérifiant l'équation :*

$$l(\pi^*) = \min_{\pi : u \rightsquigarrow v} l(\pi)$$

La fonction w prenant ses valeurs dans $\mathbb{R}$, la présence de cycles de longueur négative dans un graphe est possible. Or, s'il existe un chemin de la forme $\pi = \langle u, ..., \mathbf{c}, w_0, ..., w_k, \mathbf{c}, ..., v \rangle$ tel que $\gamma = \langle \mathbf{c}, w_0, ..., w_k, \mathbf{c} \rangle$ est un cycle de longueur négative, alors $\min_{\pi : u \rightsquigarrow v} l(\pi) = -\infty$. Comme il n'existe pas de chemin ayant une longueur infinie, il ne peut exister de plus court chemin entre u et v. Par la suite, on considérera donc uniquement les graphe sans cycle négatif afin de s'assurer de l'existence d'un plus court chemin entre n'importe quel couple de sommets du graphe dès lors qu'il existe au moins un chemin entre ces sommets. On peut alors définir la notion de distance d'une source vers un sommet quelconque du graphe.

Définition 4.5 *Soit $G = (V, E, w)$ un graphe orienté. Soit $s \in V$ un sommet* **source**. *On définit la fonction* **distance** *$d_s : V \to \mathbb{R}$ tel que :*

$$\forall v \in V, d_s(v) = \begin{cases} \min_{\pi : u \rightsquigarrow v} l(\pi) & \text{si } v \text{ est atteignable} \\ +\infty & \text{sinon} \end{cases}$$

Théorème 4.1 *Soient $G = (V, E, w)$ un graphe orienté et $s \in V$ un sommet source. Soit d_s la fonction distance. Pour tout $(u, v) \in E$ tel que $-\infty < d_s(u) < +\infty$ la condition de Bellman est satisfaite :*

$$d_s(v) \leq d_s(u) + w(u, v)$$

De plus si $\langle s, ..., u, v\rangle$ est le plus court chemin de s à v alors :

$$d_s(v) = d_s(u) + w(u, v)$$

Preuve Par l'absurde : supposons qu'il existe un arc $(u, v) \in E$ tel que $d_s(v) > d_s(u) + w(u, v)$. Soit $\nu = \langle s, ..., u\rangle$ le plus court chemin de s à u et $\mu = \langle s, ..., u, v\rangle$. On a

$$\begin{aligned} l(\nu) &= d_s(u) \\ l(\mu) &= l(\nu) + w(u, v) = d_s(u) + w(u, v) \end{aligned}$$

Par hypothèse on a $l(\mu) < d_s(v)$ ce qui contredit le fait que $d_s(v) = \inf_{\pi : s \rightsquigarrow v} l(\pi)$. □

On définit ensuite l'arbre des plus courts chemins (ou *shortest path tree*).

Définition 4.6 *Soient $G = (V, E, w)$ un graphe orienté et $s \in V$ un sommet source. On appelle* **arbre des plus courts chemins ayant s comme racine** *tout arbre $T(s) = (V', E')$ tel que :*

1. *$V' \subseteq V$ et $E' \subseteq (V' \times V') \cap E$;*
2. *$\forall v \in V'$, $-\infty < d_s(v) < +\infty$;*
3. *$\forall (u, v) \in E'$, $d_s(v) = d_s(u) + w(u, v)$.*

Ainsi un arbre des plus courts chemins $T(s)$ contient tout les sommets v de G atteignables depuis la source s ($d_s(v) < +\infty$) sans contenir de cycle de longueur négative ($-\infty < d_s(v)$). Tout chemin dans $T(s)$ est donc un plus court chemin dans G.

Il est possible d'avoir, pour une même destination, plusieurs plus courts chemins. Comme l'arbre des plus courts chemins n'en contient qu'un seul, celui-ci peut ne pas être unique.

Grâce aux définitions précédentes, il est désormais possible de définir la notion de calcul des plus courts chemins venant d'une source unique (*single-source shortest paths problem*).

Définition 4.7 *Soient $G = (V, E, w)$ un graphe orienté et $s \in V$ un sommet source. On définit* **le problème des plus courts chemins venant d'une source unique** *comme étant le calcul d'un arbre des plus courts chemins ayant s comme racine.*

De même on peut alors introduire le concept des plus courts chemins toutes paires (ou *all-pair shortest paths*).

Définition 4.8 *Soient $G = (V, E, w)$ un graphe orienté. On définit* **le problème des plus courts chemins toutes paires** *comme étant le calcul des plus courts chemins entre toutes les paires de sommets possibles du graphe.*

4.1.2 Technique de repondération

On présente ici une technique, dit de *reweighting* ou repondération, permettant, à partir d'un graphe quelconque, en modifiant le poids des arcs qui le composent, de générer un graphe orienté sans arc de poids négatif. On obtient grâce à cette technique une propriété importante : les plus courts chemins sont les mêmes dans le graphe initial et dans le graphe ainsi généré. Ceci permet de justifier l'utilisation d'algorithmes ne travaillant que sur des graphes à arcs de poids non négatifs.

Définition 4.9 *Soit $G = (V, E, w)$ un graphe orienté. Soit $h : V \to \mathbb{R}$ une fonction quelconque définie sur l'ensemble des sommets du graphe. On appellera h la fonction* **potentielle**. *On définit $\hat{w} : E \to \mathbb{R}$ la fonction de* **repondération** *de w telle que :*

$$\forall (u, v) \in E, \hat{w}(u, v) = w(u, v) + h(u) - h(v)$$

Même si la fonction h utilisée dans la définition précédente est totalement arbitraire, on obtient la propriété suivante (voir [Cormen 01] pour la démonstration).

Théorème 4.2 *Soit $G = (V, E, w)$ et $\hat{G} = (V, E, \hat{w})$ deux graphes orientés ayant le même ensemble de sommets et d'arcs et où $\hat{w}$ est la fonction de repondération de w. Un chemin π^* est un plus court chemin dans G si et seulement si π^* est un plus court chemin dans $\hat{G}$. De plus, γ est un cycle de longueur négative dans G si et seulement si γ est un cycle de longueur négative dans $\hat{G}$.*

Si la fonction potentielle est la fonction de distance d_s alors $\hat{w}$ est une fonction de repondération à valeur dans $\mathbb{R}^+$.

Théorème 4.3 *Soient $G = (V, E, w)$ un graphe orienté, $s \in V$ un sommet source et $d_s : V \to \mathbb{R}$ la fonction distance depuis s. Si $\hat{w}(u, v) = w(u, v) + d_s(u) - d_s(v)$ alors $\forall (u, v) \in E$, $\hat{w}(u, v) \geq 0$*

Preuve Le théorème 4.1 nous assure que $d_s(v) \leq d_s(u) + w(u, v)$. Ainsi :

$$0 \leq d_s(u) + w(u, v) - d_s(v) = \hat{w}(u, v)$$

□

On remarque que cette technique de repondération ayant pour fonction potentielle la fonction de calcul de la distance n'est pas utilisable pour résoudre le problème du calcul de l'arbre des plus courts chemins puisqu'il faut que les différentes distances soient déjà calculées. En revanche c'est cette technique qui est à la base des meilleurs algorithmes pour résoudre la version dynamique du problème (section 4.3) et permet de limiter ceux-ci aux graphes d'arcs de poids non négatif.

4.2 Calculs des plus courts chemins

4.2.1 Algorithme de Dijkstra

Dans cette section nous présentons un algorithme permettant de résoudre le problème du calcul de l'arbre des plus courts chemins. Compte tenu de notre problème réel qui est l'évaluation du critère d'optimisation, nous nous limitons ici aux graphes dont les arcs sont tous de poids positif ou nul. C'est l'algorithme de [Dijkstra 59] que vous détaillons ici.

L'algorithme de Dijkstra permet de calculer toutes les distances minimales et les plus courts chemins associés dans un graphe orienté $G = (V, E, w)$ depuis un sommet source s vers tout les sommets atteignables. Pour cela, l'algorithme développe itérativement tous les plus courts chemins depuis s, par ordre de distance croissante.

L'algorithme 1 décrit dans cette section s'applique sur un graphe orienté $G = (V, E, w)$ avec $w : E \rightarrow \mathbb{R}^+$, $s \in V$ étant le sommet source choisi. L'algorithme retourne l'arbre des plus courts chemins $T(s)$, c'est-à-dire une fonction de distance d_s ainsi qu'une fonction parent $p_s : V \rightarrow V \cup \{\text{nil}\}$ telle que $p_s(v) = u$ si et seulement si $(u, v) \in T(s)$. Comme la source n'a pas de parent dans T(s), on utilise la valeur *nil* pour $p_s(s)$.

Algorithme 1 Algorithme de Dijkstra

Entrées: $G = (V, E, w), s$
Sorties: $T(s)$

```
1:  pour tout v ∈ V faire
2:      d_s[v] ← +∞
3:      p_s[v] ← nil
4:  fin pour
5:  d_s[s] ← 0
6:  Q ← V
7:  tantque Q ≠ ∅ faire
8:      soit u ∈ Q tel que d_s(u) = min_{w∈Q} d_s(w)
9:      Q ← Q − {u}
10:     pour tout v ∈ Q tel que (u, v) ∈ E faire
11:         si d_s(v) > d_s(u) + w(u, v) alors
12:             d_s[v] ← d_s(u) + w(u, v)
13:             p_s[v] ← u
14:         fin si
15:     fin pour
16: fin tantque
17: retourner d_s, p_s
```

L'algorithme se décompose en deux phases :
- une phase d'initialisation lignes 1 à 6 ;
- une boucle principale lignes 7 à 16. Cette dernière permet le parcourt progressif de tous les sommets du graphe de manière croissante avec leur distance par rapport à s

(ligne 8) puis recherche tout arc sortant à destination d'un sommet non encore traité. Si une meilleure solution est trouvée, d_s et p_s sont mises à jour (lignes 12 et 13). On remarque que si le sommet n'est pas atteignable alors d_s vaut $+\infty$.

Soit $n = |V|$ et $m = |E|$, la complexité de cet algorithme est en $\mathcal{O}(m + n * \log(n))$.

4.2.2 Algorithme de Floyd-Warshall

L'algorithme de Floyd-Warshall est sans doute le plus simple des algorithmes de plus courts chemins à mettre en œuvre. En revanche, et contrairement à l'algorithme de Dijkstra, il ne résout que le problème toutes paires.

Travaillant sur des graphes orienté $G = (V, E, w)$ où w en à valeur dans $\overline{\mathbb{R}^+}$, l'algorithme repose sur la propriété suivante :

Théorème 4.4 *Soit $G = (V, E, w)$ un graphe orienté avec $w : E \to \overline{\mathbb{R}^+}$. Si $\pi = \langle v_0, ..., v_k \rangle$ est un plus court chemin de v_0 à v_k alors pour tout $i \in [0, k]$, $\langle v_0, ..., v_i \rangle$ et $\langle v_i, ..., v_k \rangle$ sont deux plus courts chemins.*

L'algorithme 2 décrit par [Floyd 62] calcule la matrice des distances minimales d entre toutes les paires de sommets du graphe. Soit n le nombre de sommets.

Algorithme 2 Algorithme de Floyd-Warshall

```
1: pour i, j = 1 à n faire
2:    si i = j alors
3:       d[i, j] ← 0
4:    sinon
5:       d[i, j] ← w(i, j) si (i, j) ∈ E, +∞ sinon
6:    fin si
7: fin pour
8: pour k = 1 à n faire
9:    pour i = 1 à n faire
10:      pour j = 1 à n faire
11:         d[i, j] ← min(d(i, j), d(i, k) + d(k, j)
12:      fin pour
13:   fin pour
14: fin pour
15: retourner d
```

Les lignes 1 à 7 permettent l'initialisation de la matrice des distances avec les données du graphe. Dans le cas d'un graphe non orienté, cette matrice peut être triangulaire. On remarque néanmoins que l'algorithme ne conserve que les distances. Il suffit alors de rajouter une instruction ligne 11 pour conserver également les chemins. Il faut néanmoins prévoir une structure de données permettant d'insérer facilement un sommet dans un chemin.

La complexité de cet algorithme est trivialement en $\mathcal{O}(n^3)$ ce qui restreint son application aux graphes de taille « raisonnable ». On rappelle que la complexité de l'algorithme de Dijkstra sur le problème toutes paires est, avec $n = |V|$ et $m = |E|$, en $\mathcal{O}(mn + n^2 \log(n))$.

En revanche une fois les matrices des distances et des chemins calculées, on accède à un chemin en $\mathcal{O}(1)$ alors que lorsqu'on utilise un arbre des plus courts chemins, l'accès se fait en $\mathcal{O}(n)$ dans le pire cas.

4.3 Algorithmes dynamiques par repondération

4.3.1 Définition du problème

Une fois calculé l'arbre des plus courts chemins pour une source donnée dans un graphe, les algorithmes dynamiques s'intéressent au calcul de ces chemins lorsque le graphe initial est modifié.

La solution la plus simple pour résoudre ce problème consiste, une fois le graphe modifié, à réutiliser l'algorithme de Dijkstra ou de Floyd-Warshall et donc à recalculer tous les différents plus courts chemins. Si cette solution a le mérite d'être simple, il est évident qu'elle est loin d'être optimale. En effet, si les changements sont nombreux et tous les chemins affectés par ces modifications alors cette solution est efficace. Mais dans le cas d'un changement « localisé », si peu de plus courts chemins sont affectés par la modification, il paraît plus efficace de ne pas tout recalculer mais seulement les chemins concernés. On parle alors de *maintien* des différents plus courts chemins.

Plusieurs études expérimentales testant différents algorithmes de calcul des plus courts chemins [Buriol 03, Demetrescu 04] ont montré que les algorithmes dynamiques basés sur la technique de repondération donnent de meilleurs résultats en terme de vitesse de calcul que le recalcul de tous les plus courts chemins.

L'idée principale des algorithmes dynamiques basés sur la technique de repondération est de maintenir la fonction de calcul des distances ainsi que les arbres des plus courts chemins après les avoir calculés une première fois sur le graphe initial en utilisant l'algorithme de Dijkstra. La fonction distance permet ainsi de définir une fonction de repondération non négative. Lors d'une mise à jour, le processus utilise un algorithme de Dijkstra sur le graphe repondéré, les plus efficaces travaillant uniquement sur le sous-graphe réellement modifié et non pas sur le graphe dans sa totalité. Les mises à jour peuvent être séparées en deux cas distincts qui seront étudiés en détails par la suite :

- l'augmentation de la valeur du poids ou la suppression d'un arc (ce qui revient à attribuer une valeur de poids infinie à l'arc concerné) ;
- la diminution de la valeur du poids ou l'ajout d'un arc (ce qui correspond à changer une valeur infinie en valeur réelle).

Parmi les meilleurs algorithmes permettant de résoudre les problèmes dynamiques, il en existe deux, [Ramalingam 96] et [King 99], qui donnent les meilleurs résultats mais que les tests effectués par [Buriol 03] et [Demetrescu 04] peinent à départager.

Il faut alors remarquer que les algorithmes cités ici sont des algorithmes écrits pour

résoudre le problème du calcul de l'arbre des plus courts chemins provenant d'une unique source. Ils ont été simplement étendus au problème des plus courts chemins toutes paires en étant exécutés $|V|$ fois sur un graphe orienté $G = (V, E, w)$. Si le calcul des $|V|$ arbres des plus courts chemins paraît produire une quantité non négligeable de redondances, il n'existe pas, à ma connaissance dans la littérature, d'algorithme spécialement écrit pour le problème toutes paires et plus efficace que la répétition, pour chaque sommet, de l'algorithme de [Ramalingam 96] ou de [King 99].

Même s'il est impossible de trouver une différence significative entre les performances des deux algorithmes en fonction des problèmes étudiés, l'algorithme de [Ramalingam 96] semble être légèrement meilleur dans la plupart des cas. De manière totalement arbitraire, c'est donc ce dernier qui sera présenté dans les sections suivantes et utilisé par la suite comme algorithme de référence.

4.3.2 Algorithme de Ramalingam et Reps

Comme annoncé précédemment, l'algorithme de Ramalingam et Reps est basé sur la résolution du problème de plus courts chemins pour une source donnée. On considère désormais qu'il existe toujours, pour chaque sommet du graphe considéré, un arbre des plus courts chemins ainsi qu'une fonction distance et qu'il faut ensuite itérer le processus $|V|$ fois pour les obtenir tous. L'algorithme travaille sur un graphe orienté $G = (V, E, w)$ où w est à valeurs dans $\overline{\mathbb{R}^+}$. La fonction d_s retourne la distance de s au sommet considéré et p_s retourne le père du sommet considéré dans le plus court chemin qui le lie à s.

L'algorithme utilise un maximier H et ainsi que diverses fonctions :

- `NonVide` qui renvoie `vrai` si le maximier considéré n'est pas vide ;
- `Ajoute` qui ajoute un élément dans un maximier ;
- `Ajuste` qui remplace le coût d'un élément du maximier considéré par sa nouvelle valeur si celui-ci y est ou l'ajoute sinon ;
- `EnleveMin` qui enlève l'élément de coût minimal d'un maximier.

À ces fonctions, vient s'ajouter la fonction `Out` (respectivement `In`) retournant, pour un sommet v donné, l'ensemble des sommets u relié à v par un arc orienté (v, u) (respectivement (u, v)).

Augmentation de poids et suppression d'arc (algorithme 3)

Supposons l'augmentation de la valeur du poids d'un arc (u, v) d'une valeur ϵ. Si (u, v) n'est pas dans $T(s)$, il n'y a rien à faire. En revanche, si (u, v) est dans $T(s)$ alors seulement les sommets inclus dans $T(v)$ peuvent changer et leur distance à s être mise à jour.

La figure 4.1 illustre la manière dont l'augmentation d'un arc (u, v) affecte l'arbre des plus courts chemins et le résultat de la découverte d'un nouveau plus court chemin. Dans ce cas, le plus court chemin de s à t change car la distance en passant par p est inférieure à la nouvelle distance qui tient compte de l'augmentation de poids. Dans ce cas, le sous-arbre de $T(s)$ ayant v comme racine $(T_v(s))$ est amputé de celui ayant t pour racine $(T_t(s))$.

Algorithme 3 Algorithme de Ramalingam et Reps : maintien des plus courts chemins lors de l'augmentation du poids d'un arc

Entrées: $G = (V, E, w), (u, v), d_s, p_s$

si $p_s(v) = u$ **alors**
  pour tout $u' \in \texttt{In}(v)$ **faire**
    si $d_s(v) = d_s(u') + w(u', v)$ **alors**
      $p_s[v] \leftarrow u'$, **Exit programme**
    fin si
  fin pour
  $\mathcal{P} = \{v\}$, $\mathcal{A} = \{v\}$
  tantque $\mathcal{P} \neq \emptyset$ **faire**
    Soit $p \in \mathcal{P}$, $\mathcal{P} \leftarrow \mathcal{P} - \{p\}$
    $d_s[p] \leftarrow +\infty$
    pour tout $q \in \texttt{Out}(p)$ **faire**
      si $p_s(q) = p$ **alors**
        pour tout $p' \in \texttt{In}(q)$ **faire**
          si $d_s(q) = d_s(p') + w(q, p')$ **alors**
            $p_s[q] \leftarrow p'$, **Exit boucle pour tout**
          fin si
        fin pour
        si $p_s(q) = p$ **alors**
          $\mathcal{P} \leftarrow \mathcal{P} \cup \{p\}$, $\mathcal{A} \leftarrow \mathcal{A} \cup \{p\}$
        fin si
      fin si
    fin pour
  fin tantque
  pour tout $p \in \mathcal{A}$ **faire**
    pour tout $o \in \texttt{In}(p)$ **faire**
      si $d_s(p) > d_s(o) + w(o, p)$ **alors**
        $d_s[p] \leftarrow d_s(o) + w(o, p)$
        $p_s[p] \leftarrow o$
      fin si
    fin pour
    si $d_s(p) \neq +\infty$ **alors**
      $\texttt{Ajoute}(p, d_s(p))$ dans H
    fin si
  fin pour
  tantque $\texttt{NonVide}(H)$ **faire**
    $p = \texttt{EnleveMin}(H)$
    pour tout $q \in \texttt{Out}(p)$ **faire**
      si $d_s(q) > d_s(p) + w(p, q)$ **alors**
        $d_s[q] \leftarrow d_s(p) + w(p, q)$
        $p_s[q] \leftarrow p$
        $\texttt{Ajuste}(q, d_s(q))$ dans H
      fin si
    fin pour
  fin tantque
fin si

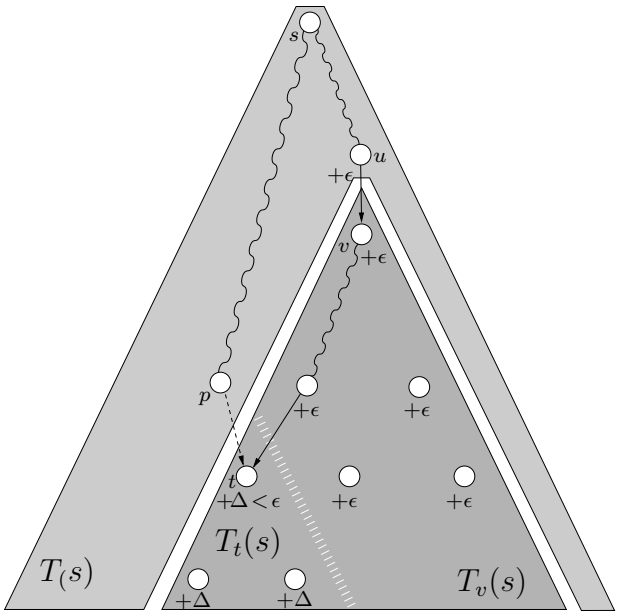

FIG. 4.1 – Évolution de l'arbre des plus courts chemins par augmentation du poids de l'arc (u, v) : le sous-arbre de racine t se détache de celui de racine v

Description de l'algorithme 3 : L'algorithme ne s'exécute que si l'arc (u, v) dont le poids a été modifié est dans $T(s)$ (ligne 1). La première étape consiste à trouver dans le voisinage de v un autre arc (u', v) fournissant un plus court chemin de même longueur (lignes 2-6).

Deux ensembles sont initialisés ligne 7. $\mathcal{P}$ permettant le parcours des sommets du graphe afin de chercher les sommets affectés par la modification et $\mathcal{A}$ contenant l'ensemble des sommets affectés. Tous deux sont initialisés par le sommet v.

Si $T(s)$ contient l'arc (u, v) dont le poids a été modifié, on recherche les sommets affectés par la mise à jour (ligne 8-23). Pour tous les sommets p dans $\mathcal{P}$, leur distance à s devient infinie et tout arc (p, q) sortant de p est traversé. Si p est le père de q dans le plus court chemin de s à q alors tous les arcs (p', q) entrant dans q sont étudiés. On tente grâce à cela de trouver une alternative aux plus courts chemins qui passaient par p puisque $d_s(p)$ est désormais infinie. Si on trouve une alternative (ligne 15), on ajoute p à $\mathcal{P}$ et on stocke l'ensemble des sommets affectés par la modification (ligne 19).

La boucle lignes 24-34 permet la mise à jour des distances des chemins à destination des sommets affectés par l'augmentation du poids de l'arc (u, v). Si cette distance peut être diminuée, le plus court chemin correspondant est mis à jour (ligne 28). Tous les sommets dont la distance a été diminuée sont insérés dans le maximier H.

Les lignes 35-44 mettent à jour de la même manière que précédemment les chemins passant par les sommets affectés.

Diminution de poids et ajout d'arc (algorithme 4)

Supposons maintenant qu'on diminue le poids d'un arc (u, v) d'une valeur ϵ. Si la distance la plus courte de s à v reste inférieure à celle passant par u alors il n'y a rien à faire. Dans le cas contraire, les distances de s aux sommets du sous-arbre de $T(s)$ ayant v pour racine sont réduites et sont donc affectées par la modification.

La figure 4.2 illustre les événements induits par une diminution de poids. Dans ce cas, un sommet de $T(s)$ extérieur au sous-arbre ayant v pour racine ($T_v(s)$) trouve un nouveau parent dans ce sous-arbre. De ce fait, le sous-arbre de $T(s)$ ayant t pour racine ($T_t(s)$) se rattache à $T_v(s)$

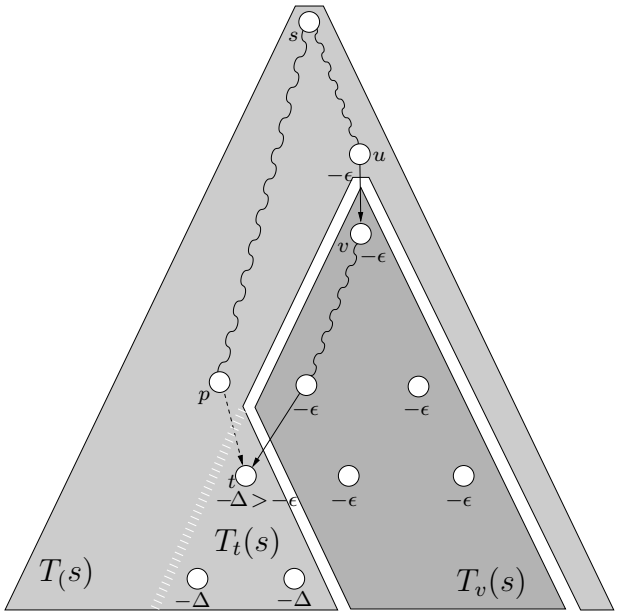

FIG. 4.2 – Évolution de l'arbre des plus courts chemins par diminution du poids de l'arc (u, v) : le sous-arbre de racine t s'attache à celui de racine v

Description de l'algorithme 4 : L'algorithme ne s'exécute que si l'arc (u, v) dont le poids a été modifié a une influence sur le plus court chemin de s à v (ligne 1). Dans ce cas on met à jour v dans $T(s)$ (lignes 2 et 3) et on initialise le maximier H.

La boucle lignes 5-14 permet la mise à jour des distances des chemins à destination des sommets affectés par la diminution du poids de l'arc (u, v). Si cette distance peut être diminuée, le plus court chemin correspondant est mis à jour. Tous les sommets dont la distance à s a été diminuée sont alors insérés dans le maximier H afin de propager la mise à jour.

Algorithme 4 Algorithme de Ramalingam et Reps : maintien des plus courts chemins lors d'une diminution du poids d'un arc

Entrées: $G = (V, E, w), (u, v), d_s, p_s$

si $d_s(v) > d_s(u) + w(u, v)$ **alors**
  $d_s[v] \leftarrow d_s(u) + w(u, v)$
  $p_s[v] \leftarrow u$
  `Ajoute`(v, d_v) dans H
  tantque `NonVide`(H) **faire**
    $p =$ `EnleveMin`(H)
    pour tout $q \in$ `Out`(p) **faire**
      si $d_s(q) > d_s(p) + w(p, q)$ **alors**
        $d_s[q] \leftarrow d_s(p) + w(p, q)$
        $p_s[q] \leftarrow p$
        `Ajuste`$(q, d_s(q))$ dans H
      fin si
    fin pour
  fin tantque
fin si

4.4 Maintien des plus courts chemins pour le problème toutes paires

Comme on l'a vu précédemment, les meilleurs algorithmes de maintien des plus courts chemins sont basés sur l'algorithme de Dijkstra. Or celui-ci permet uniquement de résoudre le problème du calcul des plus courts chemins issus d'une source unique. Il en est de même des algorithmes de maintien. De façon surprenante, il n'existe pas, à ma connaissance, d'algorithme performant permettant de maintenir des chemins dans le cadre de la résolution du problème toutes paires.

Partant d'un algorithme résolvant le problème toutes paires (section 4.2.2), cette section présente différentes solutions permettant d'en déduire un algorithme de maintien des plus courts chemins, tout d'abord par l'utilisation des invariants issus de la programmation par contraintes (section 4.4.1) puis, section 4.4.2, par l'écriture d'un algorithme *ad-hoc*.

4.4.1 Invariants

La première idée pour maintenir les plus courts chemins a été de faire appel aux invariants, issus de la programmation par contraintes. Le lecteur pourra se référer à la thèse de Nicolas Barnier [Barnier 02] pour obtenir plus de détail sur celle-ci.

Les invariants sont des équations fonctionnelles entre des variables qui doivent être maintenues quand ces variables changent de valeur. Ces équations sont des dépendances directionnelles entre les variables : dans l'invariant $z = f(x, y)$, la valeur de z doit être calculée si x et/ou y ont été modifiés.

Dans le domaine de la bureautique, la plupart des tableurs numériques fonctionnent sur ce principe. Les invariants que nous allons considérer par la suite viennent de la recherche locale et du langage de modélisation Localizer introduit par [Michel 97] où ils sont utilisés pour automatiser la mise à jour incrémentales de structures de données et des critères qui définissent un voisinage et une fonction de coût.

Un exemple d'utilisation des invariants sur une somme peut s'écrire comme ceci : soit v un invariant. Nous noterons $\underline{v}$ sa valeur avant une mise à jour et $\bar{v}$ sa valeur après la mise à jour. Supposons ensuite que $v = \sum_{i=1}^{n} v_i$. La mise à jour d'un v_i implique alors la mise à jour de v **en temps constant** : $\bar{v} \leftarrow \underline{v} - \underline{v_i} + \overline{v_i}$.

FaCiLe et invariants

Les premiers invariants utilisés ont été ceux fournis par la librairie fonctionnelle de programmation par contraintes sur les domaines finis (entiers et ensembles d'entiers) FaCiLe [Barnier 01]. Cette librairie intègre toutes les fonctionnalités standards de création et de manipulation de variables (logiques) à domaine fini, d'expressions et de contraintes arithmétiques (éventuellement non-linéaires), de contraintes globales (différence, cardinalité, tri, etc.) et de buts de recherche et d'optimisation.

Dans cette librairie, un invariant est une référence *réversible* (c'est-à-dire dont les valeurs sont restaurées lors d'un retour arrière ou *backtrack* [Barnier 02]). Il peut être :

- une constante ;
- une variable ;
- le résultat d'une fonction ayant pour paramètres d'autres invariants ;
- un attribut d'une structure de donnée dynamique (par exemple la valeur maximale d'un domaine).

L'avantage principal de l'usage de cette librairie est la possibilité de prototyper rapidement notre algorithme de maintien. En effet, en plus de fournir la structure de données, elle fournit également plusieurs moyens permettant « d'invariantiser » les différentes fonctions nécessaires au calcul (y compris le `min`). L'algorithme de maintien consiste donc à :

- « invariantiser » le graphe, c'est-à-dire rendre invariant le poids des arcs ;
- générer la même matrice des plus courts chemins que celle générée par l'algorithme de Floyd-Warshall (section 4.2.2) et utilisant le même algorithme, sauf que les éléments de cette matrice sont, dans ce cas, des invariants, eux-mêmes résultats de différentes fonctions « invariantisées ».

Ainsi, à chaque modification du poids d'un arc, les fonctions de réveil incluses dans FaCiLe permettent de ne mettre à jour que les invariants concernés, donc les plus courts chemins concernés par la modification.

Le problème rencontré alors est la non spécificité des invariants. En effet, FaCiLe étant une librairie de programmation par contraintes, les mécanismes mis en jeu ici sont bien trop lourds pour être efficaces. Vu que l'algorithme de Floyd ne fait pas de recherche, toute la structure de backtrack, attachée à chaque invariant, est ici inutile.

Ainsi, le temps de mise à jour est plus lent que le temps mis par l'algorithme de Floyd-Warshall pour recalculer tous les plus courts chemins. Nous avons donc écrit notre propre

module d'invariants.

Invariants *ad-hoc*

Un module d'invariants nécessite, dans notre cas, en plus de la structure de données qui permet de relier un invariant aux invariants dont il dépend, une fonction de réveil et de mise à jour ainsi que toutes les fonctions effectuant des calculs sur des invariants (somme, concaténation de chemin, `min`, `if then else`, etc). Les fonctions étant mieux adaptées et la structure moins lourde, les résultats sont un peu meilleurs que ceux calculés en utilisant FaCiLe mais néanmoins insuffisants.

Le problème principal de ce genre d'approche est la consommation excessive de mémoire car l'algorithme, contrairement aux algorithmes dynamiques, stocke **tous** les plus courts chemins allant de i à j et pas seulement **un** plus court chemin. Comme nous l'avons vu précédemment, supposons que le chemin $i \rightsquigarrow j$ passe par les points du graphe $\langle k_1, k_2, ..., k_n \rangle$. Il est parfaitement envisageable que la mise à jour d'un arc k_i, k_{i+1} rallonge ce chemin et que le nouveau plus court chemin soit un chemin passant par des points totalement autres que les $k_1, ..., k_n$. De même, mettre à jour un arc quelconque du graphe peut raccourcir un chemin tel que le nouveau plus court chemin ne soit plus $\langle k_1, ..., k_n \rangle$ mais $\langle p_1, ..., p_n \rangle$, un chemin différent. Il faut alors, dans le cas du maintien du chemin $i \rightsquigarrow j$ par les invariants stocker en mémoire, non seulement le plus court chemin du moment $\langle k_1, ..., k_n \rangle$ mais également l'information que $\langle p_1, ..., p_n \rangle$ pourrait être plus court. Ainsi pour chaque paire i, j du graphe, le nombre de chemins à retenir *au cas où* est beaucoup trop important et donc trop gourmand en mémoire.

Nous avons donc développé une méthode pour maintenir plus efficacement les différents chemins pour le problème toutes paires.

4.4.2 Algorithme *ad-hoc*

De la même manière que l'algorithme de Ramalingam et Reps est basé sur l'algorithme de Dijkstra, l'algorithme décrit ici est basé sur l'incrémentalité de l'algorithme de Floyd-Warshall. Contrairement aux algorithmes inspirés de celui de Dijkstra, l'idée est de créer un unique algorithme qui fonctionne à la fois pour une augmentation et/ou une diminution de poids pour pouvoir effectuer plusieurs modifications en une seule fois. En effet les algorithmes présentés précédemment ne sont capables que de traiter les modifications au cas par cas.

La boucle principale de l'algorithme de Floyd-Warshall peut s'écrire :

Soit d la matrice des distances entre les n sommets du graphe.

```
for k = 1 to  n do
    for i = 1 to  n do
        for j = 1 to  n do
            d(i,j) ← min(d(i,j), d(i,k) + d(k,j))
```

La première observation que l'on peut formuler à propos de cet algorithme est que, pour un k fixé, la longueur du chemin entre i et j ($d(i,j)$) soit reste inchangée, soit ne dépend que de la longueur du chemin entre i et k ($d(i,k)$) et de celle entre j et k ($d(k,j)$). Considérons alors $n+1$ matrices d^k ($k \in [0,n]$), la matrice 0 étant celle représentant le graphe. On a :

$$\forall k \in [1,n], \forall i,j \in [1,n] \quad d^k(i,j) = min(d^{k-1}(i,j),\ d^{k-1}(i,k) + d^{k-1}(k,j))$$

$d^k(i,j)$ est alors le plus court chemin entre i et j passant par des sommets inférieurs ou égaux à k (on suppose ici que les sommets sont étiquetés par un numéro allant de 1 à n). La matrice d^n est donc la matrice des plus courts chemins telle que celle calculée par l'algorithme de Floyd-Warshall. Or, plutôt que de calculer chacune des valeurs de chaque matrice, on peut écrire chacune d'entre elles comme le résultat d'une équation fonctionnelle dont les variables sont des valeurs de la matrice précédente. La figure 4.3 illustre la dépendance entre ces $n+1$ matrices.

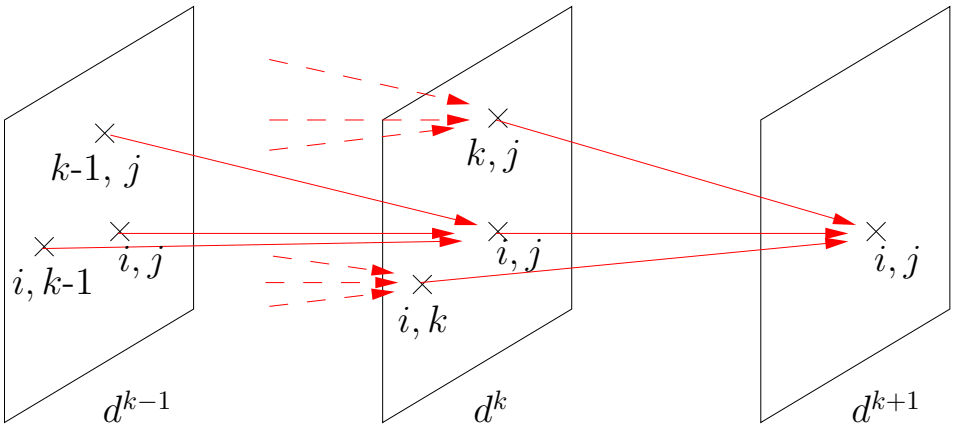

FIG. 4.3 – Dépendance entre matrices

Une fois l'initialisation des $n+1$ matrices effectuée, lorsqu'un arc voit son poids modifié, seuls les chemins dépendants de ce poids sont mis à jour .

Pour k fixé, la distance de u à v minimale par des sommets inférieure à k s'écrit :

$$d^k(u,v) = \min(d^{k-1}(u,v),\ d^{k-1}(u,k) + d^{k-1}(k,v)) \tag{4.1}$$

Lorsque le poids d'un ou plusieurs arcs est modifié, l'algorithme 5 met à jour les distances et les plus courts chemins concernés par cette modification. On notera $\mathcal{M}$ l'ensemble des arcs modifiés dans le graphe initial.

Soit k fixé. On distingue trois cas :

- $u \neq k, v \neq k$: la valeur mise à jour dans (4.1) est $d^{k-1}(u,v)$. On met donc à jour la valeur $d^k(u,v)$ si besoin est (lignes 4 - 7).
- $u = k$: la valeur mise à jour dans (4.1) est $d^{k-1}(k,v)$. Pour tout $i \in [0, n-1]$ on recalcule $d^k(i,v)$ (lignes 10 - 13).

– $v = k$: la valeur mise à jour dans (4.1) est $d^{k-1}(u, k)$. Pour tout $j \in [0, n-1]$ on recalcule $d^k(u, j)$ (lignes 17 - 20).

On recommence avec $k + 1$ en tenant compte des nouvelles valeurs mises à jour dans la matrice k.

Algorithme 5 Algorithme de maintien de plus courts chemins inspiré par l'algorithme de Floyd-Warshall

Entrées: $\mathcal{M}$ contenant l'ensemble des arcs modifiés

```
1: pour k = 1 à n + 1 faire
2:    M' ← ∅
3:    pour tout (u, v) ∈ M faire
4:       si d^k(u, v) > d^k(u, k) + d^k(k, v) alors
5:          d^k[u, v] ← d^k(u, k) + d^k(k, v)
6:          M' ← M' ∪ (u, v)
7:       fin si
8:       si u = k alors
9:          pour i = 1 à n faire
10:            si i ≠ u ∧ i ≠ v ∧ d^k(i, v) > d^k(i, k) + d^k(k, v) alors
11:               d^k[i, v] ← d^k(i, k) + d^k(k, v)
12:               M' ← M' ∪ (i, v)
13:            fin si
14:         fin pour
15:      sinon si v = k alors
16:         pour j = 1 à n faire
17:            si j ≠ u ∧ j ≠ v ∧ d^k(u, j) > d^k(u, k) + d^k(k, j) alors
18:               d^k[u, j] ← d^k(u, k) + d^k(k, j)
19:               M' ← M' ∪ (u, j)
20:            fin si
21:         fin pour
22:      fin si
23:   fin pour
24:   M ← M'
25: fin pour
```

Pour le calcul de la complexité, l'algorithme exécute n fois la boucle principale (ligne 1) et n mises à jour de distance lorsque $u = k$ ou $v = k$ (lignes 8 et 15). L'algorithme fait également autant de fois la boucle ligne 3 que le nombre d'arcs affectés par la modification, ce nombre évoluant à chaque k, sa borne supérieure est le nombre d'arcs du graphe. La complexité de cet algorithme est donc, dans le pire des cas, en $\mathcal{O}(mn^2)$ même si, en moyenne, le nombre d'arcs affectés de proche en proche est généralement nettement inférieur à m.

4.5 Conclusion

La complexité de l'algorithme de Ramalingam et Reps est en $\mathcal{O}(||\delta|| + |\delta| log |\delta|)$ avec $|\delta|$ le nombre de sommets affecté par la modification δ et $||\delta||$ le nombre de sommets affectés plus le nombre d'arcs dont au moins une des extrémités est un sommet affecté. Dans le pire des cas, la complexité est en $\mathcal{O}(m + n \log(n))$, la même que l'algorithme de Dijkstra. Dans le cas du calcul des plus courts chemins toutes paires, la complexité dans le pire des cas est en $\mathcal{O}(m * n + n^2 \log(n))$.

On se rend bien compte alors que l'algorithme de Ramalingam et Reps est de complexité moindre que celui inspiré de l'algorithme Floyd-Warshall. Néanmoins, dans certain cas (voir chapitre 7), ce dernier peut se révéler intéressant à utiliser.

Chapitre 5

Résolution par algorithmes génétiques

Les algorithmes génétiques (parfois appelés algorithmes évolutionnaires) sont inspirés du concept de sélection naturelle proposée par Charles Darwin. Le vocabulaire employé est directement calqué sur celui de la théorie de l'évolution et de la génétique. Nous parlerons donc ici d'individus (solutions potentielles), de populations, de gènes (variables), de chromosomes, de parents, de descendants, de reproductions, de croisements, de mutations, etc. Les points de l'espace de recherche sont alors les individus d'une population et la fonction à optimiser leur adaptation. Ces algorithmes font alors évoluer une population de manière itérative. Certains individus se reproduisent, d'autres mutent ou encore disparaissent et seuls les individus les mieux adaptés sont censés survivre. L'héritage génétique de chaque génération doit permettre à une population d'être de mieux en mieux adaptée, donc de correspondre de plus en plus au critère d'optimisation.

Les origines de ces algorithmes remontent aux années 60, avec les travaux de [Holland 62] sur les systèmes adaptatifs. L'ouvrage de référence [Goldberg 89] a fortement participé à leur essor. Ils sont désormais très populaires et leurs domaines d'applications sont de plus en plus variés.

Ils ont été utilisés à de nombreuses reprises pour résoudre divers problèmes de trafic aérien tel que l'optimisation du trafic au sol sur les grands aéroport [Gotteland 04], la résolution de conflits aériens [Granger 02] ou encore l'optimisation des flux de trafic aérien [Gianazza 04]. C'est pourquoi le choix d'une première méta-heuristique s'est porté sur les algorithmes génétiques.

5.1 Principes et généralités

D'une manière générale, les algorithmes génétiques sont utilisés pour rechercher les optima d'un critère d'optimisation défini sur un espace de recherche $\mathcal{E}$. Leur mise en œuvre nécessite :

- **La définition d'une fonction d'adaptation** f à maximiser, définie sur l'espace de recherche $\mathcal{E}$ et généralement à valeurs dans $[0;\ 1]$ ou $\mathbb{R}+$. Cette fonction peut également être appelée *fonction d'évaluation* ou encore *fitness*. Elle se définit en fonction du critère d'optimisation du problème : les extrema recherchés du critère doivent correspondre aux maxima de f.
- **Un codage des données**, associant à chaque point de l'espace de recherche une structure de données particulière, appelée *génotype* ou ensemble de *chromosomes*, qui caractérisera chaque *individu* de la population.
- **Une population initiale**, définie comme un ensemble d'individus, dont dériveront toutes les futures générations.
- **Des opérateurs d'évolution** de la population, permettant l'exploration, plus ou moins déterministe, de l'espace de recherche $\mathcal{E}$.
- **Un processus de sélection** des individus les mieux adaptés, qui sera appliqué sur chaque nouvelle génération d'individus.

La figure 5.1 illustre les principales étapes d'un algorithme génétique :

1. génération de la population initiale ;
2. génération d'une nouvelle population :
   (a) mesure de l'adaptation de chacune des séquences présentes ;
   (b) reproduction de chaque séquence en fonction de son adaptation. Les séquences les mieux adaptées se reproduisent mieux que les séquences inadaptées. La nouvelle population est composée des séquences après reproduction ;
   (c) croisement d'un certain nombre de paires d'individus tirées aléatoirement (voir section 5.1.4) ;
   (d) mutation d'un certain nombre d'individus choisis de manière aléatoire (voir section 5.1.4) ;
3. critère d'arrêt. Il peut être défini en fonction de plusieurs indicateurs tels que le nombre de générations, l'adaptation du meilleur individu, etc.

Comme souvent pour ce type de mécanisme, qui pourrait apparaître comme un peu magique car il peut être utilisé pour résoudre un grand nombre de problèmes, le « réglage » de paramètres des algorithmes génétiques est délicat. Le premier réglage consiste alors à bien choisir les divers paramètres.

5.1.1 Paramètres des algorithmes génétiques

La rapidité et la qualité de convergence d'un algorithme génétique sont influencées par de nombreux paramètres, souvent liés entre eux. Les principaux paramètres numériques sont :

- **La taille de la population** (N_{pop}) : plus le nombre d'individus est grand, plus la probabilité de convergence est élevée mais plus l'algorithme est lent pour calculer l'adaptation (évaluation de chaque individu) et générer la nouvelle population. La complexité de l'algorithme génétique s'exprime généralement en $\mathcal{O}(N_{pop})$.

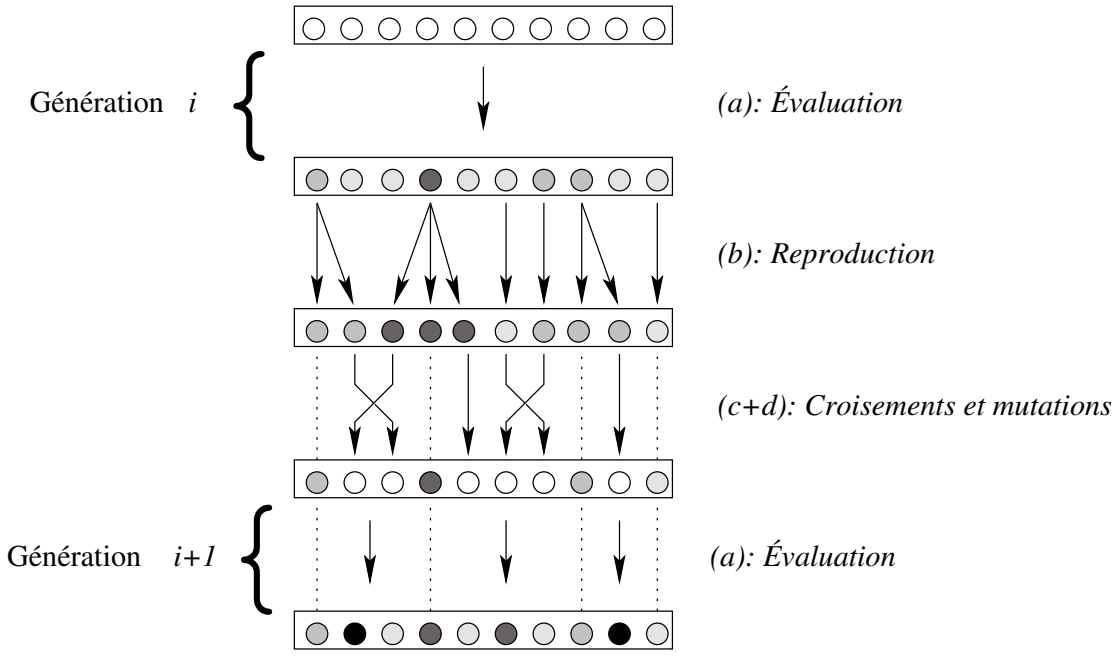

FIG. 5.1 – Une itération d'un algorithme génétique

- **Le nombre maximal de générations** effectué par l'algorithme, qui se définit le plus souvent en fonction de la taille de la population et du temps imparti pour résoudre le problème. Plus il est court et plus on risque d'être loin de la solution optimale.
- **Les taux de croisement et de mutation** qui fixent la proportion de la population qui sera, à chaque génération, influencée par la diversification. Ces taux ont généralement une influence importante sur la qualité de convergence de l'algorithme génétique mais leur valeur optimale dépend de la nature des opérateurs employés et du problème à résoudre.

Un certain nombre de paramètres plus qualitatifs sont également nécessaire pour une bonne convergence de ces algorithmes :

- **La pression sélective**, c'est-à-dire la quantité d'individus conservés lors de la reproduction, celle-ci ayant pour fonction de favoriser les individus les mieux adaptés. Sans cette pression sélective, l'algorithme ne peut converger car les individus ne sont pas forcés à converger vers un optimum. En revanche, si elle est trop élevée, la population risque de s'agglomérer trop rapidement autour des meilleurs individus et on risque alors de s'enfermer dans la recherche d'un optimum local.
- **Le niveau de déterminisme** des opérateurs de croisement et de mutation. Lorsqu'il est trop élevé, la population ne peut qu'évoluer cycliquement entre divers optima locaux de la fonction et n'a que très peu de chances d'explorer certaines régions de l'espace de recherche. Dans le cas inverse, la convergence peut être fortement ralentie, au point de rendre la méthode inefficace.

– **L'élitisme** de l'algorithme, qui consiste à forcer la conservation du ou des meilleurs individus trouvés. Cette stratégie assure la croissance de la fonction d'adaptation du meilleur individu de la population mais peut nuire à la diversité de la population.

Le réglage de ces paramètres est souvent fastidieux. Il nécessite beaucoup d'essais et dépend fortement du problème (du critère) à optimiser. Les résultats théoriques sur la convergence des algorithmes génétiques n'apportent que très peu d'éléments concluants sur la calibration des différents paramètres, qui reste de ce fait l'étape la plus difficile de leur mise en œuvre.

5.1.2 Fonction d'adaptation

Dans le cadre des algorithmes génétiques, aucune hypothèse de régularité n'est faite sur la fonction à maximiser. Elle peut ne pas être dérivable et peut même être fortement discontinue. Le calcul de la fonction d'adaptation peut ainsi être quelconque, que ce soit une simple équation ou le résultat d'un algorithme. Dans le cas de la résolution de conflits par exemple, le calcul nécessite la simulation de tout ou partie du trafic aérien sur un horizon de prédiction donné [Granger 02].

Une façon très simple de définir l'adaptation consiste donc à prendre le critère lui-même lorsqu'il doit être maximisé, ou son inverse pour un problème de minimisation.

Cependant, la qualité et surtout la rapidité de convergence de l'algorithme génétique sont influencées par l'allure de la fonction d'adaptation, notamment par le nombre de ses optima locaux. Pour cette raison, dans beaucoup de problèmes, la relation entre le critère et l'adaptation n'est pas toujours directe.

Dans le cas des problèmes avec contraintes, deux principales stratégies peuvent être utilisées :

– Lorsqu'il est possible d'assurer que tous les individus respectent les contraintes, l'adaptation ne représente que le critère [Michalewicz 91]. Il parait alors naturel de faire en sorte que la génération de la population initiale et que l'application des opérateurs d'évolution ne fournissent que des individus correspondant à une région admissible de l'espace de recherche. Cette option est parfois la plus efficace (car elle réduit directement l'espace de recherche) mais ne concerne que les problèmes avec des contraintes relativement simples (linéaires par exemple) car il faut être capable de conserver ces contraintes à travers le croisement et la mutation.
– Dans les autres cas, la fonction d'adaptation prend en compte les violations de contraintes en pénalisant l'évaluation des individus ne respectant pas tout ou partie des contraintes. Une solution classique consiste à plafonner la valeur de l'adaptation des individus en fonction du nombre de contraintes qu'ils violent (méthode dite par palier). Cette méthode a l'avantage de conserver, dans les premières générations tout au moins, des individus situés dans le domaine non admissible. En effet, ces derniers peuvent malgré tout générer des individus de bonne qualité. Certaines techniques [Joines 94] utilisent un palier évolutif, fonction par exemple de l'adaptation du meilleur élément et du nombre de générations effectuées pour conserver plus longtemps des individus hors du domaine admissible.

Lorsque les irrégularités de la fonction d'adaptation peuvent être lissées sans changer ses optima, la convergence de l'algorithme peut être accélérée. Cette situation peut notamment survenir dans les problèmes fortement contraints pour lesquels certaines contraintes sont plus restrictives que d'autres. La fonction d'adaptation peut pénaliser d'avantage leurs violations, ce qui permet d'orienter progressivement la population vers le domaine admissible.

5.1.3 Population

Codage des données

Le codage de chaque individu est une phase essentielle de l'élaboration d'un algorithme génétique et peut influencer fortement son utilisation. En effet cette phase détermine la structure de données qui sera utilisée pour coder le génotype des individus de la population mais va également influencer la définition des opérateurs d'évolution qui vont lui être appliqués. Le codage doit donc être adapté au problème à résoudre, c'est-à-dire à l'évaluation des individus, à la séparabilité des problèmes, etc.

Dans l'algorithme initial de Holland, chaque individu était codé par une séquence binaire de longueur fixe. Ce codage binaire reste aujourd'hui encore très populaire. Par la suite et pour d'autres problèmes plus structurés, le codage binaire s'est parfois révélé moins pratique ou moins efficace que des codages entiers ou réels Dans ce cas, le génotype se définit par une chaîne d'entiers ou de réels [Goldberg 89] ou encore plus généralement par une combinaison des deux. Cette forme de codage revêt plusieurs avantages :

- Les opérations de codage et de décodage sont souvent simples.
- Le génotype devient structuré et peut être décomposé en différentes parties identifiables, parfois appelées *gènes*. Cette structure peut être utilisée par des opérateurs de croisement et de mutation spécifiques afin d'augmenter les chances d'obtenir des descendants meilleurs que leurs parents, lors du renouvellement de la population.
- La gestion des contraintes du problème peut également être simplifiée lorsque celles-ci ne concernent qu'un sous-ensemble réduit de variables du problème.

Population initiale

La population initiale est celle qui va servir de population souche à l'algorithme génétique. Elle peut donc avoir une influence sur la vitesse de convergence de l'algorithme même si celle-ci n'est pas toujours maîtrisable. Dans tous les cas, l'établissement d'une « bonne » population initiale est fortement corrélé à la nature du problème à optimiser :

- Pour certains problèmes, leur nature ne permet pas de présager de la position de l'optimum dans l'espace de recherche. Ainsi un tirage uniforme dans ce dernier devrait fournir un ensemble d'individus suffisamment homogène et être réparti uniformément dans tout l'espace de solutions. Si on connaît a priori la structure de l'espace des solutions, on peut aussi scinder celui-ci en plusieurs composantes et choisir un individu dans chacun de ces sous-ensembles.

- Pour les problèmes fortement contraints dont le critère est relativement régulier, l'optimum recherché est souvent conditionné par les contraintes (il se trouve sur la frontière du domaine admissible) et l'optimum du problème sans contrainte est parfois connu. Dans ce cas, la population initiale peut être composée d'individus générés aléatoirement dans un voisinage fixé de l'optimum sans contrainte. Le non respect de ces contraintes forcera les individus à se diversifier lors des premières générations de l'algorithme grâce à l'exploration de la région où le critère est optimal.
- Lorsque des méthodes sont connues pour trouver différentes solutions non optimales mais respectant les contraintes du problème, ces solutions peuvent former autant de régions dans lesquelles seront générés les individus de la population initiale.

Dans tous les cas, le choix de la population initiale ne peux se faire arbitrairement et il faut donc procéder par *essais-erreurs*. En effet, il se peut que dans certains cas une population choisie en fonction du critère d'optimisation donne de meilleurs résultats et que dans d'autres cas, une population choisie aléatoirement soit plus efficace. Comme pour toute méta-heuristique, si les idées sont réutilisables et transposables d'un problème à un autre, rien ne garantit leur succès.

5.1.4 Génération de population

La création d'une nouvelle génération de population se fait en deux étapes :

- par **sélection**, ce qui va permettre une meilleure reproduction des bons individus par rapport aux individus moins bien adaptés ;
- par **renouvellement** (croisement et mutation), ce qui va permettre d'enrichir la population avec de nouveaux individus et de rendre l'algorithme génétique susceptible d'atteindre tous les points de l'espace d'états sans pour autant nécessairement les adresser tous lors du processus de résolution (propriété d'*ergodicité*), ceci quelle que que soit la population initiale.

Sélection

Le rôle de la sélection est d'assurer majoritairement la survie des meilleurs éléments au détriment des plus mauvais. Il est tout de même important de ne pas garder que les meilleurs éléments afin de conserver une diversité génétique suffisante. En effet même les individus les moins bien adaptés peuvent, par croisement ou mutation, permettre de générer une descendance intéressante du point de vue de la recherche d'un optimum. C'est la *pression sélective*. Plus celle-ci est grande et plus sont représentés les meilleurs individus, c'est-à-dire que les mauvais individus survivent moins bien d'une génération à une autre. Or lorsque la pression sélective n'est pas adaptée, l'algorithme génétique peut devenir inefficace :

- soit les individus « s'étalent » dans tout l'espace de recherche sans jamais progresser (pression sélective trop faible) et l'algorithme ne converge pas ou très peu ;
- soit quelques individus, bien mieux adaptés que tous les autres, sont systématiquement sélectionnés (pression sélective trop forte) et la population tend à s'homogénéiser. L'algorithme converge alors trop rapidement vers un optimum local.

Dans le cas d'une pression sélective trop faible, la solution consiste généralement à affiner les opérateurs de croisement et de mutation, pour les rendre plus déterministes par exemple. En revanche le cas inverse impose l'utilisation de diverses techniques (le *scaling* présenté section 5.2.1 et le *sharing* section 5.2.2) permettant de faire varier artificiellement la pression sélective.

Il existe de nombreuses techniques afin de réaliser cette sélection parmi lesquelles trois sont les plus utilisées.

Sélection par tournoi Deux individus sont choisis au hasard, selon une loi de probabilité uniforme et *combattent*, c'est-à-dire que l'on compare leur adaptabilité. Le plus adapté l'emporte avec une probabilité $P \in]\frac{1}{2}; 1]$, où P est un paramètre qui permet de régler la pression sélective. Il est tout à fait envisageable qu'un même individu participe à plusieurs tournois, il aura donc la possibilité de se dupliquer plusieurs fois.

Sélection par roulette Cette technique reproduit le principe du tirage aléatoire utilisé dans les roulettes de casino avec une structure linéaire. À chaque individu i est associée une probabilité d'être choisi proportionnelle à son adaptation. Supposons la fonction d'adaptation à valeur dans $\mathbb{R}^+$. On désigne par $\sum f$ la somme des adaptations de la population.

$$\text{S}i\ \sum f = 0\ a lors\ l_i = \frac{1}{N_{pop}},\ sinon\ l_i = \frac{f_i}{\sum f}$$

La nouvelle population est obtenue par N_{pop} tirages aléatoires (avec une distribution uniforme de probabilité) de réels entre 0 et 1, chaque réel tiré désignant un individu sélectionné. Avec cette méthode, les individus les mieux évalués sont statistiquement sélectionnés plus souvent.

Reste stochastique sans remplacement Pour chaque individu on détermine un nombre de clones présents dans la future population. Pour un individu i, ce nombre est donné par $E(\tau_i)$ où E désigne la partie entière et τ_i l'adaptation de i rapportée à la moyenne des adaptations de tous les individus. Le reste de la population est ensuite complété par la méthode de sélection par roulette où l'évaluation de chaque individu est donnée par le *reste stochastique* $r_i = \tau_i - E(\tau_i)$.

Parmi ces trois techniques, c'est la troisième qui est retenue. La sélection par tournoi n'est pas suffisamment efficace car elle induit bien souvent une pression sélective trop faible (même avec $P = 1$). D'un autre côté, la sélection par roulette possède un biais plus ou moins important en fonction de la taille de la population. En effet, lorsque la taille de la population est réduite, le nombre de tirages est faible et l'espérance mathématique de sélection peut ne pas être obtenue. La technique de reste stochastique tente alors de minimiser ce biais induit.

Mutation

La mutation d'un individu consiste à modifier localement son patrimoine génétique afin de générer un nouvel individu et ceci, jamais de manière déterministe. Elle est censée être suffisante pour satisfaire la contrainte d'ergodicité. Certaines implémentations n'utilisent que celui-ci [Fogel 66].

En règle générale, on procède en tirant aléatoirement un ou plusieurs gènes dans le chromosome puis on les modifie de manière aléatoire (par exemple en rajoutant un bruit gaussien) en assurant tout de même que la résultante est une solution admissible au problème (voir figure 5.2). La mutation est souvent délicate à régler en terme d'évaluation du nouvel individu ainsi créé : insignifiante et on risque de s'enfermer dans la recherche d'un optimum local, trop importante et on risque d'empêcher toute convergence de l'algorithme.

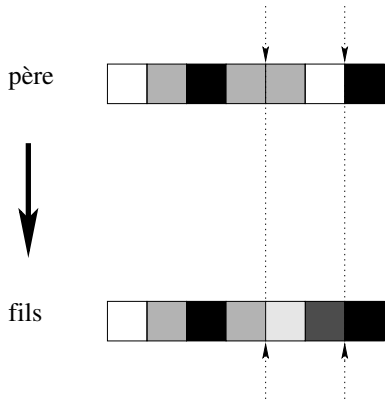

FIG. 5.2 – Un opérateur de mutation pour un algorithme génétique

Croisement

Le croisement permet la création de plusieurs individus à partir du patrimoine génétique de plusieurs parents. Quelque fois, de « bons » gènes d'un parent viennent remplacer les « mauvais » gènes d'un autre et créent des fils mieux adaptés que les parents. Contrairement à l'opérateur de mutation, le croisement peut être moins aléatoire puisqu'on peut choisir a priori quels sont les gènes qui seront croisés. Il lui est alors complémentaire et permet d'orienter les recherches.

En règle générale on crée deux enfants à partir de deux parents p et q. Dans le cas des problèmes discrets, on peut par exemple choisir aléatoirement deux gènes dans chacun des parents et échanger ensuite les deux sous-chaînes ainsi définies (figure 5.3). Dans le cas de problèmes continus, la règle la plus connue est le croisement arithmétique (de type BLX-alpha linéaire). On crée les fils p' et q' tels que :

$$\vec{p'} = \alpha * \vec{p} + (1 - \alpha) * \vec{q}$$
$$\vec{q'} = (1 - \alpha) * \vec{p} + \alpha * \vec{q}$$

où α est un coefficient de pondération aléatoire adapté au domaine de définition des gènes. On utilise ainsi souvent un intervalle de variation égal à $[-0.5; 1.5]$.

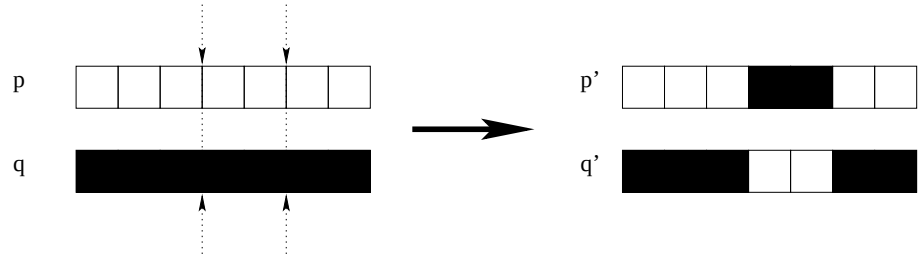

FIG. 5.3 – Un opérateur de croisement pour un algorithme génétique

De même que pour l'opérateur de mutation, il existe une grande quantité d'opérateurs de croisement dans la littérature, chacun correspondant à un type de problème. Malgré ce grand nombre, il faut toujours adapter ces derniers en fonction du problème à optimiser et à la structure utilisée pour le codage des individus. De même que pour l'établissement des paramètres de l'algorithme génétique, il est très difficile de dire quels opérateurs vont être efficaces sans tester a priori.

5.2 Améliorations et adaptations classiques

5.2.1 Scaling

La figure 5.4 (partie de gauche) illustre un cas où les techniques de sélection peuvent poser problème. En effet, si la répartition de la population est relativement homogène, on s'aperçoit que l'individu q possède une valeur d'adaptation élevée alors que celles des autres individus oscillent autour d'une valeur moyenne beaucoup plus faible. On risque alors de favoriser la duplication de q et de ne jamais atteindre le véritable optimum qu'est p.

De même, lorsqu'aucun optimum ne se détache (partie droite de la figure 5.4), il serait probablement intéressant de « grossir » artificiellement la zone entourée afin d'y chercher un individu meilleur que les autres.

Pour pallier ces deux problèmes, la technique de *scaling* (ou mise à l'échelle) permet de diminuer ou d'augmenter artificiellement les écarts d'évaluation entre les individus afin d'ajuster la pression sélective des techniques de sélection.

Deux techniques sont utilisées couramment [Michalewicz 92] :

Le scaling linéaire défini par une fonction de mise à l'échelle affine :

$$f_s(i) = a * f(i) + b \qquad \text{avec } a > 0$$

– si $a < 1$ alors la pression sélective est affaiblie et la recherche de solutions est plus exhaustive :

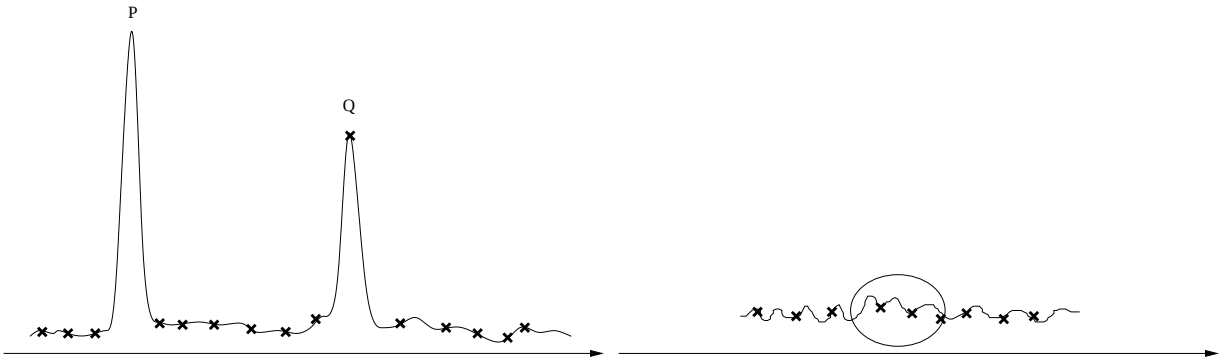

FIG. 5.4 – Exemples où la sélection risque de ne pas être efficace, dans un premier cas par mauvaise distribution de la population initiale, dans un second cas par l'absence de pic

– si $a > 1$ alors la pression sélective est renforcée et la recherche se concentre sur les individus les mieux adaptés.

La constante b n'est utilisée que pour permettre à f_s d'être à valeurs dans $\mathbb{R}+$ si la technique de sélection le requiert. Il est possible de faire varier a et b en fonction de l'indice de la génération.

Le scaling exponentiel défini par une fonction de mise à l'échelle :

$$f_s(i) = f^k(i)$$

– Pour k proche de 0, la pression sélective est fortement réduite. L'algorithme génétique se comporte alors comme une exploration aléatoire de l'espace de recherche.
– Pour k proche de 1, le scaling est inactif.
– Pour $k > 1$, la pression sélective est renforcée et seuls les meilleurs individus survivent.

De même que pour le scaling linéaire, k peut varier au fil des générations. En général, k varie de manière croissante en fonction du nombre de générations, avec par exemple (figure 5.5) :

$$k = \left(\tan\left(\frac{n}{N_{gen}+1} * \frac{\pi}{2}\right)\right)^p \text{ avec } \begin{cases} n & : \text{numéro de la génération courante} \\ N_{gen} & : \text{nombre maximal de générations} \\ p & : \text{intensité du scaling} \end{cases}$$

Dans la pratique [Granger 02, Gotteland 04, Gianazza 04], la valeur $p = 0.1$ s'est souvent montrée efficace.

5.2.2 Sharing

Le *sharing* (*partage* ou *répartition*) relativise l'adaptation des individus par rapport à leur proximité (ou densité) relative dans l'espace de recherche. L'objectif est de forcer la

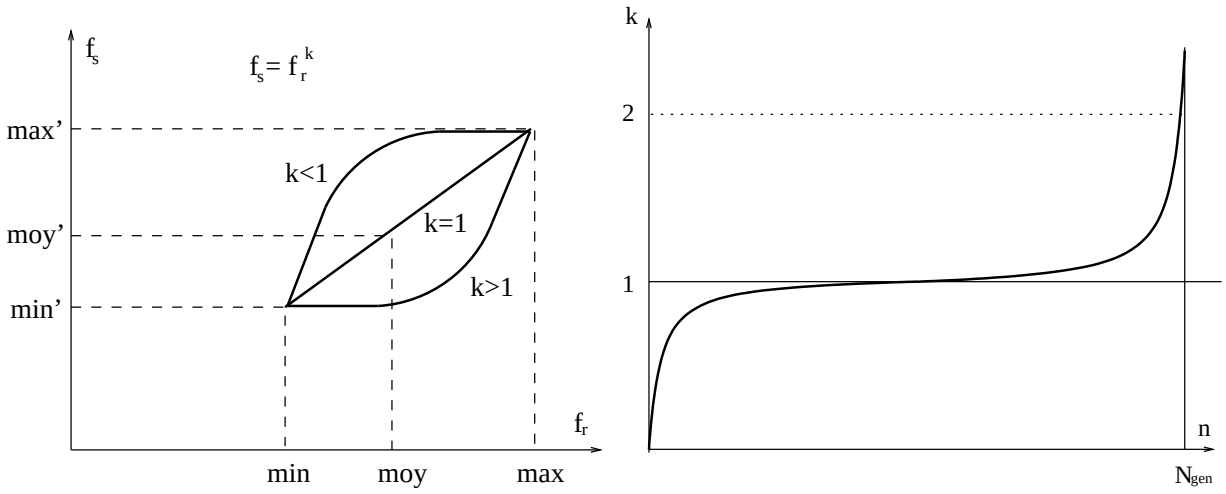

FIG. 5.5 – Fonctions de scaling exponentielle et évolution du paramètre k

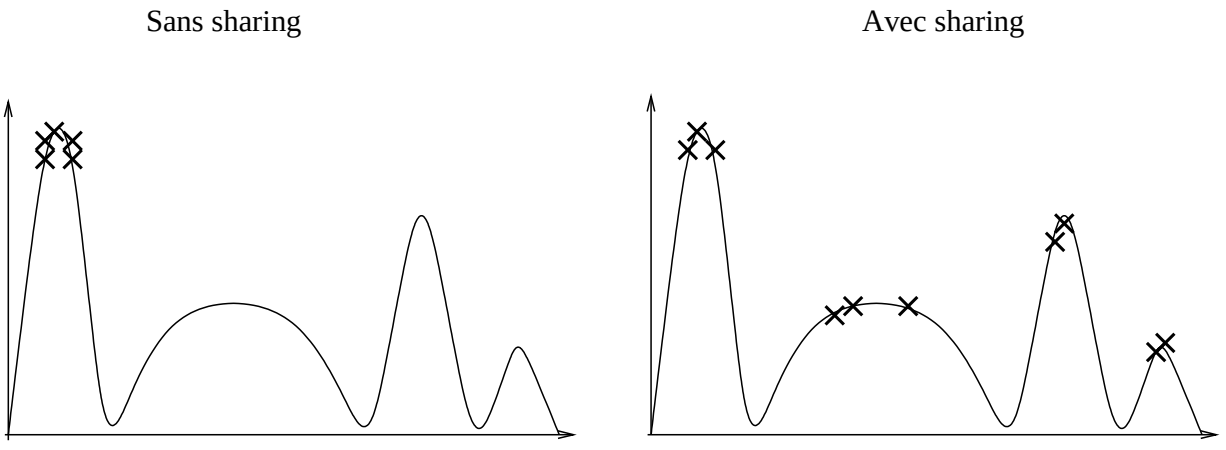

FIG. 5.6 – Objectif du sharing : répartir la population pour éviter sa concentration précoce autour d'un unique optimum

population à se répartir dans différentes régions de l'espace de recherche afin de satisfaire la contrainte d'ergodicité. Son utilisation est particulièrement recommandée avec des fonctions objectifs présentant de nombreux optima locaux afin d'éviter que la population ne se regroupe sur un seul d'entre eux (figure 5.6). Sa mise en œuvre nécessite la notion de distance sur l'espace de recherche ou entre les individus ce qui la rend souvent délicate à mettre en œuvre.

De même que le scaling, le sharing modifie l'adaptation des individus. Ce dernier pénalise un individu en fonction du taux d'agrégation de la population dans son voisinage. Pour cela, une distance d représentative des différences entre individus est nécessaire. La nouvelle fonction d'adaptation f_{sh} d'un individu i est donnée par :

$$f_{sh}(i) = \frac{f(i)}{\sum_{j=1}^{N_{pop}} S(d(x_i, x_j))}$$

$$\text{Avec : } \begin{cases} S(d) = 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} \text{ si } d < \sigma_{share} \\ S(d) = 0 \text{ si } d > \sigma_{share} \end{cases}$$

- Le paramètre σ_{share} définit la zone d'influence des individus : seuls les individus dont la distance est inférieure à σ_{share} se pénalisent mutuellement. Sa valeur doit être déterminée en fonction du problème traité et de la distance définie sur l'espace de recherche. Il est souvent utile, pour ce réglage, de normaliser les distances (entre 0 et 1 par exemple).
- Le paramètre α fixe l'intensité du sharing : plus α est grand, plus les groupes d'individus agglomérés sont pénalisés (figure 5.7).

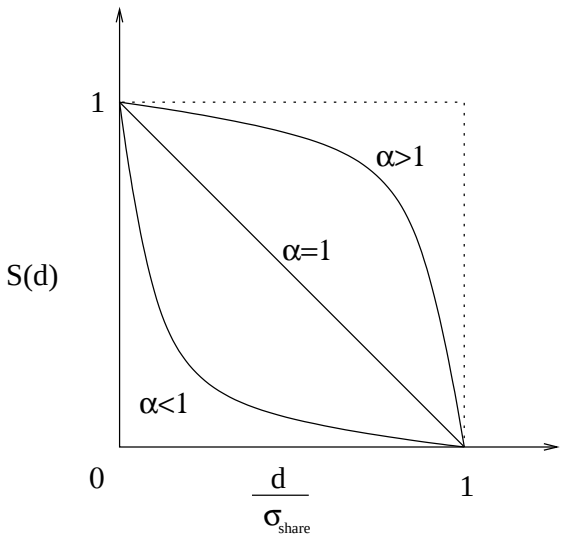

FIG. 5.7 – Allure de la fonction de sharing en fonction de son intensité

Dans la pratique, ce type de sharing donne de bons résultats mais sa complexité en $\mathcal{O}(N_{pop}^2)$ par rapport au nombre d'individus peut s'avérer pénalisante comparée aux autres

opérations de l'algorithme génétique qui s'effectuent généralement en $\mathcal{O}(N_{pop})$. C'est pourquoi le sharing *clusterisé* lui est souvent préféré.

Sharing clusterisé

Le *sharing clusterisé* [Yin 93] permet de réduire la complexité du sharing en répartissant les individus de la population par groupes de proximité appelés *clusters*. Deux paramètres $d_{\min} < d_{\max}$ définissent la manière avec laquelle sont construits ces clusters :

- Initialement, chaque individu de la population est considéré comme le centre d'un cluster dont il est l'unique élément.
- Si deux centres sont à une distance inférieures à $d_{\min}$, les deux clusters correspondant sont réunis dans un unique cluster dont le centre est le milieu des deux centres initiaux.
- Si la distance d'un individu au centre du cluster le plus proche est inférieure à $d_{\max}$, l'individu est ajouté au cluster et le centre de ce dernier devient le barycentre entre l'individu et l'ancien centre du cluster.

Cette méthode peut être implémentée en $\mathcal{O}(N_{pop} \log N_{pop})$ opérations [Yin 93]. Le sharing consiste alors à modifier l'adaptation de chaque individu i comme suit :

$$f_{sh}(i) = \frac{f(i)}{m'_i}; \qquad m'_i = n_i * \left(1 - \left(\frac{d_i}{2d_{\max}}\right)^{\alpha}\right)$$

$$\text{Avec :} \quad \begin{cases} n_i : \text{nombre d'individus dans le même cluster que } i \\ d_i : \text{distance entre } i \text{ et le centre de son cluster} \\ \alpha : \text{intensité du sharing} \end{cases}$$

Ce type de sharing s'avère aussi efficace que celui présenté précédemment. La méthode nécessite cependant le calcul de barycentres entre les points de l'espace de recherche, ce qui peut être difficile en fonction du codage utilisé pour les individus.

5.3 Convergence théorique

Les premières études théoriques sur les algorithmes génétiques se sont intéressées à leur fonctionnement avec un codage binaire. Ces résultats sont connus sous le nom de la *théorie des schémas*. Ils sont largement développés dans [Goldberg 89] et résumés dans [Alliot 92]. Ils permettent de mieux comprendre l'effet du processus de sélection et de certains opérateurs de mutation et de croisement sur les *schémas*[1] constituant le génotype des individus. Il apparaît par exemple qu'avec les opérateurs envisagés, les schémas de petite taille (définis sur un nombre restreint de bits) sont favorisés et ont plus de chance d'être reproduits au cours des générations.

[1] Un schéma est une suite du type $*110**01$ où $*$ peut être indifféremment 0 ou 1. L'ordre d'un schéma est son nombre de positions fixe (0 ou 1) et sa longueur fondamentale est la distance séparant la première position fixe de la dernière. Par exemple $*110**01$ est d'ordre 5 et de longueur fondamentale $8 - 2 = 6$

Des propriétés de convergence des algorithmes génétiques avec codage réel ont été développées par [Cerf 91]. Cette étude théorique permet notamment d'établir le comportement asymptotique des algorithmes génétiques ainsi que les cycles qui régissent la dynamique du processus. Elle démontre par ailleurs qu'un algorithme génétique peut converger sans utiliser d'opérateur de croisement.

Ces résultats théoriques ne rendent cependant pas compte du fonctionnement des algorithmes génétiques tels qu'ils sont utilisés en pratique. En effet pour obtenir une bonne vitesse de convergence il faut conjuguer une population initiale bien équilibrée avec de bons opérateurs de renouvellement, le tout couplé avec des paramètres bien étalonnés. Or changer la population initiale peut remettre en cause l'établissement des paramètres. Tout cela fait qu'il est bien souvent impossible de justifier l'emploi de telle ou telle méthode afin d'améliorer la convergence, sauf par la pratique.

5.4 Application à l'optimisation du réseau principal de routes

Dans cette section, un algorithme génétique est appliqué au problème d'optimisation du réseau principal de routes. La section 5.4.1 décrit les différentes implémentations de celui-ci et la section 5.4.2 rend compte de la mise en pratique et compare les différents choix effectués.

5.4.1 Diverses implémentations

Codage des données

La topologie du réseau de route est fixée par le concept *Sector-Less* (voir section 2.3 page 19). Hormis le rôle du contrôleur, deux caractéristiques sont nécessaires : la séparation des flux parallèles et le système de croisement des flux orthogonaux.

Lors de l'optimisation du réseau de routes, seule la notion de distance sol est intéressante. Le point de vue ATM ne vient qu'une fois une bonne solution trouvée. Il est donc possible lors de la phase d'optimisation de faire abstraction du fait qu'entre deux points de croisement plusieurs voies parallèles existent, que le réseau est en trois dimensions et qu'une méthode particulière est utilisée pour faire tourner les aéronefs. Seul l'emplacement géographique des points de croisement est donc nécessaire.

Ainsi un individu, représentant un réseau de routes, est défini par chacun des points de croisement qui composent le réseau, qu'on appellera chromosomes. Un individu possède donc 256 chromosomes. Chaque chromosome contient deux gènes qui sont les coordonnées de celui-ci dans le plan.

Population initiale

Deux types de population initiale ont été utilisés :

- Le premier et le plus simple est la répétition d'un même individu. La population est donc composée de n fois l'individu représentant le réseau initial de routes décrit section 3.2 (page 26).
- Le second type de population est composé d'individus représentant des réseaux de routes ayant la même géométrie que celui décrit section 3.2 mais ayant une position géographique au-dessus de l'Europe différente.

Parce que la population initiale doit être solution du problème, la génération aléatoire d'individus a été testée sans succès car il est pratiquement impossible de trouver des individus respectant la topologie mais également la contrainte d'angle et d'assurer ainsi que toute paire origine-destination possède au moins un chemin (donc un plus court chemin) dans chaque individu.

Enfin, pour les mêmes raisons, un dernier type de solution représentant des réseaux de routes ayant leurs points de croisement dans le voisinage géographique des flux principaux a été testée et n'a jamais donné de résultats satisfaisants.

Fonction d'adaptation

La fonction d'adaptation f de l'algorithme génétique est maximisée. Elle est donc définie comme l'inverse du critère d'optimisation présenté section 3.3.

Croisement

De même que pour la population initiale, le mélange aléatoire des chromosomes de deux parents afin d'obtenir deux enfants ne donne pas facilement des individus acceptables du point de vue de la topologie et des limitations imposées. Deux types de croisement ont tout de même été testés :

- Les enfants sont générés en remplaçant une séquence de gènes d'un père par celle de l'autre parent. Géographiquement cela consiste à découper chaque parents en deux. Le sens de découpe, horizontal ou vertical, est choisi aléatoirement de même que l'endroit de la découpe. Ensuite on échange une des deux parties avec celle provenant de l'autre parent. Il faut alors vérifier que les différents points sur la zone de coupure ne sont pas trop prêt et qu'aucun axe ne se croise. Ce type de croisement sera désigné comme étant le croisement *par découpage*.
- La seconde technique est celle présentée section 5.1.4. Les deux fils générés à partir de deux parents sont des barycentres par rapport à une constante α choisie aléatoirement dans $[-0.5; 1.5]$. Supposons que chaque chromosome soit numéroté et que cette numérotation est commune à tout les individus. Le barycentre de deux individus est défini comme le barycentre de la position dans le plan des sommets de même numéro dans chacun des deux graphes. Ce type de croisement sera désigné comme étant le croisement *arithmétique*.

Mutation

L'opérateur de mutation utilisé génère un fils en modifiant aléatoirement la valeur d'un chromosome. Géographiquement, cela revient à déplacer un point de croisement choisi au hasard dans une direction aléatoire. On s'assure alors que le point ainsi déplacé n'est ni trop proche d'un autre point, ni que ce déplacement n'implique l'intersection de deux routes en un point non prédéfini (voir section 3.4.2).

Sharing *clusterisé*

Le sharing est nécessaire pour éviter une homogénéisation prématurée de la population autour des premiers individus représentant de « bonnes » solutions. Le sharing *clusterisé* nécessite la définition de distance et de barycentre entre deux individus. Pour le calcul du barycentre, on reprend la même définition que pour l'opérateur de croisement. Pour le calcul de la distance, en supposant toujours les sommets numérotés, on définit la distance entre un individu A et un individu B comme suit :

$$D(A, B) = \sum_i d(A_i, B_i)$$

où d est la distance géométrique entre deux points.

Terminaison

Le critère d'arrêt de l'algorithme est défini par un nombre maximal d'itérations. Passé celui-ci, l'algorithme s'arrête quelle que soit l'adaptation des différents individus de la dernière population trouvée.

5.4.2 Comparaison des différentes méthodes

Le but ici n'est pas de générer la meilleure solution possible mais plutôt de comparer les vitesses de convergence ainsi que le nombre d'individus à générer pour obtenir une certaine valeur du critère, le tout en fonction des différents choix effectués pour les paramètres de l'algorithme génétique. De plus, il ne sera pas discuté ici de la qualité de la solution obtenue du point de vue ATM (voir chapitre 7 page 85).

Étant donné qu'un algorithme génétique fait appel à des fonction de tirage aléatoire, les résultats présentés ici sont obtenus en faisant la moyenne des résultats de dix exécutions de l'algorithme testé.

Taux de croisement et de mutation

En règle générale les premiers réglages s'effectuent sur les taux appliqués aux opérateurs de renouvellement car on considère que ceux-ci sont les moins à même d'être perturbés par les autres paramètres. Néanmoins ceux-ci dépendent de l'opérateur considéré. Il faut donc faire les tests pour les deux opérateurs de croisement définis précédemment.

Pour chacun des opérateurs de croisement (par découpage et arithmétique), deux séries de tests ont été réalisées. Les premiers tests (figures 5.8.a et 5.8.b) ont été effectués avec une population initiale homogène, c'est-à-dire avec le même individu dupliqué n fois, Les suivants (figures 5.8.c et 5.8.d) avec une population initiale hétérogène.

Quatre taux de renouvellement (somme du taux de croisement et du taux de mutation) ont été testés. Les figures suivantes présentent la valeur du critère défini section 3.3 (page 28) du meilleur individu trouvé après 100 générations (moyenne sur dix exécutions). Il s'agit donc de l'inverse de l'adaptation du meilleur individu. L'algorithme est appliqué avec une population de taille 100, sans scaling ni sharing.

La première observation vient confirmer le fait que chacun des paramètres de l'algorithme influe sur les autres. La rapidité de convergence est différente, non seulement en fonction des taux de croisement et de mutation utilisés mais également en fonction du type de population initiale.

Malgré ces fluctuations, on peut tout de même dégager une tendance. Plus le taux de renouvellement est important, plus la décroissance du critère d'optimisation est rapide. Quel que soit le type de population initiale et d'opérateur de croisement, un couple de valeurs semblent donner les meilleurs résultats :

- Taux de croisement : 10%
- Taux de mutation : 80%

En revanche il est difficile d'affirmer que l'opérateur de croisement arithmétique est plus efficace que celui par découpage. En effet, sachant qu'il est plus facile d'allonger un flux que de le raccourcir et qu'ainsi, en moyenne, un grand nombre de modifications dégrade plutôt que n'améliore globalement le critère, l'opérateur de croisement par découpage, en modifiant aléatoirement un grand nombre de flux, à tendance à avoir un effet négatif sur le critère. En revanche, on pouvait espérer que le croisement arithmétique allait permettre, dans certains cas, d'accentuer le déplacement de sommets dans une bonne direction ou d'atténuer ceux partant dans une mauvaise direction. Au vu des différents résultats, la différence n'est pas flagrante et il est difficile de conclure. On retrouve ainsi les réserves évoquées section 3.4.1 sur les difficultés d'écrire des opérateurs de croisement efficaces.

De plus, la meilleure valeur est obtenue avec un taux de croisement de seulement 10%, quel que soit l'opérateur utilisé. Lorsque ce taux augmente, l'adaptation du meilleur individu diminue. Ceci tendrait à montrer l'inefficacité de cet opérateur de croisement.

Choix et taille de la population initiale

A priori, la convergence d'un algorithme génétique est améliorée par augmentation de la taille de la population et du nombre de générations. Néanmoins il faut trouver un compromis entre ces deux paramètres afin de limiter le temps d'exécution. Si N_{pop} désigne le nombre d'individus et N_{gen} le nombre de générations, le temps de résolution s'exprime généralement en $\mathcal{O}(N_{pop}N_{gen})$ car l'opération la plus pénalisante est en règle générale (et c'est la cas ici) l'évaluation de chaque individu. Il est donc possible, pour un temps d'exécution fixé, de conserver la somme $N_{pop} + N_{gen}$ constante mais il faut trouver le réglage optimal par essais successifs.

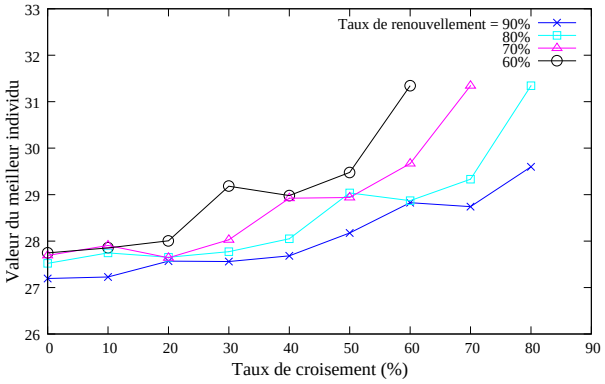

Fig 5.8.a - Croisement par découpage : influence du taux de croisement et de mutation avec une population initiale homogène de 100 individus. Évolution sur 100 générations

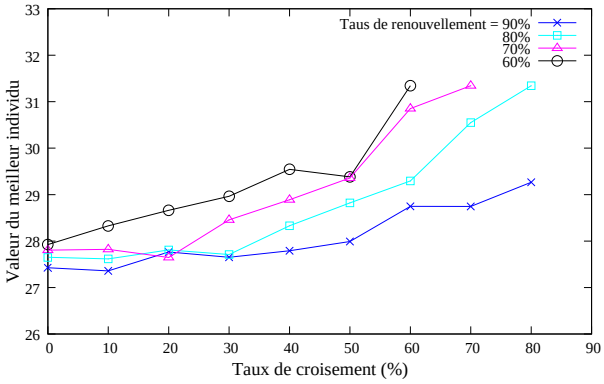

Fig 5.8.b - Croisement arithmétique : influence du taux de croisement et de mutation avec une population initiale homogène de 100 individus. Évolution sur 100 générations

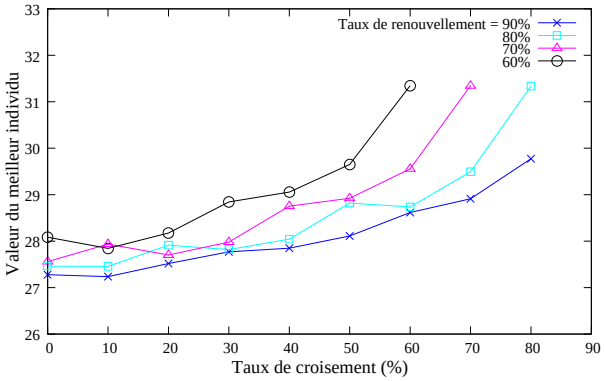

Fig 5.8.c - Croisement par découpage : influence du taux de croisement et de mutation avec une population initiale hétérogène de 100 individus. Évolution sur 100 générations

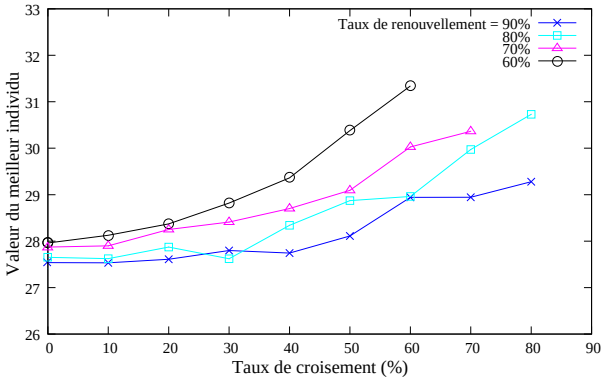

Fig 5.8.d - Croisement arithmétique : influence du taux de croisement et de mutation avec une population initiale hétérogène de 100 individus. Évolution sur 100 générations

FIG. 5.8 – Tests pour l'algorithme génétique des taux de mutation et de croisement

Dans cette optique, la vitesse de convergence est étudiée avec une population variant de 10 à 200 individus et avec un nombre de générations inversement proportionnel à la taille de la population, de 150 pour 200 individus par génération à 3000 pour 10 individus. L'évolution de la valeur de l'évaluation du meilleur individu est corrélée au nombre total d'individus explorés, donné par le produit du nombre de générations et du nombre d'individus. Les autres paramètres de l'algorithme sont fixés avec les valeurs optimales obtenues précédemment, c'est-à-dire 80-10, quel que soit le type de croisement utilisé.

Les figures 5.9.a et 5.9.b présentent les résultats des tests pour une population initiale où tout les individus sont identiques alors que les figures 5.9.c et 5.9.d présentent ceux calculés pour une population initiale hétérogène.

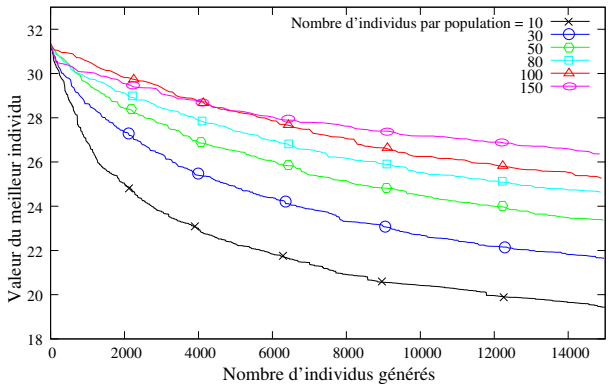

Fig 5.9.a - Croisement par découpage : influence de la taille de la population avec une population initiale homogène

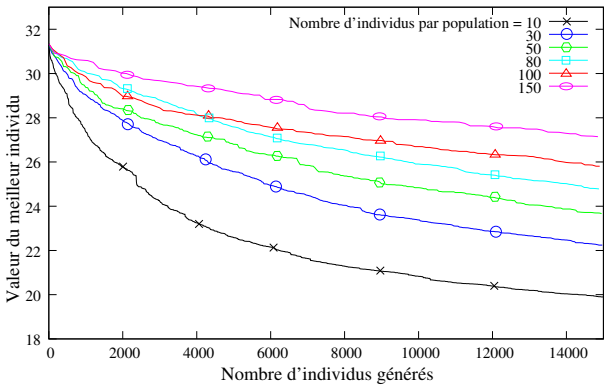

Fig 5.9.b - Croisement arithmétique : influence de la taille de la population avec une population initiale homogène

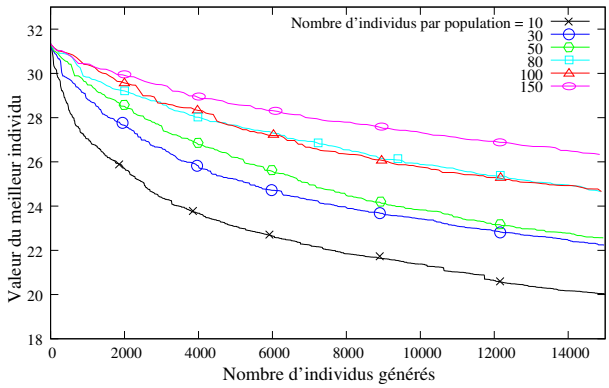

Fig 5.9.c - Croisement par découpage : influence de la taille de la population avec une population initiale hétérogène

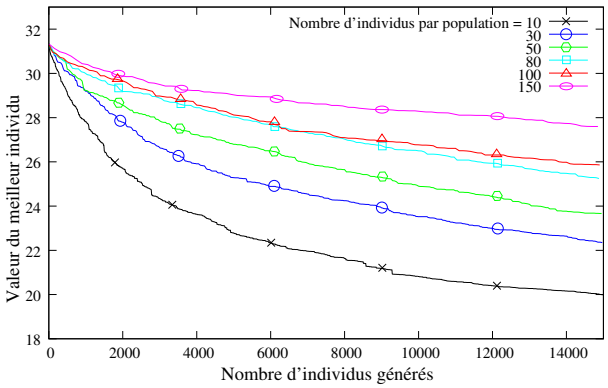

Fig 5.9.d - Croisement arithmétique : influence de la taille de la population avec une population initiale hétérogène

FIG. 5.9 – Tests pour l'algorithme génétique de différentes tailles de population

Contrairement à ce qui est généralement attendu, l'augmentation de la population tend ici à pénaliser la convergence du processus.

Quel que soit l'opérateur de croisement choisi et la population initiale, moins il y a d'individus, plus la convergence est rapide et, pour atteindre une même valeur du critère, moins il est nécessaire de générer des individus. Cela vient donc confirmer un peu plus les résultats précédents quant à l'utilité d'un algorithme génétique pour résoudre ce type de problème.

Efficacité du sharing

Différentes techniques et notamment le *sharing* permettent d'améliorer la distribution de la population dans l'espace de recherche. Sur une population peu nombreuse, cette technique ne devrait pas avoir d'effet notable mais on peut espérer que sur une population dense, cette technique accélère la convergence de l'algorithme génétique.

La figure 5.10.a donne la valeur moyenne de l'évolution du meilleur individu en fonction du nombre de générations tout d'abord avec différentes valeurs du taux de sharing défini section 5.2.2 puis sans sharing pour une population de 100 individus, un taux de mutation de 80% et un taux de croisement de 10% calculé sur dix exécutions. Les résultats sont peu probants et aucune amélioration réelle n'est visible.

Si on applique la même technique de sharing (figure 5.10.b) à une population de 10 individus avec les mêmes valeurs de taux de croisement et de mutation, celle-ci n'a que peu d'influence, ce qui est attendu au vu de la taille de la population initiale.

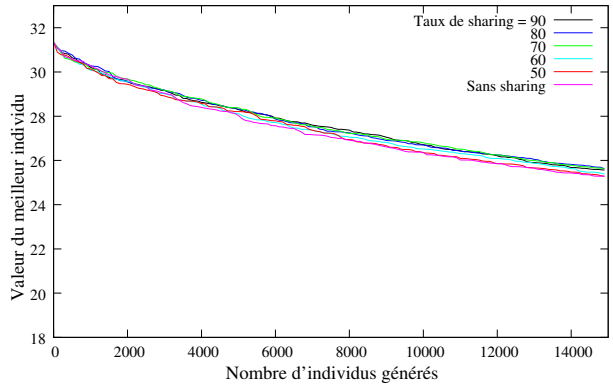

Fig 5.10.a - Influence du taux de sharing sur la vitesse de convergence pour une population initiale de 100 individus

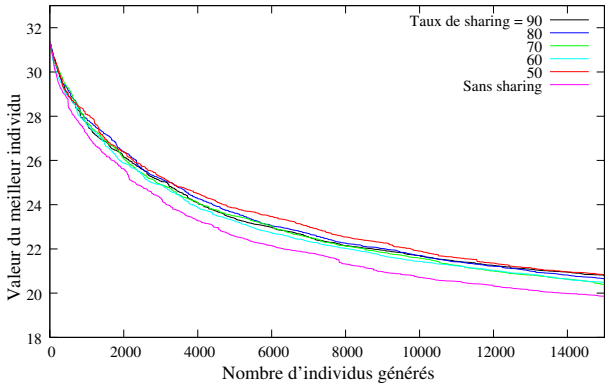

Fig 5.10.b - Influence du taux de sharing sur la vitesse de convergence pour une population initiale de 10 individus

FIG. 5.10 – Tests pour l'algorithme génétique de différents taux de sharing

Encore une fois et même en utilisant les techniques les plus abouties, il n'est pas possible sur le problème de l'optimisation du réseau principal de routes d'être réellement plus efficace que lorsqu'on utilise un algorithme sans « amélioration », c'est-à-dire, ici, sans sharing.

5.5 Conclusion

On rencontre ici un problème classique lorsqu'on tente de résoudre un problème avec une méta-heuristique : les résultats obtenus ne sont pas « mauvais » mais les techniques habituelles n'améliorent en rien de façon probante l'algorithme génétique utilisé ici. Se pose alors la question de savoir si l'utilisation de cette méta-heuristique est adaptée au problème traité. Néanmoins, l'évaluation de l'utilisation de cette méta-heuristique ne prend en compte que l'allongement des trajectoires, pas la qualité des solutions du point de vu gestion du trafic aérien. Pour connaître la réelle efficacité de l'algorithme génétique, cette étape va être déterminante.

Finalement, la remarque la plus importante, faite lors des différents tests, est l'absence de tendance forte, en particulier quant à l'évolution de la valeur du critère, ni en fonction de la population initiale ou encore, ni en fonction de l'opérateur de croisement. L'algorithme présenté ici n'est donc pas robuste.

Le principal résultat est alors que c'est l'opérateur de mutation qui permet à l'algorithme de converger vers une bonne solution. Ainsi, on peut penser que l'utilisation d'un recuit simulé donnera de meilleures solutions.

Chapitre 6

Résolution par recuit simulé

Le recuit simulé est une méta-heuristique inspirée d'un processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lent et de réchauffage (ou recuit) qui tendent à minimiser l'énergie du matériau en réorganisant les atomes qui le composent.

La description des phénomènes physiques et quantiques liés au processus de recuit s'appuie sur la statistique de Boltzmann utilisée en physique statistique pour déterminer la distribution de particules selon un ensemble de niveaux d'énergie. Elle est notamment à la base de la théorie cinétique des gaz.

Le recuit simulé s'appuie sur l'algorithme de Metropolis [Metropolis 53], qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système. Cet algorithme a été mis au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983 [Kirkpatrick 83] et indépendemment par V. Cerny en 1985 [Cerny 85].

6.1 Principes et généralités

Le principe général du recuit simulé est de parcourir de manière itérative l'espace des solutions. Partant d'une solution donnée (c'est l'état initial) on modifie celle-ci afin d'en obtenir une seconde, qui sera alors, soit acceptée et servira de base pour calculer la prochaine itération, soit refusée suivant certains critères que nous présenterons par la suite. Si cette nouvelle solution améliore le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système ; si celle-ci le dégrade, on a alors une augmentation de l'énergie du système (recuit). On effectue le même procédé de manière itérative.

Si l'on accepte uniquement les nouvelles solutions qui améliorent le critère, on tend alors à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une « mauvaise » solution permet ainsi d'explorer une plus grande partie de l'espace de recherche. Sans cela, on « plonge » vers un optimum local.

6.1.1 État initial

L'état initial peut être pris au hasard dans l'espace des solutions. À cette solution correspond une énergie initiale E_0 calculée à partir du critère que l'on souhaite optimiser. Une température initiale T_0 (généralement élevée) est également choisie. Ce choix est alors totalement arbitraire et va dépendre de la technique utilisée pour faire diminuer cette dernière.

6.1.2 Itérations

À chaque itération de l'algorithme, une modification élémentaire de la solution est effectuée. Cette modification entraîne une variation Δ_E de l'énergie du système. Si cette variation est négative (c'est-à-dire qu'elle fait baisser l'énergie du système), la solution engendrée par la modification devient la solution courante. Sinon, elle est acceptée avec une probabilité donnée.

Dans la grande majorité des cas, le choix de cette probabilité est généralement celle présentée comme étant de *Boltzmann* [Szu 87]. La probabilité d'acceptation p d'une solution est basée sur les chances d'obtenir un nouvel état ayant une *énergie* E_{k+1} généré à partir d'un état ayant une *énergie* E_k :

$$\begin{aligned} p(\Delta_E) &= \frac{e^{-\frac{E_{k+1}}{T}}}{e^{-\frac{E_{k+1}}{T}} + e^{-\frac{E_k}{T}}} \\ &= \frac{1}{1 + e^{-\frac{\Delta_E}{T}}} \\ &\simeq e^{-\frac{\Delta_E}{T}} \end{aligned}$$

6.1.3 Variation de température

Deux théories s'affrontent quant à la variation de la température :

- Pour la première, on itère en gardant la température constante. Lorsque le système a atteint un équilibre thermodynamique (au bout d'un certain nombre de changements et lorsqu'aucune modification ne fait plus évoluer le critère d'optimisation de manière notable), on diminue la température du système. On parle alors de *paliers* de température.
- La seconde théorie est basée sur la baisse de la température de manière continue. On peut alors imaginer toute sorte de lois de décroissance. La plus courante étant affine :

$$T_{i+1} = \alpha * T_i \; avec \; \alpha < 1$$

  La valeur fréquemment utilisée pour α est comprise entre $0,90$ et $0,99$.

Si la température atteint un seuil prédéfini ou que le système devient figé, l'algorithme s'arrête.

Quelle que soit la manière de faire décroître celle-ci, la température joue un rôle important. À haute température, le système doit être libre de se déplacer dans l'espace des solutions ($e^{-\frac{\Delta_E}{T}}$ proche de 1) en choisissant des solutions ne minimisant pas forcément l'énergie du système. À basse température, les modifications baissant l'énergie du système seront choisies majoritairement tout en conservant la possibilité d'en accepter de « mauvaises », empêchant ainsi l'algorithme de tomber dans un minimum local.

6.1.4 Description de l'algorithme de recuit simulé

Le pseudo-code de l'algorithme 6 correspond à une baisse continue de la température. On initialise avec une température T_0 donnée et une solution S_0 choisie dans l'espace des solutions du problème à optimiser. La fonction `energie` retourne *l'énergie* d'une solution c'est-à-dire applique le critère à optimiser (dans le cas d'une minimisation) à la solution courante. La fonction `modifie` retourne une solution différente calculée à partir de S et la fonction `reduire` applique une fonction décroissante à la température courante. On s'arrête lorsque l'on a atteint une température donnée T_{min}.

Algorithme 6 Recuit simulé

Entrées: T_0, S_0, T_{min}
Sorties: S

$S = S_0$
$T = T_0$
$E = \texttt{energie}(S_0)$
tantque $T > T_{min}$ **faire**
  $S' = \texttt{modifie}(S)$
  $E' = \texttt{energie}(S')$
  $\Delta_E = E' - E$
  si $\Delta_E \leq 0$ **alors**
    $S = S', E = E'$
  sinon
    $S = S', E = E'$ sous condition
  fin si
  $T = \texttt{reduire}(T)$
fin tantque
retourner S

Même s'il existe d'autres algorithmes de recuit simulé réputés plus rapides, notamment le *Very Fast Simulated Re-Annealing* [Ingber 89], les résultats présentés ici sont ceux des tests effectués sur l'algorithme classique de recuit simulé présenté précédemment, les tests utilisant le VFSR n'ayant pas montré de réelle amélioration.

6.1.5 Acceptation et rejet de solutions

La condition d'acceptation ou de rejet d'une nouvelle solution (ligne 11 dans l'algorithme 6) est fonction de la variation d'énergie Δ_E :

- si $\Delta_E < 0$, on améliore le critère d'optimisation et la solution est acceptée sans restriction aucune ;
- si $\Delta_E > 0$, la valeur du critère est dégradée. On tire alors un nombre p aléatoirement entre 0 et 1 en fonction d'une loi normale. Si $p \leq e^{-\frac{\Delta_E}{T}}$, on accepte la transformation.

6.2 Application au problème d'optimisation du réseau initial de routes

Dans cette section, un algorithme de recuit simulé est appliqué au problème d'optimisation du réseau de routes principales. Les premières parties (6.2.1, 6.2.2 et 6.2.3) rendent compte des différentes implémentations de celui-ci puis la dernière (6.2.4) compare les différents choix effectués.

L'utilisation de l'algorithme de recuit simulé lors de l'optimisation du réseau principal de routes correspond, dans un premier temps, au pseudo-code de l'algorithme 6, la solution initiale correspondant au réseau initial décrit section 3.2 page 26. *L'énergie* correspond alors à l'application du critère présenté section 3.3 page 28. Sa valeur initiale est alors de 31,4%.

La section 6.2.1 présente les différentes heuristiques utilisées pour la création d'une nouvelle solution (fonction `modifie` dans l'algorithme 6), la section 6.2.2 présente les différentes conditions envisagées pour l'acceptation ou le rejet de la nouvelle solution (lignes 8 et 11 dans 6) et enfin la section 6.2.3 présente les différentes lois de décroissance de la température (ligne 13).

6.2.1 Heuristiques pour la modification de la solution courante

De même que pour l'algorithme génétique, l'optimisation consiste à déformer le graphe correspondant au réseau initial de routes afin de réduire les distances à parcourir sans en changer la topologie, c'est-à-dire sans croisement de routes non prévu (sans intersection entre les arcs), sans ajout ou disparition de points de croisement (sans ajout ni disparition de sommet dans le graphe) ou encore sans chevauchement de points de croisement.

À chaque itération, afin de générer une nouvelle solution à partir de la solution courante, la première technique a été de reprendre celle utilisée pour l'opérateur de mutation de l'algorithme génétique (voir section 5.4.1 page 66), c'est-à-dire de ne déplacer géographiquement qu'un unique point de croisement par itération, c'est-à-dire de ne modifier qu'un unique sommet du graphe et donc la valeur du poids des arcs reliés à ce sommet. Une seconde heuristique permettant de déplacer plusieurs points de croisement a également été utilisée.

Le choix des sommets déplacés a également été fait selon différentes heuristiques, soit aléatoirement, soit en choisissant le point à déplacer de manière contrôlée. L'idée est alors

de choisir un point sur une des trajectoires les plus fréquentées afin de maximiser la probable réduction du critère de sélection. La répartition du nombre de vols étant fortement disproportionnée sur une journée, presque tous utilisant les mêmes axes, cette heuristique n'était pas bien adaptée à un réseau dont la prétention est de couvrir toute l'Europe. Une autre heuristique a donc consisté au choix du point de croisement à déplacer parmi les flux dont l'allongement est supérieur à la valeur courante du critère, c'est-à-dire à l'allongement moyen.

Une comparaison des différentes heuristiques est présentée section 6.2.4.

6.2.2 Acceptation et rejet de solutions

Lors de l'utilisation du recuit simulé en ne modifiant qu'un unique sommet à chaque itération (qu'il soit ou non choisi aléatoirement), l'acceptation ou le rejet de la nouvelle solution est effectué suivant la règle présentée section 6.1.5.

Lorsqu'on bouge plusieurs points, les déplacements sont considérés comme un seul et unique mouvement et on agit comme dans le cas précédent.

6.2.3 Itération et décroissance de la température

La nature même du problème, le nombre de variables qui le composent, rend le système très difficile à optimiser et encore plus à stabiliser. Au vu du nombre de flux considérés, tous les points de croisement sont parcourus par plusieurs de ces flux et le fait de bouger un point a une influence très complexe sur la valeur du critère. En effet, déplacer un point va raccourcir certains flux et dans le même temps augmenter la distance des autres flux passant par ce point. De plus, déplacer un point a toujours une influence sur le critère, d'où une grande difficulté à stabiliser le système.

Le déplacement d'un seul point ne fait jamais baisser le critère de façon dramatique (c'est-à-dire de plusieurs %). En général, un mouvement ne fait varier le critère que de quelque centième voir millième. Ainsi il est difficile d'établir une condition sur un quelconque « équilibre thermodynamique ».

Ne pouvant pas utiliser la méthode de variation de la température du système par palier, tout le problème réside alors dans la manière de faire décroître cette dernière. Trop vite et on risque de s'enfermer dans la recherche d'un optimum local, trop lentement et on ne se rapproche jamais suffisamment d'une solution satisfaisante.

6.2.4 Comparaison des différentes méthodes

De même que pour l'algorithme génétique, le but ici n'est pas de trouver la meilleure solution possible mais plutôt de comparer des vitesses de convergence en fonction des différentes heuristiques. De même, il ne sera pas discuté ici de la qualité de la solution obtenue du point de vue ATM (voir chapitre 7).

Tests sur la température

Plusieurs tests ont été réalisés en utilisant une décroissance géométrique de la température. La figure 6.1 présente la moyenne sur dix exécutions du nombre de dégradations de la valeur du critère qui ont été acceptées en fonction du nombre d'itérations (en pourcentage d'itération et en valeur) et la figure 6.2, l'évolution de la moyenne de la meilleure valeur du critère, toujours en fonction du nombre d'itérations. La variation de température est faite telle que $T_{i+1} = 0.99 * T_i$. D'autres valeurs du coefficient de décroissance, comprises entre $0,90$ et $0,99$ ont été testées mais les résultats, sensiblement identiques aux figures 6.1 et 6.2, ne sont pas présentés ici. Un test a également été réalisé avec une température nulle, c'est-à-dire qu'aucune dégradation n'a été acceptée.

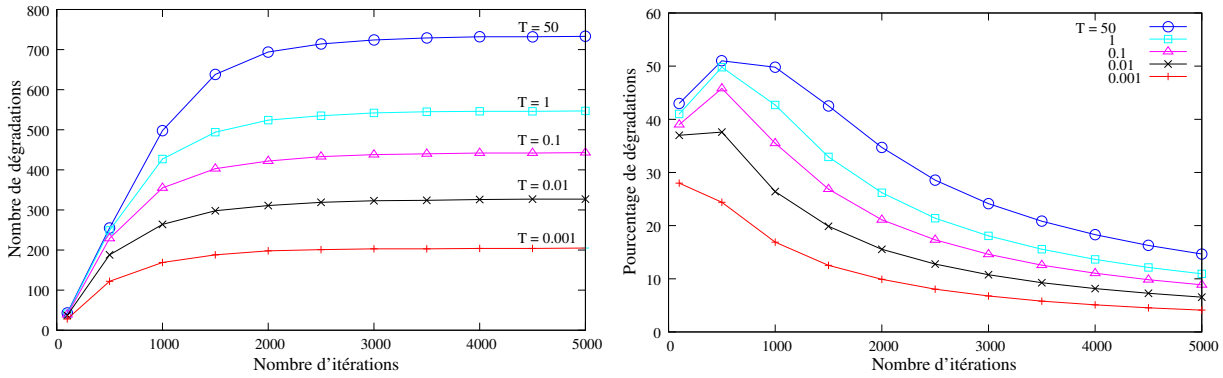

FIG. 6.1 – Tests sur la variation de la température : évolution du nombre de dégradations acceptées

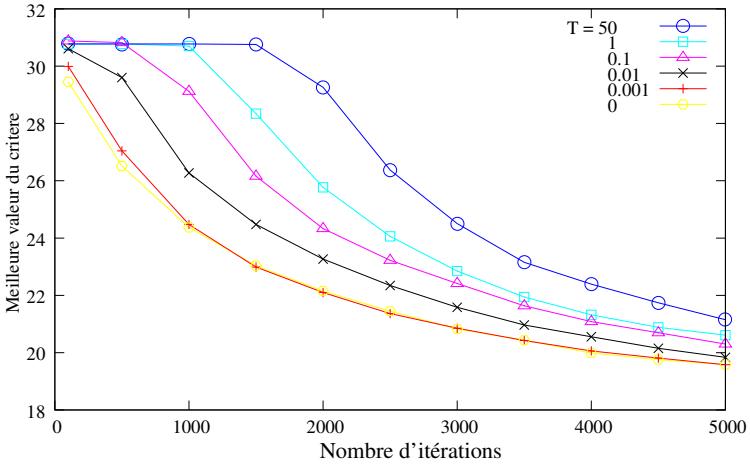

FIG. 6.2 – Tests sur la variation de la température : évolution de la meilleure valeur du critère en fonction du nombre d'itérations

Comme attendu, plus la température de départ est élevée plus le nombre de dégradations acceptées est important. De même lorsque la température est élevée, les premières étapes ont tendance à accepter beaucoup de dégradations ce qui « retarde » le début de l'amélioration de la valeur du critère (après plus de 1500 itérations pour une température de départ valant 50, après plus de 1000 itérations pour une température de 1). On remarque également que lorsqu'on n'accepte aucune dégradation (température initiale nulle), la vitesse de convergence est au moins aussi rapide qu'avec une température initiale non nulle.

Dans le cas de la décroissance continue de la température, on obtient bien le résultat attendu qui est qu'une température initiale trop élevée retarde l'optimisation du critère. En revanche le fait qu'une température trop basse risque de faire plonger l'algorithme vers un optimum local et peut ainsi empêcher le parcours intégral de l'espace des solutions n'a pas l'air d'avoir lieu ici.

Une solution adaptée à notre problème consiste alors à débuter la recherche sans accepter de dégradation pour faire baisser la valeur du critère puis effectuer un « véritable » recuit simulé une fois un certain seuil atteint.

Pour cela, un recuit « sans température ni probabilité d'acceptation » a finalement été utilisé. Il s'agit en fait de tenter d'améliorer le critère au maximum, donc d'atteindre un optimum local, puis d'accepter une ou plusieurs dégradations afin de parcourir un autre zone de l'espace des solutions. Ce recuit consiste donc à effectuer un palier mais sans condition particulière quant à la définition de « l'équilibre thermodynamique ». Dans ce cas, si le critère est amélioré, la solution est acceptée ; si le critère est dégradé, la solution est refusée systématiquement. Si au bout d'un certain nombre d'itérations, la solution courante n'est jamais améliorée, on considère que l'on a atteint un optimum local et une dégradation est alors acceptée. Ainsi on est certain de s'approcher « au plus près » d'un optimum local. Ce n'est qu'après cela que l'on cherche à explorer une autre partie de l'espace des solutions en acceptant une dégradation. C'est finalement cette solution qui fournit les meilleurs résultats présenté chapitre 7.

Différents mouvements de points

Le but des tests qui suivent est de comparer la vitesse de convergence en fonction des différentes heuristiques de choix de points. La figure 6.3 présente différents tests réalisés avec une température nulle, c'est-à-dire en n'acceptant que les solutions améliorant le critère d'optimisation. La courbe *Long flux* représente la meilleure valeur du critère en choisissant de déplacer un unique sommet du graphe appartenant à un des flux dont l'allongement est supérieur à l'allongement moyen (les tests sur le déplacement de plusieurs points appartenant aux « longs flux » donnent des résultats équivalents). Les autres courbes représentent les tests réalisés en déplaçant de deux à dix sommets du graphe par itération. Pour pouvoir comparer ces résultats, les meilleures valeurs du critère sont présentées en fonction du nombre de mouvements et non en fonction du nombre d'itérations comme précédemment. Ainsi, dans le cas de 10 mouvements par itération, 5000 mouvements ne correspondent qu'à 500 itérations.

On note alors que la meilleure solution est alors la plus simple, c'est-à-dire celle ne

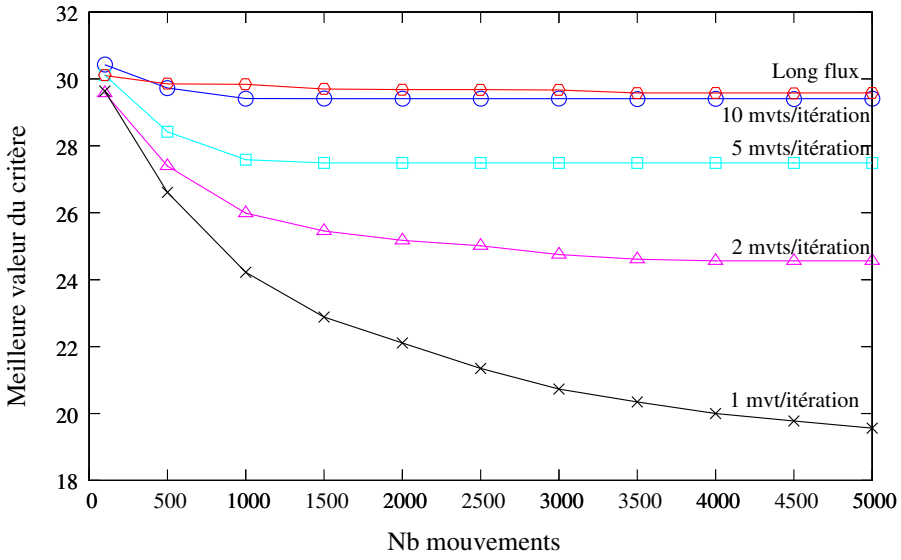

FIG. 6.3 – Tests sur les heuristiques de choix de points : évolution de la meilleure valeur du critère en fonction du nombre de déplacements de sommets réalisés

faisant qu'un seul déplacement par itération.

Dans le cas d'un choix « réfléchi » du point à déplacer parmi les flux dont l'allongement est supérieur à l'allongement moyen, le décroissance est continue mais peu rapide puisqu'en 5000 itérations, le critère est passé de 31,4% à 29,6% alors que dans le même temps, sans choix préalable, le critère atteint 19,6%. Une explication de ce phénomène est, qu'à cause des limitations décrites section 3.4.4 page 30, il est nécessaire de déplacer certains sommets n'étant jamais inclus dans les trajectoires les plus longues afin de permettre à ces derniers de se déplacer sans contrainte puisqu'il est interdit à un sommet de se retrouver dans le voisinage d'un autre sommet.

Le choix de déplacer plusieurs points par itérations pose un autre problème. Étant donné qu'en moyenne, le déplacement aléatoire d'un sommet tend plutôt à dégrader le critère d'optimisation qu'à l'améliorer, plus on déplace de points en même temps, plus on risque que la somme des déplacements unitaires dégrade le critère. Ainsi, plus on déplace de points, plus il est difficile de trouver une combinaison qui améliore le critère d'optimisation. La meilleure solution reste donc de ne bouger les points qu'un par un, itération après itération.

6.3 Conclusion

Nous venons de le voir encore une fois, la meilleure des solutions est finalement la plus simple, c'est-à-dire de ne déplacer qu'un unique sommet tout en respectant les limitations géographiques inhérentes au concept *Sector-Less*. Aucune autre heuristique tentée ne donne de meilleurs résultats.

Quant à la décroissance de la température, il parait plus efficace de ne pas utiliser un tel paramètre. De même qu'avec les algorithmes génétiques, les méthodes d'amélioration les plus abouties n'améliorent en rien les solutions trouvées à notre problème.

Chapitre 7

Évaluation des résultats

Une synthèse des travaux d'évaluation est présentée dans ce chapitre. Tout d'abord les résultats fournis par les différents algorithmes d'optimisation sont évalués du point de vue de la gestion du trafic aérien (section 7.2). La section 7.1 développe alors de quelles manières et grâce à quel outil les simulations de trafic ont été réalisées. Les sections 7.3 et 7.4 s'intéressent à l'évaluation des différents algorithmes, que ce soit de maintien de plus courts chemins ou d'optimisation, en fonction du temps de calcul et de la quantité de mémoire utilisée.

7.1 Simulation de trafic aérien

Cette section décrit la manière dont sont testés, du point de vue du trafic aérien, les réseaux de routes générés.

Une simulation de trafic réel est effectuée sur ces nouveaux réseaux en utilisant le moteur de simulateur de trafic CATS-OPAS [Alliot 97]. Ce simulateur est basé sur un temps discrétisé et évalue à chaque pas de temps [1] les positions et vitesses de tous les aéronefs. Il permet ainsi de faire voler des aéronefs en temps accéléré en routes directes ou sur n'importe quel réseau de routes. Il permet également de détecter les conflits entre aéronefs et de les résoudre en utilisant différents algorithmes [Granger 02].

7.1.1 Données d'une journée de trafic

Partant de plans de vol réels, c'est-à-dire des plans de vol déposés par les compagnies aériennes pendant une journée de trafic au dessus de l'Europe, de nouveaux plans de vol sont générés en utilisant les nouvelles balises (correspondant aux points de croisement) du réseau principal de routes utilisé. On limite les simulations sur une seule journée. Ces plans de vols contiennent :

- le type d'aéronef qui permettra de se référer à un modèle de performances ;
- l'aéroport d'origine et la balise d'entrée dans le réseau de routes principales ;

[1] Le pas de temps utilisé ici est de 15 secondes

- l'aéroport de destination et la balise de sortie du réseau de routes principales ;
- l'indicatif du vol ;
- l'heure de décollage demandée dans le plan de vol, qui pourra être modifiée de manière à ce que deux aéronefs ne décollent pas au même moment du même aéroport ;
- l'heure d'entrée, pour un aéronef arrivant de l'étranger, dans l'espace aérien européen (un aéronef venant en Europe depuis un pays extra-européen ou étant en transit au dessus de l'Europe ne sera pris en compte qu'à son entrée dans le réseau de routes ; le simulateur ne se charge pas de le faire voler avant son arrivée dans l'espace aérien européen mais calcule sa position d'entrée dans cet espace) ;
- le niveau de vol demandé ;
- le niveau de vol en entrée de l'espace aérien européen (un aéronef qui arrive de l'étranger n'est pas forcément au niveau qu'il a demandé) ;
- la vitesse de l'aéronef (ce sont généralement les compagnies qui fixent cette vitesse en fonction du « cost-index », un compromis entre le coût du carburant pour aller plus vite et le coût dû à l'augmentation du temps de trajet) ;
- la route prévue, c'est-à-dire la suite de balises que l'aéronef va utiliser.

Notons que CATS-OPAS peut simuler du trafic avec ou sans prérégulation (la prérégulation n'étant pas équivalente à l'action effectuée par la CFMU décrite chapitre 1 mais uniquement une régulation au sol, c'est-à-dire évitant que deux aéronefs ne décollent ou n'atterrissent en même temps). Pour l'application présentée dans cette thèse, c'est uniquement du trafic régulé au sol qui est observé. En effet, la régulation de la CFMU a été définie pour un espace découpé en secteur. Elle n'est donc pas compatible avec un concept de contrôle *Sector-Less*.

7.1.2 Trajectoire et détection de conflits

Pour pouvoir évaluer les nouveaux réseaux de routes, il faut pouvoir savoir à tout instant où se situent les aéronefs et quelles sont leurs trajectoires. L'évaluation va se faire en terme de charge de travail pour les contrôleurs, c'est-à-dire, ici, en nombre de conflits à résoudre. Le problème principal de cette méthode est qu'il faut prévoir une certaine incertitude quant aux positions des aéronefs. La détection de conflits doit donc tenir compte de cette incertitude.

Modèle de trajectoire

Le modèle de vol utilisé pour les aéronefs est un modèle tabulé. L'aéronef se déplace par pas de temps. Le modèle tabulé indique, en fonction du type de l'aéronef, de son altitude et de son évolution (en montée, en descente ou en croisière), la vitesse sol [2], la vitesse verticale et la consommation de l'aéronef.

Les modèles de performance aéronef proviennent de la base de données BADA (pour *Base of Aircraft DAta*) fournie par l'agence Eurocontrol. Dans cette table on peut trouver

[2]Le simulateur de trafic CATS-OPAS ne tient pas compte du vent.

67 modèles d'aéronefs différents. On considère alors, à l'aide d'une table de synonymes, que les autres appareils se comportent comme l'un de ces 67 modèles.

ACFT	FL	Croisière				Montée					Descente				
		kts	fuel(kg/min)			kts	ROCD(fpm)			kg/min	kts	fpm	kg/min	min	NM
	FL	TAS	min	nom	max	TAS	max	nom	min	fuel	TAS	ROCD	fuel	TIME	TOPD
A300	30	230	64.1	75.0	93.8	178	2060	2020	1640	213.4	230	1490	30.3	2.01	7.72
A300	40	233	64.3	75.2	94.1	212	2640	2490	2060	216.9	233	1510	29.6	2.68	10.29
A300	60	272	69.3	78.1	93.4	272	3470	2960	2350	221.0	240	1550	28.3	3.97	15.45
A300	80	280	69.8	78.7	94.2	280	3320	2810	2210	212.8	280	1760	27.0	5.10	20.75
A300	100	289	70.3	79.3	95.0	289	3160	2670	2070	204.6	289	1800	25.7	6.21	26.11
A300	120	297	70.9	80.0	95.9	356	3150	2690	2150	208.1	356	2320	24.4	7.08	31.22
A300	140	306	71.4	80.7	96.8	366	2950	2510	1980	199.9	366	2360	23.1	7.92	36.39
A300	160	377	87.5	94.5	106.7	377	2750	2320	1810	191.7	377	2400	21.8	8.76	41.63
A300	180	388	88.1	95.3	107.6	388	2540	2130	1630	183.7	388	2440	20.5	9.58	46.93
A300	200	400	88.7	96.0	108.6	400	2330	1940	1450	175.8	400	2480	19.2	10.38	52.30
A300	220	412	89.3	96.7	109.6	412	2120	1740	1270	168.0	412	2520	17.9	11.18	57.75
A300	240	425	90.0	97.5	110.6	425	1910	1540	1090	160.3	425	2550	16.6	11.96	63.31
A300	260	438	90.6	98.3	111.7	438	1690	1340	900	152.6	438	2590	15.3	12.73	68.95
A300	280	452	91.2	99.1	112.7	452	1480	1140	710	145.0	452	2620	14.0	13.50	74.70
A300	290	459	91.5	99.5	113.3	459	1370	1040	620	141.3	459	2640	13.4	13.87	77.59
A300	310	458	88.2	96.8	111.8	458	1690	1230	630	132.1	458	3590	12.1	14.43	81.85
A300	330	454	84.8	94.2	110.6	454	1450	990	380	122.8	454	3250	10.8	15.05	86.50
A300	350	450	82.1	92.4	108.2	450	1210	740	110	113.9	450	3180	9.5	15.68	91.22
A300	370	447	80.3	91.6	100.1	447	870	430	0	105.4	447	2910	8.2	16.36	96.34
A300	390	447	79.4	91.8	92.5	447	630	170	0	97.4	447	2920	6.9	17.05	101.44

TAB. 7.1 – Table de performances BADA pour l'airbus A300

Détaillons ici rapidement le fonctionnement du simulateur et des tables de performance dont un exemple se trouve dans la table 7.1. Supposons un airbus A300 volant au niveau 330 (33000 ft). Si l'aéronef est en vol stable, sa vitesse est de 454 kts et il consomme alors entre 84,8 et 110,6 kilos de fuel par minute, avec une consommation nominale de 94,2 kgs/min. S'il est en montée, sa vitesse est également de 454 kts, son taux de montée est compris entre 380 et 1450 pieds par minute avec un taux nominal de 990 ft/min et sa consommation est de 122,8 kg/min. Enfin, s'il est en descente, sa vitesse est toujours de 454 kts, son taux de descente de 3250 ft/min et sa consommation de fuel de 10,8 kg/min ; il lui reste alors 15,05 minutes avant d'atteindre le niveau 0. Enfin, l'information portée dans la dernière colonne indique qu'il doit commencer sa descente 86.5 Nm avant son aéroport de destination, s'il veut le rejoindre exactement en appliquant les taux nominaux tout au long de la descente.

Le simulateur utilise directement ces tables de performance avec un pas de discrétisation de 15 secondes. Ce pas a été choisi de manière à ce que deux aéronefs face à face ne puissent pas se croiser sans que le conflit généré ne soit détecté. Pour les aéronefs dont l'altitude n'est pas un niveau de vol entier, une interpolation linéaire entre les deux niveaux les plus proches est réalisée. Il existe également d'autres raffinements qui ne seront pas décrit ici, comme l'arrondi de la trajectoire de l'aéronef lorsqu'il atteint le niveau de vol auquel il doit se stabiliser.

On ne peut alors manquer de se poser des questions sur le réalisme de ce modèle. La réponse n'est pas simple et l'on peut même dire qu'il est extrêmement difficile de la donner aujourd'hui. Comme tout modèle tabulé, le modèle BADA occulte nombre des problèmes que pose la prévision de trajectoire. Cependant, il est le résultat de nombreuses années d'expérience. Il ne s'agit en effet pas d'un modèle basé sur des performances aéronefs ou de la dynamique du vol mais plutôt sur une « observation » de la réalité du comportement des aéronefs. On peut noter d'ailleurs certaines bizarreries si on compare ces tables aux résultats

d'équations de mécanique du vol. Ainsi, on observe un décrochage net entre les vitesses stables du niveau 140 et du niveau 160 qui n'a aucune explication en terme de mécanique du vol. On peut supposer que le modèle BADA est un modèle plausible, surtout pour les aéronefs en route et ce sont ceux-là qui nous intéressent ici plus particulièrement.

Incertitude sur les trajectoires et détection de conflit

À un instant t, si le simulateur permet facilement de connaître la position de tous les aéronefs, il est en revanche plus difficile de déterminer leurs positions futures. Or ce sont ces positions futures qui permettent de vérifier le respect des distances de séparation entre aéronefs. Dans le cas où ces distances minimales risquent d'être violées dans un futur plus ou moins proche, on parle de conflit *potentiel*.

La prévision de trajectoire doit prendre en compte des incertitudes sur les positions futures des aéronefs. On ne peut pas, dans la pratique, connaître la position future d'un aéronef pour plusieurs raisons : si la vitesse et la direction des vents sont connues, elles le sont généralement avec une certaine incertitude ; il est difficile de prévoir les taux de montée ou de descente que vont utiliser les aéronefs (le modèle tabulé donne, par exemple, pour la montée un encadrement du taux de montée utilisable par le pilote) ; les consignes compagnie en terme de vitesse ne sont pas connues ce qui rend difficile de prévoir les accélérations et décélération des aéronefs, etc.

Les pilotes sont, en général, capable de maintenir leur aéronef relativement stable à une altitude et sur une route en croisière. Néanmoins, les erreurs éventuelles sur ces deux données sont prises en compte par les normes de séparation. Les tenues de vitesse et de taux de montée ou de descente sont beaucoup plus aléatoires et ne peuvent pas être prises en compte par les normes de séparation, même pour une prévision courte de 5 minutes environ.

Pour ce faire, le simulateur ne représente pas les aéronefs par des points mais par des volumes ressemblant plus ou moins à des cylindres. À l'instant initial (voir figure 7.1), l'aéronef est représenté par un volume réduit à un point.

Dans le plan horizontal, au cours du temps et à cause de l'incertitude sur la vitesse de l'appareil, le volume se déforme dans la direction du vecteur vitesse de l'aéronef. L'aéronef est représenté par un segment de droite dont le « point avant » se déplace à la vitesse maximale possible de l'aéronef et le « point arrière » se déplace à la vitesse minimale possible. Dans la pratique, on définit un taux d'incertitude e_g (pour *ground speed error*) et le « point avant » se déplace à $v * (1 + e_g)$ où v est la vitesse nominale de l'aéronef, le « point arrière » se déplace à $v * (1 - e_g)$. Le segment s'allonge avec le temps. Lorsque le « point avant » du segment atteint une balise, il change de direction pour se diriger vers la balise suivante. L'aéronef n'est alors plus représenté par un segment mais par deux. Il peut ainsi être représenté par une liste de segments.

Dans le plan vertical c'est l'incertitude sur les taux de montée et de descente qui déforme le volume. Tous les aéronefs ont une altitude maximale et minimale. Ces altitudes sont modifiées en tenant compte du taux d'incertitude sur les taux de montée et de descente.

Ainsi il faut tenir compte de ces différentes incertitudes lorsqu'on évalue le nombre de

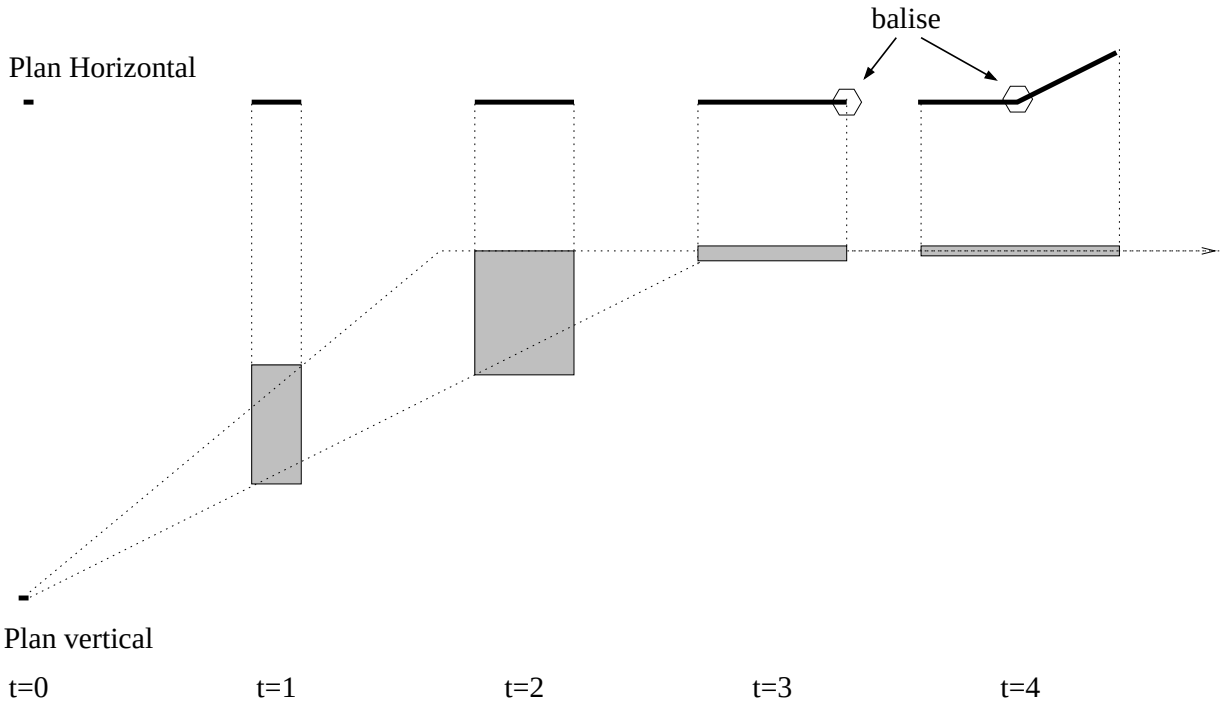

FIG. 7.1 – Modélisation de l'incertitude pour le calcul des positions futures des aéronefs

conflits potentiels pour un horizon temporel donné. Pour connaître plus en détail le cadre mathématique permettant l'évaluation du nombre de conflits potentiels, le lecteur pourra se référer à [Granger 02].

7.2 Évaluation du point de vue gestion du trafic aérien

7.2.1 Travail du contrôleur

Dans le concept *Sector-Less*, le rôle du contrôleur est de guider chaque aéronef dont il a la charge à travers le réseau principal de routes en assurant la sûreté du vol. Plusieurs flux, contrôlés potentiellement par différents contrôleurs, employant les même axes, la résolution des conflits doit alors se faire par concertation entre les différents contrôleurs impliqués. On peut décomposer le travail des contrôleurs en deux types de problèmes à résoudre :

- la résolution des conflits ;
- l'insertion d'un avion sur un axe lorsque celui-ci tourne au dessus d'un point de croisement (et donc change de niveau de vol).

Ceci n'étant pas sans rappeler le contrôle d'approche, la possibilité de mettre les avions en attente sur des hippodromes en moins (voir section 1.2.1 page 8).

Pour permettre une comparaison des résultats présentés par la suite, cette section rapporte ceux présentés par [Averty 98] dans une thèse sur l'activité des contrôleurs aériens

d'approche autour de l'aéroport de Lyon-Saint-Exupéry [3].

[Averty 98] a réalisé une étude portant sur vingt cinq contrôleurs dont l'âge se situe entre 37 et 54 ans. Les observations ont été réalisées pendant une semaine entre 18 et 21 heures, cet intervalle de temps contenant une des deux périodes de la journée (l'autre étant le matin) où le trafic à gérer est le plus conséquent. Lors de cette période, différentes analyses du comportement (mesure du stress, de la sudation, de l'anxiété, etc) ont été réalisées puis corrélées au trafic géré (analyse de l'image radar, des strips et des communications).

La charge de travail des contrôleur a été évaluée au moyen d'un indice global, appelé *indice de sollicitation* permettant de chiffrer non seulement la quantité de trafic (le nombre d'avions) géré, mais également sa « qualité » du point de vue des problèmes qui lui sont attachés, c'est-à-dire des types de problèmes à résoudre.

Il résulte de cette étude qu'un contrôleur d'approche gère en moyenne quatre aéronefs au même moment mais ce nombre peut aller jusqu'à une dizaine. Le nombre maximal de problèmes à résoudre peut aller jusqu'à six. Lorsque ces bornes maximales sont atteintes, [Averty 98] considère que la valeur de l'indice de sollicitation est telle que la charge est alors considérée comme excessive.

7.2.2 Une meilleure solution découverte

La figure 7.2 présente une parmi les meilleures solutions trouvées en terme de valeur du critère, les méta-heuristiques ne permettant pas de prouver l'optimalité d'une solution. L'allongement moyen des trajectoires est de 16,7%. Cette solution a été trouvée en utilisant le recuit simulé présenté section 6.2.4 page 79.

Les axes nord-sud sont représentés en rouge alors que les axes est-ouest paraissent en bleu.

La figure 7.3 montre un exemple d'utilisation de ce réseau principal de routes à travers la trajectoire d'un aéronef effectuant le trajet de Reykjavik en Iceland (BIKF) à Palma de Majorque en Espagne (LEPA).

7.2.3 Première évaluation

La première comparaison est faite par rapport au réseau de routes existant. Le critère d'optimisation est l'allongement moyen des trajectoires **pondérées** par le nombre d'aéronefs qui les utilisent. Si l'on s'affranchit de cette pondération, on obtient un allongement moyen de 17,99% ce qui est plus important que la valeur du critère sans pondération. Comparativement, l'allongement moyen d'une trajectoire sur le réseau utilisé à l'heure actuelle par rapport à la route directe se situe entre 7% et 11% [4]. Les nouvelles trajectoires

[3] À l'époque, l'aéroport se nommait Lyon-Satolas.

[4] On n'a pas ici de nombre unique pour l'allongement moyen car dans le système actuel, un aéronef peut parfois choisir entre plusieurs routes pour aller d'un point à un autre. Ainsi, la borne minimale est obtenue en considérant que tous les aéronefs utilisent les routes les plus courtes, la borne maximale en considérant qu'ils utilisent tous les routes les plus longues.

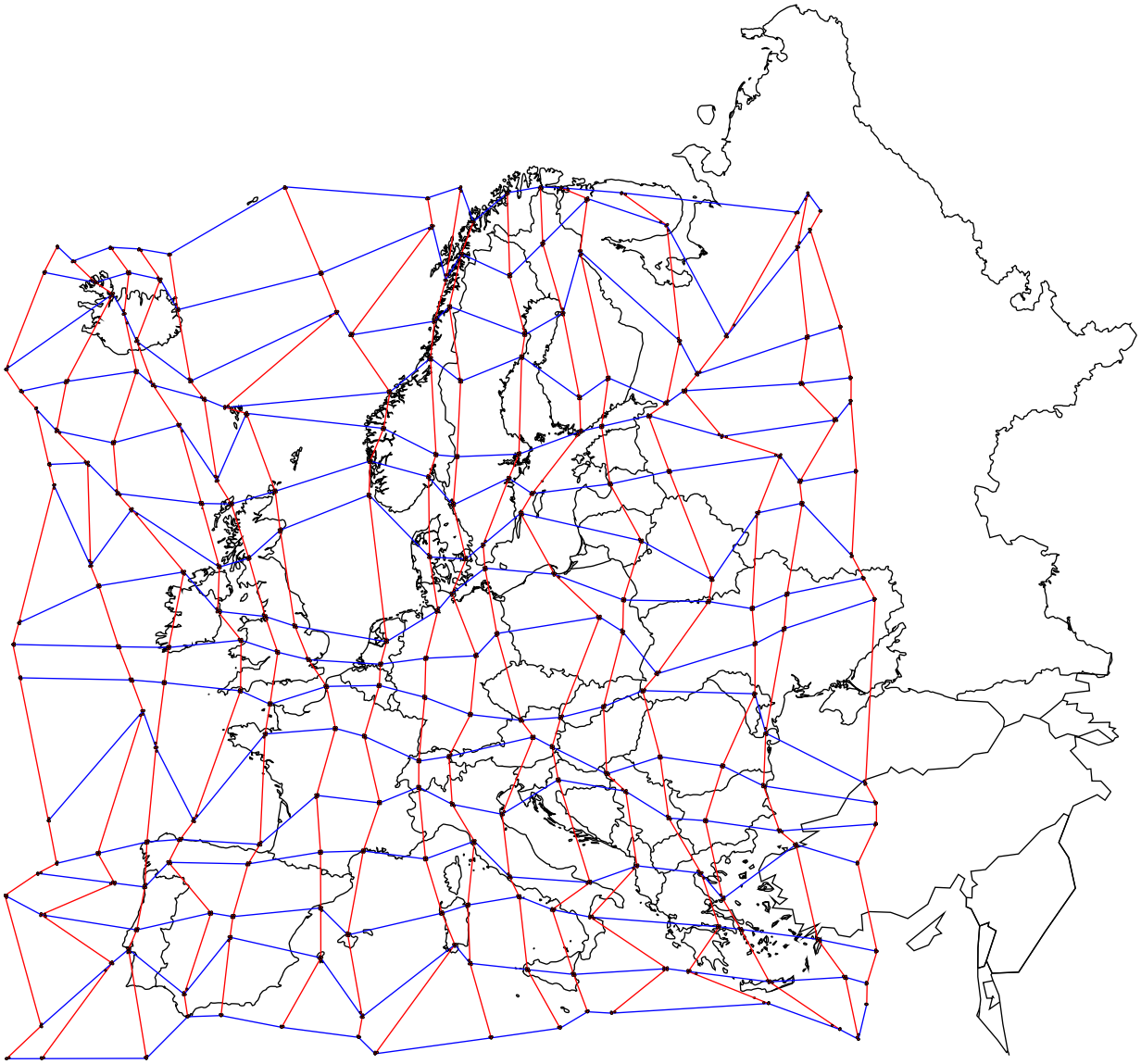

FIG. 7.2 – Meilleur réseau de routes découvert

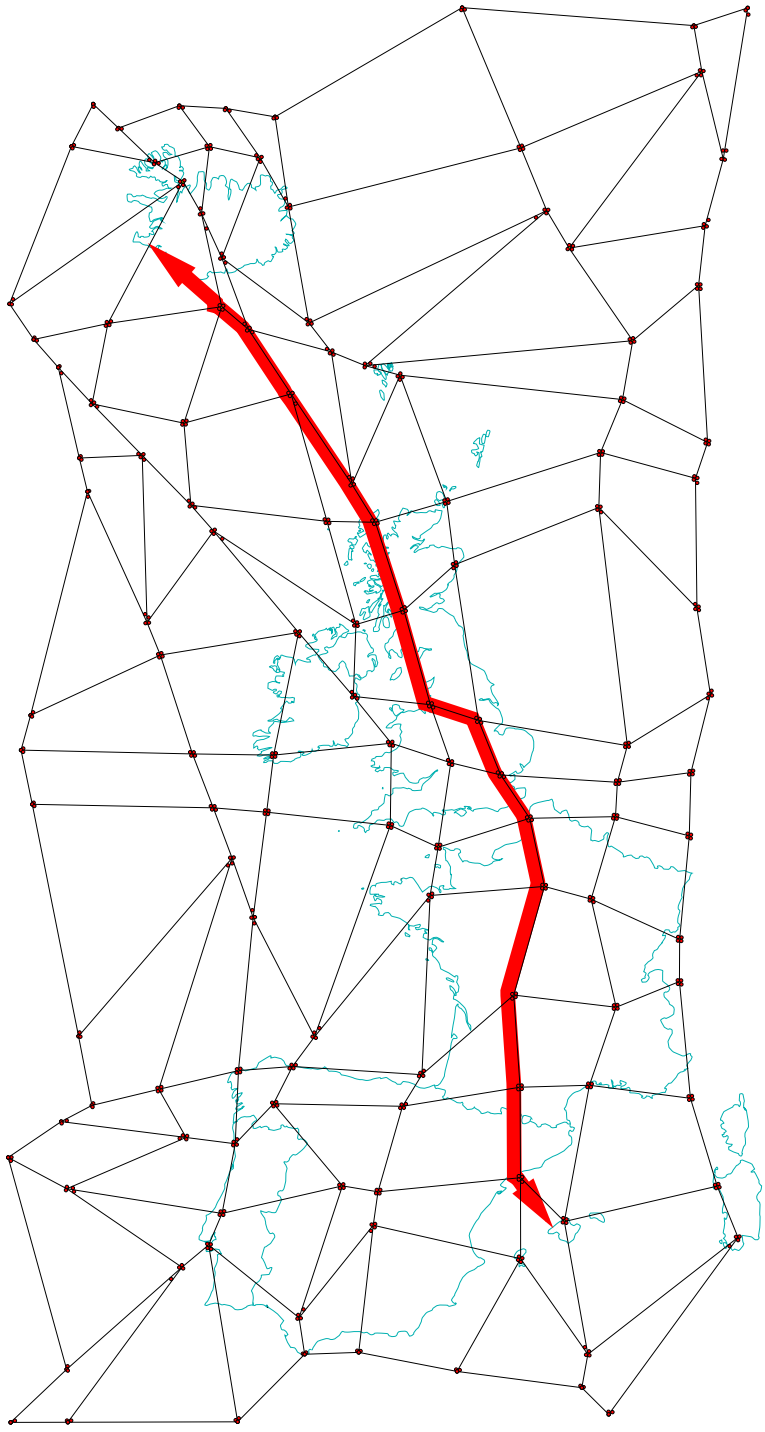

FIG. 7.3 – Un exemple de trajectoire utilisant le réseau principal de routes pour un aéronef allant de BIKF à LEPA

sont donc, en moyenne, plus longues, ce qui était attendu au vu de la simplification en terme de densité de routes imposée par le concept *Sector-Less*.

Si on regarde plus attentivement la répartition du nombre de vols en fonction de leur allongement par rapport à la route directe (figure 7.4), on s'aperçoit néanmoins que la grande majorité des allongements se situe sinon en dessous des 7%, du moins en dessous des 11% (ces bornes sont représentées en pointillé sur la figure 7.4). Environ 5% des vols sont allongés de plus de 50% et 2% de plus de 100%. Parmi ces derniers, certains ont un allongement de plus de 2000%. Ce résultat est dû au critère très strict de génération de trajectoire. En effet, tout aéronef doit emprunter le réseau de route, quelle que soit la distance à parcourir. Or lorsque deux aéroports sont proches, cela impose parfois aux aéronefs d'effectuer un large détour. De plus les Açores et les îles Canaries font partie de l'Europe de l'aviation civile. Ainsi les vols intérieurs à ces îles doivent donc passer par le réseau de routes qui, lui, ne couvre que l'Europe géographique. Ces vols sont donc contraint d'effectuer un passage au dessus du Portugal avant de revenir vers leur aéroport de destination. Il faut donc filtrer ces vols.

Ces vols, peu nombreux, sont alors considérés comme ne suivant pas le réseau principal de routes et devant faire l'objet a posteriori d'un traitement spécifique. En filtrant les vols dont l'allongement est supérieur à 100%, l'allongement moyen du réseau principal de routes est de 10,75%. En filtrant ceux dont l'allongement est supérieur à 50%, l'allongement moyen est de 9,38%, ce qui montre qu'il est possible de faire au moins aussi bien, en moyenne, que le réseau actuel à condition de lui faire subir un post traitement pour traiter les cas particuliers.

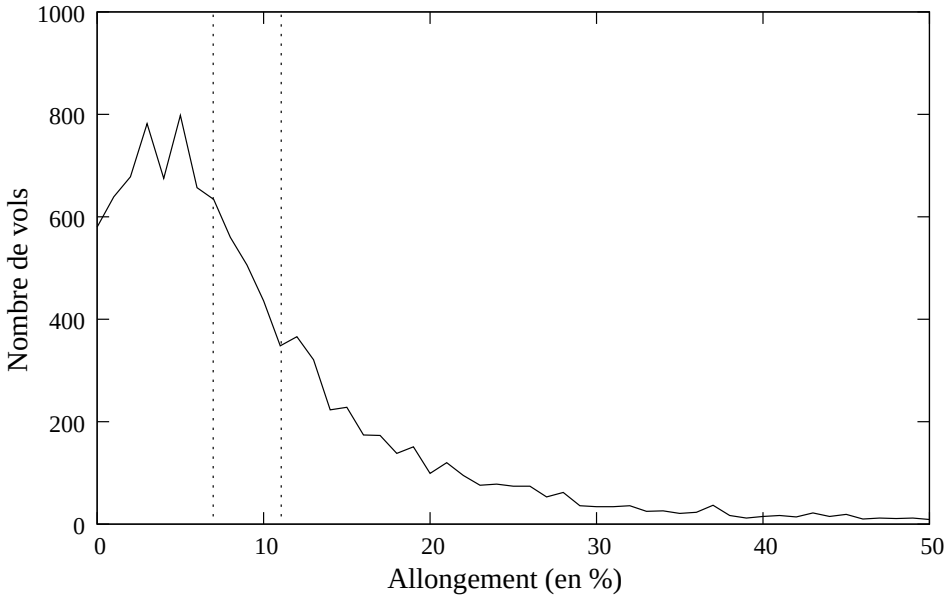

FIG. 7.4 – Répartition du nombre de vols en fonction de l'allongement de leur trajectoire par rapport à la route directe

La vérification de la cohérence du réseau de routes obtenu et donc du processus d'optimisation peut-être faite par une première évaluation « visuelle ». En effet on peut relever plusieurs points positifs quant au résultat présenté figure 7.2 :

- Les différents algorithmes d'optimisation déplacent majoritairement les points de croisement là où les aéronefs sont les plus présents. En effet on remarque qu'il n'y a, au dessus de la mer du nord ou de la mer méditerranée, aucun point de croisement, les aéronefs n'ayant pas besoin de tourner au dessus de ces zones.
- Les routes ont tendance à suivre les flux principaux (à cause du critère d'optimisation utilisé). On distingue nettement, par exemple, les axes Paris-Toulouse, Paris-Marseille ou encore Paris-Londres. Par ailleurs on remarque bien, au dessus de l'Europe continentale une réelle organisation des points de croisement en un réseau simple et structuré.
- Enfin, on assiste à un regroupement des points dans les « coins » du graphe, dû en partie au fait que les aéronefs arrivant ou partant vers « l'extérieur » suivent tous, plus ou moins, les mêmes routes et tendent à arriver par les coins. Ces regroupements multiplient alors les choix qu'ils ont pour la route à prendre.

En revanche, la structuration du réseau de routes en fonction des grands flux risque d'avoir un effet pervers. En effet, un plus court chemin étant la somme de plus courts chemins, le critère retenu mais surtout la faible densité du réseau de routes ainsi généré va avoir tendance à faire passer tous les aéronefs au même endroit. Par exemple, comme présenté figure 7.3, un aéronef allant de Reykjavik à Palma de Majorque va passer sur des segments de route déjà très fréquentés comme ici les axes Londres-Paris et Paris-Toulouse. Ceci risque donc d'augmenter le nombre de conflits dans ces zones. Malheureusement, avec un tel réseau, il n'est pas réellement possible d'offrir un chemin alternatif de longueur acceptable.

7.2.4 Évaluation en terme de charge de travail du contrôleur

Les simulations présentées dans cette section ont été réalisées en utilisant des données plans de vol européens datant de 2002. Arbitrairement c'est la journée du 21 juin qui a été retenue car c'est une des journées les plus importantes en terme de trafic aérien avec plus de 28000 vols pour une moyenne de 24000 vols par jour sur l'année 2002.

La charge de travail des contrôleurs est calculée ici en fonction du nombre de conflits potentiels à résoudre. Les conflits sont comptabilisés mais non résolus. On impose cependant trois restrictions :

- On ne compte que les conflits ayant lieu sur le réseau de routes principales puisque c'est celui-ci que l'on veut évaluer, les conflits autour des aéroports ne sont donc pas comptabilisés.
- Les conflits ayant lieu en dessous du niveau de vol 100 ne sont pas non plus comptabilisés. En effet on considère qu'en dessous de ce niveau de vol, les aéronefs sont gérés par le contrôle d'approche ou sont des vols non commerciaux.
- Plusieurs aéronefs en conflit plus d'une fois ne sont comptabilisés que si le temps entre deux conflits est supérieur à 30 secondes.

Afin d'éviter que deux aéronefs ne décollent ou n'atterrissent en même temps et au même endroit, le trafic est régulé en ordonnançant les départs et les arrivées sur chaque aéroport en imposant un espacement de 60 secondes entre deux décollages ou deux atterrissages (sauf pour Paris-Charles de Gaulle où le délai est de 30 secondes ainsi que Paris-Orly où le délai est de 45 secondes) [5].

Nombre total de conflits

La faisabilité du concept *Sector-Less* et du réseau de routes associé est tout d'abord évaluée en terme de nombre de conflits pendant une journée. Si cette valeur, d'un point de vue global, n'est pas assez significative pour permettre une évaluation précise de la charge de travail de chaque contrôleur, elle permet néanmoins d'évaluer la meilleure valeur quant à l'espacement entre deux routes parallèles de directions opposées présenté figure 2.1 page 21 (figures 7.5 et 7.6 pour les résultats). On distingue deux types de conflits :

- les conflits situés au-dessus du niveau de vol 300. Au-dessus de ce niveau on ne trouve que des aéronefs en croisière. La géométrie des conflits est donc généralement simple car elle se situe souvent dans le plan horizontal.
- les conflits « évolutifs » c'est-à-dire les conflits concernant des aéronefs dont un au moins est, soit en montée, soit en descente. Malgré l'absence d'image radar en trois dimensions, ces conflits sont souvent simple à résoudre car il suffit alors de stabiliser un des aéronefs en évolution sur un niveau donné. Néanmoins, c'est le type de manœuvres le plus coûteux en terme de consommation de carburant et de moins confortable pour les passagers.

La figure 7.5 présente le nombre total de conflits sur le réseau de routes en fonction de l'espacement entre les routes parallèles. La figure 7.6 présente le nombre de points de croisement où sont détectés des conflits, ceux où sont détectés plus de 50 conflits par jour puis ceux avec plus de 100 conflits par jour, en fonction de l'espacement entre les routes parallèles. Dans ce cas, les conflits sont uniquement ceux détectés à l'intérieur des points de croisement, c'est-à-dire lorsqu'un au moins des aéronefs en conflit se situe entre les quatre balises qui délimitent chaque point de croisement.

La norme de séparation horizontale étant de 5 NM (environ 9.3 km), il n'est pas surprenant que le nombre de conflits pour une distance de séparation inférieure à 10 km soit bien plus grand que pour toute autre valeur (figure 7.5). Ainsi on note une forte décroissance du nombre de conflits pour des valeurs situées entre 9 km et 10 km. La décroissance enregistrée pour des valeurs comprise entre 10 km et 11 km est due à l'incertitude quant à la position des aéronefs. Pour une distance comprise entre 11 km et 50 km, le nombre de conflits reste stable. Les valeurs supérieures à 50 km n'ont pas été testées car, à partir de cette taille, les points de croisement sont tellement larges qu'ils se recouvrent deux à deux, ce qui n'a plus aucun sens.

Si le nombre de croisements impactés lorsque la distance entre deux routes parallèles (qui est aussi celle de la largeur des points de croisement) est plus faible lorsque celle-ci

[5] Sur les grands aéroports (comme ceux de Paris) l'espacement entre aéronefs peut être réduit grâce à la multiplication des pistes. Au moment des simulations ces données faisaient défaut

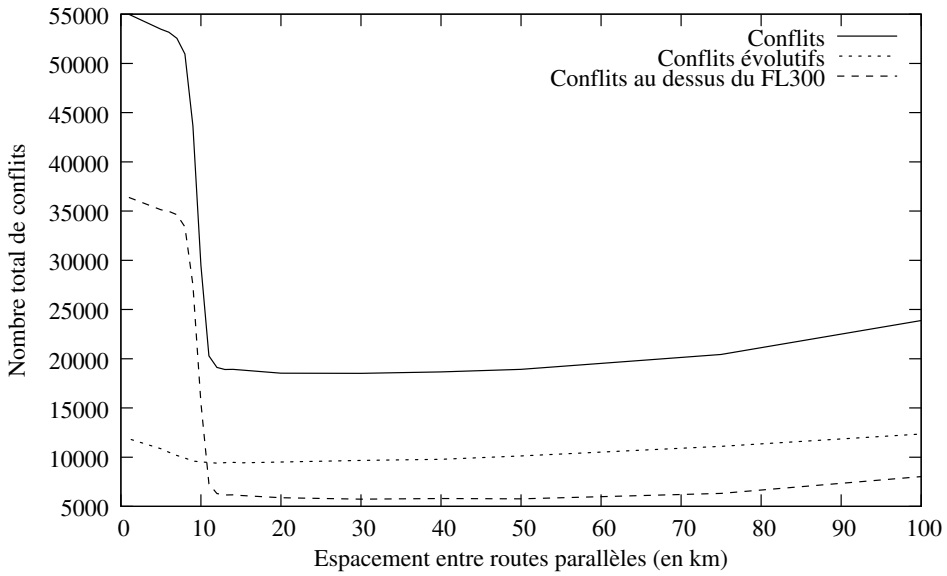

FIG. 7.5 – Évolution du nombre de conflits en fonction de l'espacement des routes parallèles

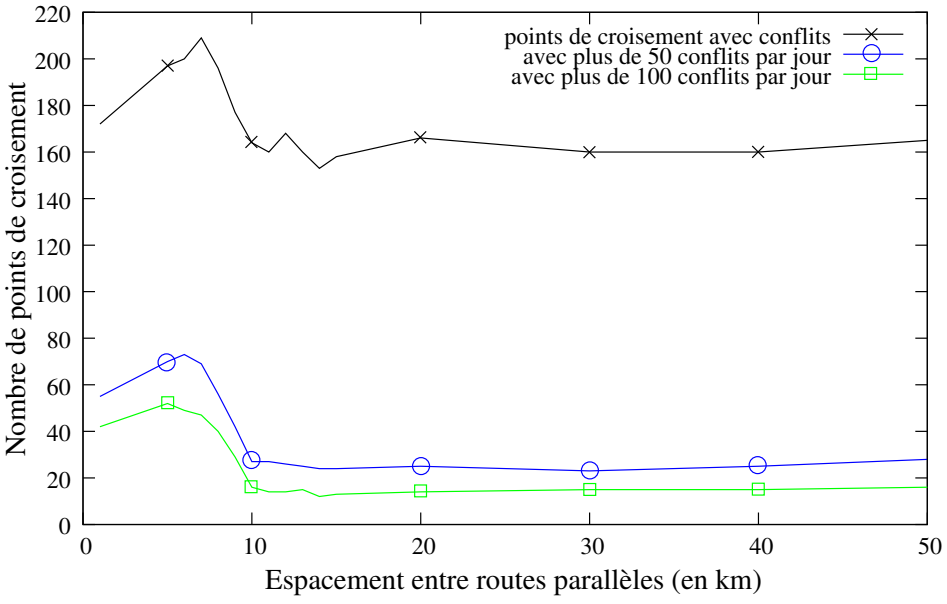

FIG. 7.6 – Évolution du nombre de conflits internes aux points de croisement fonction de l'espacement des routes parallèles

est supérieure à la norme de séparation, il apparaît que cette valeur n'a qu'une influence limitée.

Ainsi, on peut affirmer que la valeur de la séparation entre les routes parallèles importe peu. On la choisira égale à 15 km par la suite.

Conflits par point de croisement

Puisque le concept *Sector-Less* implique un contrôle de chaque aéronef depuis son aéroport de départ jusqu'à sa destination, donc un contrôle par flux, la charge des points de croisement fournit une bonne vision quant à la faisabilité du concept en terme de quantité de trafic gérable pas les contrôleurs.

Si le nombre total de conflits donne une première impression, il est important de comprendre comment ces conflits sont répartis géographiquement. Ainsi, nous ne nous intéressons plus ici qu'aux conflits qui se situent au sein des points de croisement, ceux ayant lieu le long des axes n'étant que des conflits en rattrapage (c'est-à-dire lorsqu'un aéronef va plus vite que celui qui le précède) et facilement résolu par le système, soit en régulant la vitesse des aéronefs, soit en utilisant les voies de dépassement propre à chaque routes.

Comme on peut le voir sur la figure 7.6, quelle que soit la taille des points de croisement, une vingtaine d'entre eux ont plus de 100 conflits par jour. Géographiquement (voir figure 7.7), la partie la plus chargée d'Europe est composée du sud du Royaume-Uni, de la France, du Bénélux et de l'Allemagne. On trouve ensuite les autres croisements comptant beaucoup de conflits potentiels isolés près des flux importants, par exemple un en Espagne près de Barcelone ou encore en Italie près de Milan. Enfin on trouve quelques points de croisements très chargés dans les « coins » de la grille correspondant aux points d'entrée ou de sortie des aéronefs en provenance ou à destination d'un lieu extra-européen.

Conflits par point de croisement et par fenêtre glissante

Quelles que soient les solutions générées, les points de croisement les plus chargés se situent toujours dans les mêmes zones géographiques. Nous nous intéressons, dans cette partie, à quatre croisements considérés comme représentatif du point de vue de la charge (voir figure 7.7) :

- un point de croisement aux environs de Limoges (au sud-ouest de Paris) (LMG) avec 71 conflits dans la journée et un point de croisement à l'est de Paris (PE) avec 112 conflits ; on considérera ces deux points comme représentatifs de zones chargées en terme de conflits mais dont le nombre parait gérable par des contrôleurs ;
- un point de croisement en Suisse (GNV) avec 212 conflits ainsi qu'un point de croisement au sud de l'Allemagne (FFT) avec 349 conflits ; on considérera ceux-ci comme représentatifs de zones probablement surchargées et donc difficilement contrôlable.

Le nombre total de conflits ne permettant pas d'évaluer la charge de travail des différents contrôleurs, la figure 7.8 présente le nombre instantané de conflit sur une journée. La première observation qui peut être faite est que ce nombre dépasse rarement 1, c'est-à-

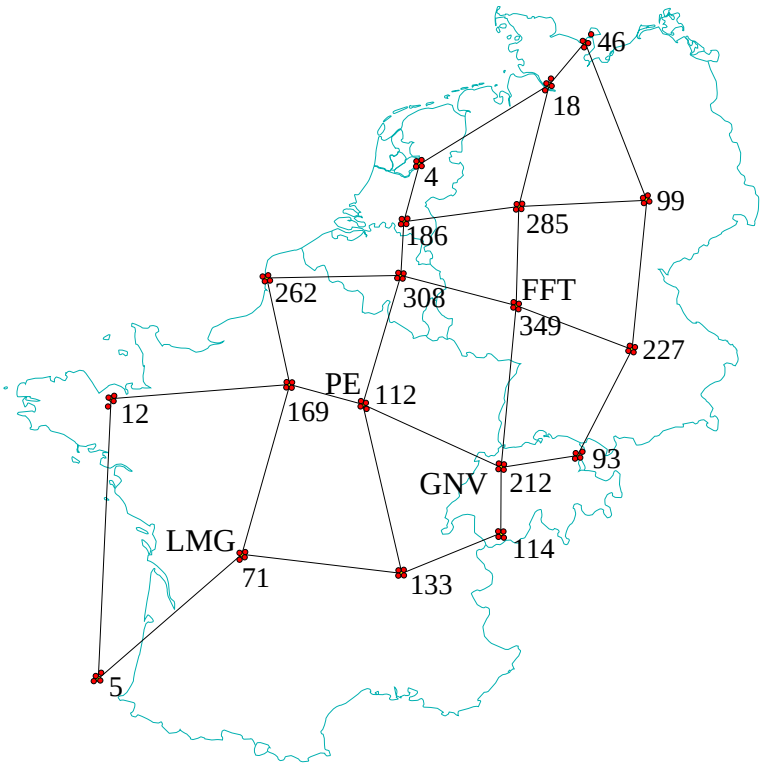

FIG. 7.7 – Nombre de conflits par points de croisement avec un espacement de 15 km entre routes parallèles au dessus de la région la plus chargée en Europe

dire qu'il n'arrive que très rarement que deux conflits indépendants (c'est-à-dire n'ayant pas d'aéronefs impliqués en commun) surviennent au même moment. Bien entendu, plus le nombre total de conflits est élevé, plus leur fréquence est importante. On se rend bien compte alors que les points de croisement les plus chargés vont être difficilement gérables par les contrôleurs.

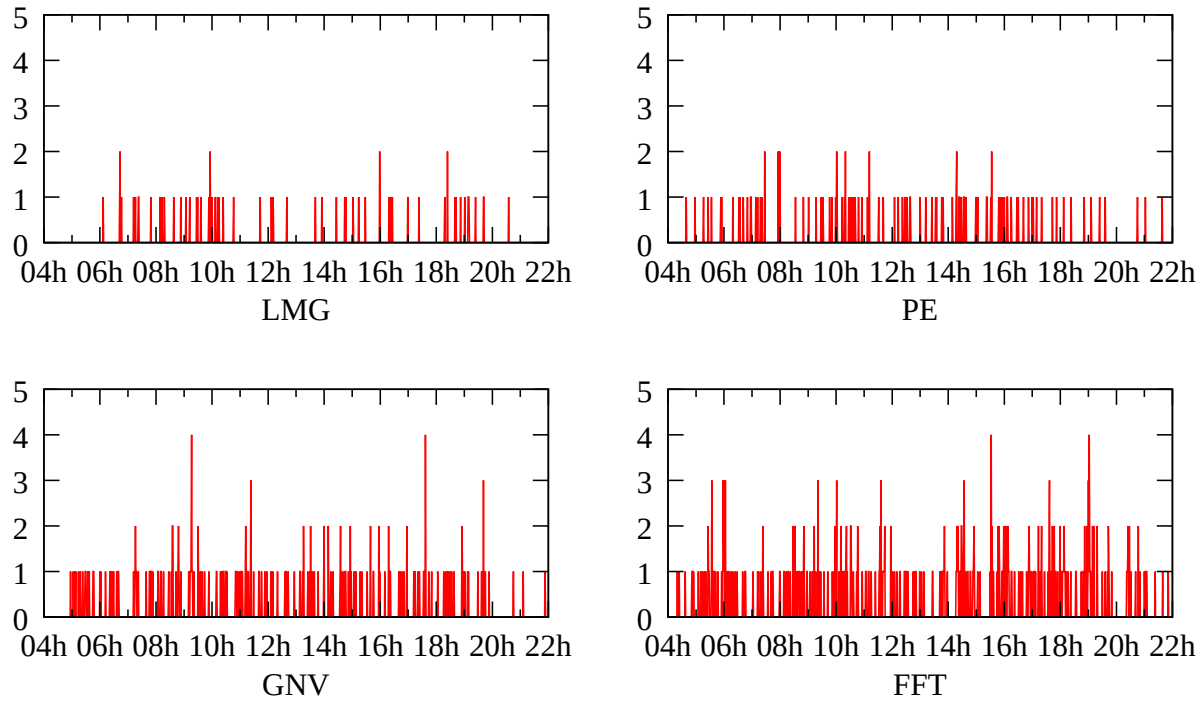

FIG. 7.8 – Nombre instantané de conflits internes aux quatre points de croisement de référence

Si le nombre instantané de conflits donne une première indication sur la charge de travail des contrôleurs, une meilleure mesure est le nombre de conflits à traiter dans les x minutes à venir. La figure 7.9 présente ainsi les résultats utilisant une fenêtre glissante de 15 minutes, c'est-à-dire le nombre de conflits à résoudre dans les 15 prochaines minutes. Sur les points de croisement ayant « peu » de conflits, ce nombre dépasse rarement 5, ce qui parait parfaitement gérable par les contrôleurs [Averty 98]. En revanche, si on considère les points de croisement ayant un nombre élevé de conflits par jour, le nombre de conflits à résoudre en moins de 15 minutes est régulièrement supérieur à 10 et parfois même supérieur 15. On se rend bien compte alors que, dans ce cas extrême, la résolution est impossible à effectuer par un opérateur humain.

Il faut maintenant rappeler le principe général d'un contrôle *Sector-Less*. Dans ce cadre-ci, un aéronef est contrôlé par un même contrôleur depuis la zone d'approche de son aéroport de départ (ou depuis son point d'entrée dans le réseau de routes s'il vient d'un pays hors zone couverte par le réseau) jusqu'à celle de son aéroport d'arrivée (ou de son

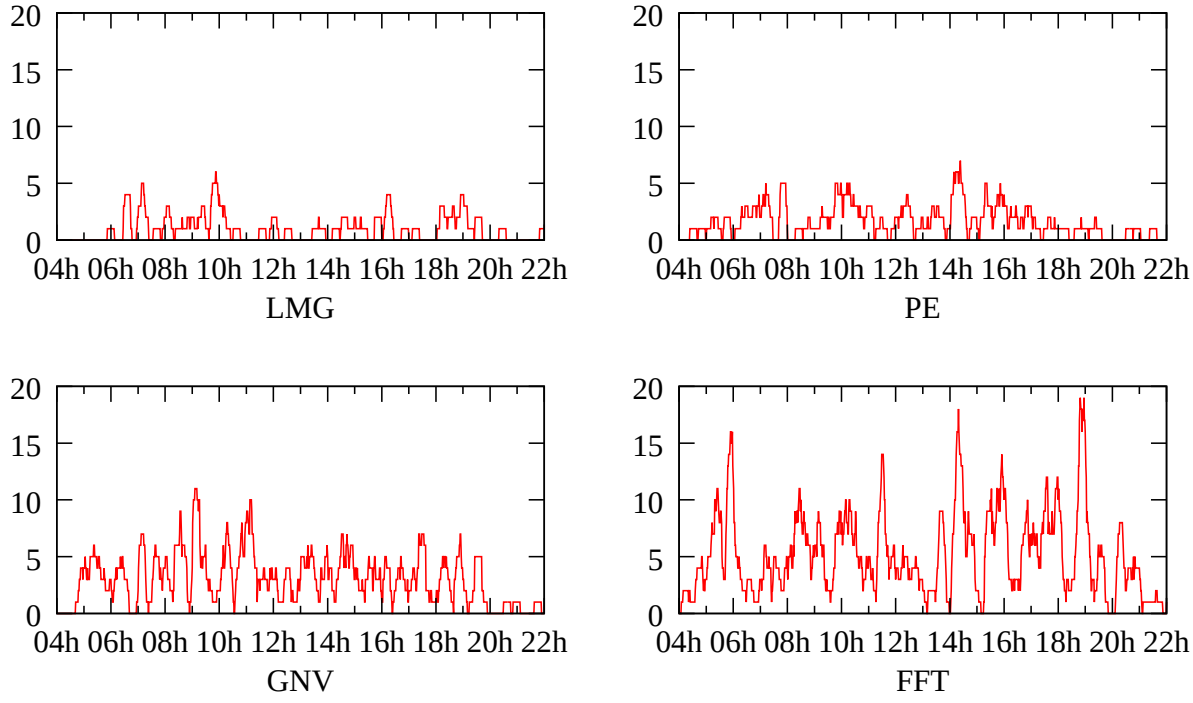

FIG. 7.9 – Nombre de conflits internes à résoudre par fenêtre glissante de 15 minutes aux quatre points de croisements de référence

départ du réseau s'il rejoint un pays hors zone). Lorsqu'il y a conflit entre deux ou plusieurs aéronefs, deux cas sont alors envisageables :

- le contrôleur contrôle tous les aéronefs en conflits et la gestion de ceux-ci n'a pas d'influence sur les autres aéronefs ; alors le conflit est facilement soluble par le contrôleur concerné.
- plusieurs contrôleurs sont concernés par le conflit ou la résolution d'un conflit peut perturber la trajectoire d'aéronefs extérieurs à celui-ci. La résolution implique donc une « négociation » entre les différents opérateurs humains concernés.

Le problème principal vient donc de savoir quels types de conflits on rencontre le plus. Étant donné que les conflits enregistrés sont uniquement dus aux aéronefs changeant de direction donc changeant de niveau et les aéronefs étant, a priori, affectés aux contrôleurs en fonction de leur flux, on risque donc, en grande majorité se retrouver avec des conflits concernant deux ou plusieurs contrôleurs. C'est bien entendu ceux-ci qui posent le plus de problèmes. En effet, si l'on s'intéresse aux points de croisement chargés, on peut imaginer un même contrôleur obligé de résoudre un conflit avec un contrôleur et un autre conflit avec un autre contrôleur. D'après [Averty 98] la charge d'un contrôleur est excessive lorsque celui-ci doit gérer six conflits au même moment. Un conflit, dans le cadre du concept *Sector-Less* pouvant, en plus, faire intervenir une négociation entre deux voire plusieurs contrôleurs, on se rend bien compte que ceci risque de devenir totalement ingérable.

Le concept *Sector-Less* montre alors toutes ses limites et parait, pour l'instant, peu transposable dans la réalité.

7.3 Évaluation des algorithmes de maintien des plus courts chemins dans un graphe

Tous les tests évoqués dans les sections qui suivent ont été effectués sur un pentium IV cadencé à 2.8 Ghz possédant 4 Go de mémoire vive.

Les algorithmes présentés chapitre 4 ont tout d'abord été testés sur des graphes planaires quelconques générés aléatoirement, puis sur des graphes planaires quelconques mais respectant la topologie décrite au chapitre 3 (un sommet est relié à au plus quatre autres sommets qui lui sont « géographiquement » adjacents) et enfin sur le problème principal traité dans ce document, c'est-à-dire la génération et l'optimisation d'un réseau principal de routes pour un contrôle *Sector-Less*.

Les résultats des tableaux 7.2 et 7.3 montrent, pour un nombre donné de variations aléatoires du graphe (modification du poids des arcs pour le tableau 7.2, « déplacement » de sommets pour le tableau 7.3), le temps CPU (en secondes) pour l'initialisation (calcul pour la première fois de tous les plus courts chemins) ainsi que le temps total d'exécution hors temps d'initialisation. La dernière ligne de chaque tableau indique la quantité (en Mo) maximale de mémoire utilisée mesurée (après l'initialisation et au total après 1000 itérations).

La comparaison se fait entre trois algorithmes différents, le but étant d'évaluer l'algo-

rithme de maintien des plus courts chemins présenté section 4.4.2 page 48. On le comparera tout d'abord à l'algorithme de Floyd-Warshall (section 4.2.2 page 40) puisque l'algorithme *ad-hoc* en est directement inspiré et ceci afin de pouvoir relever les améliorations réalisées, puis à l'algorithme de Ramalingam et Reps (section 4.3.2 page 42), ce dernier étant considéré comme la référence en terme d'efficacité.

7.3.1 Évaluation des algorithmes de maintien sur des graphes planaires aléatoires

Les premiers tests ont été réalisés sur des graphes planaires quelconques générés aléatoirement [6]. On évalue les algorithmes en fonction de plusieurs caractéristiques :

- le nombre d'itérations, c'est-à-dire le nombre de fois où l'on exécute les algorithmes de mise à jour de la structure contenant les plus courts chemins. Dans le cas de l'algorithme de Floyd-Warshall, c'est le nombre de fois où l'on exécute entièrement le processus (tous les plus courts chemins sont alors recalculés).
- le nombre de modifications de poids d'arcs par itération, c'est-à-dire le nombre d'arcs ayant la valeur de leur poids modifiée. L'algorithme de Ramalingam et Reps effectuant les mises à jour une par une, on exécute dans ce cas l'algorithme autant de fois qu'il y a d'arcs modifiés.

Le tableau 7.2 présente la moyenne des résultats des tests effectués sur dix graphes planaires aléatoires de 256 sommets. La première partie du tableau présente les résultats après n itérations, chaque itération correspondant à l'altération du poids d'un unique arc choisi au hasard ; la seconde partie présente les résultats où l'on effectue quatre modifications de poids sur 4 arcs différents choisis au hasard et enfin la dernière où l'on effectue 10 modifications sur 10 arcs différents.

Le tableau 7.3 présente la moyenne des résultats des tests effectués sur dix graphes planaires aléatoires de 256 sommets mais respectant la topologie, c'est-à-dire que les graphes sont « en forme » de grille. La première partie du tableau présente les résultats après n itérations, chaque itération étant le déplacement d'un sommet choisi au hasard donc correspond à l'altération du poids des arcs reliés à ce sommet (jusqu'à 4 arcs mis à jour) ; la seconde partie présente les résultats où l'on effectue quatre déplacements de sommets (jusqu'à 16 arcs mis à jour) et enfin la dernière où l'on effectue 10 modifications sur 10 sommets différents.

Au vu des résultats, c'est bien entendu l'algorithme de Ramalingam et Reps qui est le plus efficace, quel que soit le cas étudié. Non seulement il est bien plus rapide que tous les autres algorithmes mais de plus il consomme moins d'espace mémoire.

L'algorithme *ad-hoc* inspiré de celui de Floyd-Warshall est quant à lui plus rapide que ce dernier mais uniquement dans les cas les plus simples, c'est-à-dire lorsque l'on change le poids d'un unique arc ou lorsque l'on déplace un unique sommet donc que l'on modifie la valeur des poids des arcs qui lui sont reliés. Dans le premier cas on améliore par un

[6] Les graphes planaires quelconques ont été générés en utilisant la librairie Ocamlgraph (voir `www.lri.fr/~filliatr/ocamlgraph`).

1 modification aléatoire d'arc par itération						
	Floyd-Warshall		Ramalingam et Reps		Algorithme *ad-hoc*	
Nb itérations	init.	mises à jour	init.	mises à jour	init.	mises à jour
1	0.47	0.50	0.22	0.00	2.96	0.05
10	0.46	4.97	0.22	0.05	2.94	1.07
100	0.46	49.81	0.22	0.47	2.95	12.80
1000	0.46	502.30	0.22	4.96	2.96	155.51
RAM (Mo)	2.6	2.8	1.6	1.6	131.9	242.1
4 modifications aléatoires d'arcs par itération						
	Floyd-Warshall		Ramalingam et Reps		Algorithme *ad-hoc*	
Nb itérations	init.	mises à jour	init.	mises à jour	init.	mises à jour
1	0.47	0.50	0.22	0.02	2.96	0.27
10	0.47	4.99	0.22	0.18	2.97	4.82
100	0.47	50.26	0.22	1.90	2.96	60.22
1000	0.47	508.67	0.22	21.95	2.95	675.41
RAM (Mo)	2.6	2.8	1.6	1.6	131.9	243.7
10 modifications aléatoires d'arcs par itération						
	Floyd-Warshall		Ramalingam et Reps		Algorithme *ad-hoc*	
Nb itérations	init.	mises à jour	init.	mises à jour	init.	mises à jour
1	0.47	0.50	0.22	0.05	2.97	1.05
10	0.46	5.04	0.21	0.49	2.97	12.36
100	0.47	50.56	0.22	5.01	2.98	153.79
1000	0.47	506.23	0.22	57.99	2.94	1634.62
RAM (Mo)	2.6	2.9	1.6	1.6	131.9	244.5

TAB. 7.2 – Temps CPU en seconde (initialisation et temps des mises à jour) des algorithmes de maintien sur des graphes planaires aléatoires de 256 sommets en fonction du nombre de modifications par itération

1 sommet déplacé aléatoirement par itération (jusqu'à 4 arcs modifiés)						
	Floyd-Warshall		Ramalingam et Reps		Algorithme *ad-hoc*	
Nb itérations	init.	mises à jour	init.	mises à jour	init.	mises à jour
1	0.32	0.33	0.23	0.04	2.78	0.08
10	0.31	3.26	0.23	0.53	2.77	1.26
100	0.31	33.05	0.23	4.56	2.77	11.27
1000	0.32	331.78	0.23	45.39	2.78	142.09
RAM (Mo)	2.0	2.3	1.6	1.6	130.2	237.2
4 sommets modifiés aléatoirement par itération (jusqu'à 16 arcs modifiés)						
	Floyd-Warshall		Ramalingam et Reps		Algorithme *ad-hoc*	
Nb itérations	init.	mises à jour	init.	mises à jour	init.	mises à jour
1	0.32	0.33	0.23	0.24	2.80	0.51
10	0.31	3.33	0.23	1.98	2.72	4.45
100	0.32	32.87	0.23	18.38	2.75	53.45
1000	0.31	331.53	0.23	178.69	2.75	604.86
RAM (Mo)	2.0	2.3	1.6	1.6	130.2	237.8
10 sommets modifiés aléatoirement par itération (jusqu'à 40 arcs modifiés)						
	Floyd-Warshall		Ramalingam et Reps		Algorithme *ad-hoc*	
Nb itérations	init.	mises à jour	init.	mises à jour	init.	mises à jour
1	0.32	0.33	0.23	0.53	2.76	1.14
10	0.32	3.40	0.24	4.89	2.81	10.91
100	0.31	33.14	0.23	45.74	2.82	135.58
1000	0.32	330.98	0.23	445.96	2.76	1405.91
RAM (Mo)	2.0	2.3	1.6	1.6	130.2	238.2

TAB. 7.3 – Temps CPU en seconde (initialisation et temps des mises à jour) des algorithmes de maintien sur des graphes planaires aléatoires de 256 sommets respectant la topologie en fonction du nombre de modifications par itération

facteur 3, dans le second par un facteur 2. En revanche dès que le nombre de modifications devient important, l'algorithme de maintien n'est plus efficace et il est alors plus rapide de recalculer tous les plus courts chemins.

7.3.2 Évaluation des algorithmes de maintien sur le cas réel

La différence majeure lors de l'évaluation du cas réel avec les évaluations faites précédemment est, en plus de l'environnement (lecture des données, optimisation, génération de plans de vol, etc.), l'ajout de la contrainte d'angle imposée à tous les aéronefs (voir section 3.4.4 page 30). Or il n'est pas possible d'intégrer cette contrainte au sein même de l'algorithme de maintien de Ramalingam et Reps. En effet, les différents algorithmes de maintien des plus courts chemins utilisant la structure d'arbre des plus courts chemins, maintiennent cette structure en tenant compte du fait qu'un plus court chemin est une somme de plus courts chemins. Considérons le plus court chemin allant de i à k passant par le sommet j $\langle i, j, k\rangle$. Supposons alors que j est modifié tel que $\widehat{ijk}$ ne vérifie plus la contrainte d'angle mais en conservant $\langle i, j\rangle$ et $\langle j, k\rangle$ comme les plus courts chemins allant respectivement de i à j et de j à k. $\langle i, j, k\rangle$ n'est plus, au regard de la contrainte d'angle, le plus court chemin allant de i à k. On ne peut donc plus construire l'arbre des plus courts chemins ayant pour racine i.

Insertion de la contrainte angulaire dans le graphe

Pour contourner cette difficulté, la contrainte a été « insérée » dans le graphe lui-même. Chaque sommet du graphe a ainsi été dupliqué en huit sommets identiques afin de pouvoir gérer les angles de virages. La figure 7.10 présente cette duplication. Les flèches représentent les sens des arcs. En réalité ces huit sommets se superposent géographiquement.

Ainsi, lorsque l'on déplace un sommet, en calculant les différents angles qu'il forme avec son voisinage et en supprimant les arcs correspondant aux trajectoires ne respectant pas les contraintes, il devient impossible de calculer un plus court chemin ne respectant pas la contrainte angulaire.

Par exemple, comme illustré figure 7.11 supposons qu'un déplacement implique que l'angle $\widehat{ACB}$ devient inférieur à la limite imposée (dans le cas des simulations, on prend cette limite égale à 90 degrés), il suffit alors de supprimer les arcs $\overrightarrow{1,8}$ et $\overrightarrow{7,2}$. Si l'angle redevient supérieur à la limite angulaire, ces arcs sont rajoutés. Pour ne pas fausser le calcul de distance, ces arcs « internes » sont de poids nuls. Sachant que les algorithmes de maintien gèrent parfaitement l'ajout ou la suppression d'arc, cette modification n'a pas d'influence sur le code de ces algorithmes. En revanche la taille du graphe à parcourir est, elle, huit fois supérieure.

Le problème de cette solution est qu'il est alors difficilement possible d'utiliser l'algorithme *ad-hoc*. En effet, il faudrait pouvoir stocker 2048 matrices $(256 * 8)$ de taille respective 2048 x 2048. Cela est bien entendu possible à condition de posséder un machine ayant plus de 64 G0 de mémoire vive...

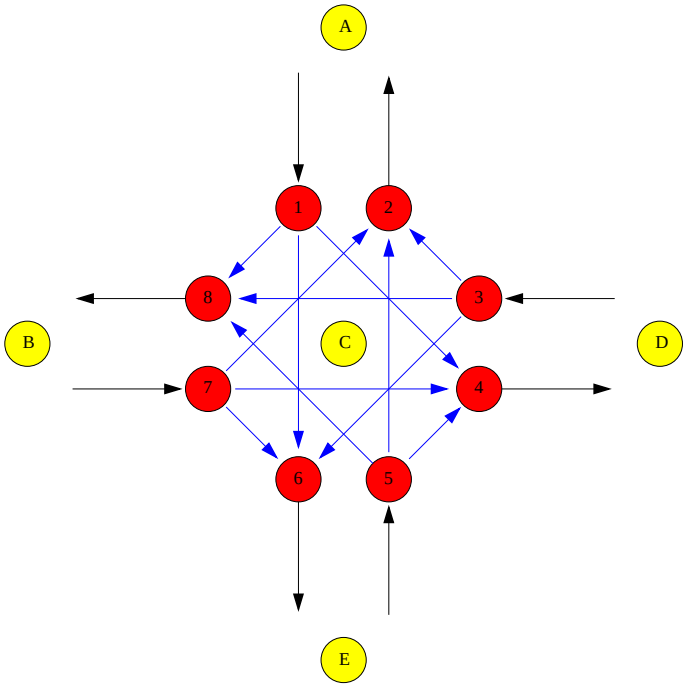

FIG. 7.10 – Duplication de sommet afin de permettre l'insertion d'un contrainte angulaire dans un graphe

Contrainte angulaire partielle

Une autre solution consiste alors à utiliser notre algorithme en *valeur approchée* quant au calcul des plus courts chemins. Contrairement à l'algorithme de Ramalingam et Reps, car il travaille sur les arbres de plus courts chemins, il est possible d'insérer une contrainte d'angle *partielle* au sein même de l'algorithme *ad-hoc* et ainsi éviter la duplication des sommets. En revanche le résultat trouvé ne sera alors qu'une approximation de la véritable valeur car certaines trajectoires seront plus longues que le véritable plus court chemin. Il faudra alors faire subir au résultat un post traitement afin de généré les bons plans de vol à partir des véritables plus courts chemin respectant la contrainte d'angle, ceci ne pouvant être fait qu'en dupliquant chacun des sommets comme précédemment.

On ajoute la contrainte partielle aux lignes 4, 10 et 17 de l'algorithme 5 page 50. Pour un i et un j fixés, l'algorithme évalue si un plus court chemin ne passerait pas par k. Connaissant le plus court chemin de i à k passant par des sommets inférieurs à k, soit i' le sommet tel que $\langle i...i'k\rangle$ est ce plus court chemin. Connaissant le plus court chemin de k à j passant par des sommets inférieurs à k, soit j' le sommet tel que $\langle kj'...j\rangle$ est ce chemin. On peut alors facilement contraindre $\widehat{i'kj'}$ à être supérieur à la limite fixée.

La contrainte n'est alors que partielle. En effet, l'algorithme de Floyd et sa version de maintien des plus courts chemins développée ici sont basés sur deux propriétés : l'incrémentalité qui assure de parcourir tous les sommets du graphe mais également le fait qu'un plus

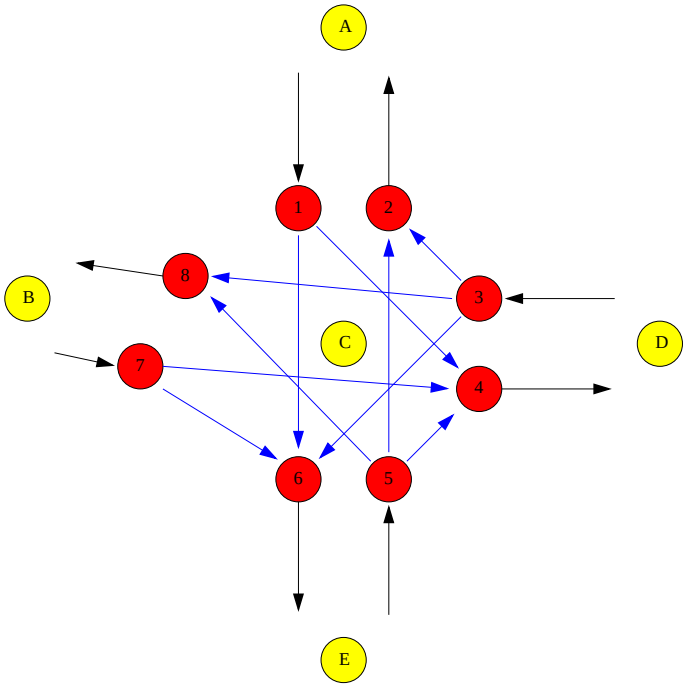

FIG. 7.11 – Modification de graphe entraînant le non respect de la contrainte angulaire et ainsi la suppression d'un arc « interne »

court chemin est la somme de plus courts chemins. Or en rajoutant une contrainte angulaire on perd cette seconde propriété à cause de l'incrémentalité.

La figure 7.12 présente un cas où cette propriété est violée. Calculons le plus court chemin pour aller de 1 à 6. L'algorithme parcourt les différents sommets dans l'ordre croissant :

- Le premier testé est le sommet numéro 2. Comme il n'existe pas de plus court chemin entre 2 et 6 passant par des sommets inférieurs à 2, le passage par ce sommet n'est pas valide.
- On tente par 3. Le plus court chemin de 1 à 3 passe par 2. Comme l'angle $\widehat{2,3,6}$ est inférieur à 90 degrés, le passage par ce sommet n'est pas valide.
- Par 4. Pas de plus court chemin entre 4 et 6 passant par des sommets inférieurs à 4.
- Par 5. Le plus court chemin de 1 à 5 passe par 2. Comme l'angle $\widehat{2,5,6}$ est inférieur à 90 degrés, le passage par ce sommet n'est pas valide.
- Par 7. Le plus court chemin de 1 à 7 passe par 4. Celui de 7 à 6 passe par 4 et 5. $\widehat{4,7,4}$ est inférieur à 90 degrés, le passage par ce sommet n'est pas valide.
- Par 8. Le plus court chemin de 1 à 8 passe par 2 et 5. Celui de 8 à 6 passe par 5. $\widehat{5,8,5}$ est inférieur à 90 degrés, le passage par ce sommet n'est pas valide.
- Par 9. Le plus court chemin de 1 à 9 passe par 2, 5 et 8. Le passage par ce sommet est valide.

Ainsi le plus court chemin trouvé pour aller de 1 à 6 avec la contrainte angulaire partielle

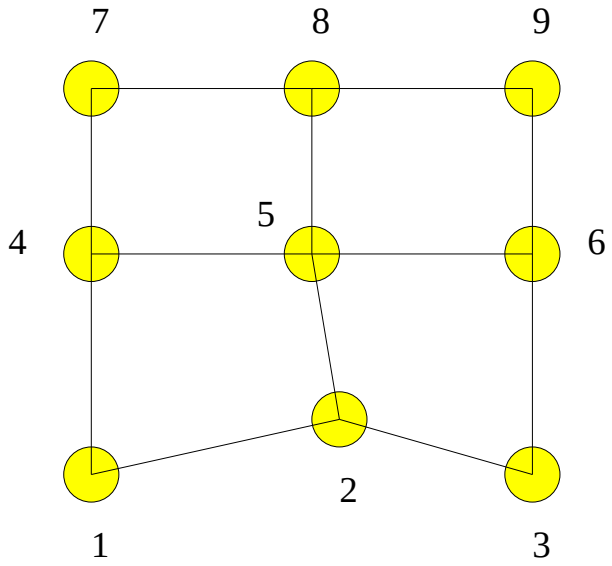

FIG. 7.12 – Exemple de graphe où la contrainte angulaire partielle empêche de trouver le plus court chemin

est $\langle 1, 2, 5, 8, 9, 6\rangle$. Or le véritable plus court chemin est $\langle 1, 4, 5, 6\rangle$ mais celui-ci n'a pas été détecté car au moment de tester le passage par 4, il n'existait pas de plus court chemin allant de 4 à 6 par des sommets inférieurs à 4. On obtient donc un plus court chemin *approché*.

Néanmoins il est tout à fait possible d'utiliser cette contrainte angulaire pour optimiser le réseau de route. La valeur du critère ne sera donc qu'une valeur approchée et les mouvements rejetés car dégradant le critère potentiellement bon. Un fois l'optimisation terminée, il suffira d'établir les véritables plans de vol en utilisant la technique de duplication des sommets.

Le tableau 7.4 présente les temps d'exécution de chacun des algorithmes de plus courts chemins en fonction du nombre d'itérations du recuit simulé, un unique sommet étant déplacé à chaque itération. Les valeurs représentent la moyenne sur dix exécutions du temps CPU passé par le recuit simulé à déplacer les sommets, calculer/maintenir les plus courts chemins et calculer le critère qui est présenté ici (on s'affranchit alors de la lecture des donnés et autres fonctions secondaires). Dans le cas de l'algorithme de Floyd-Warshall ou de notre algorithme, on fait un calcul en valeur approchée, dans le cas de l'algorithme de Ramalingam et Reps, on duplique les sommets donc le calcul se fait en valeur exacte.

À cause de la grande taille du graphe sur lequel s'exécute l'algorithme de Ramalingam et Reps, le temps d'initialisation est très important (plus d'une centaine de seconde) comparativement au temps entre deux itération (entre 2 et 3 secondes).

Par comparaison avec les tests sur un graphe aléatoire, on pouvait s'attendre à un gain d'environ 50% (voir tableau 7.3) pour 1000 itérations en utilisant l'algorithme de maintien des plus courts chemins par rapport à l'algorithme de Floyd-Warshall or on n'obtient

Nb itérations	Floyd-Warshall	Ramalingam et Reps	Algorithme *ad-hoc*	
	GC normal	GC normal	GC normal	GC augmenté
1	0.82	119.21	0.99	0.86
10	8.39	144.51	9.59	8.11
100	83.45	352.93	76.77	62.35
1000	848.60	3212.48	760.64	612.32
RAM (Mo)	10.9	155.8	247.9	802.6

TAB. 7.4 – Tests sur le cas réel, en valeur approchée pour les Floyd, en valeur exacte pour Ramalingam et Reps

(résultats de gauche intitulés *GC normal*, dernière colonne dans 7.4) qu'un gain de 10%. Le problème vient alors de la quantité de mémoire utilisée, notamment lors du calcul du critère. Les éléments stockés en mémoire utilisés pour ce calcul sont nombreux et importants (256 matrices de taille 256 x 256). En profilant l'exécution de l'algorithme, on observe alors que 30% du temps est utilisé par les fonctions du *récupérateur de mémoire* (ou *garbage collector*). En augmentant la taille de la mémoire qui lui est réservée (*space overhead*) (résultats de droite intitulés *GC augmenté*, dernière colonne dans 7.4), on réduit considérablement le temps de calcul (on obtient une amélioration de l'ordre de 30% entre l'algorithme de Floyd-Warshall et celui *ad-hoc*) mais la quantité de mémoire utilisée augmente fortement passant de 250 Mo à 800 Mo. L'augmentation de la taille de l'espace mémoire réservé dans le cas de l'algorithme de Ramalingam et Reps n'a que peu d'influence sur la vitesse de l'algorithme.

Si la meilleure solution présentée précédemment (section 7.2.2) à été trouvée en utilisant l'algorithme de Ramalingam et Reps, les solutions générées en utilisant la valeur approchée du critère, une fois calculés les plus court chemins par duplication, sont équivalente en terme d'allongement moyen des trajectoires sans pondération. En fait, l'erreur de l'approximation n'est, en moyenne, que de l'ordre de 3%. Il y a donc un réel avantage en terme de temps de calcul à utiliser, dans notre cas, l'algorithme développé dans cette thèse. En revanche la consommation de mémoire induite est alors non négligeable. On retrouve ainsi le compromis commun à beaucoup d'applications entre quantité de mémoire et temps de calcul.

7.4 Évaluation des algorithmes d'optimisation

Il est difficile de comparer, du point de vue du temps de calcul les deux méta-heuristiques. En effet si l'on tente de tester l'algorithme génétique en l'associant à l'algorithme de Ramalingam et Reps, les paramètres étant ceux « optimaux » trouvés chapitre 5 donc avec une population de 10 individus, la consommation de mémoire vive est de l'ordre de 2 Go. Le temps de calcul pour passer d'une génération à une autre est alors tellement long et le nombre de générations étant importants, cet algorithme devient alors totalement inutilisable. Dans le cas de l'algorithme *ad-hoc*, le test n'a même pas pu être effectué par manque

de mémoire sur la machine de test...

On peut alors tenter de comparer les deux méta-heuristiques sur des graphes aléatoires mais dans ce cas, on s'aperçoit très rapidement de la supériorité du recuit simulé. L'opérateur de croisement défini précédemment n'étant pas réellement efficace, c'est à dire ne générant que très rarement de bon individus.

Conclusion et perspectives

Du point de vue algorithmique

Nous avons proposé dans cette thèse un nouvel algorithme de maintien des plus courts chemins permettant de résoudre le problème dynamique du calcul des plus courts chemins toutes paires.

Dans le cas particulier qui nous intéresse, où le calcul des plus courts chemins est soumis à une contrainte géométrique, les algorithmes classiques ne sont pas assez performants. En revanche, le nouvel algorithme développé dans cette thèse permet de calculer rapidement une valeur approchée de ce problème sous contraintes. Si les chemins maintenus par cet algorithme ne sont pas les plus courts, leur valeur est suffisamment proche pour être utilisable lors de la génération et de l'optimisation, en terme de distance, du graphe qui nous intéresse.

Néanmoins cet algorithme est nettement plus lent et plus gourmand en mémoire que les meilleurs algorithmes connus lorsqu'il est utilisé sur des graphes sans contrainte géométrique. Si cet algorithme n'est donc, pour l'instant, utile uniquement qu'au cas par cas, il ouvre plusieurs perspectives dans le maintien des plus courts chemins sous contrainte et notamment par l'élaboration d'un nouvel algorithme qui permettrait, non plus de calculer les plus courts chemins en valeur approchée mais en valeur exacte.

Du point de vue ATM

Nous avons évalué dans cette thèse la faisabilité d'un nouveau concept de gestion du trafic aérien, le concept *Sector-Less*, entre autre par la création, l'optimisation et l'évaluation d'un réseau principal de routes adapté.

La création et l'optimisation du réseau principal de routes montre qu'il est possible de générer un réseau moins dense mais au moins aussi bon que le réseau existant, en terme de distances à parcourir pour les aéronefs. Ceci peut avoir des conséquences positives, même pour le système actuel, et peut permettre d'améliorer le réseau de routes existant. En canalisant mieux le trafic actuel, il est peut être possible d'augmenter la capacité du système.

Néanmoins, le type de contrôle effectué dans le concept *Sector-Less* ressemble plus ou moins à celui effectué par le contrôle aérien militaire français [Rivière 02]. Les contrôleurs militaires français sont responsables de plusieurs aéronefs pour une zone géographique donnée (correspondant généralement à un quart de la France), ces aéronefs leur étant

affectés de manière à ce que deux aéronefs allant dans une même direction soient gérés par le même contrôleur. Lorsqu'un aéronef est potentiellement en conflit avec un second, une discussion s'engage entre les deux contrôleurs concernés, ceux-ci se trouvant dans une même salle. Il en résulte un marchandage permanent entre les différents contrôleurs. Pour avoir assisté à leur travail lors d'une journée particulièrement chargée (le centre de Tours alors visité était responsable ce jour là de toute la moitié ouest de la France), on se rend bien compte que ceci n'est absolument pas transposable dans l'aviation civile. En fait, afin de limiter les risques, un contrôleur militaire ne peut pas gérer plus de trois mouvements [7] et l'état major recommande fermement de ne pas dépasser deux mouvements hors période de surcharge exceptionnelle. Avec plus de 25000 vols par jour en moyenne, cette limitation du nombre de vols apparaît difficilement transposable dans le cas du contrôle civil.

Ainsi, même si le concept *Sector-Less*, en plus du système de contrôle par flux, fournit un aménagement de l'espace aérien composé, entre autre, d'un réseau de routes adapté et performant en terme de distances à parcourir pour les aéronefs, au vu des simulations effectuées ici, il apparaît difficile d'utiliser celui-ci sans modification majeure. Pour permettre son utilisation, le concept requiert notamment une affectation « intelligente » des aéronefs aux contrôleurs en fonction de leurs trajectoires et notamment en fonction des aéronefs avec lesquels ils risquent d'être en conflit puisqu'il est plus facile pour un contrôleur de gérer seul ces aéronefs que de devoir trouver un compromis avec une autre personne. Malgré cela, rien n'indique, pour l'instant, que ce concept est viable.

[7] Les contrôleurs militaires gèrent une patrouille composée de plusieurs avions comme un seul et unique vol d'où l'appellation de mouvement

Glossaire

ATFM	Air Traffic Flow Management, 13
ATM	Air Traffic Management, 5, 16, 17, 19, 48, 61
BADA	Base of Aircraft DAta, 84, 85
CASA	Computer Assisted Slot Allocation, 14
CFMU	Central Flow Management Unit, 13, 14, 84
FMS	Flight Management Display, 6
FREER	Free-Route Experimental Encounter Resolution, 18, 19
GPS	Global Positionning System, 6
IFR	Instruments Flight Rules, 5, 6, 8
OACI	Organisation de l'Aviation Civile Internationale, 8
RVSM	Reduced Vertical Separation Minima, 15
SRN	Secondary Route Network, 20
TACAN	Tactical Air Navigation system, 6
TMA	Terminal Maneuvring Area, 8, 19
TRN	Trunk Route Network, 20, 22
VFR	Visual Flight Rules, 5, 6, 8
VOR	VHF Omnidirectional Range, 6

Bibliographie

[Alliot 92] Jean-Marc Alliot et Thomas Schiex. *Intelligence Artificielle et Informatique Théorique*. Cepadues, 1992. ISBN : 2-85428-324-4. *Cité p.* 65

[Alliot 97] JM Alliot, JF Bosc, N Durand et L Maugis. Cats : A complete air traffic simulator. *16th DASC*, 1997. *Cité p.* 85

[Alliot 03] Jean-Marc Alliot et Dominique Colin de Verdiere. Atm : 20 ans d'effort et perspectives. *Symposium de l'Academie Nationale de l'Air et de l'Espace : vers l'automatisation du vol et sa gestion*. 2003. *Cité p.* 19

[Averty 98] Philippe Averty. *Les effets de la charge de trafic sur le niveau d'activation psychophysiologique du contrôleur aérien*. Thèse : Institut de psychologie, Université Lyon 2, juillet 1998. *Cité p.* 89, 90, 99, 101

[Barnier 01] Nicolas Barnier et Pascal Brisset. *FaCiLe : A Functional Constraint Library*. ENAC/CENA, (`www.recherche.enac.fr/opti/facile`), 2001. *Cité p.* 47

[Barnier 02] Nicolas Barnier. *Application de la programmation par contraintes à des problèmes de gestion du trafic aérien*. Thèse : Institut National Polytechnique de Toulouse, décembre 2002. *Cité p.* 46, 47

[Buriol 03] L.S. Buriol, M.G.C. Resende et M. Thorup. *Speeding up dynamic shortest path algorithms*. Rapport technique TD-5RJ8B, AT&T Labs Research, 2003. *Cité p.* 41

[Cerf 91] Raphaël Cerf. Asymptotic convergence of genetic algorithms. *Mathematics Subject Classification. 60F10, 60J10, 92D15.*, 1991. *Cité p.* 29, 66

[Cerny 85] Cerny. A thermodynamical approach to the travelling salesman problem : An efficient simulation algorithm. *Optimization Theory Applications*, 1985. *Cité p.* 75

[CFMU 00] Eurocontrol CFMU, Brussels. *Basic CFMU Handbook - General & CFMU Systems*, 6.0 edition, février 2000. *Cité p.* 13

[Cormen 01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest et Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, USA, 2001. *Cité p.* 38

[Demetrescu 01] Camil Demetrescu. *Fully Dynamic Algorithms for Path Problems on Directed Graphs.* Thèse : Università degli studi di Roma "La Sapienza", 2001. *Cité p.* 36

[Demetrescu 04] C. Demetrescu, S. Emiliozzi et G. Italiano. Experimental analysis of dynamic all pairs shortest path algorithms. *15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04).* 2004. *Cité p.* 41

[Diestel 00] Reinhard Diestel. *Graph Theory.* Springler-Verlag, New-York, 2000. *Cité p.* 36

[Dijkstra 59] Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1959. *Cité p.* 39

[Dréo 03] J. Dréo, A. Pétrowski, P. Siarry et E. Taillard. *Métaheuristiques pour l'optimisation difficile.* Eyrolles, 2003. *Cité p.* 29

[Duong 96] V. Duong, E. Hoffman, J.P. Nicolaon, L. Floc'hic et A. Bossu. *Extended Flight Rules to Apply to the Resolution of Encounters in Autonomous Airborne Separation.* Rapport technique, Eurocontrol, septembre 1996. *Cité p.* 18

[Duong 97a] Vu Duong et Eric Hoffman. Conflict resolution advisory in autonomous aircraft operations. *Procceedings of the 16th IEEE Digital Avionics System Conference.* Irvine Ca, 1997. Eurocontrol. *Cité p.* 18

[Duong 97b] Vu N Duong, Eric Hoffman et Jean-Pierre Nicolaon. Initial results of investigation into autonomous aircraft concept (freer-1). *Procceedings of the 1rst USA/Europe Seminar.* Saclay France, 1997. Eurocontrol/FAA. *Cité p.* 18

[Duong 98] Vu N Duong et Pierre Faure. On the applicability of the free-flight mode in european airspace. *Procceedings of the 2nd USA/Europe Seminar.* Orlando Fl, 1998. Eurocontrol/FAA. *Cité p.* 18

[Duong 01] Vu Duong, Gilles Gawinowski, Jean-Pierre Nicolaon et Darren Smith. Sector-Less Air Traffic Mangement. *4th USA/Europe Air Traffic Management R&D Seminar.* Santa Fe, décembre 2001. *Cité p.* 1, 19, 20, 26, 28

[Durand 99] N. Durand, J.M. Alliot et G. Granger. Peut-on supprimer le contrôle au sol. *La recherche*, avril 1999, vol 13, n°3. *Cité p.* 18

[era 05] *Erato : objectif 2008.* Rapport technique, Direction Technique et de l'Innovation, 2005. *Cité p.* 17

[Floyd 62] R. W. Floyd. ACM Algorithm 97 : Shortest path. *j-CACM*, juin 1962, vol 5, n°6, p 345. *Cité p.* 40

[Fogel 66] L.J Fogel, A.J Owens et M.J Walsh. *Artificial Intelligence Through Simulated Evolution.* Wiley and sons. NY, 1966. *Cité p.* 60

[Gawinowski 03a] Gilles GAWINOWSKI, Jean NOBEL, Didier DOHY, Jean-Yves GRAU et Vu DUONG. Bridging the predictive and adaptative issues in air traffic management : the synchronous paradigm. *22nd Digital Avionics Systems Conference*. Indianapolis, 2003. *Cité p.* 24

[Gawinowski 03b] Gilles GAWINOWSKI, Jean NOBEL, Didier DOHY, Laurent GUICHARD, Jean-Pierre NICOLAON et Vu DUONG. Operational Concepts For SuperSector. *5th USA/Europe Air Traffic Management R&D Seminar*. Budapest, juin 2003. *Cité p.* 24

[Gianazza 04] D. GIANAZZA. *Optimisation des flux de trafic aérien*. Thèse : Institut National Polytechnique de Toulouse, novembre 2004. *Cité p.* 26, 53, 62

[Goldberg 89] David GOLDBERG. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989. ISBN : 0-201-15767-5. *Cité p.* 53, 57, 65

[Gotteland 04] J.B GOTTELAND. *Optimisation du trafic au sol sur les grands aéroports*. Thèse : Institut National Polytechnique de Toulouse, novembre 2004. *Cité p.* 8, 53, 62

[Granger 01a] G. GRANGER, N. DURAND et J.M. ALLIOT. Optimal resolution of en route conflicts. *4th ATM R and D Seminar*. Santa Fe, décembre 2001. *Cité p.* 19

[Granger 01b] G. GRANGER, N. DURAND et J.M. ALLIOT. Token allocation strategy for free-flight conflict solving. *IJCAI'01*. Seattle, août 2001. *Cité p.* 19

[Granger 02] G. GRANGER. *Détection et résolution de conflits aérien : modélisation et analyse*. Thèse : Institut National Polytechnique de Toulouse, novembre 2002. *Cité p.* 19, 53, 56, 62, 85, 89

[Holland 62] John HOLLAND. Outline for a logical theory of adaptative systems. *Journal for the Association of Computing Machinery*, 1962, vol 3. *Cité p.* 53

[Ingber 89] L. INGBER. Very fast simulated re-annealing. *Mathematical Computer Modelling*, 1989, vol 12, n°8, p 967–973. *Cité p.* 77

[ITA 00] ITA. *Costs of air transport delay in Europe*. Rapport technique, Institut du Transport Aérien, novembre 2000. *Cité p.* 15

[Joines 94] J.A. JOINES et C.R. HOUCK. On the use of non stationary penalty functions to solve nonlinear constrained optimization problems with GAs. *Proceedings of the IEEE ICEC 1994*. 1994, p 579–584. *Cité p.* 56

[King 99] Valerie KING. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. *IEEE Symposium on Foundations of Computer Science*. 1999, p 81–91. *Cité p.* 41, 42

[Kirkpatrick 83] S. KIRKPATRICK, C.D. GELATT et M.P. VECCHI. Optimization by simulated annealing. *Science*, mai 1983, vol 220, n°4598, p 671–680. *Cité p.* 75

[Marsh 05] David Marsh et Agnieska Wegner. *Air traffic statistics and forecasts (STATFOR) : Medium-term forecast of flight movements (2005-2011).* Rapport technique, Eurocontrol, mai 2005. *Cité p.* ix, 1, 5, 15

[Metropolis 53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller et E. Teller. Equation of state calculations by fast computing machines. *Chemical Physics*, 1953, vol 21, n°6, p 1087–1092. *Cité p.* 75

[Michalewicz 91] Z Michalewicz et C.Z Janikov. Handling constraints in genetic algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithm.* Edited by Belew R.K et Booker L.B. ICGA, Morgan Kaufmann, 1991, p 151–157. *Cité p.* 56

[Michalewicz 92] Z. Michalewicz. *Genetic algorithms+data structures=evolution programs.* Springer-Verlag, 1992. ISBN : 0-387-55387-. *Cité p.* 61

[Michel 97] Laurent Michel et Pascal Van Hentenrick. Localizer : a modeling language for local search. *Third Conference on Principles and Practice of Constraint Programming.* 1997. *Cité p.* 47

[MIT 95] MIT, NASA et AATT. *Free flight case study.* Rapport technique, 1995. *Cité p.* 18

[Ramalingam 96] G. Ramalingam et Thomas W. Reps. An incremental algorithm for a generalization of the shortest-path problem. *J. Algorithms*, 1996, vol 21, n°2, p 267–305. *Cité p.* 41, 42

[Rivière 02] Thomas Rivière. *Le contrôle aérien militaire.* Rapport technique, CENA, 2002. *Cité p.* 19, 111

[Rivière 03] Thomas Rivière. A first approach in generating the trunk route network. *Innivative Research Activity Report*, 2003. *Cité p.* 2

[Rivière 04a] Thomas Rivière. Generating a european route network for sectorless. *Proceedings of the 1st International Conference on Research in Air Transportation.* Zilina, Slovakia, novembre 2004. *Cité p.* 2

[Rivière 04b] Thomas Rivière. Redesign of the european route network for sectorless. *Proceedings of the 23rd Digital Avionics Systems Conference.* Salt Lake City, octobre 2004. *Cité p.* 2

[Rivière 05a] Thomas Rivière. Shortest path in planar graph and air route network. *4th Eurocontrol Innovative Research Workshop & Exhibition.* décembre 2005. *Cité p.* 2

[Rivière 05b] Thomas Rivière et Pascal Brisset. Plus courts chemins dans un graphe planaire et création d'un réseau de routes aériennes. *JFPC'2005 : Premières Journées Francophones de Programmation par Contraintes.* Lens,France, juin 2005. *Cité p.* 2

[RTCA 95] RTCA. *Report of the RTCA Board of Directors select committee on free flight.* Rapport technique, 1995. *Cité p.* 18

[SIA 03] *Réglementation de la circulation aérienne*. Service de l'Information Aéronautique, 2003. *Cité p.* 7

[Szu 87] H. Szu et R. Hartley. Fast simulated annealing. *Physics Letters A*, juin 1987, vol 122, p 157–162. *Cité p.* 76

[Villiers 04] Jacques Villiers. *L'Automatisation du contrôle de la circulation aérienne pour un « ciel unique » novateur et efficace, comment et vers quoi avancer ensemble*. Rapport technique, Institut du Transport Aérien, 2004. *Cité p.* 18

[Yin 93] X Yin et N Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. *Proceedings of the Artificial Neural Nets and Genetic Algorithms*. Edited by Springer-Verlag. 1993. *Cité p.* 65