Techniques d’optimisation stochastique appliquées à certains problèmes du trafic aérien

Jean-Marc Alliot

27 octobre 1996

Table des matières

In a forest a fox bumps into a little rabbit, and says, "Hi, junior, what are you up to? "

"I'm writing a dissertation on how rabbits eat foxes," said the rabbit.

"Come now, friend rabbit, you know that's impossible! "

"Well, follow me and I'll show you." They both go into the rabbit's dwelling and after a while the rabbit emerges with a satisfied expression on his face.

Comes along a wolf. "Hello, what are we doing these days? "

"I'm writing the second chapter of my thesis, on how rabbits devour wolves."

"Are you crazy? Where is your academic honesty? "

"Come with me and I'll show you." As before, the rabbit comes out with a satisfied look on his face and a diploma in his paw. Finally, the camera pans into the rabbit's cave and, as everybody should have guessed by now, we see a mean-looking, huge lion sitting next to some bloody and furry remnants of the wolf and the fox.

The moral: It's not the contents of your thesis that are important — it's your PhD advisor that really counts.

Introduction

Ce document présente le bilan de quatre années d'études et de recherches effectuées par toute une équipe de thésards et d'étudiants de DEA dont j'ai fort modestement assuré, partiellement, la direction et l'encadrement. Ils ont implanté, corrigé, amélioré, découvert, et sans eux, nombre des idées que j'ai pu essayer d'exploiter n'auraient jamais dépassé le stade de la feuille de papier.

Ce document est donc le résultat d'un travail d'équipe. Je le revendique comme tel, et j'en suis particulièrement heureux. Il est vrai que si l'on considère l'Habilitation à Diriger des Recherches comme un doctorat d'Etat déguisé, cette monographie manque complètement son but : on aura peine à trouver plus d'une vingtaine de pages décrivant une recherche purement et exclusivement personnelle. Si, au contraire, on considère aussi l'HDR comme la démonstration d'une capacité à animer une équipe de recherche, alors les quelques pages qui suivent essaient de remplir cet office.

Ce document se compose de deux parties que l'on peut lire de façon presque indépendante.

La première partie présente un certain nombre de résultats de caractère général sur les techniques génétiques. Cette partie comporte deux chapitres :

- le premier chapitre sera consacrée aux algorithmes génétiques, avec une présentation des résultats théoriques existants, puis la description de tous les raffinements (scaling[1], sharing, clustering, parallélisme, etc.) indispensables à un fonctionnement efficace.
- dans le second chapitre, nous développerons sur quelques exemples classiques l'utilisation des algorithmes génétiques et nous les comparerons à d'autres techniques, locales (simplex, BFGS), globales déterministes (programmation par intervalles) ou stochastiques (recuit).

La seconde partie présentera l'application des techniques génétiques aux problèmes du trafic aérien à travers un certain nombre d'exemples :

- construction de trajectoires optimales pour la résolution de conflits en route
- optimisation de la sectorisation de l'espace et de la répartition de flux
- résolution réactive de conflits à court terme

Bien entendu, ce document est incomplet. Certains travaux réalisés, comme l'optimisation des chaînes sécurité dans les aérogares (étude pour le Service des Bases Aériennes), l'optimisation des redevances aéroportuaires (pour Aéroport De Paris), ou la réflexion sur l'optimisation des créneaux

[1] Nous emploierons dans tout ce document les termes anglais qui se sont (malheureusement?) imposés au fil des années, dont ceux développés spécifiquement par notre équipe. On pourrait certes remplacer "scaling" par "mise à l'échelle", "sharing" par "répartition forcée des éléments" ou clustering par "regroupement par paquets". Le texte n'y gagnerait pas nécessairement en lisibilité... Notons également que nous employons abusivement le mot aléatoire chaque fois que nous désignons un processus aléatoire dont le tirage est uniforme. La nature du processus est précisée chaque fois qu'il ne s'agit pas d'un tirage uniforme.

de décollage, n'ont pu y trouver place. Cependant, nous pensons qu'il reflète l'esprit général du travail que nous effectuons et souhaitons effectuer : appliquer une méthodologie scientifique aux problèmes du trafic aérien.

Partie I

Résultats généraux

Numerical optimization can be likened to a kangaroo searching for the top of Mt. Everest. Everest is the global optimum, but the top of any other really high mountain such as K2 would be nearly as good.

When training a neural network, initial weights are usually chosen randomly, which means that the kangaroo may start out anywhere in Asia. If you know something about the scales of the inputs, you may be able to get the kangaroo to start near the Himalayas. However, if you make a really stupid choice of distributions for the random initial weights, or if you have really bad luck, the kangaroo may start in South America.

In standard backprop or stochastic approximation, the kangaroo is blind and has to feel around on the ground to make a guess about which way is up. He may be fooled by rough terrain unless you use batch training. If the kangaroo ever gets near the peak, he may jump back and forth across the peak without ever landing on the peak. If you use a decaying step size, the kangaroo gets tired and makes smaller and smaller hops, so if he ever gets near the peak he has a better chance of actually landing on it before the Himalayas erode away. In backprop with momentum, the kangaroo has poor traction and can't make sharp turns.

With Newton-type (2nd order) algorithms, the Himalayas are covered with a dense fog, and the kangaroo can only see a little way around his location. Judging from the local terrain, the kangaroo make a guess about where the top of the mountain is, and tries to jump all the way there. In a stabilized Newton algorithm, the kangaroo has an altimeter, and if the jump takes him to a lower point, he backs up to where he was and takes a shorter jump. If the algorithm isn't stabilized, the kangaroo may mistakenly jump to Shanghai and get served for dinner in a Chinese restaurant. (I never claimed this analogy was realistic.) In steepest ascent with line search, the fog is very dense, and the kangaroo can only tell which direction leads up. The kangaroo hops in this direction until the terrain starts going down again, then chooses another direction.

Notice that in all the methods discussed so far, the kangaroo can hope at best to find the top of a mountain close to where he starts. There's no guarantee that this mountain will be Everest, or even a very high mountain.

In simulated annealing, the kangaroo is drunk and hops around randomly for a long time. However, he gradually sobers up and tends to hop up hill.

In genetic algorithms, there are lots of kangaroos that are parachuted into the Himalayas (if the pilot didn't get lost) at random places. These kangaroos do not know that they are supposed to be looking for the top of Mt. Everest. However, every few years, you shoot the kangaroos at low altitudes and hope the ones that are left will be fruitful and multiply.

From all this we can conclude that the best methods to find the Mount Everest are (in order):

1. *to know where it is*
2. *to have a map on which you can find it*
3. *to know someone who knows where it is or who has a map*
4. *to send (a) kangaroo(s) to search for it*

and even if you have to send a kangaroo, it is useful if you know at least:

1. *where the mountain range is in which the Mount Everest may be and*
2. *how to bring your kangaroo to that mountain range.*

Warren Searle

Chapitre 1

Les Algorithmes Génétiques

1.1 Principes généraux

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

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

1. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques. Le codages binaires ont été très utilisés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles.

2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

3. Une fonction à optimiser. Celle-ci retourne une valeur de $\Re^+$ appelée *fitness* ou fonction d'évaluation de l'individu.

4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.

5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

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

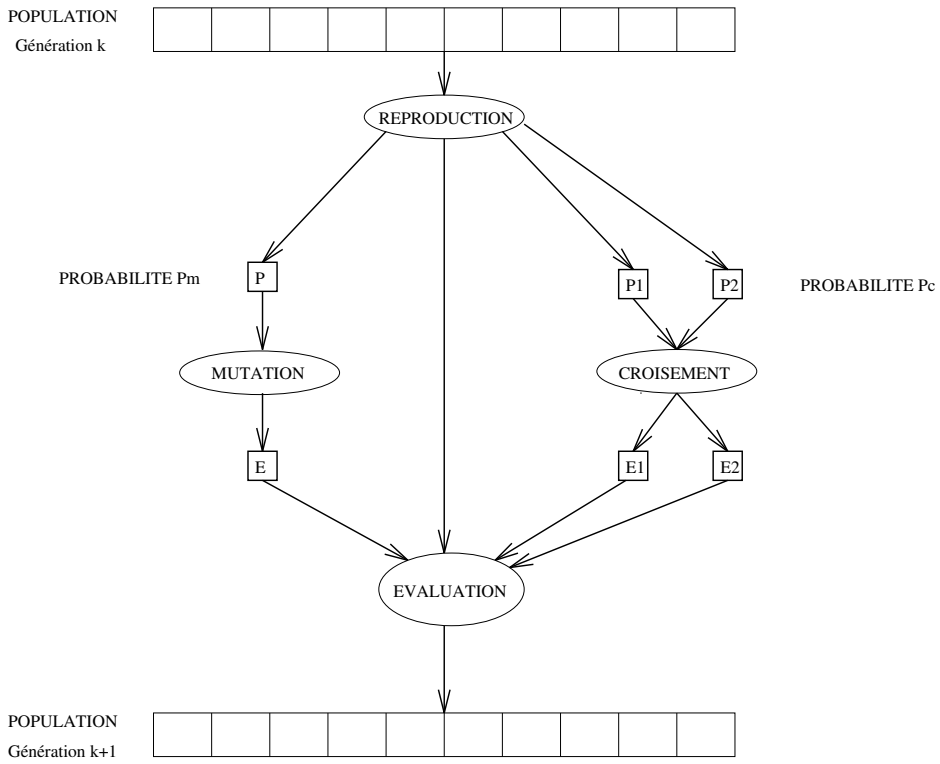


Figure 1.1: Principe général des algorithmes génétiques

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la figure 1.1 : on commence par générer une population d'individus de façon aléatoire. Pour passer d'une génération k à la génération $k+1$, les trois opérations suivantes sont répétées pour tous les éléments de la population k. Des couples de parents P_1 et P_2 sont sélectionnés en fonction de leurs adaptations. L'opérateur de croisement leur est appliqué avec une probabilité P_c (généralement autour de 0.6) et génère des couples d'enfants C_1 et C_2. D'autres éléments P sont sélectionnés en fonction de leur adaptation. L'opérateur de mutation leur est appliqué avec la probabilité P_m (P_m est généralement très inférieur à P_c) et génère des individus mutés P'. Le niveau d'adaptation des enfants ($C1$,C_2) et des individus mutés P' sont ensuite évalués avant insertion dans la nouvelle population. Différents critères d'arrêt de l'algorithme peuvent être choisis :

- Le nombre de générations que l'on souhaite exécuter peut être fixé à priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.

- L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement.

Nous allons maintenant détailler chacun de ces points.

1.2 Description détaillée

1.2.1 Codage des données

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

Cependant, ce type de codage n'est pas toujours bon comme le montrent les deux exemples suivants :

- deux éléments voisins en terme de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un codage de Gray.
- Pour des problèmes d'optimisation dans des espaces de grande dimension, le codage binaire peut rapidement devenir mauvais. Généralement, chaque variable est représentée par une partie de la chaîne de bits et la structure du problème n'est pas bien reflétée, l'ordre des variables ayant une importance dans la structure du chromosome alors qu'il n'en a pas forcément dans la structure du problème.

Les algorithmes génétiques utilisant des vecteurs réels [Gol91, Wri91] évitent ce problème en conservant les variables du problème dans le codage de l'élément de population sans passer par le codage binaire intermédiaire. La structure du problème est conservée dans le codage.

1.2.2 Génération aléatoire de la population initiale

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état en veillant à ce que les individus produits respectent les contraintes [MJ91]. Si par contre, des informations à priori sur le problème sont disponibles, il parait bien évidemment naturel de générer les individus dans un sous-domaine particulier afin d'accélérer la convergence. Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités. Il est clair qu'il vaut mieux, lorsque c'est possible ne générer que des éléments de population respectant les contraintes.

1.2.3 Gestion des contraintes

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

Il peut cependant être intéressant de conserver, tout en les pénalisant, les éléments non admissibles car ils peuvent permettre de générer des éléments admissibles de bonne qualité. Pour de nombreux problèmes, l'optimum est atteint lorsque l'une au moins des contraintes de séparation est saturée, c'est à dire sur la frontière de l'espace admissible.

Gérer les contraintes en pénalisant la fonction fitness est difficile, un "dosage" s'impose pour ne pas favoriser la recherche de solutions admissibles au détriment de la recherche de l'optimum ou inversement.

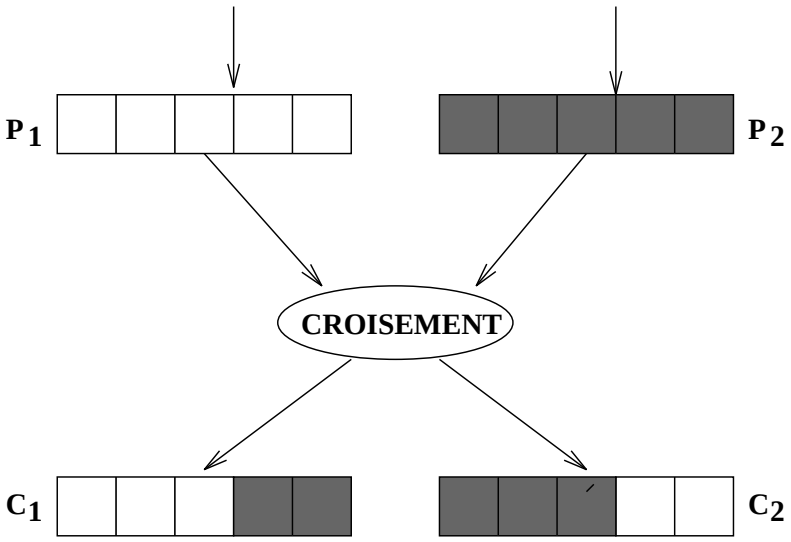


Figure 1.2: Slicing crossover

Disposant d'une population d'individus non homogène, la diversité de la population doit être entretenue au cours des générations afin de parcourir le plus largement possible l'espace d'état. C'est le rôle des opérateurs de croisement et de mutation.

1.2.4 Opérateur de Croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants.

Initialement, le croisement associé au codage par chaînes de bits est le croisement à découpage de chromosomes (slicing crossover). Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C_1 et C_2 (voir figure 1.2).

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

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets. Pour les problèmes continus, un croisement "barycentrique" est souvent utilisé : deux gènes $P_1(i)$ et $P_2(i)$ sont sélectionnés dans chacun des parents à la même position i. Ils définissent deux nouveaux gènes $C_1(i)$ et $C_2(i)$ par combinaison linéaire :

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

où α est un coefficient de pondération aléatoire adapté au domaine d'extension des gènes (il n'est pas nécessairement compris entre 0 et 1, il peut par exemple prendre des valeurs dans l'intervalle $[-0.5, 1.5]$ ce qui permet de générer des points entre, ou à l'extérieur des deux gènes considérés).

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

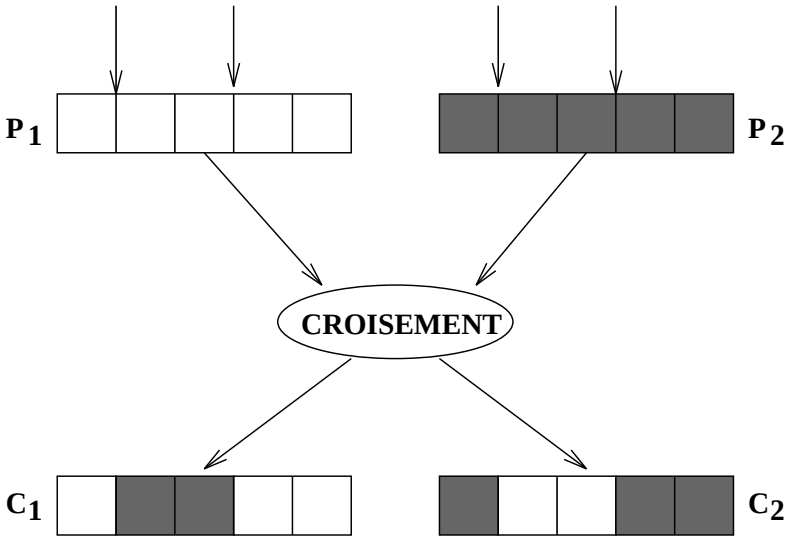


Figure 1.3: Slicing crossover à 2 points

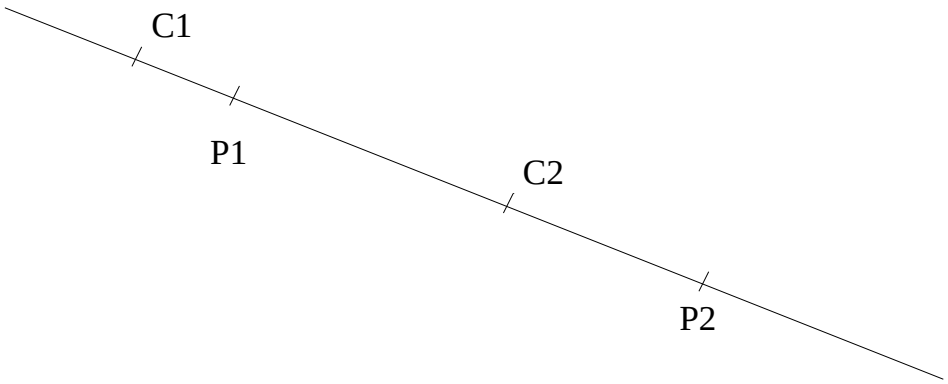


Figure 1.4: Croisement barycentrique

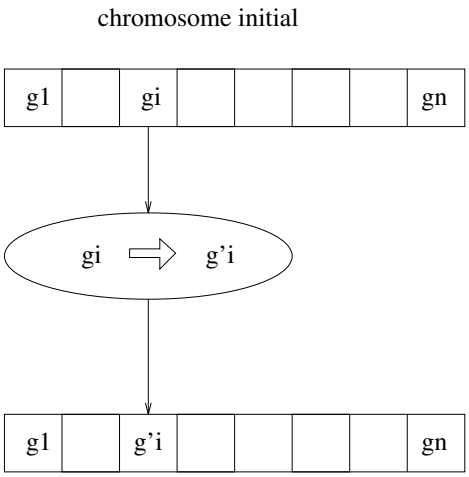


Figure 1.5: Principe de l'opérateur de mutation

$$\begin{cases} \vec{C}_1(i) = \alpha\vec{P}_1(i) + (1-\alpha)\vec{P}_2(i) \\ \vec{C}_2(i) = (1-\alpha)\vec{P}_1(i) + \alpha\vec{P}_2(i) \end{cases}$$

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

1.2.5 Opérateur de mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implantations fonctionnent de cette manière [FOW66]. Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur sur le plan théorique.

Pour les problèmes discrets, l'opérateur de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (voir figure 1.5). Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale.

Dans les problèmes continus, on procède un peu de la même manière en tirant aléatoirement un gène dans le chromosome, auquel on ajoute un bruit généralement gaussien. L'écart type de ce bruit est difficile à choisir a priori. Nous discutons ce problème de façon plus détaillée, en présentant une amorce de solution, dans la section 1.4.2.

1.2.6 Principes de sélection

A l'inverse d'autres techniques d'optimisation, les algorithmes génétiques ne requièrent pas d'hypothèse particulière sur la régularité de la fonction objectif. L'algorithme génétique n'utilise notamment pas ses dérivées successives, ce qui rend très vaste son domaine d'application. Aucune hypothèse sur la continuité n'est non plus requise. Néanmoins, dans la pratique, les algorithmes génétiques sont sensibles à la régularité des fonctions qu'ils optimisent.

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

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent. Dans le cadre de notre travail, les deux principes de sélection suivants ont été testés et évalués:

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

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

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

- Pour chaque élément i, on calcule le rapport r_i de sa fitness sur la moyenne des fitness.
- Soit $e(r_i)$ la partie entière de r_i, chaque élément est reproduit exactement $e(r_i)$ fois.
- La *roulette wheel selection* précédemment décrite est appliquée sur les individus affectés des fitness $r_i - e(r_i)$.

Compte-tenu du fait que des faibles populations seront utilisées par la suite, ce principe de sélection s'avèrera le plus efficace dans les applications pratiques et sera donc utilisé par la suite.

1.3 Améliorations classiques

1.3.1 Introduction

Les processus de sélection présentés sont très sensibles aux écarts de fitness et dans certains cas, un très bon individu risque d'être reproduit trop souvent et peut même provoquer l'élimination complète de ses congénères; on obtient alors une population homogène contenant un seul type d'individu. Ainsi, dans l'exemple de la figure 1.6 le second mode M_2 risque d'être le seul représentant pour la génération suivante et seule la mutation pourra aider à atteindre l'objectif global M_1 au prix de nombreux essais successifs.

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

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

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

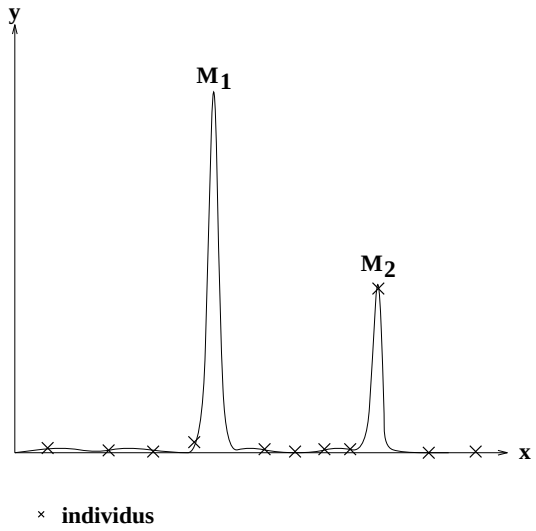


Figure 1.6: Exemple où les sélections classiques risquent de ne reproduire qu'un individu

1.3.2 Scaling

Le *scaling* ou mise à l'échelle, modifie les fitness afin de réduire ou d'amplifier artificiellement les écarts entre les individus. Le processus de sélection n'opère plus sur la fitness réelle mais sur son image après scaling. Parmi les fonctions de scaling, on peut envisager le scaling linéaire et le scaling exponentiel. Soit f_r la fitness avant scaling et f_s la fitness modifiée par le scaling.

Scaling linéaire

Dans ce cas la fonction de scaling est définie de la façon suivante [Mic92] :

$$f_s = af_r + b$$

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

En règle générale, le coefficient a est inférieur à un, ce qui permet de réduire les écarts de fitness et donc de favoriser l'exploration de l'espace. Ce scaling est statique par rapport au numéro de génération et pénalise la fin de convergence lorsque l'on désire favoriser les modes dominants.

Scaling exponentiel

Il est défini de la façon suivante [Mic92] (voir figure 1.7):

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

où n est la génération courante.

- Pour k proche de zéro, on réduit fortement les écarts de fitness ; aucun individu n'est vraiment favorisé et l'algorithme génétique se comporte comme un algorithme de recherche aléatoire et permet d'explorer l'espace.

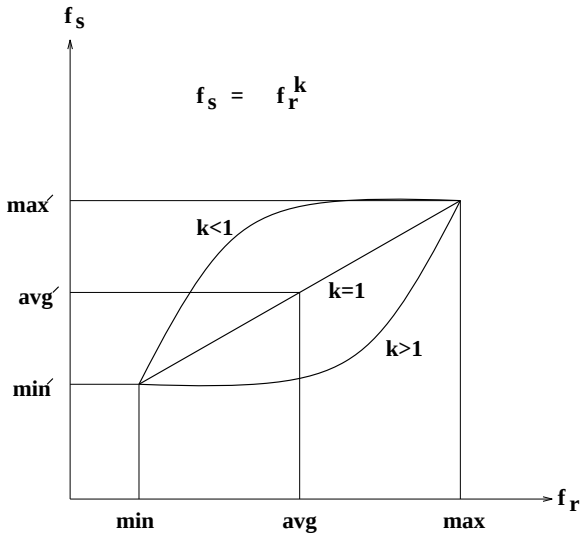


Figure 1.7: Fonction de scaling exponentielle

- Pour k proche de 1 : le scaling est inopérant.
- Pour $k > 1$ les écarts sont exagérés et seuls les bons individus sont sélectionnés ce qui produit l'émergence des modes.

Dans la pratique, on fait généralement varier k des faibles valeurs vers les fortes valeurs au cours des générations. Pour cela on peut utiliser la formule suivante :

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

n étant la génération courante, N le nombre total de générations prévues, p un paramètre à choisir. Le choix de $p = 0.1$ s'est avéré pertinent dans les applications. L'évolution de k en fonction de la génération n est donnée par la figure 1.8.

Ce dernier principe de scaling donne effectivement de meilleurs résultats sur nos problèmes que le scaling linéaire et sera donc systématiquement utilisé. Dans le cas des fonctions objectifs multi-modes présentant des optimaux quasi-équivalents, cette technique de scaling, en amplifiant les écarts de fitness en fin de convergence, va effectivement favoriser le mode dominant mais aussi masquer les modes sous-optimaux qui peuvent tout de même présenter un intérêt. Le scaling permet donc une bonne exploration de l'espace d'état mais ne favorise pas la répartition des individus sur les différents modes de la fonction objectif.

1.3.3 Sharing

Introduction

L'objectif du sharing est de répartir sur chaque sommet de la fonction à optimiser un nombre d'individus proportionnel à la fitness associée à ce sommet. La figure 1.9 présente deux exemples de répartitions de populations dans le cas d'une fonction à cinq sommets : le premier sans sharing, le second avec sharing.

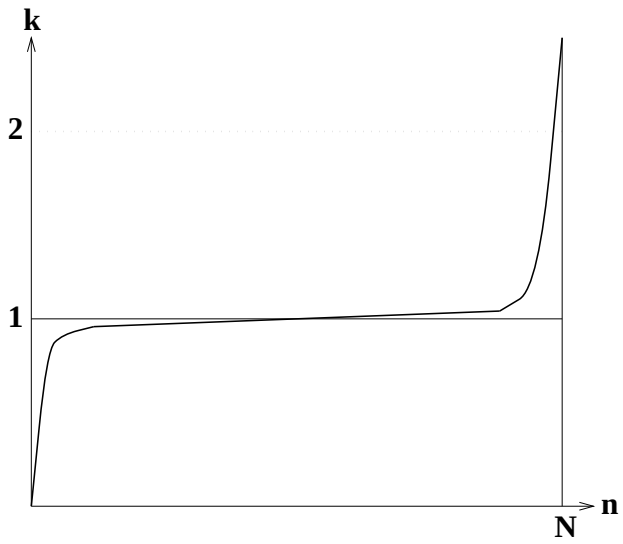


Figure 1.8: Allure de l'évolution de k en fonction des générations

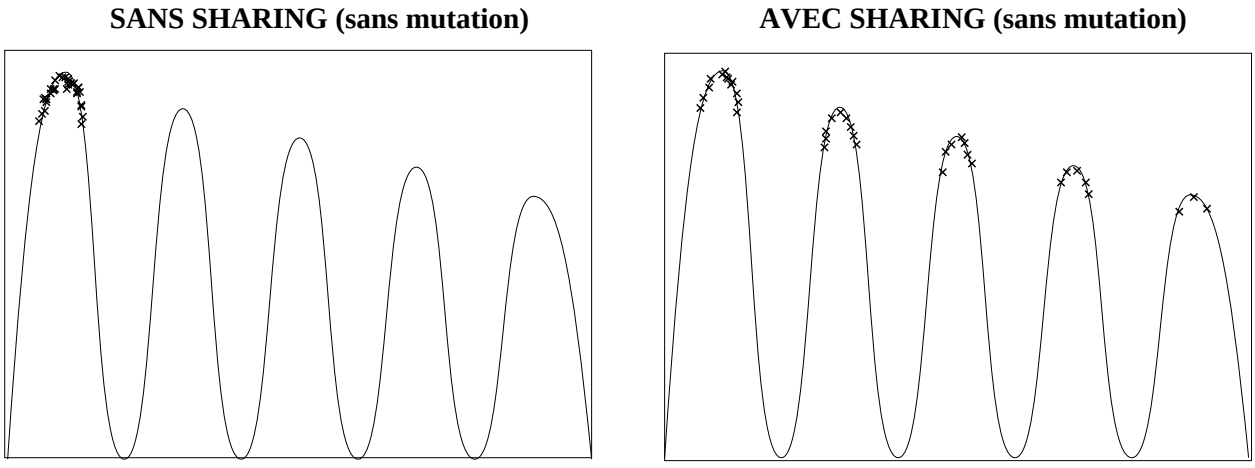


Figure 1.9: Objectif du sharing

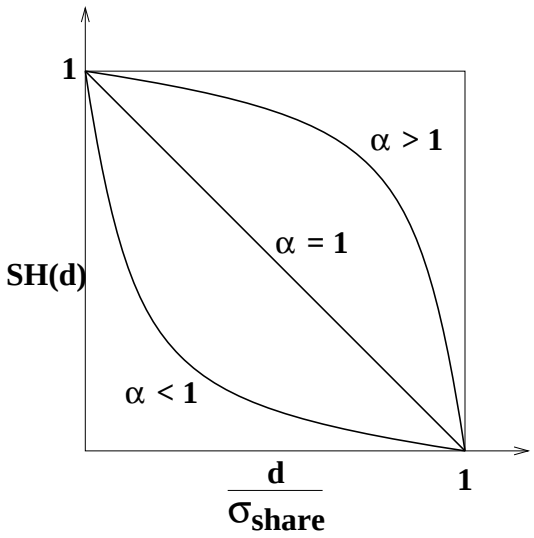


Figure 1.10: Allure de $S(\frac{d}{\sigma_{share}})$

Principe

De la même façon que le scaling, le sharing consiste à modifier la fitness utilisée par le processus de sélection. Pour éviter le rassemblement des individus autour d'un sommet dominant, le sharing pénalise les fitness en fonction du taux d'agrégation de la population dans le voisinage d'un individu. Il requiert l'introduction d'une notion de distance. Dans la pratique, il faut définir une distance indiquant la dissimilarité entre deux individus. Cette distance est alors utilisée pour calculer la nouvelle fitness de la façon suivante :

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

avec

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

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

Le paramètre σ_{share} permet de délimiter le voisinage d'un point et dépend du problème traité. La figure 1.10 donne l'allure de $S(d)$ pour différentes valeurs de α. Suivant la valeur donnée à α le sharing sera plus ou moins efficace. Ainsi pour $\alpha < 1$, on pénalise les groupes très agglomérés.

Dans la pratique ce type de sharing donne effectivement de bons résultats mais au prix de N^2 calculs de distances entre chromosomes à chaque génération pour une population de taille N. Or les algorithmes génétiques induisent une complexité en N sans sharing et le fait de passer en N^2 peut être pénalisant, notamment pour N grand.

Pour réduire ce nombre, on utilise un sharing "clusterisé".

1.3.4 Sharing clusterisé

Pour effectuer ce type de sharing [YG93], on commence par identifier les différents "clusters" d'individus dans la population. Ce dernier utilise deux paramètres d_{min} et d_{max} respectivement pour fusion-

ner des clusters ou en créer de nouveaux. Initialement, chaque individu de la population est considéré comme le centre d'un cluster. On applique alors successivement les deux principes suivants :

- si deux centres sont à une distance inférieure à d_{min}, on fusionne ces derniers dans un cluster unique dont le centre résultant est le barycentre des deux centres initiaux.

- un nouvel individu est agrégé à un cluster si sa distance au centre le plus proche est inférieure à d_{max}. Dans ce cas, on recalcule le centre du cluster global. Sinon, on crée un nouveau cluster contenant ce seul individu.

Ce principe de fusion-agrégation permet d'obtenir un nombre de clusters fluctuant avec la répartition des individus dans l'espace d'état. On applique ensuite le principe de sharing en modifiant les fitness de la façon suivante :

$$f'_i = \frac{f_i}{m'_i};\ m'_i = n_c \left(1 - \left(\frac{d_{ic}}{2d_{max}}\right)^{\alpha}\right);$$

avec

- n_c : nombre d'individus contenus dans le cluster auquel appartient l'individu i.

- α : coefficient de sensibilité.

- d_{ic} : distance entre l'individu i et le centre du cluster c.

On montre que ce type de sharing induit une complexité en $O(N \log N)$[YG93] pour des résultats tout à fait comparables à ceux fournis par le sharing classique. Dans la pratique, on remarque que le réglage des coefficients d_{min} et d_{max} est assez délicat car l'efficacité de ces derniers dépend essentiellement de la connaissance a priori des distances inter-modes dans l'espace d'état, distance qu'il est très difficile d'estimer. Nous présentons dans la section 1.4.3 une technique permettant de calculer automatiquement ces quantités.

1.3.5 Algorithmes génétiques et recuit simulé

Introduction

Les algorithmes génétiques et le recuit simulé étant deux techniques d'optimisation stochastique travaillant sur les mêmes types de problèmes, il est logique de chercher à les associer afin de tirer partie de leurs avantages respectifs. Après plusieurs évaluations de ces deux techniques sur les mêmes problèmes test, on remarque que le recuit simulé converge généralement plus vite vers la solution optimale lorsque le problème est de taille raisonnable. Toutefois, il ne donne qu'une solution dans le cas des problèmes multi-modes, ceci confirme les résultats donnés dans [IR92a]. A l'inverse, les algorithmes génétiques fournissent plusieurs solutions quasi-optimales mais au prix d'un temps de convergence généralement plus long. Il semble alors naturel d'associer ces deux techniques afin d'améliorer la convergence des algorithmes génétiques.

Il y a eu de nombreuses tentatives de fusion entre les algorithmes génétiques et le recuit simulé, les travaux les plus intéressants étant ceux de Mahfoud et de Goldberg [MG92].

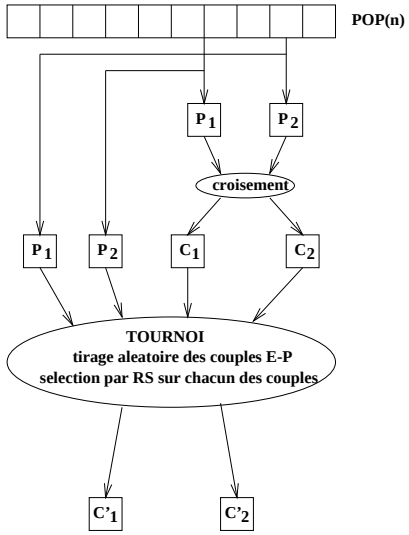


Figure 1.11: Principe du croisement avec recuit simulé

Principe du croisement avec recuit simulé

Pour appliquer ce principe de croisement, on commence par sélectionner deux parents P_1 et P_2 dans la population (voir figure 1.11). On applique ensuite l'opérateur de croisement classique qui génère deux enfants C_1 et C_2. Un tournoi est alors effectué entre les parents et les enfants pour lequel les deux vainqueurs sont sélectionnés par le schéma de recuit suivant. On considère l'individu C_1. On tire ensuite aléatoirement un des deux parents, soit P_i ce parent :

- si C_1 est meilleur que $P_i \Rightarrow C_1$ est sélectionné.
- sinon C_1 est sélectionné avec la probabilité :

$$P = e^{-\left(\frac{|f_{C_1} - f_{P_i}|}{C_n}\right)}$$

où $c(n)$ est un coefficient décroissant en fonction de la génération courante (n).

On fait de même pour C_2 avec le parent restant et l'on détermine ainsi deux individus C'_1 et C'_2.

L'évolution de la variable $C(n)$ se fait de la façon suivante. On utilise un schéma de recuit standard géométrique à un palier de basculement. Pratiquement, on calcule trois "températures" dont les valeurs dépendent de la connaissance des écarts min et max des fitness de la population initiale.

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

où $\Delta f_{min}, \Delta f_{max}$ représentent les écarts minimum et maximum des fitness de la population initiale. Le schéma géométrique fait évoluer la température courante de la façon suivante :

$$\begin{cases} C_0 = C_s \\ C_{n+1} = \alpha_1 C_n \text{ pour } C_s > C_n > C_x; \\ C_{n+1} = \alpha_2 C_n \text{ pour } C_x > C_n > C_f; \end{cases}$$

avec $0 < \alpha_1 < \alpha_2 < 1$.

Pour chaque palier, on calcule le nombre d'itérations de stabilisation à l'aide des formules :

$$N_1 = \frac{ln\left(\frac{C_x}{C_s}\right)}{ln\,\alpha_1} \quad N_2 = \frac{ln\left(\frac{C_f}{C_x}\right)}{ln\,\alpha_2}$$

Ces deux formules, nous permettent de calculer le nombre total de générations pour un problème donné.

Ce même principe de recuit simulé a été essayé sur l'opérateur de mutation en faisant un schéma de recuit entre l'individu muté et l'individu d'origine mais il ne produit pas l'effet escompté. En effet, on peut supposer que dans ce cas le recuit simulé réduit le brassage de la population provoqué par la mutation en limitant l'espace exploré aux zones qui améliorent statistiquement la fitness en "interdisant" les domaines qui la dégradent. L'exploration de du domaine admissible est fragilisée. Ce constat sur les problèmes que nous avons testés ne permet pas de généraliser.

1.3.6 Recherche multi-objectifs

Introduction

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

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

Technique employée

La technique que nous employons dérive directement des travaux de Jeffrey Horn et Nicholas Nafpliotis ([HN93]). Le principal changement induit concerne le processus de sélection : en multi-objectifs, comment va t-on décider qu'un individu est meilleur qu'un autre[3] ? On introduit alors une variante de la notion de dominance que l'on définit ainsi : on peut par exemple décider que l'élément E_i domine E_j si le nombre des valeurs contenues dans son vecteur d'adaptation qui sont supérieures aux valeurs correspondantes dans E_j dépasse un certain seuil. A partir de là, la technique proposée pour effectuer la sélection est simple : on tire deux individus au hasard ainsi qu'une sous-population[4] à laquelle ils n'appartiennent pas et qui va servir à les comparer.

Trois cas se présentent alors :

- si le premier élément domine tous ceux de la sous-population et que ce n'est pas le cas pour le second alors le premier sera sélectionné.
- inversement si seul le second domine l'ensemble de la sous-population alors c'est lui qui sera conservé.

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

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

- restent maintenant deux possibilités : soit les deux sont dominés, soit les deux dominent. On ne peut se prononcer sans ajouter un autre test, c'est pourquoi dans ce cas il est fait usage d'un nouveau type de sharing qui opère sur l'espace objectif.

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

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

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

f_i^k désignant la k-ième composante du vecteur adaptation de l'élément i. Le paramètre p permet de faire varier la forme et la taille du voisinage. A l'intérieur des voisinages ainsi définis dans l'espace objectif, il suffit de compter le nombre d'individus pour favoriser les zones les moins denses et de cette façon maintenir la diversité de la population. Ainsi la figure 1.12 montre comment les voisinages sont choisis autour des individus de la région de Pareto lorsque ceux-ci ne peuvent être départagés sans sharing.

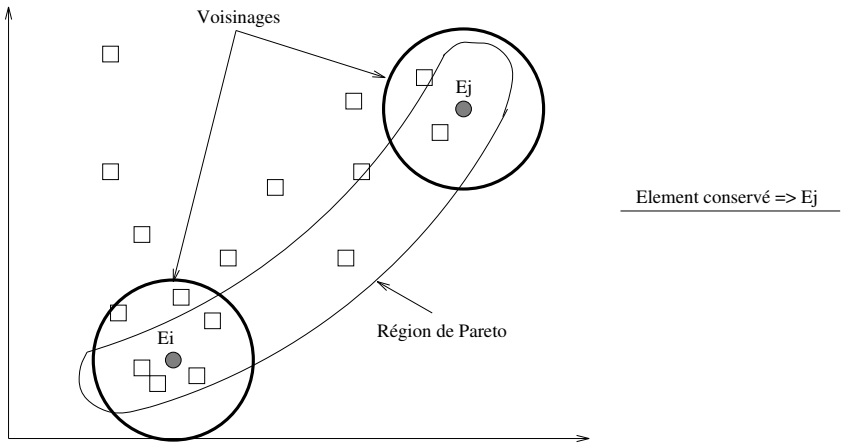


Figure 1.12: Surface de Pareto et voisinages

1.3.7 Association des AG avec des méthodes locales

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

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

l'algorithme génétique ne reste pas le meilleur élément après amélioration par l'algorithme local de tous les meilleurs éléments de clusters.

Une autre technique consiste à utiliser un algorithme local associé à l'algorithme génétique pour calculer la fitness d'un élément. On peut, par exemple dans un espace fortement combinatoire, rechercher avec l'algorithme génétique les zones intéressantes de l'espace en générant les éléments, l'adaptation de chaque élément étant calculée par un programme d'optimisation local (linéaire par exemple). Un exemple de ce type est donné dans la section 3.7.

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

1.4 Autres améliorations

Les raffinements décrits dans cette section ont été développés au sein de notre équipe, et sont donc, en principe, originaux. Notons que l'algorithme génétique que nous avons développé implante l'intégralité des techniques présentées dans la section précédente, celles présentées dans cette section, ainsi que les différents types de parallélismes décrits dans la section 1.5. Nous pensons qu'il s'agit d'un des programmes les plus efficaces actuellement.

1.4.1 Croisement adapté pour les fonctions partiellement séparables

Introduction

Dans beaucoup de problèmes d'optimisation, la fonction à optimiser dépend de nombreuses variables, et peut se décomposer comme somme de sous-fonctions prenant en compte une partie des variables seulement ([GT82]). Les algorithmes génétiques s'avèrent être de bons outils d'optimisation globale en raison de leur capacité à sortir des minima locaux. Néanmoins, leurs performances sont souvent pénalisées par le caractère très aléatoire des opérateurs de croisement et de mutation. Pour remédier à ce phénomène, certains évolutionnistes utilisent des heuristiques qui permettent de favoriser les "bons" croisements ou les "bonnes" mutations et d'éviter les "mauvaises". Nous présentons ici un opérateur de croisement efficace, utilisable dans des problèmes où la fonction à optimiser peut se décomposer en somme de sous-fonctions positives ne dépendant pas de toutes les variables du problème.

Après une présentation formelle du type de problèmes auxquels s'adresse cette méthode, nous présenterons l'opérateur de croisement. Une illustration de l'efficacité de cette méthode sera ensuite proposée au travers de plusieurs exemples simples comportant jusqu'à une cinquantaine de variables. Enfin, [DAN94] propose une application de cet outil au classique problème du Voyageur de Commerce, largement étudié dans la littérature, aussi bien pour son intérêt pratique qu'en raison de sa grande complexité [Bra90, GL85, GGRG85, HGL93, OSH89, WSF89]. En effet, pour tester la performance d'un algorithme d'optimisation globale, le problème du Voyageur de commerce, NP-Complet, permet d'offrir de nombreux exemples de très grande taille comportant beaucoup d'optima locaux dont on connaît les solutions.

Problèmes concernés

Il s'agit de problèmes ou fonctions partiellement séparables, à savoir des problèmes où la fonction F à optimiser dépend de plusieurs variables x_1, x_2, .., x_n, et peut se décomposer en somme de fonctions F_i positives ne dépendant pas de toutes les variables. On remarquera que bon nombre de fonctions multiplicatives peuvent se transformer en fonction additives grâce à des transformations simples. On s'intéresse donc aux fonctions pouvant s'écrire :

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

Soit E_k l'ensemble des indices i tels que x_k soit une variable de F_i :

$$E_k \quad = \quad \{i / x_k \in variables\ de\ F_i\}$$

Pour définir notre opérateur de croisement, nous allons introduire pour chaque variable x_k sa "fitness locale" $G_k(x_1, x_2, .., x_n)$ définie comme suit :

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

La fitness locale associée à une variable isole sa contribution. Le but de cette section est de montrer comment utiliser cette fitness locale pour définir un opérateur de croisement permettant de réduire la taille des populations et de converger plus rapidement vers la solution optimale. Ces opérateurs sont particulièrement efficaces lorsqu'on les combine avec le sharing.

L'opérateur de croisement

L'opérateur de croisement consiste, à partir de deux chromosomes parents, à créer deux nouveaux chromosomes.

Le codage que nous adopterons pour appliquer notre croisement consiste à représenter le chromosome par la liste brute des variables du problème. S'il s'agit de bits, nous aurons une liste de bits, s'il s'agit de variables réelles, nous aurons une liste de réels. On pourra éventuellement avoir un panachage de différents types de variables. L'idée intuitive est la suivante : dans le cas d'un problème complètement séparable, optimiser le problème global peut se réduire à l'optimisation de chaque variable séparément. Nous essayons d'adapter ce raisonnement au cas de fonctions partiellement séparables. La technique que nous proposons consiste donc à retenir pour chaque variable, lors de la conception des fils, celle des deux parents qui a la meilleure fitness locale, ceci à un facteur Δ près. Par exemple, dans le cas où l'on cherche un minimum pour F, pour la $k^{\text{ième}}$ variable, si :

$$G_k(pere_1) \quad < \quad G_k(pere_2) - \Delta$$

alors le fils 1 se verra affecter la variable x_k du père 1. Si par contre :

$$G_k(pere_1) \quad > \quad G_k(pere_2) + \Delta$$

il se verra affecter la variable x_k du père 2. Si enfin :

$$\|G_k(pere_1) - G_k(pere_2)\| \quad \leq \Delta$$

la variable x_k du fils 1 sera choisie aléatoirement.

Lors du croisement, on souhaite créer deux fils à partir de deux parents. Si l'on applique la même stratégie pour le fils 2 que celle décrite ci-dessus pour le fils 1, les deux fils risquent de se ressembler fortement, surtout si Δ est faible. Ceci n'est pas souhaitable. On peut donc pour le deuxième fils choisir une autre valeur de Δ ce qui permet d'être plus ou moins déterministe, ou alors, comme pour les techniques de croisement classiques choisir le complémentaire du fils 1, à savoir affecter au fils 2 la variable des deux pères non affectée au fils 1.

On peut remarquer qu'il est facile d'introduire un opérateur de mutation qui s'appuie sur le même principe et modifie avec une plus forte probabilité les variables ayant une mauvaise fitness locale.

1.4.2 Mutation adaptative

Un des gros inconvénients de l'algorithme génétique est sa mauvaise convergence locale. Il est possible de raffiner le comportement des AGs en utilisant une technique que nous qualifions de "mutation adaptative" dans tous les problèmes à variable réelle.

Le principe est le suivant : dans les problèmes à variable réelle, l'opérateur de mutation consiste généralement à ajouter un bruit gaussien à l'élément de population concerné. Le problème est de bien choisir ce bruit gaussien. S'il est trop petit, les déplacements dans l'espace sont insuffisants en début de convergence, et l'algorithme peut rester bloqué dans un optimum local. Si le bruit est trop fort, l'AG trouvera certes une zone contenant l'optimum, mais sera incapable de converger localement. Il s'agit donc de modifier le bruit au fil des générations. Pour ce faire, on calcule l'écart moyen pour chacune des coordonnées entre le meilleur élément à la génération n et tous les autres éléments : ce nombre est alors utilisé comme écart type du bruit gaussien sur la coordonnée en question à la génération $n+1$.

L'efficacité de cette méthode est démontrée dans la section 2.1.1.

1.4.3 Clustering adaptatif

Nous avons vu dans la section 1.3.4 une technique de sharing clusterisé permettant de diminuer la complexité du sharing traditionnel. Cependant, comme nous l'avons souligné, les quantités d_{min} et d_{max} sont difficiles à calculer à priori. Nous présentons ici un algorithme permettant de calculer automatiquement ces quantités de façon adaptative pendant l'exécution de l'algorithme génétique.

Au départ on initialise le processus avec $d_{moy} = 0$ et $\Delta = 2.0$. Ensuite à chaque génération on utilise ce qui suit :

1. Calcul de d_{min} et d_{max} :
$$\begin{cases} d_{max} = \frac{d_{moy}}{\Delta} \\ d_{min} = \frac{d_{max}}{3} \end{cases}$$

2. Évaluation de la distance moyenne des individus par rapport aux centroïdes des clusters :
$$d_{moy} = \frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{N_c} d(E_i, C_j) \right)}{n N_c}$$
si l'on note C_j le centre du groupe j et N_c le nombre total de clusters.

3. on compte ensuite le nombre de clusters dont le meilleur individu a une fitness supérieure à un certain pourcentage[5] de celle du meilleur élément de la population , on le note N_{opt}

[5] Egal au taux de sharing.

4. on met à jour le paramètre Δ en utilisant la règle :

$$\begin{cases} \Delta = \Delta * 1.05 \text{ si } \frac{N_{opt}}{N_c} > S_1 \text{ et } \Delta < 100 \\ \Delta = \Delta * 0.95 \text{ si } \frac{N_{opt}}{N_c} < S_2 \text{ et } \Delta > 1 \\ \Delta \text{ reste inchangé dans les autres cas} \end{cases}$$

Les valeurs S_1 et S_2 désignent deux seuils : si beaucoup de clusters sont suffisamment bons (c'est à dire quand leur proportion dépasse S_1) alors on diminue d_{min} et d_{max} pour accroître le nombre total de groupes et ainsi chercher d'autres optima. Au contraire, s'ils sont peu nombreux alors on va diminuer la quantité de clusters en augmentant les deux distances afin de rechercher des zones intéressantes de l'espace d'état.
Les valeurs utilisées habituellement pour les seuils sont $S_1 = 0.85$ et $S_2 = 0.75$.

Cette technique donne d'excellents résultats pour rechercher les optima multiples de fonctions réelles.

1.5 Parallélisme

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

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

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

1.5.1 Parallélisme par îlots

L'idée ici est de faire fonctionner plusieurs algorithmes génétiques en parallèle avec une population *réduite* pour chacun. Le programme maître lance N occurrences du programme d'algorithmes génétiques appelées esclaves sur des machines différentes en passant à chacune les paramètres nécessaires à leur bon fonctionnement comme le montre la figure 1.13.

Ensuite chaque processus fait évoluer sa population indépendamment jusqu'à ce qu'il décide (selon une probabilité fixée à l'avance) de rassembler ses meilleurs individus pour en transmettre une certaine quantité (proportionnelle à la taille de la population) à une autre machine de son choix. La machine réceptrice intègre alors ces nouveaux éléments dans sa propre population en éliminant les moins bons de ses individus. L'intérêt du parallélisme par îlots est qu'il offre la possibilité de travailler sur de grandes populations (n_0) tout en donnant des résultats dans un temps raisonnable puisque

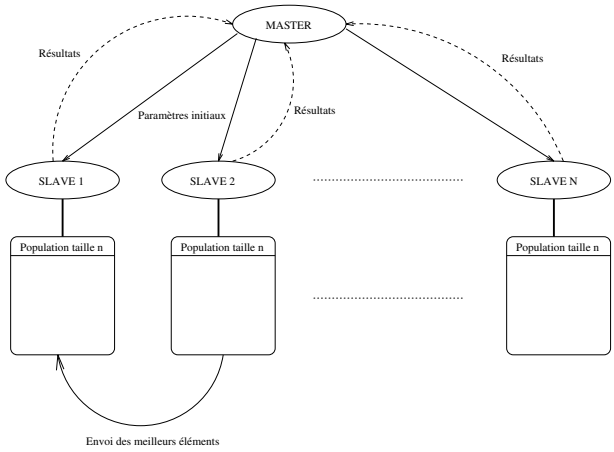


Figure 1.13: Principe de fonctionnement du parallélisme par îlots

la durée nécessaire est à peu de choses près celle qu'il faudrait pour une population de taille $\frac{n_0}{N}$ si N est le nombre d'ordinateurs disponibles et si l'on néglige les temps de communication.

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

Des tests ont été faits par Yann LeFablec [LeF95]pour déterminer l'efficacité de la méthode (voir figure 1.14). Il ne faut s'attacher qu'à la forme générale de la courbe, dans la mesure où les charges locales des machines peuvent modifier de façon sensible les résultats. L'efficacité de la méthode est cependant largement mise en évidence.

1.5.2 Parallélisation des calculs

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

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

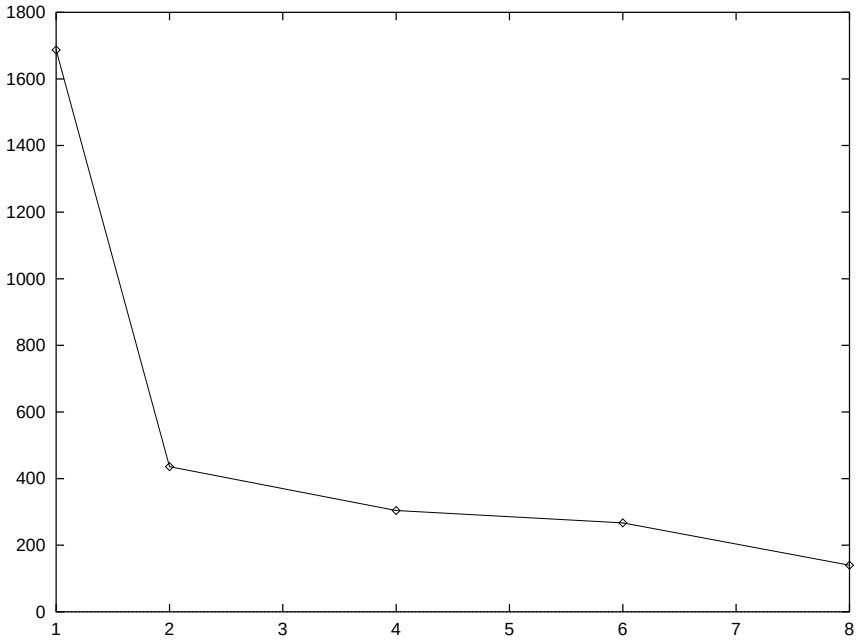


Figure 1.14: Évolution des temps de calcul

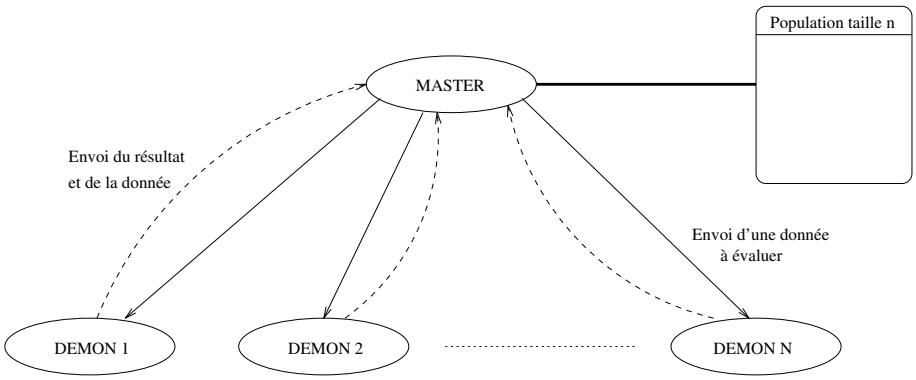


Figure 1.15: Principe de fonctionnement de la parallélisation des calculs

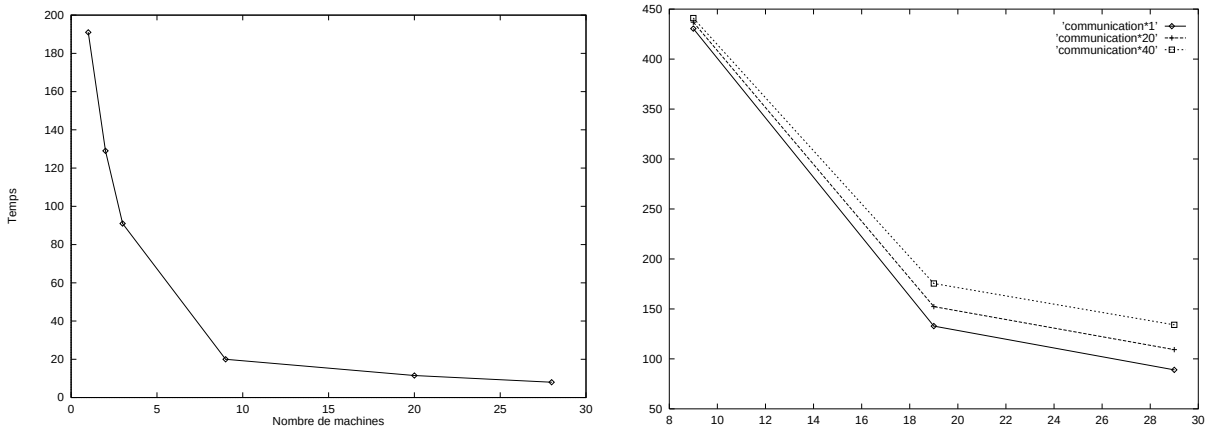


Figure 1.16: Évolution du temps de calcul

La courbe 1.16 montre comment évolue le temps de calcul en fonction du nombre de machines et du temps de communications. On constate que plus on utilise de machines, plus la pénalisation due aux communications est forte.

1.5.3 Conclusion

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

1.6 Résultats théoriques sur les algorithmes binaires

Historiquement, les algorithmes génétiques binaires sont les plus anciens et ont donc été les plus étudiés sur le plan théorique. Goldberg [Gol89a] a largement développé cette théorie résumée dans [AS92]. Nous allons tout d'abord les étudier avant de présenter les modélisations théoriques basées sur les chaînes de Markov.

1.6.1 Définitions fondamentales

Définition 1.1 (Séquence) *On appelle séquence A de longueur $l(A)$ une suite $A = a_1 a_2 \dots a_l$ avec $\forall i \in [1, l], a_i \in V = \{0, 1\}$.*

En codage binaire, les chromosomes sont des séquences.

Définition 1.2 (Schéma) *On appelle schéma H de longueur l une suite $H = a_1 a_2 \dots a_l$ avec $\forall i \in [1, l], a_i \in V^+ = \{0, 1, *\}$.*

Une $*$ en position i signifie que a_i peut être indifféremment 0 ou 1.

Définition 1.3 (Instance) *Une séquence $A = a_1 \dots a_l$ est une instance d'un schéma $H = b_1 \dots b_l$ si pour tout i tel que $b_i \neq *$ on a $a_i = b_i$.*

Ainsi, $H = 010 * 0101$ est un schéma et les séquences 01000101 et 01010101 sont des instances de H (ce sont même les seules).

Définition 1.4 (Position fixe, position libre) *Soit un schéma H. On dit que i est une position fixe de H si $a_i = 1$ ou $a_i = 0$. On dit que i est une position libre de H si $a_i = *$.*

Définition 1.5 (Ordre d'un schéma) *On appelle ordre du schéma H le nombre de positions fixes de H. On note l'ordre de H $o(H)$.*

Par exemple, le schéma $H = 01 * *10 * 1$ a pour ordre $o(H) = 5$, le schéma $H' = * * * * * 101$ a pour ordre $o(H') = 3$. On remarquera qu'un schéma H de longueur $l(H)$ et d'ordre $o(H)$ admet $2^{(l(H)-o(H))}$ instances différentes.

Définition 1.6 (Longueur fondamentale) *On appelle longueur fondamentale du schéma H la distance séparant la première position fixe de H de la dernière position fixe de H. On note cette longueur fondamentale $\delta(H)$.*

Ainsi, le schéma $H = 1 * *01 * **$ a pour longueur fondamentale $\delta(H) = 5 - 1 = 4$, le schéma $H' = 1******1$ a pour longueur fondamentale $\delta(H') = 8-1 = 7$, et pour le schéma $H'' = **1*****$ nous avons $\delta(H'') = 3 - 3 = 0$.

Définition 1.7 (Adaptation d'une séquence) *On appelle adaptation d'une séquence A une valeur positive que nous noterons $f(A)$.*

f est la fonction objectif ou fitness du problème à résoudre.

Définition 1.8 (Adaptation d'un schéma) *On appelle adaptation d'un schéma H la valeur*

$$f(H) = \frac{\sum_{i=1}^{2^{(l(H)-o(H))}} f(A_i)}{2^{(l(H)-o(H))}}$$

où les A_i décrivent l'ensemble des instances de H, c'est-à-dire la moyenne des adaptations de ses instances.

1.6.2 Effets de la reproduction

Soit un ensemble $S = \{A_1, \ldots, A_i, \ldots, A_n\}$ de n séquences de bits tirées aléatoirement. Durant la reproduction, chaque séquence A_i est reproduite avec une probabilité :

$$p_i = \frac{f(A_i)}{\sum_{i=1}^{n} f(A_i)}$$

Supposons qu'il y ait à l'instant t un nombre $m(H,t)$ de séquences représentant le schéma H dans la population S. A l'instant $t+1$, statistiquement, ce nombre vaut :

$$m(H, t+1) = m(H,t).n.\frac{f(H)}{\sum_{i=1}^{n} f(A_i)}$$

Posons

$$\overline{f_t} = \frac{\sum_{i=1}^{n} f(A_i)}{n}$$

$\overline{f_t}$ représente la moyenne de l'adaptation des séquences à l'instant t. La formule précédente devient :

$$m(H, t+1) = m(H, t)\frac{f(H)}{\overline{f_t}}$$

Posons $c_t(H) = \frac{f(H)}{\overline{f_t}} - 1$. On obtient alors :

$$m(H, t+1) = (1 + c_t(H))m(H, t)$$

Il est donc clair qu'un schéma, dont l'adaptation est au-dessus de la moyenne, voit son nombre de représentants augmenter, suivant une progression qui est de type géométrique si nous faisons l'approximation que $c_t(H)$ est constant dans le temps. Alors :

$$m(H, t) = (1 + c(H))^t.m(H, 0)$$

Si la reproduction seule était en jeu, les schémas forts élimineraient très rapidement les schémas faibles.

1.6.3 Effets des croisements

Nous allons nous intéresser à la probabilité de survie $p_s(H)$ d'un schéma H lors d'une opération de croisement. Par exemple, soit le schéma $H = **10*1**$. Supposons qu'une instance de ce schéma soit croisée avec une autre instance. Quelle est la probabilité pour que la séquence résultante soit encore une instance de H ? Il est impossible de répondre exactement à la question, tout au plus peut-on donner une borne inférieure de cette valeur. Il est clair que H ne sera pas détruit si le site de croisement qui est tiré au sort est inférieur à 3 (avant le premier 1) ou s'il est supérieur à 6 (après le dernier 1)[6].

On voit donc immédiatement qu'une borne inférieure de la probabilité de détruire un schéma H est $\delta(H)/(l-1)$. Donc la probabilité de survie dans un croisement est $1 - \delta(H)/(l-1)$. Si d'autre part on ne croise qu'une fraction p_c de séquences dans une population donnée, la probabilité de survie est donnée par :

$$p_s \geq 1 - p_c\frac{\delta(H)}{l-1}$$

De ce résultat et des résultats précédents découle la loi d'évolution d'une population[7] :

$$m(H, t+1) \geq m(H, t)(1 + c_t(H))(1 - p_c\frac{\delta(H)}{l-1})$$

1.6.4 Effets des mutations

Soit p_m la probabilité de mutation d'un bit dans une séquence. Dans un schéma H, seules les positions fixes peuvent être détruites. La probabilité de survie d'un bit étant $1 - p_m$, la probabilité de survie d'un schéma H contenant $o(H)$ positions fixes est $(1 - p_m)^{o(H)}$. La probabilité de mutation étant supposée petite devant 1, un développement limité au premier ordre donne une probabilité de survie égale à $1 - o(H)p_m$.

L'équation finale s'écrit donc :

$$m(H, t+1) \geq m(H, t)(1 + c_t(H))(1 - p_c\frac{\delta(H)}{l-1} - o(H)p_m)$$

[6] Dans tous les autres cas, H peut être (ou ne pas être) détruit.

[7] Les croisements et les mutations se font sur la population reproduite, et non sur la population initiale.

1.6.5 Conclusion sur le codage binaire

Des calculs précédents découlent deux résultats :

- les schémas dont la longueur fondamentale est petite sont plus favorisés que les autres, lors de la génération d'une nouvelle population.

- les schémas dont l'ordre est petit sont plus favorisés que les autres, lors de la génération d'une nouvelle population.

Ces résultats conditionnent la qualité du codage des données. En effet, les schémas qui codent les données "intéressantes" pour le problème considéré doivent avoir un ordre et une longueur fondamentales faibles[8], alors que les données "sans intérêt" doivent être codées par des schémas qui ont un ordre et une longueur fondamentale élevés.

Les résultats présentés ci-dessus ne sont qu'une introduction à la théorie des schémas, il existe de nombreux approfondissements. On trouvera dans les références suivantes les principaux modèles théoriques sur la théorie des schémas et ses extensions : [Gol89c, Vos91, BM93, Gol89b, BG87, CS88, BM94, DM92, SGE91]

1.7 Modélisation par chaîne de Markov

Cette dernière approche est la plus satisfaisante tant sur le plan mathématique, que sur celui de la modélisation, les différents opérateurs étant présentés comme "perturbant" un processus Markovien représentant la population à chaque étape. Ici encore il apparaît que seul l'opérateur de mutation est important, le croisement pouvant être totalement absent.

Les principaux résultats asymptotiques portant directement sur les algorithmes génétiques, ont été développés par R. Cerf [Cer94] sur la base des travaux de Catoni [Cat90] et de Trouvé [Tro93]. Ces travaux se fondent sur la théorie des petites perturbations aléatoires d'un processus dynamique de type Markovien. Plus particulièrement, la théorie de Freidlin et Wentzell [FW83] constitue la pierre d'angle de ces études. Nous donnons ici, quelques uns des résultats de la dynamique des algorithmes génétiques développés par Cerf. Nous les présentons simplifiés et dans un cadre restreint, laissant le lecteur intéressé se reporter à la difficile lecture des références citées.

Nous travaillerons sur la base d'un codage binaire, P représentant le nombre de bits utilisés pour le codage. La fonction d'évaluation, f est définie sur l'espace $E = \{0,1\}^P$ à valeurs dans $\Re^+$. Le problème est donc de localiser l'ensemble des maxima globaux de f , ou, à défaut, de trouver rapidement et efficacement des régions de l'espace où se situent ces maxima.

Comme nous l'avons vu, l'algorithme génétique est un algorithme stochastique itératif qui opère sur des ensembles de points, et qui est bâti à l'aide de trois opérateurs: mutation, croisement et sélection, que nous présentons plus formellement à présent.

1.7.1 Description rapide d'un algorithme génétique

Soit N la taille (fixe) de la population, notons X_k la population de la génération k : il s'agit d'une matrice $X_k = (X_k^1, X_k^2, .. X_k^N)$ de E^N dont les N éléments sont des chaînes de bits (chromosomes)

[8] Les données intéressantes sont bien entendu les données qui sont proches de la solution optimale. Un bon codage des données implique donc d'avoir une *idée* de la forme de l'optimum.

de taille P. Le passage de la génération k à la génération $k+1$, c'est à dire de X_k à X_{k+1} se décompose en trois étapes :

$$X_k \overset{Mutation}{\longrightarrow} Y_k \overset{Croisement}{\longrightarrow} Z_k \overset{Selection}{\longrightarrow} X_{k+1}$$

Chacune de ces étapes peut être modélisée formellement.

Mutation $X_k \longrightarrow Y_k$

L'opérateur considéré ici est le suivant : pour chaque composante de chaque élément X_k^i, une variable de Bernouilli de paramètre P_m est tirée indépendamment et suivant le résultat l'élément binaire examiné est changé ou non. (0 est changé en 1 et 1 en 0).

La probabilité P_m de mutation doit être préalablement choisie et est généralement faible.

Comme nous le verrons par la suite, cet opérateur joue un rôle clé dans la convergence de l'algorithme génétique.

Croisement $Y_k \longrightarrow Z_k$

L'opérateur étudié ici est l'opérateur à un point (slicing crossover). Ici encore, la probabilité de croisement P_c est fixée initialement. Pour construire la population Z_k, $N/2$ couples sont formés à partir de la population Y_k (par exemple en appariant les individus consécutifs de Y_k, ou bien en choisissant au hasard et uniformément des individus dans Y_k). Pour chaque couple, une variable de Bernoulli de paramètre P_c est tirée pour décider si le croisement a lieu. Si c'est le cas, un site de coupure est tiré au hasard, et les segments finaux des deux chromosomes sont échangés.

Une nouvelle paire d'individus est ainsi obtenue (identique à l'ancienne s'il n'y a pas eu de croisement) et est stockée dans la population Z_k. En général, le paramètre P_c est choisi grand.

Remarquons que les opérateurs de mutation et de croisement ne font pas intervenir la fonction f, ce sont des opérateurs stochastiques d'exploration. C'est le troisième et dernier opérateur, la sélection, qui guide la population vers les valeurs élevées de la fonction f.

Sélection $Z_k \longrightarrow X_{k+1}$

Les N individus de la population X_{k+1} sont obtenus après sélection des individus de Z_k. On conserve ainsi les "meilleurs" individus de Z_k, indépendamment à l'aide d'une distribution de probabilité qui favorise les individus de Z_k les mieux adaptés.

Le choix le plus fréquent est l'unique distribution telle que la probabilité d'un individu soit proportionnelle à son adaptation, i.e la probabilité de sélection de l'individu Z_k^i est :

$$P_i = P(Z_k^i) = \frac{f(Z_k^i)}{\sum_{j=1}^{N} f(Z_k^j)}$$

En tirant les individus dans la population Z_k conformément aux probabilités P_i, on constitue la nouvelle génération X_{k+1}.

1.7.2 Modélisation

La présentation rapide des opérateurs nous permet de modéliser la suite des $(X_k)_{k\in\mathcal{N}}$ en une chaîne de Markov, d'espace d'états $E = \left(\{0,1\}^P\right)^N$. L'algorithme génétique ne doit donc pas être interprété comme une procédure d'optimisation mais plutôt comme une marche aléatoire dans l'espace d'état, attirée vers les fortes valeurs de f .

La propriété première de cette formalisation est que la loi de X_k est déterminée de manière unique par :

- la loi de la génération initiale X_0
- le mécanisme de transition de X_k à X_{k+1}, mécanisme scindé en trois étapes détaillée précédemment.

Ce mécanisme de transition possède toutefois des propriétés essentielles qui font l'intérêt et la puissance de cette formalisation, (voir [Cer94]) :

- Il est homogène (c'est à dire indépendant de la génération, k, considérée)
- Il est irréductible, (la probabilité de joindre deux points quelconques de l'espace d'état, en un nombre fini de générations est non nulle) soit :

  $$\forall x, y \in E \qquad \exists r \in \mathcal{N} \qquad P\left[X_{k+r} = y \mid X_k = x\right] > 0$$

  Le mécanisme permet donc d'explorer tout point de l'espace d'état, avec une probabilité non nulle.
- Il est apériodique, cette hypothèse n'est cependant pas fondamentale.

Ces propriétés permettent de conclure à l'ergodicité de la chaîne de Markov, et à l'existence d'un processus limite.

Théorème 1.1 *Une chaîne de Markov homogène irréductible apériodique d'espace d'états fini est ergodique et possède une unique mesure de probabilité stationnaire ou invariante.*

Cette mesure stationnaire correspond à la loi régissant l'équilibre du processus, elle est définie , pour tout y, comme :

$$\mu(y) = \lim_{k \to infty} P\left[X_k = y \mid X_0 = x\right]$$

Nous savons également que tout élément de l'espace d'état est de probabilité non nulle pour cette mesure.

Toutefois, si ce résultat nous permet de savoir qu'il existe une dynamique de fond de l'algorithme génétique, il nous reste à en déterminer les propriétés, l'influence des opérateurs (et des paramètres associés) qui jouent un grand rôle dans le processus.

Pour cela nous introduisons les notations suivantes :

Si $x = (x_1, ..., x_N)$ est un élément de E^N et i un point de E, nous noterons :

$$\begin{aligned}
\widehat{f(x)} &= \widehat{f}(x_1, ..., x_N) = \max\left\{f(x_i) : 1 \leq i \leq N\right\} \\
\widehat{x} &= \left\{x_k \in \arg\max f(x)\right\} \\
[x] &= \left\{x_k \ : \ 1 \leq k \leq N\right\}
\end{aligned}$$

De manière générale, les lettres z, y, z, u, v.. désignent des populations, i.e. des éléments de E^N, et les lettres i, j des points de E.

Processus de fond (X_k^{infty})

C'est à partir de ce processus de fond qu'est reconstitué l'algorithme génétique en étudiant ses perturbations aléatoires par les différents opérateurs. Il est défini comme processus limite, lorsque les perturbations ont disparu. C'est également une chaîne de Markov sur E^N dont le mécanisme de transition est très simple puisque correspondant à la situation limite suivante :

Les N composantes de X_{k+1}^{infty} sont choisies indépendamment et suivant la loi uniforme sur l'ensemble $\widehat{X_k^{infty}}$.

- Les individus dont l'adaptation n'est pas maximale en k, sont éliminés et n'apparaissent pas dans la génération $k+1$,
- Les individus dont l'adaptation est maximale, ont des chances de survies égales

Cette chaîne est tout d'abord piégée dans l'ensemble S des populations ayant la même adaptation (ou ensemble des population d'*équi-adaptation*),

$$S = \left\{ x = (x_1, ..., x_N) \in E^N \quad : \ f(x_1) = f(x_2) = \cdots = f(x_N) \right\}$$

Cette population représente les *attracteurs* de la chaîne (voir 1.7.3 plus loin), puis elle est absorbée par une population uniforme, de sorte que :

$$\forall x \in E^N \qquad P\left[\exists x_i \in \widehat{x} \quad \exists K \quad \forall k \geq K \quad X_k^{infty} = x_i \mid X_0^{infty} = x_{ini} \right] = 1$$

Lorsque la population est devenue uniforme et en l'absence ici de perturbations, il ne se passe plus rien.

Ceci peut également se traduire en définissant les populations uniformes comme les *états absorbants* de la chaîne X_k^{infty}. Nous allons maintenant étudier la situation où ce processus est perturbé.

Processus perturbé (X_k^l)

La modélisation proposée par Cerf, part du processus de fond (X_k^{infty}),décrit ci-dessus, qui est perturbé aléatoirement, les perturbations sont indicées par le paramètre l. La chaîne de Markov (X_k^{infty}) devient donc une suite de chaînes de Markov (X_k^l), dont le mécanisme de transition est donné par la succession des transformations générées par les opérateurs.

$$X_k^l \quad \overset{Mutation}{\longrightarrow} \quad U_k^l \quad \overset{Croisement}{\longrightarrow} \quad V_k^l \quad \overset{Selection}{\longrightarrow} \quad X_{k+1}^l$$

Il nous faut pour cela modéliser précisément les opérateurs.

Mutation $X_k^l \longrightarrow U_k^l$

Les mutations sont définies comme de petites perturbations aléatoires indépendante des individus, de la population X_k^l. Il est assez naturel d'introduire la probabilité $p_l(i,j)$ de transition[9] de mutation entre les points i et j de E, comme un noyau Markovien p_l.

Trivialement on a :

$$\forall i \in E \qquad \sum_{j \in E} p_l(i,j) = 1$$

[9] C'est la probabilité $P_l(i,j)$ pour un point i de E de se transformer par mutation en un point j de E

Sur la chaîne X_k^l, la probabilité de transition entre les points x et u de E^N est :

$$P\left[U_k^l = u \mid X_k^l = x\right] = p_l(x_1, u_1) \cdot p_l(x_2, u_2) \cdots p_l(x_N, u_N)$$

Plus précisément et afin d'analyser la dynamique de (X_k^l) lorsque l tend vers l'infini, nous reportons ici les hypothèses sur le mode et la vitesse de convergence des probabilités de transition. Pour cela nous supposons l'existence d'un noyau irréductible, α, sur E, i.e. : $\forall\ i, j \in E, \exists\ i_o,\ i_1, \cdots,\ i_r$ (c'est à dire un chemin dans E) tels que $i_0 = i$ et $i_r = j$ tels que :

$$\prod_{0 \leq s \leq r-1} \alpha(i_k, i_{k+1}) > 0$$

L'hypothèse d'irréductibilité du noyau α est essentielle, elle assure que tout point de l'espace est potentiellement visitable.

La vitesse de convergence du noyau p_l, est caractérisée par le réel positif $\mathbf{a}$, tel que p_l admette le développement suivant :

$$\forall i, j \in E \qquad \forall s \qquad p_l(i,j) = \begin{cases} \alpha(i,j) \cdot l^{-\mathbf{a}} + o(l^{-s}) & si\ i \neq j \\ \\ 1 - \alpha(i,j) \cdot l^{-\mathbf{a}} + o(l^{-s}) & si\ i = j \end{cases} \tag{1.1}$$

La condition de positivité de $\mathbf{a}$ nous permet de faire disparaître les perturbations lorsque l tend vers l'infini.

$$\forall i, j \in E \qquad \lim_{l \to infty} p_l(i,j) = \delta(i,j) = \begin{cases} 0 & si\ i \neq j \\ 1 & si\ i = j \end{cases} \tag{1.2}$$

Croisement $U_k^l \longrightarrow V_k^l$ Ici encore l'opérateur est modélisé comme effectuant de petites perturbations aléatoires sur des couples de la population U_k^l. Ces couples sont ici formés par les éléments successifs de la population, les transitions sont gérées par le noyau Markovien q_l sur $E \times E$, cette fois, de sorte que :

$$P\left[V_k^l = v \mid U_k^l = u\right] = q_l\left((u_1, u_2) \cdot (u_3, u_4) \cdots (u_{N-1}, u_N)\right)$$

Pour ce noyau q_l nous supposerons l'existence d'un noyau irréductible β sur $E \times E$, la vitesse de convergence est alors paramétrée par le réel positif $\mathbf{b}$ tel que :

$$\forall\,(i_1, j_1) \in E \times E \qquad \forall\,(i_2, j_2) \in E \times E \qquad \forall s$$

$$q_l\left((i_1, j_1), (i_2, j_2)\right) = \begin{cases} \beta\left((i_1, j_1), (i_2, j_2)\right) \cdot l^{-\mathbf{b}} + o(l^{-s}) & si\ \ (i_1, j_1) \neq (i_2, j_2) \\ \\ 1 - \beta\left((i_1, j_1), (i_2, j_2)\right) \cdot l^{-\mathbf{b}} + o(l^{-s}) & si\ \ (i_1, j_1) = (i_2, j_2) \end{cases} \tag{1.3}$$

L'évanouissement asymptotique des croisements est également imposée par la positivité de $\mathbf{b}$:

$$\forall i_1,\ i_2,\ j_1,\ j_2 \in E \qquad \lim_{l \to infty} q_l\left((i_1, i_2)(j_1, j_2)\right) = \delta(i_1, i_2) \cdot \delta(j_1, j_2) \tag{1.4}$$

Sélection $V_k^l \longrightarrow X_{k+1}^l$ C'est l'opérateur le plus compliqué et également le plus important puisqu'il permet la convergence vers les optima de f.

Il est modélisé à l'aide d'une fonction de sélection F_l dont Cerf nous donne une définition précise, pouvant être résumée par :

$$\begin{aligned} F_l \quad : \quad & \{1, \cdots, N\} \times (\Re^+)^N \quad \longrightarrow [0,1] \\ & (i,\ f_1, f_2, \cdots f_N) \qquad \longrightarrow F_l\,(i,\ f_1, f_2, \cdots, f_N) \end{aligned}$$

telle que :

1. $F\,(\cdot, f_1, f_2, \cdots f_N)$ est une probabilité sur $\{1, \cdots, N\}$
2. Cette probabilité est indépendante de l'indexation des $f_1, f_2, \cdots f_N$ (on peut permuter les f_i)
3. La probabilité favorise les éléments i associés à des valeurs élevées (i.e.)

$$Si \quad f_1 \geq f_2 \geq \cdots \geq f_N \qquad Alors$$

$$F_l\,(1,\ f_1, f_2, \cdots, f_N) \geq F_l\,(2,\ f_1, f_2, \cdots, f_N) \geq \cdots \geq F_l\,(N,\ f_1, f_2, \cdots, f_N)$$

Cet outil nous permet d'écrire la probabilité de transition correspondant à la dernière étape.

$$P\left[X_{k+1}^l = x \mid V_k^l = v\right] = \prod_{r=1}^{N} \Upsilon_l(x_r, v_r)$$

Ceci signifie que la probabilité de transition est le produit des probabilités sur chacune des N composantes de E.

La probabilité Υ_l entre deux composantes (x_r, v_r) est donnée par :

$$\Upsilon_l(x_r, v_r) = \sum_{k\,:\,v_k = x_k} F_l\,(k,\ f(v_1), f(v_2), \cdots, f(v_N))$$

De même que pour les autres opérateurs, la fonction de sélection doit être choisie et sa vitesse de convergence caractérisée :

$$F_l\,(i,\ f_1, f_2, \cdots, f_N) = \frac{\exp\left(\mathbf{c} \cdot f_i \cdot \ln(l)\right)}{\sum\limits_{r=1}^{N} \exp\left(\mathbf{c} \cdot f_r \cdot \ln(l)\right)} \tag{1.5}$$

Ce choix correspond bien à une probabilité de sélection avantageant les fortes adaptations au détriment des faibles, le réel positif c indexant cette fonction.

Le mécanisme de sélection opérant sur le processus de fond (X_k^{infty}), correspond à la fonction de sélection F_{infty} définie par :

$$F_{infty}\,(k, f(x_1), f(x_2), \cdots, f(x_N)) = \frac{\mathbf{1}_{\widehat{x}}(x_k)}{card(\widehat{x})}$$

C'est à dire, la loi uniforme sur l'ensemble $\widehat{x} = \{x_k \in \arg\max f(x)\}$

La suite $(F_l)_{l\in\mathcal{N}}$ des fonctions de sélection tend vers cette loi uniforme

$$\begin{array}{l} \forall x \in E^N \qquad \forall k \\ \lim_{l\to infty}\ F_l\left(k, f(x_1), f(x_2), \cdots, f(x_N)\right) = F_{infty}\left(k, f(x_1), f(x_2), \cdots, f(x_N)\right) \end{array} \tag{1.6}$$

Les conditions 1.2, 1.4, et 1.6 nous permettent d'assurer que le mécanisme de transition de la chaîne (X_k^l) converge vers celui du processus de fond (X_k^{infty}) :

$$\forall y, z \in E^N \qquad \lim_{l\to infty}\ P\left[X_{k+1}^l = z \mid X_k^l = y\right] = \ P\left[X_{k+1}^{infty} = z \mid X_k^{infty} = y\right]$$

C'est également en ce sens que l'on interprète la chaîne (X_k^l) comme une perturbation de la chaîne (X_k^{infty}).

Les vitesses de convergence intervenant dans chacun des opérateurs jouent un rôle important. La formulation proposée en (1.1), (1.3) et (1.5), permet un ajustement équitable de ces vitesses (elles sont logarithmiquement du même ordre) de sorte qu'aucun opérateur ne domine les autres dans la dynamique. lorsque l tend vers l'infini, les conditions (1.2), (1.4), et (1.6) nous permettent d'assurer que le mécanisme de transition de la chaîne (X_k^l) converge vers celui du processus de fond (X_k^{infty}), et on a :

$$\forall y, z \in E^N \qquad \lim_{l\to infty}\ P\left[X_{k+1}^l = z \mid X_k^l = y\right] = \ P\left[X_{k+1}^{infty} = z \mid X_k^{infty} = y\right]$$

La chaîne (X_k^l) se comporte alors comme le ferait (X_k^{infty}). La théorie de Freidlin-Wentzell nous donne les outils pour simplifier l'étude de ces processus à temps continu.

1.7.3 Application de la théorie de Freidlin et Wentzell

Principe

Soit le système différentiel de $\Re^N$ satisfaisant les équations déterministes :

$$\left\{ \begin{array}{l} dx_t = b(x_t)\ dt \\ x_0 = x_{ini} \end{array} \right. \tag{1.7}$$

Sous de "bonnes" hypothèses, il existe une solution (trajectoire) unique, $x(t)$ à l'équation (1.7) et à la condition initiale associée. l'une des préoccupation immédiates est de savoir si cette solution va, ou non, tendre vers un équilibre (qui n'est pas forcément unique). Et si oui, quel en est l'ensemble de stabilité. L'équilibre est défini comme une fonction constante x^* telle que $x^* = \lim_{t\to infty} x_t$, et l'ensemble de stabilité comme l'ensemble $K(\ x^*)$ des points de départ qui mènent à cet équilibre[10]. On peut élargir cette notion, d'équilibre et de stabilité, par celles, très proches, d'attracteur et de bassin d'attraction.

[10] L'ensemble de stabilité de l'équilibre x^* est :

$$K(x^*) = \left\{ x_{ini} \in \Re^N,\ t.q.\ pour\ x_t\ solution\ de\ 1.7\ ;\ \lim_{t\to infty} x_t = x^* \right\}$$

Pour chaque équilibre on définit ainsi son ensemble de stabilité. Cet équilibre est stable s'il contient un voisinage de l'équilibre, et instable s'il existe des points de départ infiniment proche de l'équilibre qui ne mènent pas à celui-ci.

Un attracteur du système est le voisinage compact K_i d'un point visité une infinité de fois, et le bassin d'attraction l'ensemble des points de départ qui mènent à cet attracteur. Nous supposerons que $\Re^d$ possède un nombre fini d'attracteurs $K_1, \cdots, K_r$.

La théorie de Freidlin-Wentzell étudie l'évolution du système 1.7 lorsqu'il subit des perturbations Brownièenes, d'intensité ε. Le système déterministe 1.7 devient alors un système différentiel stochastique.

$$\begin{cases} dX_t^\varepsilon = b(X_t^\varepsilon)\ dt + \varepsilon d\omega_t \\ X_0^\varepsilon = x_{ini} \end{cases} \tag{1.8}$$

Le processus $(X_t^\varepsilon)_{t\in\Re^+}$ est maintenant un processus stochastique perturbé par le mouvement brownien $(\omega_t)_{t\in\Re^+}$ et dépendant de ε. La situation change alors puisque les perturbations brownienne permettent au processus de s'échapper de n'importe quel bassin d'attraction, et en fait le processus les visite tous.

De plus, le processus est ergodique et admet une unique mesure de probabilité invariante, i.e.

$$\forall \mathcal{B}\ borelien\ \in \Re^d \qquad \lim_{t\to infty} P\left[X_t^\varepsilon \in \mathcal{B} \mid X_0^\varepsilon = x_{ini}\right] = \mu^\varepsilon(\mathcal{B})$$

existe et $\mu^\varepsilon(\mathcal{B})$ est la probabilité de présence du processus dans le Borélien $\mathcal{B}$, lorsque le système a atteint son état d'équilibre. Cette probabilité μ^ε est invariante avec le point de départ x_{ini}.

Lorsque les perturbations cessent, le processus se comporte comme dans 1.7 et reste presque sûrement au voisinage $V(K_1 \cup \cdots \cup K_r)$ des attracteurs, tandis que la probabilité de présence dans n'importe quel Borélien $\mathcal{A}$ disjoint de $K_1 \cup \cdots \cup K_r$ disparaît.

$$\begin{aligned} \lim_{\varepsilon\to 0} \mu^\varepsilon\left(V(K_1 \cup \cdots \cup K_r)\right) &= 1 \\ \lim_{\varepsilon\to 0} \mu^\varepsilon\left(\mathcal{A}\right) &= 0 \end{aligned}$$

Le résultat principal de Freidlin et Wentzell repose sur l'équivalence du processus $(X_t^\varepsilon)_{t\in\Re^+}$ à temps continu et espace d'état $\Re^d$ et du processus $(Z_n^\varepsilon)_{n\in\mathcal{N}}$ à temps discret et espace d'état fini $\{1, \cdots .r\}$ décrivant les visites au $n$ième attracteur.

La construction précise de $(Z_n^\varepsilon)_{n\in\mathcal{N}}$, n'est pas reportée ici mais nous en donnons un aperçu afin de mieux comprendre ce dernier processus.

- Si x_{ini} est "proche" de l'attracteur K_h alors $Z_o^\varepsilon = h \in \{1, \cdots .r\}$
- puis le processus, sous l'influence de (ω_t), est attiré par K_s et $Z_1^\varepsilon = s$
- etc..

La chaîne de Markov[11] ainsi créée a pour espace d'états $\{1, \cdots .r\}$, est irréductible, et possède une unique mesure de probabilité invariante ν^ε.

Théorème 1.2 *L'étude du comportement asymptotique de la mesure μ^ε est "équivalente" à l'étude du comportement asymptotique de la mesure ν^ε*

[11]La nature Markovienne de ω_t, nous permet de montrer qu'il s'agit bien là d'une chaine de Markov.

Nous passons sous silence l'étude des probabilité de transition $P\left[Z_n^\varepsilon = i \mid Z_n^\varepsilon = j\right]$ de la chaîne $(Z_n^\varepsilon)_{n\in\mathcal{N}}$ qui s'écrivent comme des intégrales sur l'ensemble des fonctions ϕ qui lient les attracteurs K_i et K_j, laissant le lecteur intéressé se reporter à la lecture de Freidlin et Wentzell, ou de Cerf.

Notons toutefois que ces probabilité de transition s'écrivent :

$$P\left[Z_n^\varepsilon = i \mid Z_n^\varepsilon = j\right] \underset{\ln}{\sim} \exp -\frac{V(i,j)}{\varepsilon^2}$$

où $V(i,j) = \inf\{V(\phi), \phi(\cdot) \text{ continue } [0.1] \mapsto \Re^d, \phi(0) \in K_i, \phi(T) \in K_j\}$ et $V(\phi) = \int_0^1 \left|\dot{\phi}(t) - b(\phi(t))\right|^2 dt$.
est une constante associée à ϕ et caractérisant sa vitesse de convergence.

La quantité $V(i,j)$ ou *coût de communication,* mesure le coût de passage de l'attracteur K_i à l'attracteur K_j.

Les intensités de transitions de la chaîne $(Z_n^\varepsilon)_{n\in\mathcal{N}}$, nous ouvrent la voie pour déterminer la mesure invariante ν^ε.

Mesure invariante ν^ε

Les outils qui permettent de déterminer cette mesure invariante ont été développés, une nouvelle fois par Freidlin et Wentzell, nous aurons besoin de certains d'entre eux.

Définition 1.9 *Soit i un élément de $\{1,...,r\}$. Un $i-$graphe sur $\{1,...,r\}$ est un graphe g sur $\{1,...,r\}$ possédant les propriétés suivantes :*

- $\forall j \neq i$, *le graphe g contient une unique flèche issue de j*
- *Il existe un chemin dans g qui mène de j à i*
- *g ne contient pas de flèche issue de i*

Il s'agit donc d'un graphe sans cycles formé de chemins qui aboutissent en i. On note $G(i)$ l'ensemble des $i-$graphes sur $\{1,...,r\}$.

Définition 1.10 *La fonction d'énergie virtuelle W est la fonction de $\{1,...,r\}$ dans $\Re^+$ définie par :*

$$\forall i \in \{1,...,r\} \qquad W(i) = \min_{g\in G(i)} \sum_{(\alpha\to\beta)\in g} V(\alpha,\beta)$$

A cette fonction est associé l'ensemble W^* des minima globaux de W.

Finalement, la mesure invariante ν^ε est caractérisée par :

$$\forall i \in \{1,...,r\} \qquad \nu^\varepsilon(i) \underset{\ln}{\sim} \exp -\frac{W(i) - W(W^*)}{\varepsilon^2}$$

où $W(W^*) = \min\{W(i) \ : \ i \in \{1,...,r\}\}$.

Le comportement asymptotique de ν^ε (et par la même occasion de μ^ε) est donc connu : la mesure ν^ε se concentre sur les attracteurs dont l'indice est dans W^* et décroît vers zéro à la vitesse $\exp -\frac{Cste}{\varepsilon^2}$ pour les autres attracteurs. Il existe donc un sous-ensemble de W^* de l'ensemble des attracteurs sur lequel se concentre la mesure invariante du processus.

$$\lim_{\varepsilon\to 0} \lim_{t\to infty} P\left[X_t^\varepsilon \in V\left(\bigcup_{i\in W^*} K_i\right) \mid X_0^\varepsilon = x_{ini}\right] = 1$$

La dynamique du processus est donc caractérisable.

Dynamique du processus

Dans sa thèse, Cerf nous donne une très claire interprétation de la hiérarchie des cycles qui caractérisent la dynamique du processus. Supposons que le processus soit initialement dans le bassin d'attraction de K_1. Il quitte K_1 au bout d'un temps fini. Parmi toutes les trajectoires de sortie, il en existe une plus "probable" que les autres, qui l'amène vers un nouvel attracteur; par exemple K_2 puis, bientôt K_3. L'ensemble des attracteurs étant par hypothèse fini, le processus finit par revisiter un attracteur formant un cycle d'ordre 1 sur lequel le processus tourne longtemps, très longtemps. Englobons maintenant ces trois attracteurs dans une boîte. Comme toujours, les perturbations browniennes finissent par pousser le processus hors de cette boîte, et ici encore, il existe une trajectoire de sortie canonique qui fait tomber le processus dans un nouveau bassin d'attraction, ou plus généralement, dans un autre cycle d'ordre 1.

Les cycles d'ordre 1 sont aussi en nombre fini, et le processus finit par revisiter un cycle d'ordre 1: un cycle d'ordre 2 est alors formé, dans lequel le processus reste piégé très longtemps. En continuant de la sorte, il est possible de construire toute une hiérarchie de cycles qui épuise l'ensemble des attracteurs et fournit une image très précise de la dynamique asymptotique du processus. A chaque transition entre cycles est associée une constante qui caractérise la difficulté de la transition.

Enfin, lorsque ε décroît avec le temps ($\varepsilon = \varepsilon(t)$ est une fonction de t qui tend en décroissant vers 0), nous obtenons un processus de Markov inhomogène (le mécanisme de transition dépend du temps).

- Si $\varepsilon(t)$ décroît très lentement, de sorte qu'à chaque instant la loi de X_t soit proche de l'état d'équilibre associé au niveau de perturbation $\varepsilon(t)$, la situation ne change pas fondamentalement. La loi limite correspond à la limite de la suite des lois d'équilibre.

- Si au contraire $\varepsilon(t)$ décroît très rapidement, le processus risque de rester piégé dans certains sous-ensembles d'attracteurs: plus précisément, dans la hiérarchie des cycles, certaines transitions ne pourront être effectuées qu'un nombre fini de fois, alors que d'autres, plus "faciles", seront réalisées une infinité de fois avec probabilité 1. La loi limite dépend alors fortement du point de départ.

La hiérarchie des cycles permet ainsi de décrire les dynamiques possibles de (X_t) en fonction de la vitesse de décroissance de $\varepsilon(t)$.

Résultats de convergence

Lorsque l croit vers l'infini, les perturbations affectant le processus (X_k^l) diminuent de sorte que cette chaîne se comporte, presque sûrement, comme la chaîne (X_k^{infty}). Plus précisément, nous savons que les attracteurs de la chaîne (X_k^{infty}) sont les populations d'équi-adaptation S et les populations uniformes (attracteurs stables). La chaîne (X_k^l) va donc être attirée par ses attracteurs, en commençant par l'ensemble S.

La théorie de Freidlin et Wentzell nous permet de reporter cette étude sur celle de la chaîne des (Z_k^l) des visites successives de (X_k^l) à l'ensemble S. Nous poserons donc $Z_k^l = X_{T_k}^l$ où T_k est l'instant de la $k$ième visite de (X_k^l) dans S.

Les probabilités de transition de la chaîne (Z_k^l), sont estimées à l'aide des opérateurs définis en 1.7.2 et selon le schéma développé ci-dessus. Les fonctions de coût de communication $V(i,j)$ et d'énergie virtuelle W sont définies et estimées.

Nous savons alors que la suite des mesures stationnaires de la chaîne (X_k^l) se concentre sur l'ensemble W^* des minima de W :

$$\forall x_{ini} \in E^N \qquad \lim_{l \to infty} \lim_{k \to infty} P\left[X_k^l \in W^* \mid X_0^l = x_{ini}\right] = 1$$

L'un des principaux résultats indique qu'il existe une taille de la population de (X_k^l), (taille critique) telle que les maxima de f soient atteints asymptotiquement avec la probabilité 1.

Taille critique

Supposons fixés l'espace d'état E, la fonction d'adaptation f, les noyaux de transition de mutation α et de croisement β, ainsi que les constantes positives gérant les trois opérateurs $\mathbf{a}$, $\mathbf{b}$, et $\mathbf{c}$.

Théorème 1.3 *(Cerf 1993)*
Il existe une valeur critique N^*, telle que lorsque la taille de la population de l'algorithme génétique dépasse N^*, l'ensemble f^* des maxima globaux de f, contient l'ensemble W^*.
Cette taille critique N^* , dépend fortement de l'espace d'état E, de la fonction d'adaptation f, des noyaux de transition de mutation α et de croisement β, ainsi que des paramètres $\mathbf{a}$, $\mathbf{b}$, et $\mathbf{c}$.

Une borne grossière, mais lisible de N^* est :

$$N^* \leq \frac{\mathbf{a}R + \mathbf{c}(R-1)\Delta}{\min(\mathbf{a},\ \frac{\mathbf{b}}{2},\ \mathbf{c}\delta)}$$

où :

- R est le nombre minimal de transition permettant de joindre deux points arbitraires de E par mutation
- Δ et δ sont des paramètres d'échelle :
  - $\Delta = \max\{|f(i) - f(j)| : i, j \in E\}$ paramètre mesurant les écarts maximaux de f
  - $\delta = \min\{|f(i) - f(j)| : i, j \in E,\ f(i) \neq f(j)\}$ mesurant les écarts minimaux de f.

Il est intéressant de relever que le résultat est obtenu sans faire intervenir l'opérateur de croisement, qui n'est donc pas indispensable. L'exploration par mutation et la sélection suffisent à assurer la convergence vers f^* ([Zhi91]).

Ce premier résultat nous indique que dès que $N \geq N^*$, la suite des mesures stationnaires de la chaîne (X_k^l) se concentre asymptotiquement sur f^* lorsque l tend vers l'infini. On peut dans une étape suivante faire évoluer l , et donc l'intensité des perturbations, en fonction de la génération. Nous obtenons alors une chaîne de Markov inhomogène $(X_k^{l(k)})$ dont le mécanisme de transition dépend alors de la génération k.

Vitesse de convergence

Le principal problème est de savoir si cette chaîne inhomogène peut avoir un comportement proche de celui de la chaîne homogène (X_k^l), et si oui, sous quelles conditions. La vitesse de croissance de $l(k)$ vers l'infini, est bien évidemment, au centre du débat.

- Si $l(k)$ croît "lentement", alors la loi de X_k sera proche de la loi stationnaire $\mu^{\varepsilon(l(n))}$ de niveau de perturbation $\varepsilon(l(n))$ associé à $l(n)$.
- Si $l(k)$ croît "rapidement", alors X_k risque de rester piégé dans des bassins d'attraction ne correspondant pas aux maxima de f, l'intensité des perturbations devenant trop faible pour pouvoir s'en échapper.

La vitesse recherchée se situe entre ces deux extrêmes, permettant à X_k de s'échapper des "mauvais" bassins d'attraction (ne correspondant pas à des maxima de f) et de rester piégé dans le "bon" (celui des points de f^*).

La vitesse de convergence de la suite $l(k)$ est caractérisée par l'*exposant de convergence*[12], λ.

Définition 1.11 *L'exposant de convergence λ de la suite $l(k)$ est l'unique réel λ tel que :*

$$\sum_{k\in\mathcal{N}} l(k)^{-\theta} \quad \begin{array}{ll} \rightarrow & converge\ pour\ \theta > \lambda \\ \rightarrow & diverge\ pour\ \theta < \lambda \end{array}$$

Deux conditions pour la colonisation de f^* sont également données par Cerf, l'une nécessaire, l'autre suffisante.

Théorème 1.4 *Condition nécessaire pour la colonisation de f^**

Pour que :

$$\forall x_{ini} \in E^N \qquad P\left[\exists K \quad \forall k \geq K \quad [X_{T_k}] \subset f^* \mid X_0 = x_{ini}\right] = 1$$

c'est à dire, pour que la chaîne $Z_k = X_{T_k}$ des visites successives des attracteurs soit piégée dans f^* après un nombre fini K de transitions, il est nécessaire que l'exposant de convergence λ de la suite $l(k)$ appartienne à l'intervalle $]\phi, \psi[$.

Les constantes ϕ et ψ sont des caractéristiques du problème,l'intervalle $]\phi, \psi[$ est alors non vide pour N assez grand.

Théorème 1.5 *Condition suffisante pour la colonisation de f^**

Il existe deux constantes η et ρ telles que si l'exposant de convergence λ de la suite $l(k)$ appartient à l'intervalle $]\eta, \rho[$, alors :

$$\forall x_{ini} \in E^N \qquad P\left[\exists K \quad \forall k \geq K \quad [X_{T_k}] \subset f^* \ , \ \widehat{X_k} \subset f^* \mid X_0 = x_{ini}\right] = 1$$

ce qui signifie qu'après un nombre fini de transitions, nous avons presque sûrement, la situation suivante :

- *la chaîne X_{T_k} est piégée dans f^*,*
- *la population X_k contient toujours un ou des individus appartenant à f^*.*

[12] Egalement appelé *rayon de convergence.*

En guise de conclusion

D'autres résultats existent, tant dans le travail de Cerf, que dans la littérature citée dans cette section. Ils demandent cependant un investissement supplémentaire dans la compréhension des outils développés. Le but de cette section était de convaincre le lecteur que l'étude théorique des algorithmes génétiques donne (déjà) de substantiels résultats. De nombreuses interrogations demeurent cependant concernant les relations réelles entre les différents paramètres caractérisant l'algorithme génétique et les choix pratiques de ceux-ci. Dans ce domaine, la pratique devance encore la théorie, même si les mécanismes commencent à être plus clairs. Il reste également à étudier les nombreux raffinements (tels que le scaling, sharing, le clustering, l'élitisme) indispensables dans la pratique.

Chapitre 2

Méthodes et problèmes

Introduction

Nous allons dans ce chapitre nous intéresser à un certain nombre de problèmes classiques pouvant être résolus par algorithmes génétiques, et nous allons nous efforcer de comparer les algorithmes génétiques, quand cela est possible, à d'autres méthodes également applicables.

2.1 Optimisation de fonctions à variables réelles

Dans cette section, nous allons étudier le fonctionnement des AG pour l'optimisation de fonctions à variables réelles et les comparer à d'autres méthodes, soit locales (simplex [NM65, JKP72], BFGS [Bro69, Fle70, Gol70, DS83]), soit globales (programmation par intervalles [Han92, RR95], recuit [AK89, Egl92, Ing89]). Certaines de ces fonctions ont été déjà étudiées par Lester Ingber [IR92b], afin de mesurer les efficacités respectives des algorithmes génétiques et du recuit simulé (variante VFSR[1] (ou ASA[2])). Ingber avait utilisé un algorithme génétique classique, GAUCSD [SG91], qui travaillait avec les traditionnelles chaînes de bits.

Nous sommes conscients que le grand absent de cette section est le branch and bound stochastique, mais nous ne disposions pas des compétences suffisantes pour tester cette méthode.

Enfin, il faut également remarquer la difficulté qu'il y a à comparer deux méthodes d'optimisation. Nous avons choisi comme indicateur le nombre d'évaluations de la fonction, Il est clair cependant que certains algorithmes peuvent calculer moins de fois la fonction à optimiser, tout en étant plus lents a l'exécution. Notons tout de même qu'en ce qui concerne BFGS, chaque appel à la fonction pour estimer le gradient est considéré comme un appel de fonction à part entière, et comptabilisé comme tel. Cela désavantage BFGS sur les fonctions pour lesquelles le gradient peut être estimé directement.

2.1.1 Fonction de Rosenbrock et Chebyquad

Nous allons nous intéresser à la minimisation de la fonction :

$$f_2(x, y) = 100(x^2 - y)^2 + (1 - x)^2$$

sur l'intervalle $[-2, 2]$.

[1] Very Fast Simulated Reannealing
[2] Adaptive Simulated Annealing

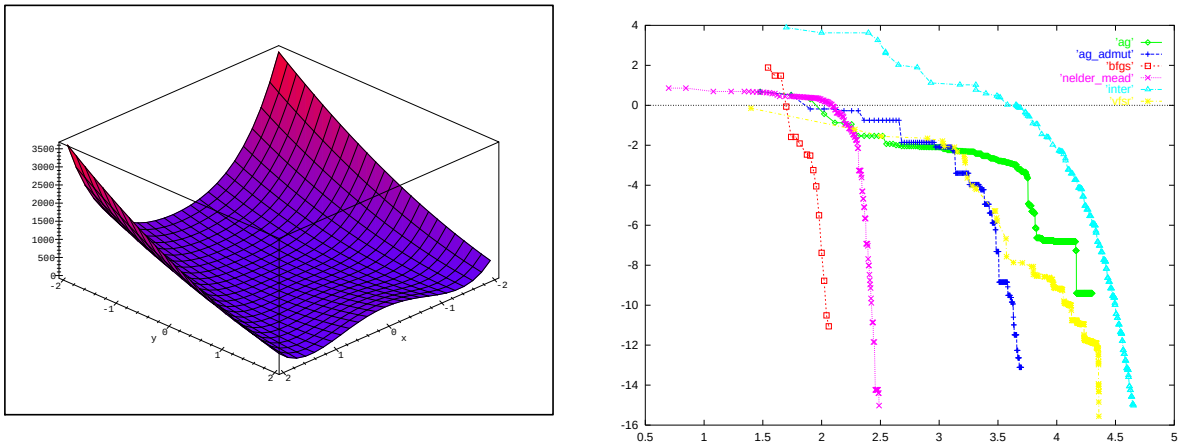


Figure 2.1: Fonction de Rosenbrock

Cette fonction peut être optimisée par des méthodes classiques. Une étude élémentaire (dérivées partielles) montre qu'il existe un seul minimum, le point $(1, 1)$ et que la valeur du minimum en ce point est 0. D'autre part, notre algorithme de simplex donne une solution à 10^{-6} en un nombre réduit d'évaluations de fonction, en partant d'un point quelconque de l'espace de recherche.

La fonction est assez difficile à optimiser : le premier terme de la fonction définit une quadrique dont les bords sont très abrupts, dont le fond est plat, défini par la parabole $y = x^2$. C'est sur cette courbe qu'il faut ensuite optimiser le second terme, sachant que toute petite perturbation de x entraîne immédiatement une forte variation du premier terme si y ne varie pas simultanément pour que le point (x, y) reste sur la parabole $y = x^2$. La figure 2.1 permet de mieux comprendre le phénomène.

On a représenté sur la figure 2.1 (à droite) l'évolution de la valeur de f_2 en fonction du nombre d'évaluations effectuées par l'algorithme génétique avec mutation adaptative et sans mutation adaptative, ainsi que du recuit simulé (VFSR), d'un algorithme de simplex (Nelder-Mead), de la méthode BFGS et d'une arithmétique des intervalles. L'échelle est logarithmique sur les deux axes. On a utilisé un croisement barycentrique et une mutation sous forme de bruit gaussien pour l'algorithme génétique. On constate que l'algorithme génétique utilisant la mutation adaptative obtient des résultats bien supérieurs à ceux de l'algorithme génétique standard. Les paramètres de VFSR ont été "adaptés" de façon à donner un résultat aussi bon que possible.

Si l'on utilise la programmation par intervalle, les résultats sont excellents en terme d'itérations. L'algorithme trouve l'optimum à 10^{-16} près avec 901 itérations pour la fonction $f(x)$ et 1803 itérations pour la fonction $F(X)$. Cependant, il faut nuancer ce résultat. En effet, le calcul d'une itération de la fonction $F(X)$ (qui calcule l'image d'un intervalle) est environ 25 fois plus longue que le calcul d'une itération de $f(x)$. Nous avons donc représenté sur la figure 2.1 le résultat de (25*IterFX+Iterfx) et non la somme du nombre des itérations. Il est clair qu'un meilleure implantation de l'arithmétique des intervalles permettrait d'améliorer considérablement les résultats.

Pour conclure notons que :

- Les méthodes locales sont plus efficaces d'environ 2 ordres de magnitude, si on les compare aux méthodes globales, lorsqu'elles sont applicables. On en vient donc à la remarque de bon sens qu'il vaut toujours mieux appliquer une méthode locale lorsque cela est possible.

- Les résultats pour le recuit sont quasiment identiques à ceux présentés par L. Ingber dans

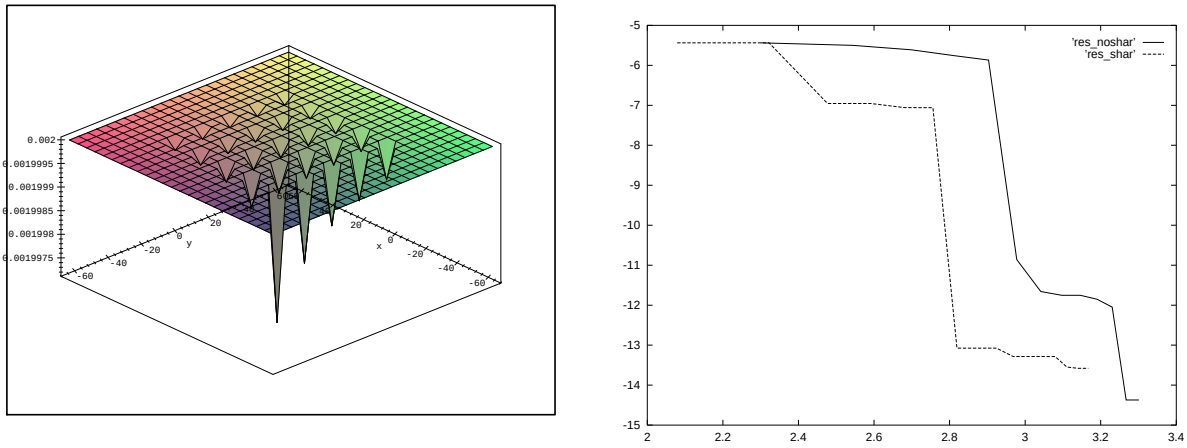


Figure 2.2: Fonction de DeJong

[IR92b]. Les résultats pour l'AG utilisant des réels sont un peu meilleurs que ceux de l'AG à chaîne de bits utilisé par Ingber. En revanche, si l'on introduit la mutation adaptative, l'AG devient plus efficace que le recuit.

2.1.2 Fonction de De Jong

L'optimisation de la fonction f_5 de DeJong [DeJ75] est un des exemples les plus classiques de l'optimisation globale. Cette fonction a la forme suivante :

$$
\begin{aligned}
f_5(x_1,x_2) &= \frac{1}{500+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ji})^6}} \\
a_{j1} &= \{-32,-16,0,16,32,-32,-16,0,16,32,-32,-16, \\
&\quad 0,16,32,-32,-16,0,16,32,-32,-16,0,16,32\} \\
a_{j2} &= \{-32,-32,-32,-32,-32,-16,-16,-16,-16,-16, \\
&\quad 0,0,0,0,0,16,16,16,16,16,32,32,32,32,32\}
\end{aligned}
$$

Nous présentons sur la figure 2.2 la forme de cette fonction ainsi que les résultats obtenus par l'algorithme génétique avec et sans (clustering+sharing). Sur la partie gauche de la figure, on représente la convergence vers l'optimum en format log/log, avec en abscisse le logarithme du nombre d'évaluations, et en ordonnée le logarithme de l'écart à l'optimum. Le clustering améliore les performances, car il permet de maintenir une meilleure diversité de population et sort rapidement l'algorithme d'un optimum local. La convergence est réalisé en moyenne en 1500 évaluations de fonctions avec le clustering. Les résultats sont très proches de ceux obtenus par VFSR [IR92b].

A ce sujet, on peut rapidement rappeler les résultats présentés par David Fogel dans [Fog95]. Les résultats de notre AG sont meilleurs que ceux obtenus par Fogel avec un mécanisme dit d' "Evolutionary Computation[3]". Au demeurant, l'ensemble des résultats présentés par Fogel à l'appui de sa thèse (l' "Evolutionary Computation" est supérieur aux mécanismes génétiques standard) est plus que discutables. Il nous a été possible en effet d'obtenir sur l'ensemble des fonctions de test qu'il présente

[3]Le terme "Evolutionary Computation" désigne des algorithmes de type génétique mais qui n'utilisent que la mutation.

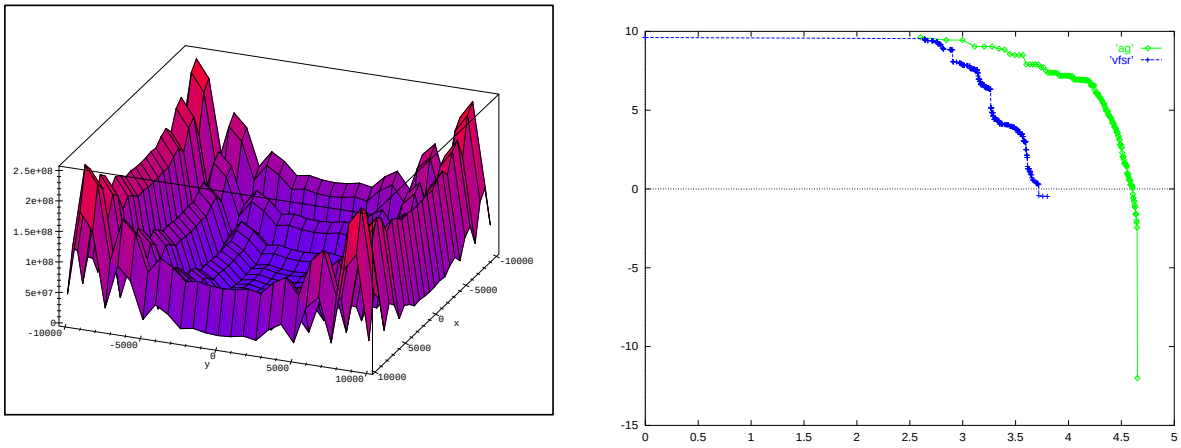


Figure 2.3: Fonction de Corana représenté pour $N = 2$ (à gauche). Résultats de convergence pour $N = 12$ (à droite)

de meilleurs résultats que ses algorithmes. En ne prenant en compte que des algorithmes génétiques binaires primitifs, et non des algorithmes à codage réel qu'il semble mal connaître, il commet une erreur qui fausse totalement sa présentation. D'autre part, les fonctions de test qu'ils utilisent ne présentent que peu, ou pas du tout, d'intérêt sur le plan de l'optimisation globale : elles sont bien trop simples (de même d'ailleurs que la fonction de De Jong).

2.1.3 Fonction de Corana

Cette fonction est présentée en détail dans [CMMR87]. Nous en utilisons ici la restriction que présente Ingber. On doit optimiser la fonction sur le compact $[-10000, 10000]^N$.

$$
\begin{aligned}
f_0(x_1, \ldots, x_N) &= \sum_{i=1}^{N} \begin{cases} 0.15\, d_i (0.05\, S(z_i) + z_i)^2 & \text{si } |x_i - z_i| < 0.05 \\ d_i\, x_i^2 & \text{sinon} \end{cases} \\
z_i &= 0.2\, \lfloor |x_i/0.2| + 0.49999 \rfloor\, S(x_i) \\
S(z_i) &= \begin{cases} 1 & \text{si } z_i > 0 \\ 0 & \text{si } z_i = 0 \\ -1 & \text{si } z_i < 0 \end{cases} \\
d_{i \bmod 4} &= \{1.0, 1000.0, 10.0, 100.0\}
\end{aligned}
$$

Cette fonction est intéressante à plus d'un titre. D'une part, elle présente la propriété d'avoir 10^{5N} optima locaux. Pour $N = 4$, cela signifie déjà qu'un algorithme utilisant un millième de second pour parcourir chaque minimum mettrait un temps supérieur à l'âge de l'univers pour trouver l'optimum global (tous les points du compact $[-0.05, 0.05]^N$ sont optimaux). D'autre part cette fonction est, pour L. Ingber, un des meilleurs tests possibles pour un algorithme d'optimisation global. Il nous a donc semblé intéressant de nous mesurer à VFSR sur cet exemple. La figure 2.3 représente la fonction pour $N = 2$. Il faut noter que pour les deux algorithmes concernés, la convergence ne présente aucune difficulté pour $N = 4$ et $N = 8$. A partir de $N = 12$, le problème devient plus difficile. Les paramètres par défaut de VFSR ne permettent pas de faire converger l'algorithme simplement.

Avec l'aide de L. Ingber, il nous a été possible de trouver un ensemble de paramètres faisant converger VFSR. Il s'agit en fait d'utiliser une technique particulière, le "Quenching" qui consiste à

	Appels à f	Appels à F
$N = 1$	31	63
$N = 2$	440	881
$N = 3$	1755	3511
$N = 4$	4979	9959
$N = 5$	38994	77989
$N = 6$	525193	1050387

Tableau 2.1: Programmation par intervalles appliquée à la fonction de Corana

accélérer le schéma de recuit, au risque de tomber d'ailleurs dans un optimum local. Sur 10 tests faits avec VFSR, 5 d'entre eux trouvent la solution optimale et 5 restent bloqués dans un optimum local. L'algorithme génétique trouve toujours la meilleure solution avec une population de 400 éléments. En revanche, lorsqu'il converge, l'algorithme de recuit est nettement plus rapide que l'algorithme génétique (voir figure 2.3 un exemple nominal de convergence des deux algorithmes).

Nous avons alors décidé d'augmenter la valeur de N pour observer le comportement de chacun des deux algorithmes lorsque le problème se durcit. Pour l'algorithme génétique, il suffit d'augmenter le nombre d'éléments de population jusqu'à ce que l'algorithme converge. La limite avec l'AG classique se trouve autour de $N = 24$. Nous ne sommes pas parvenus à faire converger VFSR pour $N = 20$.

La fonction de Corana étant complètement séparée, il est intéressant de lui appliquer la méthode d'optimisation pour les fonctions partiellement séparables mise au point par Nicolas Durand présentée dans la section 1.4.1. Pour appliquer l'AG avec les opérateurs adaptés, on définit trivialement la fitness locale comme suit :

$$G(x_i) \quad = \quad \begin{cases} 0.15\, d_i (0.05\, S(z_i) + z_i)^2 & \text{si } |x_i - z_i| < 0.05 \\ d_i\, x_i^2 & \text{sinon} \end{cases}$$

Puisque la fonction est totalement séparée, on pouvait s'attendre à ce que l'opérateur adapté se révèle particulièrement efficace, ce que confirment les tests : avec une population de 400 éléments sur 100 tests, l'AG utilisant le croisement adapté trouve toujours le minimum global avant la génération 100 pour $N = 1000$.

Enfin, nous avons également testé la programmation par intervalle. Les résultats sont présentés sur le tableau 2.1. On remarque que l'on ne peut trouver de solution pour $N > 6$, en raison de l'occupation mémoire. En fait, la fonction de Corana est très mal adaptée à la programmation par intervalles, car elle présente de nombreux plateaux. Ces plateaux peuvent être subdivisés en de nombreux intervalles pour lesquels les bornes de la fonction sont les mêmes. Cela provoque une explosion du nombre d'intervalles que l'estimateur ne parvient pas à "couper".

2.1.4 Fonction de Griewank

Cette fonction ressemble à la précédente mais présente une difficulté supplémentaire : elle n'est pas séparable. En revanche, elle ne présente pas de plateaux, ce qui la rend beaucoup plus facile pour la programmation par intervalles.

La fonction de Griewank est définie comme suit :

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

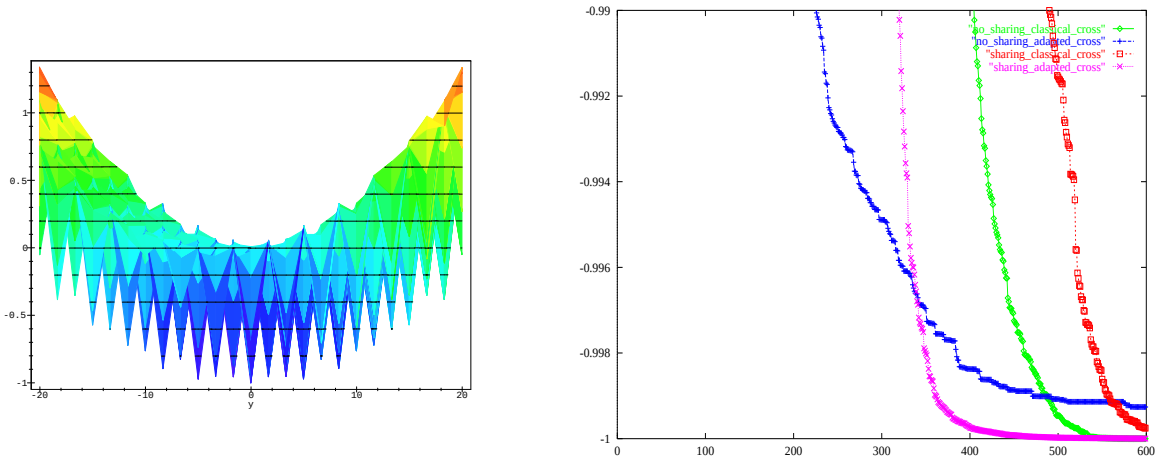


Figure 2.4: $F(x+y, x+y, ...x+y)$ pour $x \in [-\Pi, \Pi]$ et $y \in [-20, 20]$ et valeur moyenne de la meilleure fitness en fonction de la génération courante

Nous nous intéresserons ici au cas $n = 10$. Cette fonction n'est pas représentable facilement mais on peut par exemple tracer $F(x+y, x+y, ...x+y)$ pour x variant entre $-\Pi$ et Π et y variant entre -20 et 20 (voir figure 2.4). On observe alors sur cette coupe que le minimum de la fonction concernée se situe en 0, mais que la fonction a de nombreux minima locaux proches et éloignés de 0.

Nous allons montrer sur cet exemple combien l'utilisation d'une fitness locale peut être utile, bien que la fonction ne soit pas complètement séparable. On définit la fitness locale comme suit :

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

On effectue alors 4 séries de tests :

- sans sharing avec un croisement classique de type barycentrique.
- sans sharing avec le croisement adapté.
- avec sharing avec un croisement classique de type barycentrique.
- avec sharing et le croisement adapté.

Dans chaque cas, l'algorithme génétique est testé une centaine de fois avec 600 générations et 100 individus (qui représente environ 36000 évaluations de fonction). On dit que l'algorithme génétique a convergé lorsqu'il a trouvé une solution, qui, par une méthode locale simple, permet d'atteindre l'optimum global. On peut d'ailleurs souligner ici l'importance de l'association AG/méthodes locales qui permet seule de trouver l'optimum de la fonction.

Le tableau 2.2 donne le nombre minimal, maximal et moyen de générations nécessaires pour que l'algorithme génétique converge. Le tableau donne également l'écart-type de ce nombre de générations ainsi que la distance moyenne (en norme infinie) de la solution optimale trouvée à la solution optimale absolue, (0).

Les résultats ci-dessus montrent que la convergence de l'algorithme est 2 fois plus rapide avec le croisement adapté qu'avec un croisement classique. On observe également que le sharing ralentit légèrement la convergence de l'algorithme quel que soit le croisement utilisé. Ainsi, l'intérêt de

type de croisement	min	max	moy	σ	dist à 0
classique	76	294	179.44	178.47	0.00
adapté	42	204	92.33	91.81	2.03
classique avec sharing	79	357	248.87	247.32	0.06
adapté avec sharing	42	186	96.57	95.95	0.01

Tableau 2.2: 600 générations et 100 éléments de population

l'opérateur de sharing n'est pas évident dans ce cas. Cependant, les performances de l'algorithme ne sont pas dégradées par l'opérateur de sharing lorsque le croisement adapté est utilisé. On observe même que l'opérateur de sharing permet d'obtenir une solution après convergence proche de 0 alors que sans l'opérateur de sharing, la meilleure solution après convergence reste éloignée de 0.

La figure 2.4 donne les valeurs moyennes des meilleures fitness suivant les 4 tests. Cette figure confirme le bon comportement de l'opérateur adapté avec sharing.

Dans chacune de ces 4 séries de tests, il est possible d'observer l'effet du croisement de la façon suivante : à chaque génération, on effectue 60 croisements, parmi ceux-ci, certains donnent soit un individu optimal (au sens défini précédemment), soit un individu acceptable, soit un individu qui sera immédiatement rejeté. Les courbes 2.5, 2.5, 2.6 et 2.6 donnent les répartitions de ces catégories pour un test extrait de chacune des 4 séries de tests.

On observe sur les 4 figures que lorsque l'algorithme génère des solutions optimales, il génère également des solution rejetées (tellement mauvaises que l'opérateur de sélection ne les conserve pas). C'est la difficulté de ce problème (les solutions optimales sont proches de solutions très mauvaises). On observe sur la figure 2.5 que l'opérateur de croisement adapté génère un nombre très important de solutions rejetées. Il est intéressant de constater que l'opérateur de sharing permet d'atténuer cet effet.

Des tests ont été effectués avec VFSR. Nous ne sommes pas parvenus à faire converger le recuit, qui tombe systématiquement dans un minimum local. Il est possible que notre maîtrise de VFSR soit insuffisante pour arriver au résultat. Cependant, du fait que les résultats que nous sommes parvenus à obtenir avec VFSR pour les trois fonctions précédentes sont proches de ceux présentés par Ingber, nous pensons fortement que la fonction de Griewank est difficile à optimiser pour un algorithme de recuit simulé.

La programmation par intervalle, en revanche, parvient à résoudre le problème avec une précision de 10^{-5} en 180026 évaluation de fonction pour $f(x)$ et 360053 évaluations pour $F(X)$. Il faut noter qu'il n'y a là aucun besoin d'appliquer une méthode locale en fin de convergence. Les résultats sont donc réellement intéressants.

Dans le cas présent, une évaluation de $F(X)$ est équivalente (en temps) à 10 évaluations de $f(x)$. On a représenté sur la figure 2.7 l'évolution de la valeur de f en fonction de (10*IterFX+Iterfx), les échelles étant là encore logarithmiques sur les deux axes.

2.2 Problèmes de type combinatoire

On peut appliquer les algorithmes génétiques sur les problèmes purement combinatoires, tels les problèmes SAT ou CSP [DS89, HD94], ainsi qu'au classique problème d'optimisation combinatoire qu'est le voyageur de commerce. C'est cet exemple que nous présentons ici. Notons cependant, avant de présenter cet exemple, que nous avons également testé les algorithmes génétiques sur des problèmes de satisfaction de contraintes purs, comme le classique problème du zèbre. Les

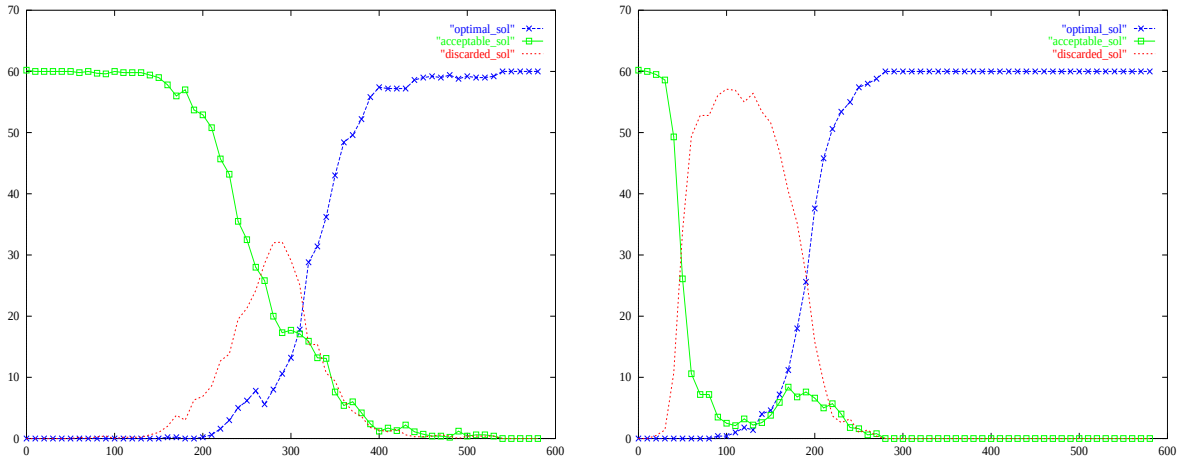


Figure 2.5: Croisement classique et croisement adapté sans sharing

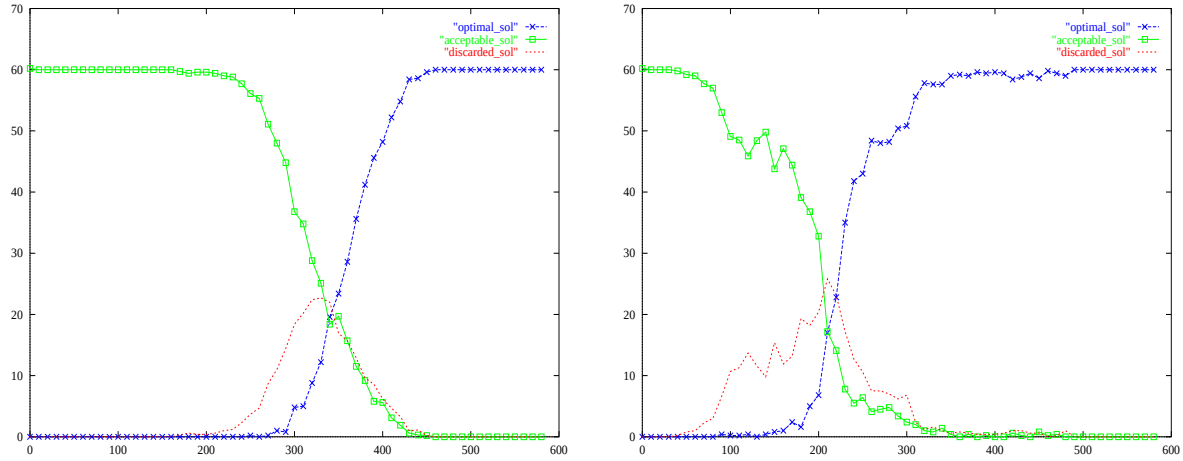


Figure 2.6: Croisement classique et croisement adapté avec sharing

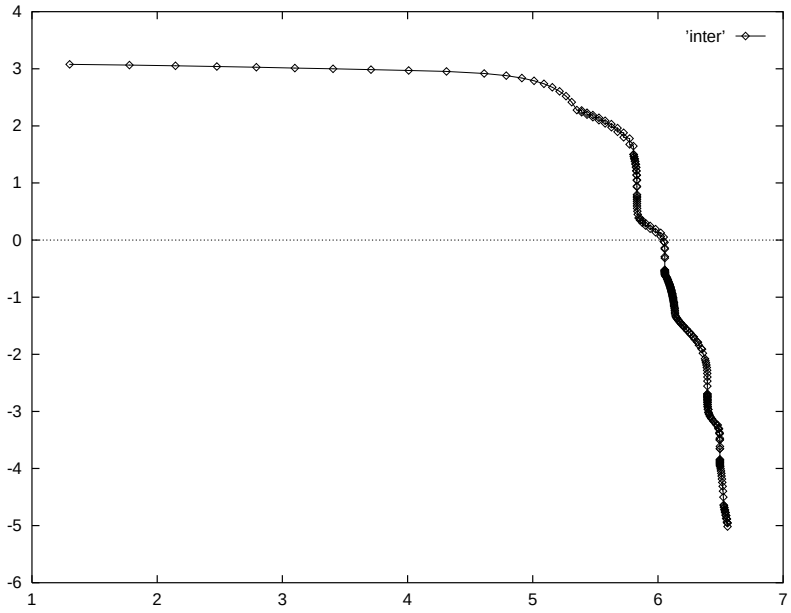


Figure 2.7: Convergence d'une méthode par intervalles sur la fonction de Griewank

résultats [Hue94] montrent que les méthodes "classiques" de satisfaction de contraintes restent plus efficaces que les AG.

2.2.1 Codage des données

Le problème du voyageur de commerce peut se coder de diverses façons, tenant compte des positions respectives des villes les unes par rapport aux autres ou non. Avec les méthodes de croisement classiques à un ou plusieurs points l'ordre dans lequel les villes sont codées a une influence sur la convergence. Pour notre algorithme, cet ordre n'a aucune importance. Les données seront donc codées sous forme d'une liste d'entiers représentant l'indice de la ville suivante. Ainsi le code $bcdea$ représente le chemin $abcdea$. Pour faciliter l'utilisation des divers opérateurs, nous aurons besoin de connaître le prédécesseur de la ville dans laquelle on se trouve. Pour éviter une recherche coûteuse en temps du prédécesseur, nous introduisons dans le codage une redondance : à chaque ville on associe l'indice de son successeur et celui de son prédécesseur. Ainsi le code (be, ca, db, ec, ad) représente le chemin $abcdea$.

Un tableau de distances est initialisé en début d'algorithme afin de limiter les calculs. Un deuxième tableau est créé dans lequel pour chaque ville, on classe dans l'ordre les villes les plus proches. Ceci permettra lors des opérations de croisement et de mutation de sélectionner rapidement des villes pas trop éloignées.

2.2.2 Opérateur de croisement

Au lieu de définir une seule fitness locale G_i, pour des raisons de symétrie, chaque variable ou ville sera dotée de deux fitness locales f_s et f_p. Soit n_{prof} un nombre compris entre 1 et la longueur totale du chromosome. Pour chaque ville à l'intérieur d'un chromosome, on détermine la longueur du chemin lorsque l'on parcourt n_{prof} villes dans le sens direct (c'est à dire en prenant la ville suivante à chaque fois). La fitness locale f_s représente donc cette valeur. On peut faire de même en parcourant

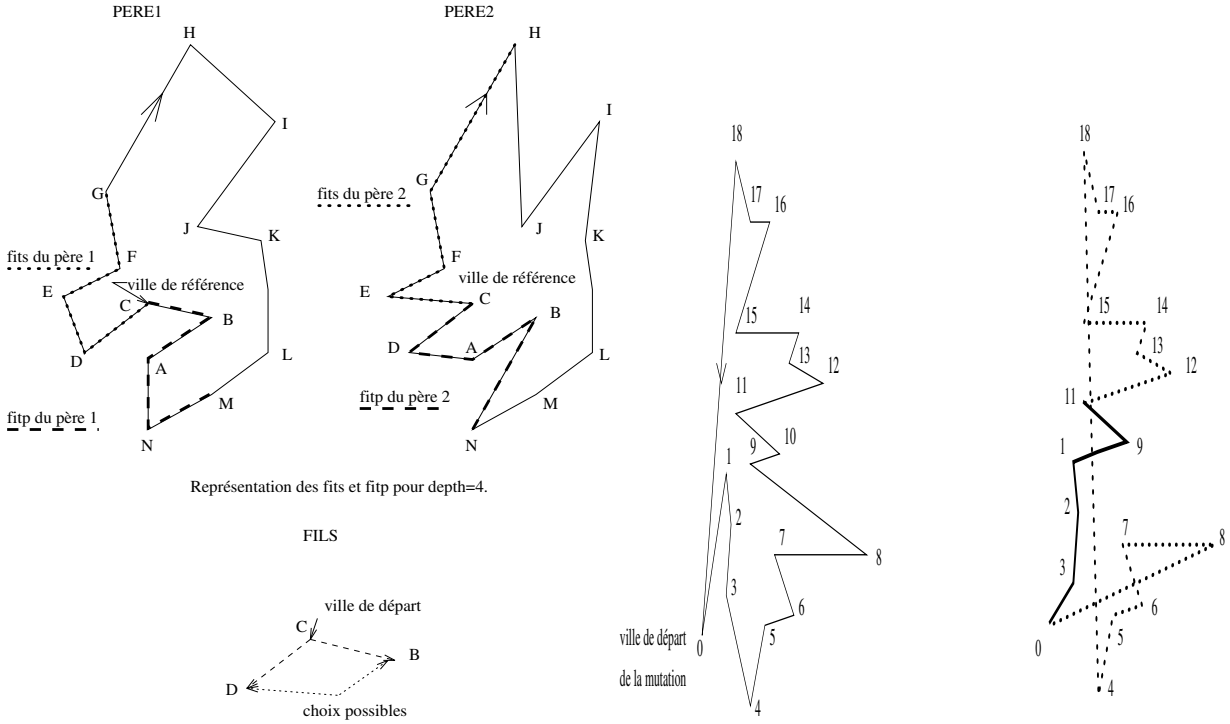


Figure 2.8: Stratégie de croisement et de mutation

les villes dans le sens opposé (c'est à dire en considérant chaque fois la ville précédente). On affectera à f_p cette valeur. Ainsi, notre codage s'enrichit. Il comporte désormais quatre listes au lieu de deux. Les deux premières listes permettent de définir l'ordre des villes dans le cycle, les deux dernières sont composées des fitness locales de chaque ville.

Le principe de notre croisement est le suivant : on choisit une ville au hasard qui constitue le point de départ de notre croisement. Le processus suivant est répété n fois où n est le nombre total de villes : pour choisir la ville suivante (voir figure 2.8), on va comparer les valeurs f_p et f_s des deux parents, représentant quatre chemins possibles. Parmi ces quatre chemins, certains parcourent des villes déjà utilisées. On pénalise alors les f_p, f_s des chemins correspondants. Le choix de la ville suivante se fera en considérant le chemin le plus court, à une certaine valeur Δ près. Si le chemin le plus court est distant de moins de Δ d'un autre, la ville suivante est choisie au hasard entre ces deux villes. On retrouve bien là le principe de croisement détaillé précédemment. La valeur Δ a pour but de ne pas rendre l'algorithme trop déterministe. Dans le cas où les quatre villes suivantes possibles ont déjà été utilisées, on choisira de façon heuristique la ville la plus proche disponible. Pour construire le deuxième fils, on effectue la même opération en prenant un point de départ différent.

Pour nos simulations, nous avons choisi Δ constante et valant la distance moyenne entre les villes. On peut imaginer que Δ décroisse au fur et à mesure des générations. Par contre, on a intérêt à choisir n_{prof} petit en début de convergence et de le faire croître de façon à prospecter de plus en plus loin dans les chemins possibles. Nous prendrons par exemple n_{prof} proportionnel au nombre de générations effectuées et variant entre 1 et $\frac{n}{5}$.

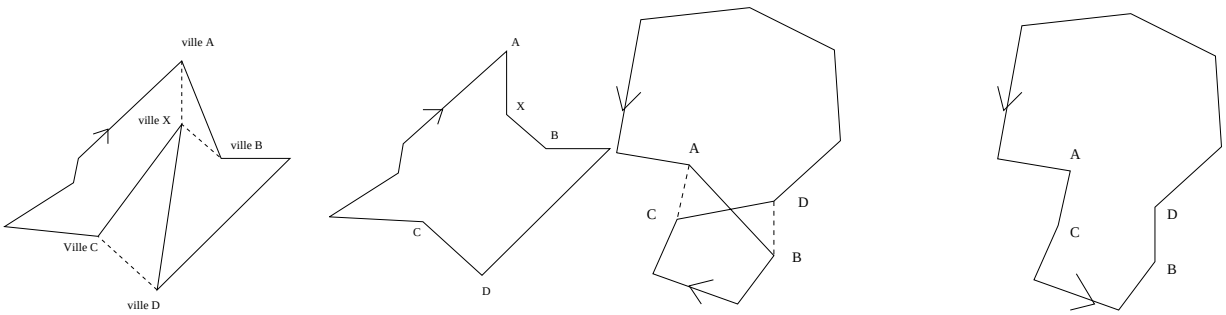


Figure 2.9: Les opérateurs de correction Loc_1 et Loc_2.

2.2.3 Opérateur de mutation

L'opérateur de mutation utilisé est très simple. Cependant, son rôle est utile pour parcourir le plus largement possible l'espace de recherche. Observons le tableau donnant pour chaque ville les villes les plus proches dans l'ordre des distances croissantes. Il est clair que si toutes les villes sont reliées à la ville la plus proche, on a un chemin optimal. En général, un certain nombre de villes ne sont pas reliées aux villes les plus proches au profit d'autres. Nous appellerons ville de rang k une ville dont la ville suivante est la $k^{\text{ième}}$ dans l'ordre croissant des villes les plus proches. Il est évident que passer d'une solution courante à la solution optimale du problème améliore au moins le rang d'une des n villes considérées. L'opérateur de mutation découle de cette observation. Son principe est de relier quelques villes à leurs villes voisines les plus proches et de compléter ensuite le chemin en parcourant l'ancien chemin dans l'ordre d'apparition des villes. La figure 2.8 donne un exemple de mutation sur un problème à 19 villes. L'une d'entre elles est choisie au hasard. La mutation consiste à rechercher cinq fois la ville la plus proche non parcourue, dans l'exemple, on parcourt les villes $0, 3, 2, 1, 9, 11$, et l'on complète ensuite le graphe avec les villes dans leur ancien ordre d'apparition.

2.2.4 Opérateurs locaux

Les opérateurs de croisement et de mutation précédemment décrits entraînent la formation de chemins imparfaits que l'on peut corriger trivialement. Nous introduisons ici deux opérateurs de correction que nous appellerons loc_1 et loc_2 et que nous appliquerons à tous les chromosomes avant chaque évaluation.

L'opérateur Loc_1

Cet opérateur consiste, pour chaque ville X (voir figure 2.9), à comparer la somme des coûts des arcs $XC + XD - CD$ avec les coûts $XA + XB - AB$ où A et B sont des villes successives proches de X. Si la deuxième somme est inférieure à la première, on reliera X aux villes A et B.

L'opérateur Loc_2

Cet opérateur a pour but de "déboucler les boucles". Pour chaque arc AB (voir figure 2.9, on cherche des villes consécutives proches C et D telles que la somme $AB + CD$ soit supérieure à $AC + BD$. On déboucle alors la boucle.

Il est à noter que ces deux opérateurs ne nécessitent pas que la distance entre les villes respecte l'inégalité triangulaire. Il n'est par ailleurs pas question de faire une recherche exhaustive de tous

Pb considéré	lin105	kroa100	kroa200
Taille du problème	105	100	200
Taille de la pop	50	50	100
Valeur du minimum	14379	21282	29368
Nb min de gens	2	2	40
Nb moy de gens	14	6	110
Nb max de gens	30	43	268
Temps min (en sec)	1	2	130
Temps moy (en sec)	6	3	441
Temps max (en sec)	12	17	1418

Tableau 2.3: Résultats numériques sans sharing du problème du voyageur de commerce.

les arcs pouvant être concernés mais seulement de regarder les villes proches de chaque ville ou arc concerné. Le coût de ces deux opérateurs est donc simplement proportionnel au nombre de villes considérées dans le problème.

2.2.5 Résultats numériques

Pour tous les résultats évoqués ci-dessous, les populations initiales sont construites de façon aléatoire. L'algorithme a été testé sur un certain nombre de problèmes classiques dont on connaît les solutions optimales. Pour chacun de ces problèmes, les paramètres des simulations sont les suivants :
Probabilité de croisement : 0.6
Probabilité de mutation : 0.15

Pour chacun de ces problèmes, nous avons fait tourner l'algorithme une cinquantaine de fois. Les simulations ont été réalisées sur HP720 sans méthode de sharing pour limiter le temps de calcul. L'algorithme a toujours trouvé la solution optimale. Les résultats sont présentés dans le tableau 2.3. La première ligne donne la référence du problème [Rei91]. La deuxième ligne donne sa taille. On donne ensuite la taille de la population utilisée pour résoudre le problème, la valeur numérique de l'optimum, le nombre minimum, moyen et maximum de générations pour résoudre le problème. Le temps minimum, moyen et maximum pour atteindre l'optimum.

Les résultats sont très bons pour des problèmes de l'ordre de 100 villes et tout à fait encourageants pour 200 villes. Ils montrent l'intérêt de cette approche locale utilisée pour le croisement. Cependant, l'algorithme génétique reste loin des méthodes les plus efficaces connues actuellement dans ce domaine.

2.3 Parallélisme et apprentissage

Nous allons présenter dans cette section une application des algorithmes génétiques à l'apprentissage d'une fonction d'évaluation pour un programme d'Othello. Il faut noter que des techniques d'apprentissage bayesiennes sont utilisées avec succès pour Othello depuis BILL [LM90], jusqu'aux programmes les plus modernes comme LOGISTELLO [Bur94]. Cependant, ces méthodes nécessitent de très grandes bases de données contenant des milliers de parties, et elle ne sont réellement efficaces que sur le jeu d'Othello, car la partie se termine toujours en un nombre fixe de coups. L'apprentissage bayésien ne s'étend pas à des jeux comme les échecs. Nous allons décrire ici une technique basée sur

500	-150	30	10	10	30	-150	500
-150	-250	0	0	0	0	-250	-150
30	0	1	2	2	1	0	30
10	0	2	16	16	2	0	10
10	0	2	16	16	2	0	10
30	0	1	2	2	1	0	30
-150	-250	0	0	0	0	-250	-150
500	-150	30	10	10	30	-150	500

Tableau 2.4: Valeur statique des cases

les AG qui peut s'étendre à tous les programmes de jeux reposant sur des algorithmes de type α-β.

Des tentatives ont été faites d'appliquer des techniques telles que la co-évolution [SG94], des réseaux de neurones appris par AG [MM94], ou des stratégies évolutives [MM93] à Othello. Mais ces tentatives, pour intéressantes qu'elles soient, n'ont pas donné de résultats particulièrement concluants. Le niveau de jeu de [SG94] était relativement faible, [MM94] ne parvenait pas à améliorer le niveau de jeu s'il utilisait des fonctions d'évaluation de bonne qualité comme fonctions test, et [MM93] ne parvint pas à améliorer la stratégie classique à Othello (un terme positionnel plus un terme de mobilité).

Nous allons, en ce qui nous concerne, partir d'un programme d'Othello et utiliser les AG pour améliorer la fonction d'évaluation (ces résultats sont également présentés dans [All95]).

2.3.1 Introduction

Le point de départ de ce travail est un programme d'Othello développé par l'auteur il y a quelques années. Il faut noter que ce programme a été testé contre de bons joueurs français (en particulier contre le numéro 10 français), et a obtenu des résultats fort honorables[4].

Ce programme a une structure extrêmement simple. Il utilise un algorithme α-β, avec une recherche en profondeur itérative et des fenêtres de coupure. Le programme déclenche une recherche exhaustive 11 à 14 demi-coups avant la fin de la partie, en fonction du temps restant et de la puissance de la machine. La fonction d'évaluation est composée de trois termes, que nous allons détailler.

Le premier terme de la fonction d'évaluation est purement statique ; on utilise la table 2.4 comme référence. Pour chaque disque sur le tableau de jeu, on additionne la valeur de la case correspondante si le disque est de la couleur du programme, ou on retranche cette valeur si le disque appartient à l'adversaire. Cette table, en raison des symétries, comporte 10 paramètres.

Le second terme a pur but de raffiner cette évaluation grossière. En effet, quand un coin du bord est déjà pris, les trois cases immédiatement adjacentes peuvent être occupées sans aucun risque, et leur valeur est donc ramenée à 0. Enfin, toujours lorsqu'un coin est occupé, tous les disques du bord de la même couleur et connectés à ce coin reçoivent un bonus. Par exemple, sur la figure 2.10, on voit que trois disques reçoivent un bonus. La valeur de ce bonus est un autre paramètre de la fonction d'évaluation.

Enfin, le troisième partie de la fonction d'évaluation est le facteur de mobilité ; on calcule pour chaque disque le nombre de libertés : il s'agit du nombre de cases libres autour de ce disque. On additionne ensuite les libertés de chaque disque pour un joueur donné (par exemple, sur la figure 2.10,

[4]Les meilleurs programmes d'Othello sont "presque" imbattables par un joueur humain.

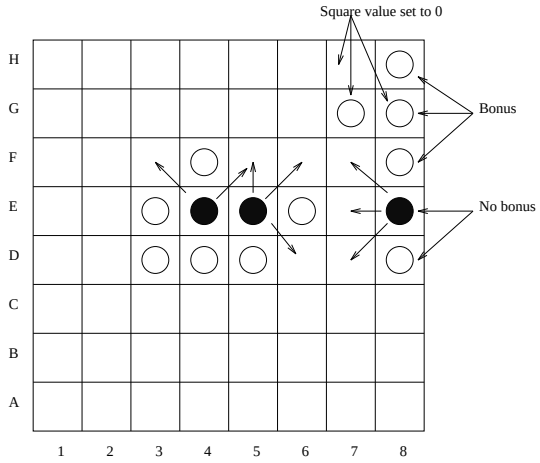


Figure 2.10: Evaluation des libertés et des bords

le joueur noir a 8 libertés). Le score de mobilité est la différence des libertés des deux joueurs. Ce score est multiplié par un coefficient avant d'être ajouté à la valeur de la fonction d'évaluation.

La fonction d'évaluation comporte donc 12 (10+1+1) paramètres. Les valeurs initialement prises par ces paramètres étaient [500, -150, -250, 30, 0, 1, 10, 0, 2, 16] pour les valeurs des cases, +30 pour le bonus de connexion, et 8 pour la pénalité de mobilité[5]. Ces valeurs ont été choisies de façon arbitraire. Elles "semblent" raisonnables, mais rien ne permet de penser qu'elles sont "optimales".

Nous allons maintenant décrire comment un AG peut-être utilisé pour améliorer les valeurs de ces coefficients.

2.3.2 Principes généraux

L'espace de recherche de l'algorithme génétique est l'ensemble des coefficients de la fonction d'évaluation. Cependant, comme la fonction d'évaluation est linéaire, on peut arbitrairement fixer la valeur d'un des coefficients. On ne garde alors que 11 paramètres au lieu de douze (dans le cas présent, nous avons fixé la valeur d'un angle à 500, car nous sommes sûrs qu'un angle doit avoir une pondération positive).

Le croisement utilisé est de type barycentrique. Un nombre α est tiré aléatoirement dans l'intervalle $[-0.5, 1.5]$ and un nombre i dans l'intervalle $[1, 11]$. On croise alors le ith élément de chaque chromosome. Le nouveau $i$ème composant de chaque chromosome vaut :

$$c'_{1i} = \alpha c_{1i} + (1 - \alpha) c_{2i}$$

$$c'_{2i} = (1 - \alpha) c_{1i} + \alpha c_{2i}$$

Le programme utilise également du sharing. La distance utilisée est la distance euclidienne sur l'espace des coefficients.

2.3.3 Calcul de la fitness

La première idée pour évaluer la fitness consiste à faire jouer chaque programme contre un programme de référence et à noter les résultats. Cependant, du fait que les résultats sont purement probabilistes,

[5] Il faut noter, qu'à Othello, le but est d'avoir le moins de libertés possibles, une liberté étant une case potentielle d'attaque pour l'adversaire.

il faudrait à chaque génération que chaque programme joue un très grand nombre de parties à des profondeurs d'évaluation différentes pour que le résultat soit fiable. Le temps d'évaluation devient alors complètement prohibitif. Nous avons donc décidé d'utiliser une technique mixte. Tout d'abord, un programme qui n'est pas modifié d'une génération à l'autre conserve la mémoire des matchs déjà joués. De cette façon, la valeur de l'intervalle de confiance sur la valeur de la fitness s'améliore au fil des parties.

La méthodologie choisie est donc la suivante pour chaque élément de la population :

- quatre positions de départ sont générées aléatoirement. Pour chacune de ces positions il y a au moins quatre disques (les quatre disques centraux), et au plus quatorze disques (les dix disques supplémentaires sont placés dans le sous damier central de 4×4).
- pour chacune de ces quatre positions, le programme joue alors une fois avec les noirs et une fois avec les blancs. Pour chacune de des deux parties, on calcule le nombre $s = n_b - n_n$ avec n_b le nombre de disques blancs et n_n le nombre de disques noirs à la fin de la partie. Puis on calcule la différence $s1 - s2$. Si le résultat est positif, on dit que le nouveau programme a gagné. Si le résultat est négatif, il a perdu. Si la différence est nulle, il y a match nul.
- on répète cette opération en faisant jouer le programme a trois profondeurs fixes différentes (0, 1 et 2).

On peut alors choisir comme fitness :

$$f = \frac{N_{victoires} + N_{nuls}/2 + D/1000}{N_{parties}}$$

où D représente les différences cumulées des pions en fin de partie.

Pourtant, cette technique n'est pas satisfaisante. On voit intuitivement qu'il est plus intéressant d'avoir un programme qui a gagné 95 parties sur 96 qu'un programme qui a gagné 12 parties sur 12. La confiance que nous avons sur le premier est forte, alors qu'elle est beaucoup plus faible pour le second. Nous avons alors défini l'estimateur suivant :

Si la probabilité de gagner une partie sur une jouée est notée p alors celle de gagner m parties sur n jouées est donnée par la loi binomiale :

$$p(m,n) = C_n^m p^m (1-p)^{n-m}$$

Ensuite, si n parties sont jouées, il y a $n+1$ résultats possibles[6] et dans l'espace des probabilités ils sont tous équiprobables, on en déduit :

$$\int_0^1 p(m,n)dp = \frac{1}{n+1}$$

Cette valeur f sera utilisée pour calculer l'adaptation de l'élément concerné ; mathématiquement, l'adaptation sera donc telle que :

$$\int_f^1 (n+1)\left(C_n^m p^m (1-p)^{n-m}\right) dp = 0.95$$

Construisons alors la fonction $h(p,m,n) = (n+1)p(m,n)$, elle va servir à établir un intervalle de confiance que nous fixerons à 95% : on désire trouver la valeur f à partir de laquelle il y a 95% de

[6] On gagne 0,1,. . .,$n-1$ ou n parties soit donc $n+1$ résultats distincts.

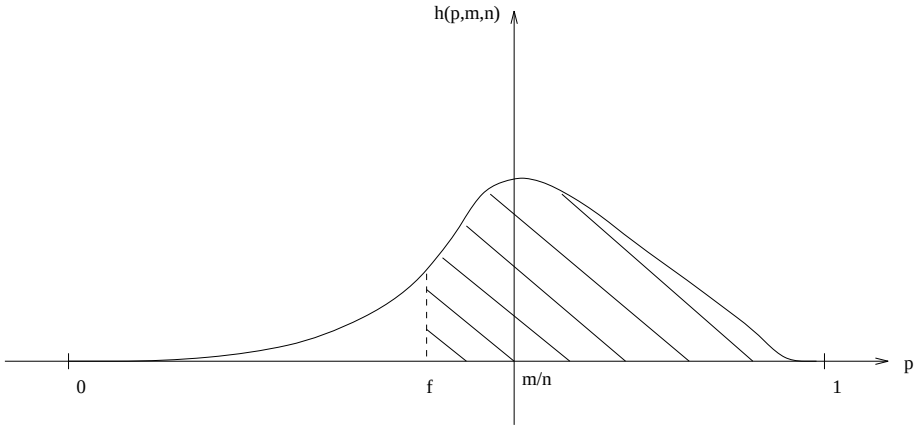


Figure 2.11: Courbe $h(p, m, n)$

chance pour que la fitness soit supérieure à f. On cherche donc la valeur telle que 95% de la surface se situant sous la courbe (représentée figure 2.11) se trouve après f.

Les valeurs de la fitness sont assez profondément modifiées. Ainsi, par exemple, $f(12, 12) = 0.79$, $f(23, 24) = 0.83$, $f(44, 48) = 0.82$ et $f(65, 72) = 0.83$. Cette technique, qui peut se généraliser à d'autres problèmes où les calculs de fitness sont statistiques, a donné d'excellents résultats.

2.3.4 Résultats expérimentaux

Pour traiter ce problème, la mise en œuvre du parallélisme a été indispensable. En effet, le temps d'évaluation d'un élément à chaque génération est de l'ordre de 15 secondes. Sachant que la fitness est évolutive, et que l'on utilise le recuit, il faudrait compter environ 18 minutes par génération pour une population de 40 éléments si l'on n'utilise pas le parallélisme. Les deux formes de parallélisme (simple et par îlots) ont été essayées[7]. Le parallélisme par îlot a donné les meilleurs résultats, sans doute parce qu'il provoque un clustering forcé.

On a tout d'abord développé un programme dit de niveau 1, qui a appris à jouer contre le programme de référence (niveau 0). Ce programme a ensuite été testé extensivement (sur 400 matchs) à toutes les profondeurs de 0 à 5, et sur 100 matchs à la profondeur 6. A partir de ce programme de niveau 1, on a développé un programme de niveau 2 qui était testé contre les programmes de niveau 1 et de niveau 0. Les résultats des différents tests sont présentés dans la table 2.5.

Il est remarquable de constater que, bien que l'apprentissage ne soit fait qu'aux profondeurs 0, 1 et 2, les "nouveaux" programmes battent les anciens de façon impressionnante jusqu'à la profondeur 6.

2.3.5 Conclusion

Les travaux décrits ci-dessus ont été poursuivis par l'auteur de ces lignes durant ses moments libres (hélas trop peu nombreux). Une nouvelle version du programme a vu le jour. Cette version utilise une fonction d'évaluation totalement différente : elle est bâtie sur une technique de "reinforcement learning" qui est appliquée à la reconnaissance de formes spécifiques sur les bords et dans les coins. Les coefficients de la fonction sont toujours appris par AG.

[7]Pour une présentation plus détaillée, en particulier sur les speedups obtenus, on peut se reporter à [LeF95].

Prof	G	P	N	G	P	N	G	P	N
0	282	98	20	188	194	18	274	107	19
1	252	101	47	285	82	33	266	82	41
2	278	78	44	303	55	42	243	120	37
3	281	75	44	298	50	52	292	61	47
4	280	75	45	320	46	34	320	46	41
5	286	68	46	329	40	31			
6	72	20	8						

Tableau 2.5: Résultats des programmes développés par AG

	Méthodes locales	Recuit	AG	Prog. Inter.
Optimum trouvé	oui	non	non	oui
Parallélisme	difficile	difficile	Intrinsèque	Facile
Optima multiple	non	non	oui	oui
Optimisation globale	non	oui	oui	oui
Nombreuses variables	efficace	efficace	efficace	inefficace

Tableau 2.6: Comparaison de différentes méthodes d'optimisation

Ce programme a été engagé sur le serveur international d'Othello (IOS), sous le nom d'`otage`. Il occupe actuellement la 4ème place en Blitz et la 9ème place en partie classique sur un ensemble d'environ 500 joueurs classés. Ses résultats contre certains des bons programmes mondiaux (`Bugs`, `Yapp`, `Hannibal`) indiquent que ce classement de 9ème doit approximativement correspondre à sa valeur réelle.

Il serait intéressant de poursuivre le travail sur `otage`, en particulier en raffinant l'algorithme de recherche, en introduisant des tables de hash, en lui faisant utiliser le temps de l'adversaire, en utilisant une bibliothèque d'ouvertures, en améliorant la qualité de la reconnaissance de patterns, etc. Le potentiel de progression du programme est, à mon avis, encore important.

2.4 Conclusion

Il est toujours délicat de conclure une étude de ce type. En effet, pour bien juger de la supériorité d'une méthode sur une autre méthode, il nous faudrait parfaitement connaître les deux. Il est en effet trop facile pour un spécialiste de la méthode A de démontrer la supériorité de ses algorithmes, simplement parce qu'il sait parfaitement utilisé son outil, alors qu'il ne maîtrise pas la méthode B à laquelle il se compare. Il faut donc être excessivement prudent lors de comparaison de plusieurs méthodes différentes sur un même problème. D'autre part, les problèmes traités sont fort différents les uns des autres, et l'on ne peut être que nuancé dans ses conclusions.

Nous pensons cependant que l'on peut retenir un certain nombre d'enseignements du travail que nous avons effectué au cours des dernières années ; les algorithmes génétiques sont efficaces lorsque :

- Il faut trouver plusieurs optima.
- La structure du problème permet d'utiliser au mieux le croisement, en y incluant par exemple la notion de fitness partielle et de séparabilité.

- Le temps de calcul de chaque fitness est élevé, et justifie donc l'emploi du parallélisme.
- il est possible de coupler l'algorithme génétique avec des méthodes locales.
- Il n'est pas indispensable de trouver l'optimum, mais seulement une "bonne solution".

En revanche, ils se révèlent relativement inefficaces dans les cas suivants :

- le problème est à variable réelle, et relativement simple ; il faut alors toujours essayer une méthode de convergence locale, ou une technique de recuit.
- le problème est purement combinatoire, en particulier du type programmation sous contraintes, il faut utiliser les techniques adaptées (CSP, etc.) plutôt que des AGs. Le travail fait par Denis Huet sur ce sujet [Hue94] est particulièrement instructif.

Le tableau 2.6 présente un résumé de ces éléments. De façon générale, les algorithmes génétiques ne doivent être employés que lorsque les méthodes plus classiques ne peuvent être utilisées. Il ne s'agit en aucun cas d'une panacée universelle susceptible de couvrir tous les domaines de l'optimisation.

Partie II

Applications aux problèmes du trafic aérien

When comfortably cruising in an airliner, sipping a glass of cold champagne and looking out of the window into the endless blue sky, you hardly ever see another man-made flying object. At the same time, the aviation paper that you are reading speaks about the capacity crisis in ATC. So, there seems to be a contradiction : the skies are empty, but the ATC radar screens are full.[...] In aviation, the introduction of new technologies in the air and on the ground have progressed out of step.

Carlos Garcia-Avello et Sip Swierstra

Chapitre 3

Optimisation de la résolution de conflits

Introduction

Le but premier du contrôle du trafic aérien est d'assurer la sécurité des aéronefs. Avant toute chose, nous allons introduire le vocabulaire indispensable à la description du problème :

Contrôle en route : il s'agit du contrôle à l'extérieur des zones entourant les aéroports (dans ces zones, on parle de *contrôle d'approche*). Les avions utilisent alors le plus souvent des routes préétablies, les *couloirs aériens* (en anglais *airways*). Ces couloirs sont des *tubes* de sections rectangulaires. Historiquement, les routes ont été définies comme étant les segments de droite reliant les *balises*. Ces balises étaient généralement des moyens de radionavigation vers lesquels se dirigeaient les avions. C'est souvent autour de ces balises qu'apparaissent des *conflits* ; en effet, elles se situent souvent aux intersections des différentes routes.

Ces routes contournent les parties d'espace réservées aux militaires et permettent aux contrôleurs d'avoir une visualisation plus aisée de la situation spatiale des avions.

Plan de vol : il contient tous les éléments indicatifs décrivant le vol prévu pour un avion. Ces informations sont déposées avant le départ auprès des services ATC avec, notamment, les éléments suivants :

- l'heure de départ ;
- le niveau de vol[1] demandé pour la croisière ;
- la route prévue : elle est décrite par une série de balises.

Séparations : on définit une distance horizontale, qui est exprimée en milles nautiques[2] (Nm), la *séparation horizontale*, et une distance verticale, qui, elle, est exprimée en pieds[3] (ft) : la *séparation verticale*. On dit que deux avions sont *séparés* quand la distance qui sépare leurs projections sur un plan horizontal est supérieure à la séparation horizontale OU quand la distance qui sépare leurs projections sur un plan vertical est supérieure à la séparation verticale. Ces *séparations* peuvent être différentes selon le type de zone dans laquelle se trouvent les avions, ou selon les outils de contrôle dont on dispose. Dans le cas du trafic *en route*, qui nous occupe

[1] Le niveau de vol ou altitude-pression est l'altitude de l'avion exprimée en centaines de pieds. Par exemple le FL290 (FL pour Flight Level) correspond à une altitude de 29000 pieds soit environ 9000 m.

[2] 1 mille nautique vaut 1852 m (1 minute d'angle sur un méridien).

[3] 1 pied vaut $30,48$ cm.

ici, la séparation horizontale est généralement de 8 Nm en France mais devrait bientôt descendre à 5 Nm. La séparation verticale vaut 1000 ft au dessous du FL 290 et 2000 au dessus.

Conflit élémentaire : deux avions sont dits en *conflit* quand ces avions ne sont plus *séparés*, c'est à dire quand les distances séparant leurs projections sur un plan horizontal et sur un plan vertical sont inférieures respectivement à la séparation standard horizontale ET à la séparation standard verticale. De façon plus générale, si l'on se fixe une durée T, deux avions seront dits en conflit potentiel pendant T si durant le temps T ils ont une probabilité non nulle d'être en conflit.

Cluster : un *cluster* d'avions est une fermeture transitive d'avions en conflits potentiels. Si l'avion A est en conflit potentiel avec l'avion B et si l'avion B est en conflit potentiel avec les avions C et D, alors on dit que les avions A, B, C et D appartiennent au même cluster. On trouvera par la suite l'expression *conflit à n avions* qui signifie en fait cluster à n avions.

Le rôle du contrôleur est donc de prévenir les risques de conflit en donnant aux avions des ordres de contrôle (changement de cap ou d'altitude, etc.) permettant d'éviter les conflits.

Mais la gestion du trafic aérien ne se réduit pas à la seule action du contrôleur sur les avions. Il s'agit en fait d'un système complexe que l'on peut décrire comme une succession de cinq filtres agissant à des niveaux différents :

Organisation à long terme : c'est le filtre le plus grossier. Son but n'est pas au sens strict d'éviter des conflits mais plutôt d'organiser le trafic de façon macroscopique à moyen et long terme (supérieur à 6 mois). On peut citer à titre d'exemple, les schémas d'orientation de trafic, les mesures du Comité des horaires ou encore, les accords inter-centres et les accords avec les militaires qui permettent aux civils d'utiliser leurs zones aériennes pour écouler les pointes de trafic du vendredi après-midi .

Organisation à court terme : on parle souvent de pré-régulation. Il consiste à organiser une journée de trafic j la veille $(j-1)$ ou l'avant-veille $(j-2)$. On dispose pour cela de données relativement précises :

- les plans de vol déjà connus ;
- la capacité de contrôle que peut offrir chaque centre en fonction des effectifs qui seront présents le jour j ;
- le nombre d'avions maximum pouvant se trouver dans un même secteur au même instant, encore appelé capacité du secteur ;
- les données des années et des semaines précédentes. En effet, le trafic aérien est relativement répétitif : le lundi ressemble au lundi de la semaine passée, le pont de l'Ascension ressemble à celui de l'année passée, etc. Ceci permet de prévoir où vont apparaître les congestions, la capacité qu'il va falloir mettre en œuvre pour répondre à la demande, voire éventuellement les mesures plus contraignantes à prendre.

Ce filtre a non seulement une action macroscopique car il organise les flux de trafic en fonction des capacités offertes mais également une action ponctuelle sur chaque avion en gérant les créneaux de décollage[4]. Effectué au niveau national jusqu'en 1995, ce filtrage est maintenant fait au niveau européen pour obtenir une meilleure coordination. C'est le rôle de la CFMU[5].

[4]Un créneau de décollage est une fenêtre temporelle pendant laquelle l'avion est autorisé à décoller.

[5]Central Flow Management Unit

Régulation en temps réel : il consiste à organiser les différents flux en tenant compte des événements du jour. Il s'agit plutôt de mesures d'ajustement qui prennent en compte des événements encore mal connus la veille. Ainsi, le trafic transatlantique est mal connu à $j-1$ mais est bien connu entre 3 à 6 heures avant son arrivée. La fraction de la capacité offerte qui avait été réservée par la pré-régulation peut alors être adaptée. On peut envoyer sur d'autres centres les avions supplémentaires ou augmenter le nombre de vols non transatlantiques traités par le centre s'il y a moins de vols transatlantiques que prévu. De la même façon, on peut ré-allouer des créneaux horaires non utilisés pour une raison quelconque (retard ou incident technique) ou tenir compte de la météo du jour (terrains inaccessibles par exemple). Ce rôle est en général joué par les FMP[6] dans chaque centre.

Tactique : c'est le dernier filtre de la chaîne du contrôle aérien : l'action du contrôleur sur son secteur. Le temps moyen passé par un avion dans un secteur est de l'ordre d'une quinzaine de minutes. La visibilité du contrôleur est un peu supérieure puisqu'il dispose des plans de vol quelques minutes avant l'entrée de l'avion dans le secteur.

La détection et surtout la résolution des conflits ne sont absolument pas automatisées. Les contrôleurs sont donc entraînés à reconnaître des types de conflits et à appliquer à ces derniers des manœuvres connues. Le contrôleur ne peut éviter les conflits qu'en jouant sur la trajectoire des avions. Pour cela, il dispose de quatre types d'ordres :

- des déviations de cap ;
- des paliers intermédiaires pour les avions en montée ou en descente ;
- des changements de niveaux de vol.
- des mesures de régulation de vitesse (essentiellement sur des avions en descente) ;

Cependant les résolutions proposées ne sont pas toujours optimales. En outre, une fois une manœuvre d'évitement choisie, le contrôleur doit également surveiller les avions impliqués pour vérifier que tout se passe bien. Tout ceci se fait au détriment de la charge[7] de travail du contrôleur et affecte donc la capacité du secteur dont il est responsable.

Urgence : Ce filtre n'est censé intervenir que lorsque le système de contrôle est absent ou a été défaillant. L'objet de ce filtre n'est plus de séparer les avions (il est trop tard dans la plupart des cas) mais plutôt d'éviter une collision présumée. La prédiction temporelle est inférieure à la minute et varie entre 25 et 40 secondes. Il est trop tard pour que le contrôleur intervienne puisque l'on estime qu'il lui faut entre 1 et 2 mn pour analyser une situation, trouver une solution et la communiquer aux avions. Actuellement, cette fonction est assurée par le TCAS[8] qui détecte les avions environnants et donne un ordre de résolution au pilote (pour le moment dans le plan vertical).

Ce filtre doit résoudre les conflits non prévisibles comme par exemple dans le cas où un avion dépasse un niveau de vol donné par le contrôle ou dans le cas d'un accident technique qui dégraderait notablement les performances de l'avion.

[6] Flow Management Position

[7] La charge de travail prend en compte des tâches de surveillance, communication, détection et résolution des conflits.

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

Depuis 1980, le contrôle du trafic aérien connaît une progression régulière, qui fait craindre une saturation de l'espace. De plus, le nombre de données à traiter augmentera encore avec la mise en place de nouveaux moyens de communication entre le sol et les avions (Data-Link). Parallèlement, les avions s'équipent de moyens sophistiqués de navigation (FMS[9], GPS[10]) permettant des tenues plus précises de trajectoires.

Dans ce cadre, il faut s'interroger sur l'évolution des techniques de contrôle pour les années à venir. On se retrouve alors pris entre deux positions caricaturales : l'automatisation complète ou le refus complet de toute automatisation. Les tenants de l'une et l'autre approche ont des discours qui relèvent plus de l'émotionnel que du rationnel. On peut cependant penser que :

- Le refus de toute automatisation est une politique à courte vue ; l'opérateur humain est actuellement le goulot d'étranglement du système de contrôle. On peut certes espérer améliorer ses capacités de traitement en améliorant la présentation des informations ; mais si l'augmentation du trafic se poursuit dans les mêmes proportions, il faudra déléguer au moins partiellement les tâches de résolution et de surveillance à la machine.

- Une automatisation complète est irréaliste dans le contexte actuel, pour tout un ensemble de raisons ; certaines ont un caractère politique ou social : la transition d'un contrôle où l'homme a une responsabilité complète à un contrôle où il n'aurait qu'une activité de supervision pose de sérieux problèmes éthiques. D'autre part, il n'existe pour l'instant aucun système capable de résoudre l'ensemble des problèmes techniques qui se posent dans ce cadre.

La voie moyenne consiste sans doute à penser un système qui, dans un premier temps, serait susceptible de fournir une aide à la détection et à la résolution des conflits, et pourrait, à la demande de l'opérateur, effectuer automatiquement certaines résolutions. Le Centre d'Etudes de la Navigation Aérienne s'est intéressé à cette hypothèse dans le cadre de certains de ses projets, comme SPECTRA[Pla93].

D'autres projets, comme ARC-2000 (développé par le Centre Expérimental Eurocontrol de Brétigny), ont exploré des voies plus futuristes, tout en souhaitant utiliser dans les systèmes de contrôle actuels les algorithmes et concepts développés par des équipes de recherche pour les systèmes de contrôle du futur. Les problèmes d'évolution des systèmes de contrôle sont discutés plus en détail dans [DAM93], et l'on trouvera un large panorama des différents projets d'automatisation dans [Dur96].

Dans tous les cas, il faut savoir prévoir les trajectoires des avions, détecter les conflits, regrouper ces conflits par paquets, et donner des manœuvres simples les résolvant, etc. Ces problèmes sont aujourd'hui identifiés, mais les techniques à employer pour les résoudre le sont moins. On peut globalement classer les approches employés pas les différentes équipes en deux catégories, qui ne sont d'ailleurs pas sans rappeler deux écoles de pensée du monde de l'Intelligence Artificielle :

- la première approche consiste à construire un modèle, dit "cognitif", du contrôleur, spécialement dans ses modes d'appréhension des paramètres du conflit, et d'utiliser ce modèle dans un calculateur afin de construire des outils de filtrage de l'information et d'assistance électronique au contrôleur. Cette approche (à laquelle l'auteur de ces lignes participa fort modestement à ses débuts [LA91], alors qu'il terminait sa thèse de doctorat au sein de l'équipe "formalisation du raisonnement" à l'Institut de Recherche en Informatique de Toulouse) possède le grand avantage d'être facilement admissible pour un contrôleur actuellement en activité : en effet, il n'aura pas à changer son mode opératoire, puisque la machine sera censée le reproduire. Pourtant, cette

[9] Flight Management System
[10] Global Positionning System

problématique souffre directement des limitations propres aux approches dites “expertes” : coût très élevé de développement et de maintenance, problème de généralisation des maquettes au cas général, dégradation brutale de l’expertise aux limites du domaine, faillibilité du système expert, etc. Ces problèmes ont été mis en évidence dans de nombreux domaines, et sont largement présentés et discutés dans la littérature spécialisée depuis de nombreuses années maintenant ([Dre84, Dre79, Dre87, Fox90, Dav82]), nous n’y reviendrons pas ici[11]. Ils semblent néanmoins justifier que l’on n’applique ce type d’approche qu’aux problèmes sur lesquels les approches plus classiques ne sauraient donner de résultats probants.

- La seconde approche, que nous qualifierons de pragmatique, consiste à étudier mathématiquement le problème et à appliquer les algorithmes susceptibles de donner les meilleurs résultats possibles sans trop se soucier de savoir si ces algorithmes reproduisent ou non le raisonnement humain. On peut d’ailleurs rappeler ce qu’écrivent [AVAD52] : “Une première légende prétend qu’il faut être un expert pour réaliser un bon programme. Pourtant, le développement d’un programme est un problème cybernétique qui demande au moins autant de travail sur les algorithmes que de connaissance du domaine. Cette dernière peut même avoir une influence négative, en raison de sa non formalisation. Une seconde légende dit qu’un ordinateur devrait penser “comme un être humain”, et que ceci est un dogme absolu. Pourtant, si le but est de faire résoudre à un ordinateur des problèmes complexes, nous nous compliquons la tâche de façon injustifiée en tentant d’abord de résoudre le problème bien plus complexe de le faire raisonner comme nous.”

  L’approche pragmatique a été employé dans AERA-III [Cel90, SPSS83, NFC$^+$83, Nie89b, Nie89a, PA91] aux Etats-Unis. De la même façon, la philosophie du système ARC-2000 [K$^+$89, FMT93, MG94] du Centre Expérimental Eurocontrol est trés proche de la nôtre. Comme l’écrivent [DFMN95] : “Le démonstrateur ARC-2000 est basé sur une approche algorithmique, par opposition à une approche de type système expert, en ce qui concerne le problème de détection et de résolution de conflit. Les expérimentations ont montré qu’il était possible d’adopter une approche algorithmique de façon efficace en temps réel et avec une charge de trafic très importante. Il s’agit d’un avantage par rapport à l’approche “système expert”, dans la mesure où cela permet d’avoir un moyen plus standardisé et plus “certifiable” de résoudre les problèmes”. Le travail que nous avons réalisé depuis 1992, et que nous allons présenter dans ce chapitre, s’inscrit également dans cette lignée.

Il faut également signaler que l’on a souvent trop tendance à confondre le but que l’on souhaite atteindre (automatisation complète, partielle ou simple aide au contrôleur) avec les techniques qu’il faut employer pour arriver à ses fins. Ce n’est pas parce que l’on souhaite fournir une aide au contrôleur, et non automatiser, que seule les méthodes “cognitives” sont applicables. On peut, par exemple, constater que l’aide la plus appréciable par tous de l’informatique a certainement été la machine à calculer, un objet fort peu cognitif. Je me plais souvent à rappeler ce petit texte d’un éminent spécialiste de l’IA, Jonathan Schaeffer : “je considère l’esprit humain comme une machine, comme je considère mon ordinateur comme une machine. Ces deux machines ont leurs forces et leurs faiblesses. Mon cerveau est par exemple très fort pour la reconnaissance de formes et de situations, alors que mon ordinateur est très bon pour répéter des tâches et effectuer des calculs. En revanche, j’ai un sérieux problème à additionner un million de nombres en une second, et mon ordinateur n’est pas très doué pour le langage naturel, ou la reconnaissance de situations. Étant donné que chaque machine a des

[11] Fort peu modestement, nous suggérons au lecteur intéressé mais pressé de se reporter à [AS92] pages 25–39, 358–361 et 463–484 pour un résumé des divers arguments.

capacités différentes, il faut essayer d'exploiter les forces des machines et non leurs faiblesses." Ainsi, les algorithmes issus du système ARC-2000 sont en cours d'intégrations dans un système d'aide à la résolution de conflits [DFMN95] : "Les principes généraux de résolution d'Arc-2000 ont prouvé leur grand efficacité. Ils sont actuellement appliqués de façon très satisfaisante à l'intérieur d'un système d'aide centré sur l'opérateur humain, appelé Highly Interactive Problem Solver". Il n'y a pas opposition entre des algorithmes permettant de construire automatiquement des solutions, et des systèmes susceptibles de fournir de l'aide au contrôle. Il faut simplement remplir deux conditions : construire des algorithmes efficaces et fiables, quelle que soit leur structure, et présenter correctement l'information issue de ces algorithmes, de façon à la rendre directement utilisable. Si on peut discuter certains choix d'ARC-2000 (en particulier la non prise en compte des incertitudes), on ne peut en revanche que souscrire à la méthodologie appliquée.

3.1 Etude du trafic

3.1.1 Simulation du trafic

La première étape, avant de s'attaquer à la résolution de conflits, est de disposer d'un outil permettant de simuler le trafic afin de pouvoir étudier la sensibilité à chacun des paramètres, et de façon à pouvoir évaluer les différentes méthodes de résolution. Nous disposons actuellement d'un simulateur de trafic (Banc de Test) permettant de "rejouer" une journée complète du trafic français, en utilisant des données plan de vol établies à partir des archives du CAUTRA (GOETHE). Il fait voler les avions selon leur route prévue ou en route directe. Le modèle avion utilisé est très simple[12] (figure 3.43). Il utilise en entrée le niveau de vol de l'avion ainsi que son attitude (montée, descente, en palier). Le modèle avion produit alors en sortie la vitesse sol, ainsi que le taux de montée ou de descente. On ne peux pas intervenir sur les taux de montée ou de descente, ni modifier la vitesse de l'avion.

Les avions suivent soit la route déposée dans le plan de vol (routes indirectes), soit une ligne droite entre le premier et le dernier point de leur route déposée (routes directes). A titre d'exemple, le simulateur introduit un écart de 4 mn par rapport à un vol réel Paris-Toulouse, phase d'atterrissage non comprise. L'écart moyen sur l'ensemble des vols, par rapport aux durées prévues, est de 13 s d'avance par vol (écart-type 6'53") en routes indirectes, et 2'52" d'avance (écart-type 8'14") en routes directes, du fait du raccourcissement des trajectoires. Par ailleurs, il est possible de simuler un accroissement de la densité de trafic en comprimant les dates de décollage.

Enfin, une procédure de réattribution aléatoire des niveaux de vol permet d'augmenter le nombre de niveaux disponibles lorsque la séparation est réduite. Cette redistribution se fait par ajout d'une gaussienne d'écart-type 1500 ft sur le niveau de vol demandé, puis attribution du niveau effectif immédiatement inférieur à la valeur obtenue. Il n'y a pas doublement de l'espacement des niveaux au-dessus du FL 295.

A intervalles réguliers, le simulateur enregistre le nombre d'avions en vol par tranche de niveau, détecte les conflits selon les normes de séparation fixées, et enregistre les positions des avions aux dates de début et de fin de conflit. Un utilitaire permet de reconstituer les clusters, c'est-à-dire les ensembles d'avions en conflit 2 à 2 par fermeture transitive.

Le temps de simulation est d'environ 3 mn pour une journée de trafic, avec détection des conflits.

Les résultats sont observés dans une fenêtre temporelle d'étude, sur laquelles on va déterminer les clusters et extraire des données statistiques. L'utilisation d'une fenêtre est motivée par l'accroissement des incertitudes sur les positions prévues des avions. La planification des trajectoires ne peut-être

[12] Il s'agit en fait du modèle CAUTRA

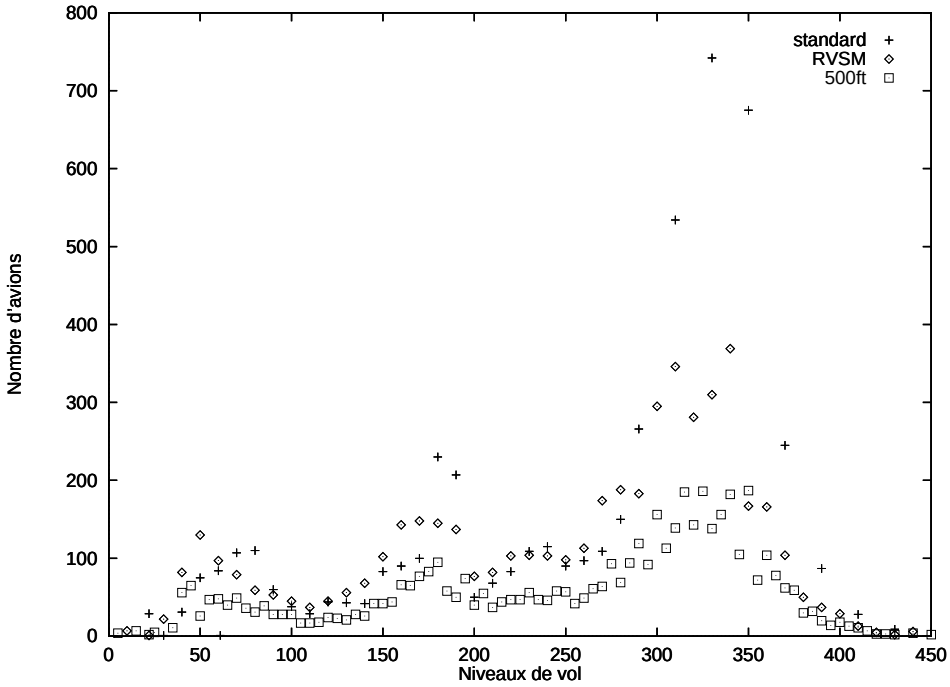


Figure 3.1: Distribution du trafic

efficace qu'en deçà d'un certain horizon temporel. Les clusters (ensembles d'avions en interaction) sont définis par fermeture transitive sur les conflits survenus dans la fenêtre.

Jean-François Bosc [Bos94a, Bos94b, Bos94c, Bos95a, Bos95b, Bos95c, Bos96a, Bos96b] a utilisé le banc de test en vue de déterminer l'influence sur le trafic, en termes de nombre de conflits et de clusters, des différents paramètres, que nous avons fait varier de la manière suivante :

- séparations horizontales (NM) : 1, 2, 4, 6, 8, 10
- séparations verticales (ft) : 300, 500, 800, 1000, 1300, 1500, 1800, 2000 (sans doublement au-dessus du FL195)
- facteur de compression de temps : 1, 1.25, 1.5, 1.75, 2, 2.5, 4

soit $6^3 = 216$ simulations, effectuées en routes indirectes, en routes directes sans résolution, et en routes directes avec résolution. On utilise une redistribution aléatoire des niveaux de vol afin de tenir compte de la variation du nombre de niveaux disponibles lorsqu'on change la séparation verticale.

Par ailleurs, afin d'étudier l'influence de l'horizon temporel, nous avons utilisé une fenêtre d'étude de longueur variable, prise dans une période de trafic important et stable (typiquement entre 13 et 14 h). La durée des fenêtres varie de 10 à 60 mn par pas de 10 mn. La figure 3.1 montre la répartition des vols par niveau dans l'échantillon initial, avec utilisation des RVSM, et avec réduction des séparations à 500 ft.

Les valeurs mesurées dans chaque fenêtre sont les suivantes :

- nombre d'avions en vol,
- nombre de conflits,

- nombre de conflits stable-stable, stable-évolutif, et évolutif-évolutif,
- nombre de clusters,
- moyenne, écart-type et maximum du nombre de conflits par cluster,
- moyenne, écart-type et maximum du nombre d'avions par cluster,
- minimum du nombre d'avions dans les clusters dont le nombre de conflits est égal au maximum,
- moyenne et maximum de l'écart-type par cluster du nombre de conflits par avion. Ces valeurs fournissent une indication sur la structure des clusters.

D'autre part, on mesure sur la journée complète les valeurs suivantes :

- nombre de conflits (total et détaille en stable-stable, stable-évolutif, et évolutif-évolutif),
- temps de vol moyen,
- délai moyen et écart-type.

Enfin, on mesure pour chaque avion les heures de départ et d'arrivée et le retard, afin de fournir une loi d'évolution des délais pris dans la fenêtre, ce qui est plus réaliste qu'une observation sur la journée. En effet, les résultats sur la journée sont faussés par la répartition irrégulière du trafic.

3.1.2 Résultats

Les mesures relevées en routes directes et indirectes sont présentées dans les tables 3.1 (trafic actuel) et 3.2 (accroissement de 150% de la densité de trafic). Les deux dernières colonnes donnent les écarts en pourcentage dûs aux routes directes. On peut noter les points suivants concernant les routes directes par rapport aux routes indirectes :

- réduction de 6 à 7% du volume de trafic. Ceci est directement lié à la réduction des temps de vol permise par les routes directes (6,4% en moyenne sur la journée).
- réduction très nette du nombre de conflits et de clusters. Cette réduction est beaucoup moins marquée pour les conflits entre deux avions évolutifs.
- réduction de la taille des clusters (nombre d'avions et de conflits).

On remarque toutefois que le gain en nombres de conflits obtenu en routes directes s'explique en grande partie par l'augmentation implicite du volume d'espace disponible, qui entraîne une diminution de la densité réelle du trafic.

La faible réduction du nombre de vols amenée par les RVSM s'explique par le fait que les avions dont le niveau de vol est modifié volent en général plus bas, et donc gagnent un peu de temps dans les phases de montée et de descente. Par contre le gain en nombre de conflits est important, du fait que les niveaux les plus chargés sont situés au-dessus du FL195 (voir figure 3.1).

On constate d'autre part que l'introduction des séparations RVSM touche essentiellement les conflits stable-stable, ce qui correspond à ce que l'on pouvait attendre. Les conflits impliquant au moins un avion évolutif sont rarement affectés.

Routes	ind		dir		écart (%)	
Séparation verticale	STD	RVSM	STD	RVSM	STD	RVSM
Nb de vols dans la fenêtre	225	221	210	207	-6.7	-6
Nb de clusters	49	42	33	21	-33	-50
Nb de conflits	69	56	40	30	-42	-46
stable-stable	22	9	11	4	-50	-56
stable-évolutif	35	35	20	14	-43	-60
évolutif-évolutif	12	12	9	12	-25	0
Moy du nb de conflits par cluster	1.41	1.33	1.21	1.43	-14	+7.5
Ecart-type du nb de conflits par cluster	0.70	0.64	0.54	0.73	-23	+14
Max du nb de conflits par cluster	3	3	3	3	–	–
Moy du nb d'avions par cluster	2.35	2.31	2.18	2.24	-7	-3
Ecart-type du nb d'avions par cluster	0.62	0.60	0.46	0.43	-26	-28
Max du nb d'avions par cluster	4	4	4	3	–	-25
Moy écart-type nb conflits par avion	0.12	0.11	0.06	0.11	-50	–
Max écart-type nb conflits par avion	0.87	0.87	0.50	0.82	-43	-6

Tableau 3.1: Trafic actuel.

Routes	ind		dir		écart (%)	
Séparation verticale	STD	RVSM	STD	RVSM	STD	RVSM
Nb de vols dans la fenêtre	569	559	531	521	-6.7	-6.8
Nb de clusters	224	202	170	139	-24	-31
Nb de conflits	491	355	271	215	-45	-39
stable-stable	181	72	84	39	-54	-46
stable-évolutif	200	179	99	96	-50	-46
évolutif-évolutif	110	104	88	80	-20	-23
Moy du nb de conflits par cluster	2.19	1.76	1.59	1.55	-27	-12
Ecart-type du nb de conflits par cluster	4.29	1.96	1.43	1.26	-67	-36
Max du nb de conflits par cluster	60	18	13	8	-78	-55
Moy du nb d'avions par cluster	3.00	2.63	2.51	2.50	-16	-5
Ecart-type nb d'avions par cluster	3.28	1.59	1.14	1.09	-65	-31
Max du nb d'avions par cluster	47	15	11	8	-77	-47
Moy écart-type nb conflits par avion	0.23	0.16	0.16	0.15	-30	-6
Max écart-type nb conflits par avion	1.27	1.50	1.20	1.16	-5.5	-23

Tableau 3.2: Accroissement de 150%.

Volume de trafic

On va tout d'abord s'intéresser aux valeurs mesurées dans la fenêtre temporelle. On obtient dans tous les cas (routes indirectes, et routes directes avec et sans résolution) une corrélation entre compression de temps et volume de trafic (nombre d'avions en vol dans la fenêtre) tout à fait conforme aux prévisions :

$$N_{avions} = K_1 C^{\alpha}$$

avec $\alpha = 1$, C étant la compression de temps. L'écart relevé sur la valeur de α est inférieur à 1.5%, et le taux de corrélation est de 99.9%. La taille de la fenêtre n'a donc pas d'incidence, tout au moins sur le trafic moyen observé dans la fenêtre. Il ne semble pas y avoir d'effet de bord (chute du trafic en fin de fenêtre) sensible, même pour les fenêtres les plus longue (jusqu'à 1 h) avec une forte compression de temps (qui réduit la durée de la plage de fort trafic).

Conflits et clusters

On remarque que les différents paramètres sont indépendants les uns des autres. On peut s'attendre à ce que le nombre de conflits augmente avec chacun des paramètres, et soit nul pour des valeurs nulles de ceux-ci. On cherche donc à modéliser le nombre de conflits sous la forme suivante :

$$N_{conf} = K C^c H^h V^v I^i$$

C étant la compression de temps, H la norme de séparation horizontale, V la norme de séparation verticale, et I l'intervalle d'étude.

Routes indirectes : Les résultats sont présentés dans la table 3.3. On remarque tout d'abord d'excellents coefficients de corrélation, qui montrent que le modèle choisi est adéquat.

On constate que le nombre de conflits augmente à peu près linéairement en fonction du carré de la densité de trafic, de la séparation verticale, et de l'intervalle d'étude. En ce qui concerne la séparation horizontale, le coefficient est un peu inférieur aux prévisions (les modèles, notamment [Ale70, Bos94c], prévoient une évolution linéaire en fonction de la norme, ou inversement proportionnelle au nombre de niveaux). Ceci est vraisemblablement dû aux conflits comprenant au moins un avion évolutif, pour lesquels la séparation verticale a peu d'incidence. On retrouve d'ailleurs une valeur proche de 1 pour les conflits stable-stable. De plus, la distribution des avions sur les niveaux de vol est imparfaite. On remarque quelques anomalies (écarts entre niveaux de même parité) sur la figure 3.1.

La valeur élevé du coefficient de la norme horizontale pour les conflits stable-évolutif, que l'on retrouve en routes directes (sans résolution), n'a pu être expliquée. Toutefois, [EO83] prévoit une composante en H^2 (correspondant au volume parcouru par le disque de rayon H) lorsque les vitesses verticales ne sont pas toutes identiques. Mais il reste a expliquer le coefficient plus faible pour les conflits stable-stable. De même pour le coefficient inférieur à 1 sur la taille de fenêtre d'observation, puisqu'on a vu qu'il ne semblait pas y avoir d'effet de bord.

Routes directes Les valeurs obtenues sont données dans la table 3.4. On obtient encore une corrélation très élevée, et les coefficients sont à peu près similaires à ceux relevés en routes indirectes. L'anomalie sur la norme horizontale est plus marquée.

Par ailleurs, l'évolution du nombre de conflits en fonction d'un paramètre lorsqu'on laisse les 3 autres fixes est visualisée sur les figures 3.2 et 3.3. Les valeurs fixées sont de 1 pour la compression

	Conflits				Clusters
	total	stab-stab	stab-évol	évol-évol	
c	1.95	2.24	1.81	1.91	1.61
h	0.99	0.68	1.30	0.87	0.68
v	0.75	0.92	0.71	0.69	0.52
i	0.90	0.91	1.08	0.67	0.80
r^2	0.98	0.94	0.96	0.95	0.95

Tableau 3.3: Conflits et clusters en routes indirectes

	Conflits				Clusters
	total	stab-stab	stab-évol	évol-évol	
c	2.16	2.04	2.07	2.08	1.81
h	1.34	1.00	1.48	1.28	1.05
v	0.85	0.96	0.86	0.72	0.69
i	0.92	0.86	0.93	0.81	0.83
r^2	0.97	0.92	0.94	0.93	0.93

Tableau 3.4: Conflits et clusters en routes directes sans résolution

de temps (trafic actuel), 6 NM pour la norme horizontale, 1000 ft pour la norme verticale (RVSM), et 30 mn pour la durée de la fenêtre. Sur la figure 3.2, le trafic standard correspond à une densité de 1.

L'évolution du nombre de clusters en fonction d'un paramètre lorsqu'on laisse les 3 autres fixes est visualisée sur les figures 3.4, et 3.5.

Enfin l'évolution du retard moyen (en secondes) observé dans la fenêtre apparaît sur la figure 3.6. La norme verticale et la durée de la fenêtre n'ont quasiment pas d'incidence. Sur chaque figure, une courbe correspond à l'ensemble des configurations de densité et de norme, l'autre est limitée à celles pour lesquelles au moins les deux tiers des conflits sont résolus.

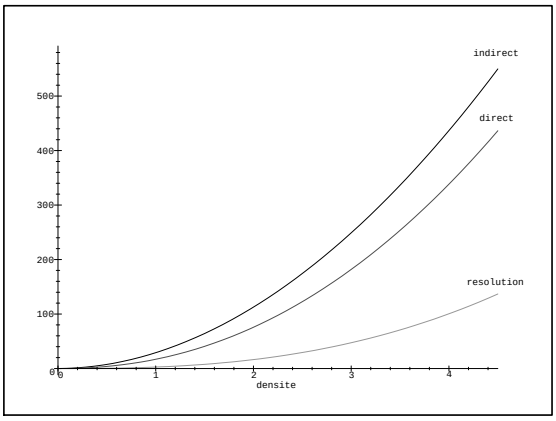
3.1.3 Conclusion

L'étude précédente montre que l'accroissement du nombre de conflit en fonction de l'accroissement du trafic est quadratique, alors que la diminution des normes de séparation n'a qu'une influence linéaire. La diminution des normes de séparation ne saurait donc être une panacée pour diminuer le nombre de conflits. Le problème de la résolution de conflits va donc se poser de façon de plus en plus aigu dans les années à venir.

3.2 Approche mathématique

Il est toujours indispensable de tenter de résoudre analytiquement un problème et d'évaluer sa complexité[13] avant de se lancer dans une approche de quelqu'autre forme que ce soit. Le principal travail effectué sur ce thème l'a été par Nicolas Durand dans le cadre de sa thèse [Dur96]. Ce travail a d'ailleurs fait l'objet de plusieurs publications [DAAS94, DAN94, DAN96b, DCA96, DAN96a, DA96].

[13] Au sens de la théorie de la complexité bien entendu...



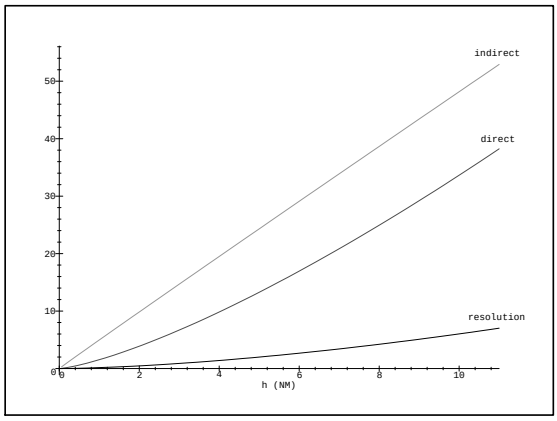
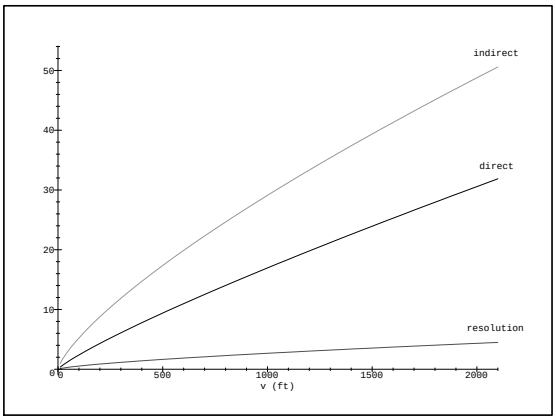


Figure 3.2: Influence de la densité et de la norme horizontale sur le nombre de conflits.



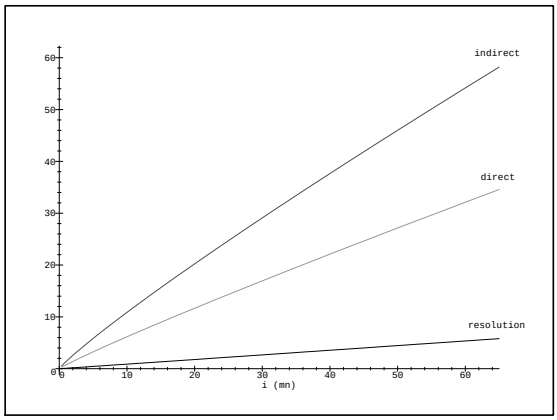
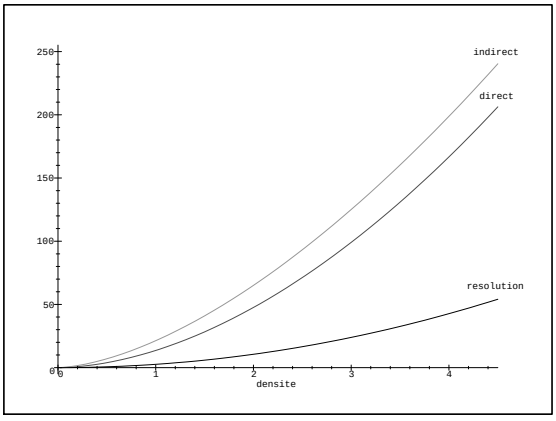


Figure 3.3: Influence de la norme verticale et de l'intervalle sur le nombre de conflits.



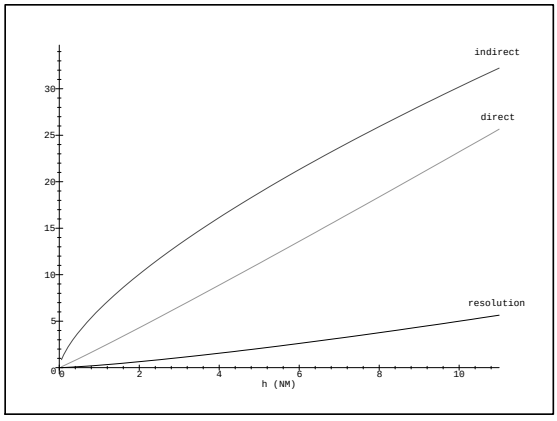


Figure 3.4: Influence de la densité et de la norme horizontale sur le nombre de clusters.

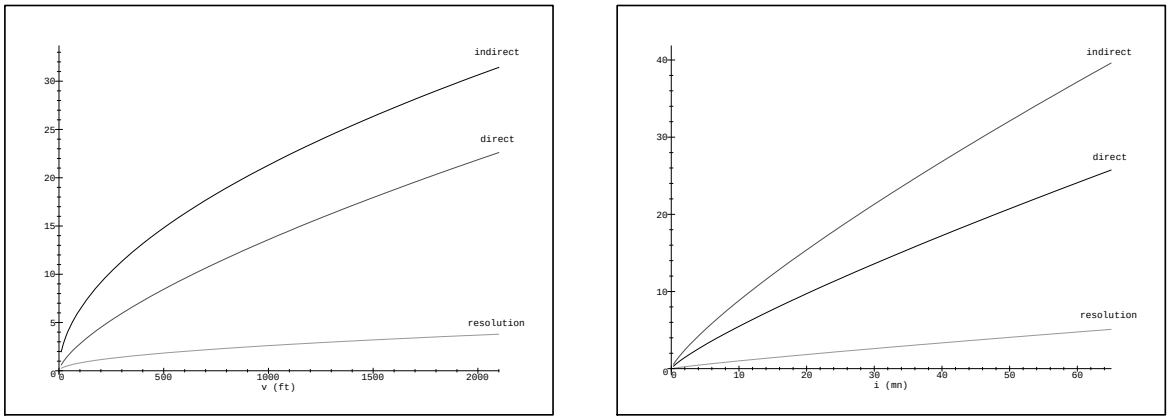


Figure 3.5: Influence de la norme verticale et de l'intervalle sur le nombre de clusters.

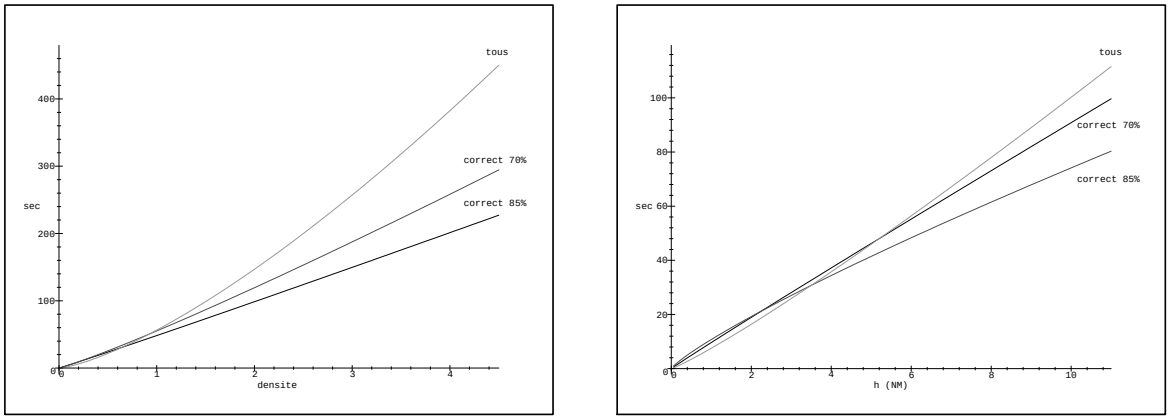


Figure 3.6: Influence de la densité et de la norme horizontale sur les retards.

3.2.1 Conflit entre des avions non contraints en vitesse

Résolution par une méthode de commande optimale

L'aéronautique utilise depuis longtemps maintenant les outils du Contrôle Optimal pour optimiser les trajectoires d'un avion [Cle90, MZ88, BH91, ME88, Sch90]. L'application de ces techniques au problème de la résolution de conflit de deux avions non contraints en vitesse est intéressante, mais montre rapidement ses limites, en raison de la forte complexité du problème.

On essaie de construire une résolution minimisant la consommation des avions, que l'on suppose proportionnelle à l'intégrale de la vitesse au carré. Il est alors possible de montrer que le problème possède deux propriétés intéressantes :

- Le centre de gravité des n avions a un mouvement rectiligne et uniforme.
- Dans le referentiel barycentrique :

$$\forall t \in [t_0, t_f], \sum_{i=1}^{n} u_{ig}(t) \wedge x_{ig}(t) \quad = \quad Cte$$

  où u_{ig} représente le vecteur vitesse et x_{ig} le vecteur position de l'avion i dans le référentiel barycentrique.

Dans le cas général de n avions, l'utilisation de ces deux invariants ne permet pas de résoudre le problème. En revanche, si l'on restreint le problème à deux avions, on peut déterminer analytiquement la solution optimale : les avions, pour s'éviter, tournent autour de leur isobarycentre à vitesse constante, isobarycentre qui se déplace lui-même à vitesse constante. On montre également que l'optimum dépend de l'éloignement en temps des avions par rapport au point de croisement mais pas de la vitesse de chacun des avions, et on peut construire une interprétation géométrique permettant de déterminer immédiatement le meilleur minimum.

Il faut cependant savoir que, dans la réalité, les avions sont fortement contraints en vitesse, ce qui rend caduc tout espoir de résolution "optimale" en terme de consommation.

Faisabilité de la résolution en vitesse

La régulation en vitesse, aujourd'hui pratiquée principalement dans les phases de descente et d'approche a l'avantage de ne pas changer le support de la trajectoire des avions. Néanmoins, elle nécessite des tenues de vitesses rigoureuses et des temps d'anticipation variables selon l'angle de conflit et sa gravité. Nous analysons ici de façon théorique quels sont les temps d'anticipations nécessaires pour une régulation en vitesse, en supposant que les tenues de trajectoires $4\,D$ des avions sont parfaites.

La figure 3.7 représente le temps d'anticipation minimum pour résoudre un conflit entre deux avions ayant la même vitesse (rapport des vitesses, $r = 1$). On représente en ordonnée la capacité e mesurée en % qu'un avion a de ralentir ou d'accélérer et en abscisse l'angle a de convergence des deux avions. Les courbes iso-temps parcourent les points pour lesquels le temps d'anticipation minimal est constant. La courbe intérieure correspond à un temps d'anticipation de 5 minutes. En allant de l'intérieur vers l'extérieur, le temps augmente par pas de 1 minute.

On observe ainsi que pour $e \geq 15\%$ et des angles de trajectoires entre 10 et 120 degrés, le temps d'anticipation reste inférieur à 10 minutes dans le pire des cas (c'est à dire lorsque les avions sont prévus au même instant au point de croisement des trajectoires). La résolution en vitesse peut alors

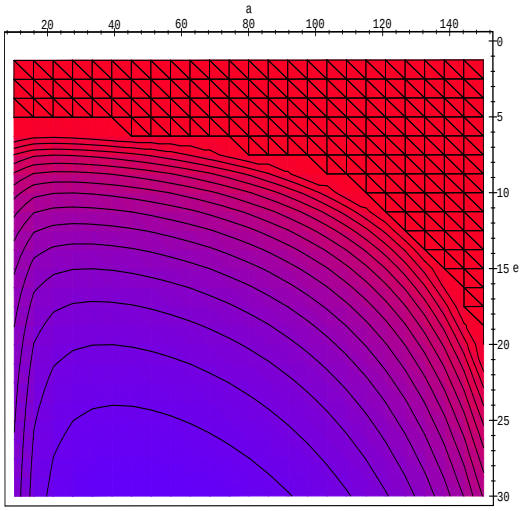


Figure 3.7: Temps d'anticipation en minutes en fonction de e et de a ($r = 1$).

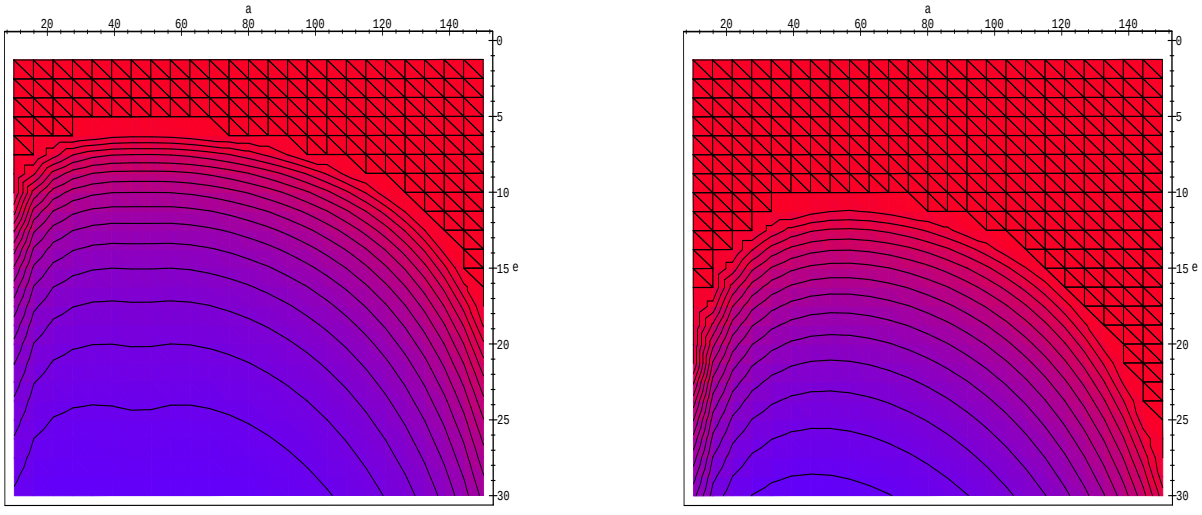


Figure 3.8: Temps d'anticipation en minutes en fonction de e et de a, à gauche : $r = 1.5$ - à droite : $r = 0.5$.

s'appliquer de façon relativement systématique. En revanche, si la marge d'accélération ou de ralentissement des deux avions est inférieure à 5%, on ne peut effectuer de résolution en vitesse si l'on ne prend pas une marge de plus de 20 minutes.

La figure 3.8 représente pour des rapports de vitesse $r = 1.5$ et $r = 0.5$ le temps d'anticipation nécessaire toujours en fonction de l'angle a des trajectoires et de e. On observe que pour $r > 1$ (on intervient alors sur l'avion le plus lent) les résultats précédents restent bons. Si l'on intervient sur l'avion le plus rapide, il faut intervenir plus tôt.

L'observation des courbes précédentes montre qu'une régulation en vitesse est envisageable pour une majorité de conflits dans la mesure où la marge de régulation sur les avions est suffisamment importante. Ainsi, si l'on peut faire varier la vitesse des avions de plus de 15%, on peut espérer résoudre en vitesse bon nombre de conflits.

Nous n'avons pas jusqu'à maintenant évoqué les problèmes d'incertitude sur les tenues de vitesse des avions. Il est clair que si les avions ne peuvent tenir leur vitesse de façon précise, la régulation en vitesse devient difficile. Dans les hypothèses que nous avons choisies, nous avons toujours considéré les conflits dans le pire des cas, à savoir lorsque les deux avions sont supposés passer par le point d'intersection de leurs trajectoires au même instant. Il est clair que si deux avions sont déjà partiellement séparés, une régulation en vitesse peut permettre de les séparer complètement facilement. Néanmoins, plus on anticipe la régulation, plus l'incertitude joue un rôle important. Ainsi, à 400 nœuds, une incertitude de 1% pendant 20 minutes représente 1.33 nautiques. Aujourd'hui peu d'avions sont équipés de $FMS - 4D$ permettant de telles précisions. Ceci explique pourquoi les contrôleurs n'utilise la régulation en vitesse qu'en phase de descente, les marges de manœuvres étant alors plus importantes.

3.2.2 Conflit entre deux avions contraints en vitesse

Résolution par une méthode de commande optimale

On va donc maintenant s'intéresser au cas de deux avions contraints en vitesse. Il s'agit du cas le plus proche de la réalité en matière de contrôle en route. On minimisera ici l'allongement de la trajectoire. La condition nécessaire de résolution permet de décrire les différentes phases de trajectoires. Lorsque la contrainte de séparation n'est pas saturée, les trajectoires des avions sont rectilignes uniformes, leurs caps sont modifiés uniquement lorsque la contrainte de séparation est saturée. Ce résultat reste valable pour le cas de n avions. La contrainte sur les vitesses ne permet pas d'aller plus loin dans la résolution analytique du conflit. De plus, lorsqu'un conflit à deux avions se présente, il n'est pas "rentable" en terme de coût pour le contrôleur aérien de dévier les deux avions. En conséquence, dans la suite, la trajectoire d'un des deux avions sera privilégiée (elle sera maintenue rectiligne et uniforme).

On peut, lorsque l'on privilégie un des deux avions en lui assurant une trajectoire rectiligne uniforme jusqu'à son objectif, s'intéresser à l'allongement de la deuxième trajectoire engendré par le conflit. On peut étudier le cas simple de 2 avions se croisant perpendiculairement et volant à la même vitesse. On notera d le rapport (norme de séparation) / (distance au point de conflit).

Dans le repère de l'avion non dévié, on peut (voir figure 3.9) représenter l'origine et la destination de l'avion dévié, ainsi que la zone interdite pour l'avion dévié, à savoir le cercle de centre l'origine et de rayon d.

On sait déjà qu'à l'optimum, l'avion dévié rejoint ce cercle en ligne droite en un point repéré par l'angle θ_1, y reste jusqu'à θ_2 et ensuite rejoint sa destination en ligne droite. On va chercher alors à exprimer les valeurs de x_+ et x_- correspondant aux deux cas possibles : l'avion dévié passe devant l'avion fixe, ou derrière celui-ci. On obtient alors les équations implicites donnant les valeurs des

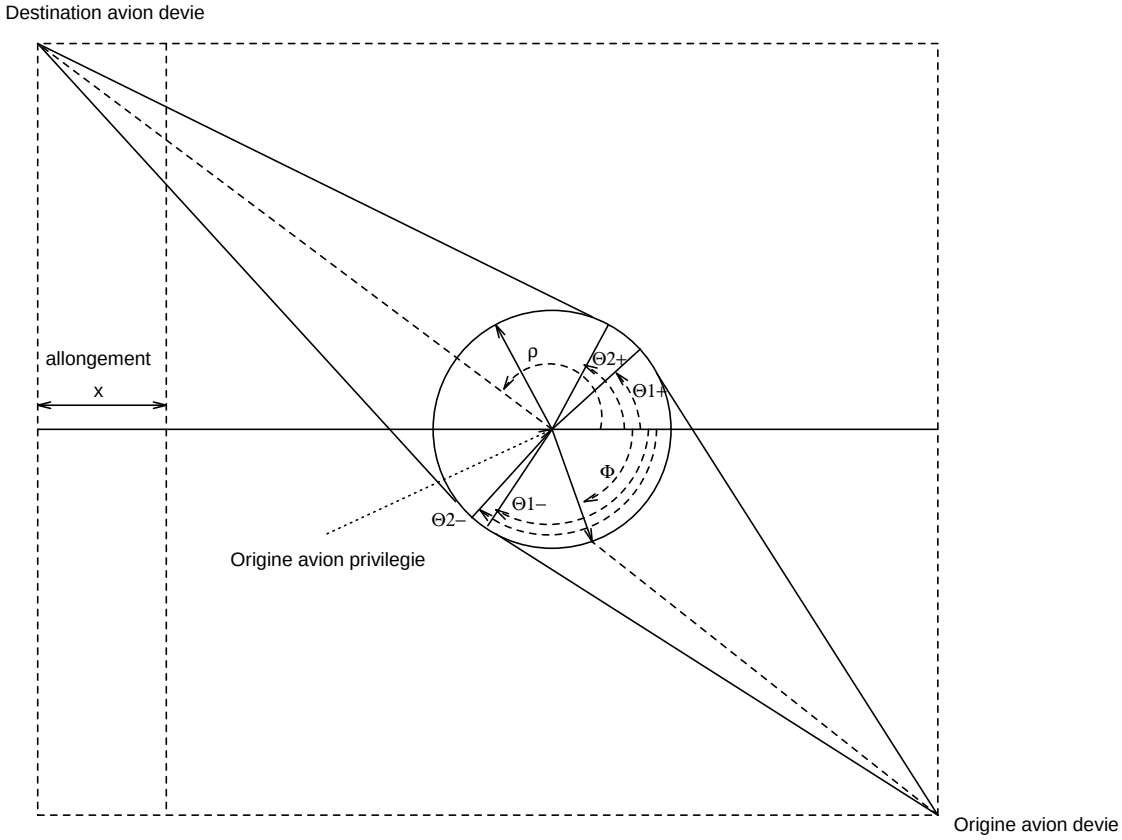


Figure 3.9: Trajectoires de l'avion dévié dans le repère lié à l'avion non dévié.

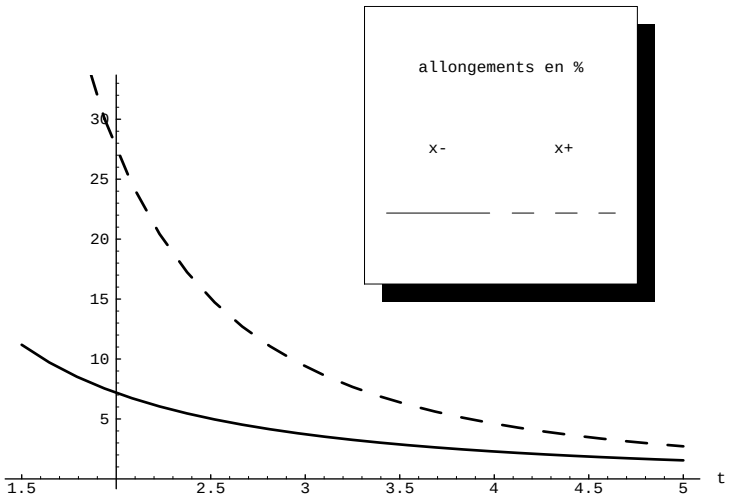


Figure 3.10: Comparaison des allongements minimaux (en $\%$) pour t supérieur à 1.5.

optima :

$$2 + x_+ = \frac{1}{1 - \frac{d}{\sqrt{2-d^2}}} + \frac{2 + 2\,x_+ + x_+^2}{2\left(1 + x_+ + \frac{d}{\sqrt{2+2\,x_+ + x_+^2 - d^2}}\right)} + \frac{d}{2}\log\left(\frac{(1+d)\left(\sqrt{2 + 2\,x_+ + x_+^2 - d^2} - 1\right)}{(1 + x_+ - d)\left(\sqrt{2-d^2} - 1\right)}\right)$$

$$2 + x_- = \frac{1}{1 + \frac{d}{\sqrt{2-d^2}}} + \frac{2 + 2\,x_- + x_-^2}{2\left(1 + x_- - \frac{d}{\sqrt{2+2\,x_- + x_-^2 - d^2}}\right)} + \frac{d}{2}\log\left(\frac{(1+d)\left(\sqrt{2 + 2\,x_- + x_-^2 - d^2} + 1\right)}{(1 + x_- - d)\left(\sqrt{2-d^2} + 1\right)}\right)$$

Ceci nous définit des fonctions implicites $x_+(d)$ et $x_-(d)$. Si l'on pose $d = \frac{1}{t}$, on peut représenter les pourcentages d'allongements de trajectoires en fonction de l'anticipation de la résolution du conflit (voir figure 3.10). L'unité de temps est alors le temps mis par un avion pour parcourir la valeur d'une norme de séparation. On observe que les pourcentages d'allongement de trajectoire restent très faible tant que t est supérieur à 2. Dans notre exemple, la trajectoire optimale est toujours celle qui passe derrière l'avion non dévié.

On peut assez facilement étendre l'étude précédente au cas de deux avions convergent non plus à angle droit mais sous un angle quelconque. Toujours en ne bougeant qu'un seul avion, on peut déterminer les deux équations implicites qui nous donnent l'allongement de trajectoire du deuxième avion dans le cas où celui-ci passe devant ou derrière.

Enfin, on peut représenter en fonction de l'angle d'incidence et du rapport des vitesses des deux avions, l'allongement minimal obtenu, pour une anticipation de, par exemple, cinq fois la norme de séparation (voir figure 3.11). On observe alors que les situations les plus pénalisantes dans le cas où les vitesses sont voisines et les angles d'incidence faibles, ce qui reste conforme aux résultats précédents ainsi qu'aux résultats décrits par AERA 3.

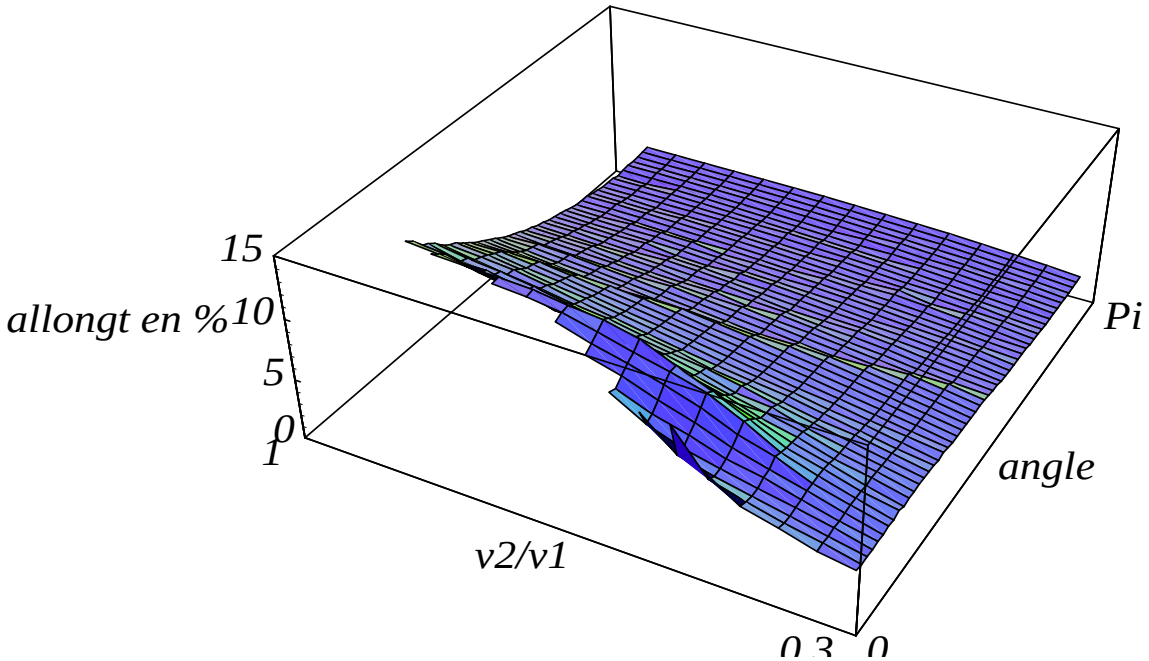


Figure 3.11: Allongements minimaux en fonction de l'angle d'incidence et du rapport des vitesses pour une anticipation valant deux fois la norme de séparation.

Approche numérique

Nous présentons ici les solutions optimales (fournies par à un algorithme d'optimisation locale[14]) de certains conflits à deux avions.

La figure 3.12 donnent six façon de résoudre un conflit à cinq avions. 6 points de départ différents ont été donnés à LANCELOT pour la recherche de la solution optimale donnant 6 solutions localement optimales situées dans des composantes connexes différentes. Il existe probablement bien plus que six composantes connexes disjointes pour ce problème, mais le but de ce paragraphe n'est pas de les chercher toutes. On remarquera que les solutions, apparemment très différentes, génèrent des allongements moyens très voisins.

On notera que le temps de calcul nécessaire à la convergence de LANCELOT est très long : pour les exemples à 5 avions présentés ci-dessous, plusieurs heures sont nécessaires. Envisager une résolution en temps réel avec ce type d'algorithme ne parait pas raisonnable.

3.2.3 Complexité théorique

Nous venons de constater que le problème de résolution de conflits est difficile à résoudre, même numériquement, dès que le nombre d'avions augmente. Il est en fait possible d'établir un résultat général concernant la complexité du problème, dans le cas où un des avions a un cap et une vitesse maintenus fixes, et où la norme et la direction du vecteur vitesse de l'avion mobile ne peuvent changer qu'en un nombre fini d'instants[15].

On peut alors montrer que l'espace des trajectoires admissibles pour l'avion mobile se compose de deux composantes connexes indépendantes, qui correspondent l'une aux trajectoires obtenues quand l'avion mobile passe devant l'avion dit "fixe", et l'autre au cas ou l'avion mobile passe derrière l'avion fixe au point de conflit.

Dans le cas où l'on s'intéresse à un conflit comprenant n avions on se retrouve face à $\frac{n\,(n-1)}{2}$ conflits potentiels. L'espace des solutions admissibles contient donc $2^{\frac{n\,(n-1)}{2}}$ composantes connexes. Il est clair que dans la pratique, compte-tenu des performances des avions, toutes les composantes connexes n'ont pas besoin d'être explorées. Néanmoins, la présence théorique d'autant d'espaces disjoints et l'impossibilité de connaître à priori lequel contient la solution optimale nous a orienté vers des méthode d'optimisation globale plutôt que vers des méthodes locales.

3.2.4 Conclusion

La modélisation par Contrôle Optimal que nous avons abordée dans cette première partie permet de dégager certaines propriétés sur les trajectoires optimales et sur la structure de l'espace des solutions admissibles.

Nous retiendrons tout d'abord que les trajectoires non contraintes sont rectilignes et uniformes à l'optimum. Dans la pratique, la durée de saturation des contraintes, compte-tenu de la vitesse élevée des avions et de la faible densité de trafic, est très faible comparée à la durée de vol sans saturation de contrainte (dans la mesure où les conflits ne sont pas à angle trop faible). En conséquence, la plupart des trajectoires optimales peuvent être approchées par des trajectoires rectilignes par morceaux (c'est

[14] L'algorithme d'optimisation local utilisé est (LANCELOT)[CGT92]. Pour pouvoir l'appliquer, les trajectoires d'avions ont été discrétisées en 20 points. Les contraintes de séparation des avions sont appliquées en chaque point de discrétisation. Dans les exemples présentés, les avions sont toujours contraints en vitesse. La norme de séparation est de 8 nautiques et les avions parcourent en moyenne 120 nautiques à une vitesse de 400 nœuds.

[15] On pose également comme hypothèse que l'avion mobile ne "tournera" pas autour de l'autre avion, ce qui effectivement ne se produit pas souvent dans la réalité.

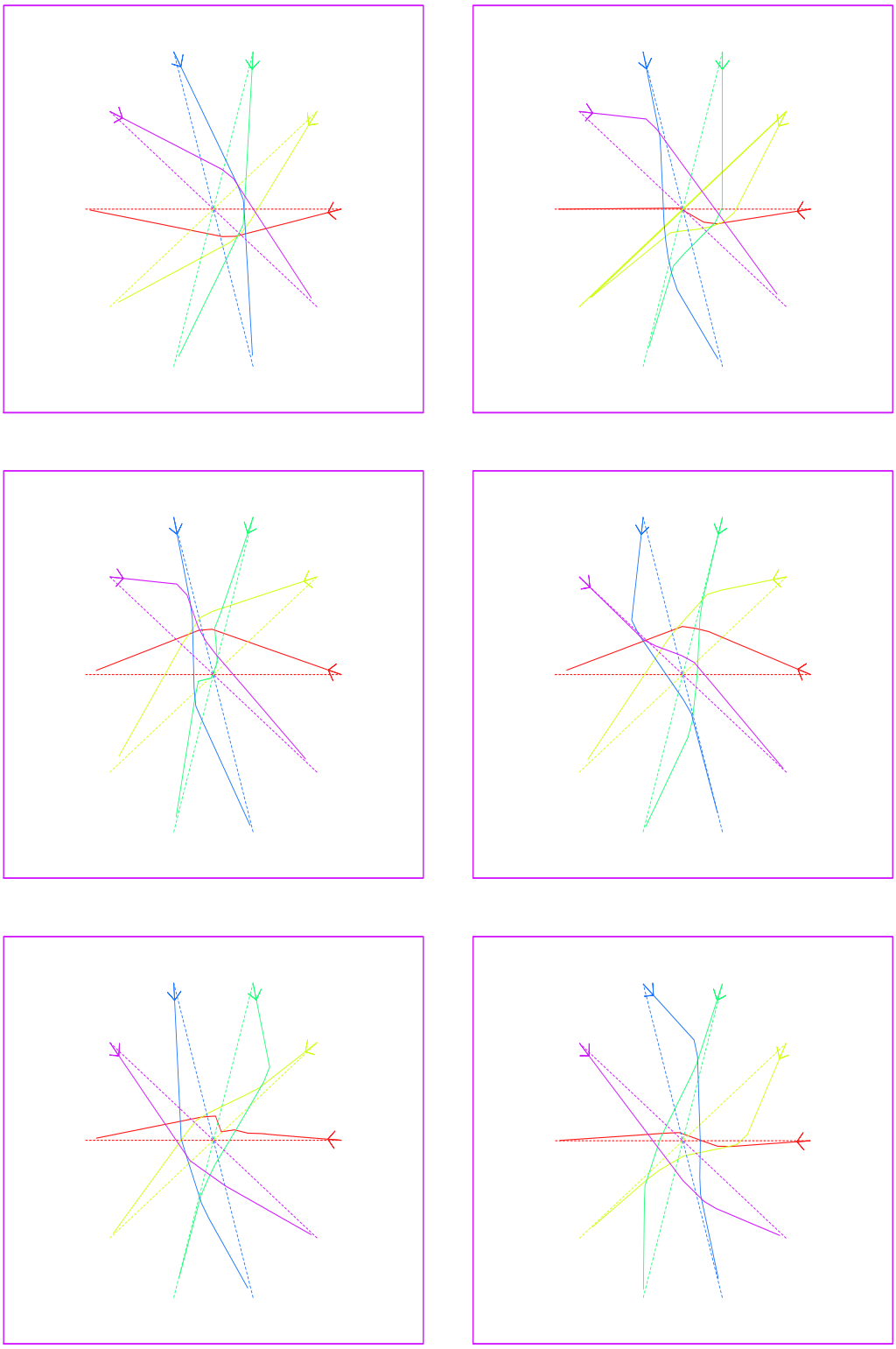


Figure 3.12: De haut en bas et de gauche à droite, allongements moyens : 2.16, 2.21, 2.22, 2.20, 2.19 et 2.17 degrés.

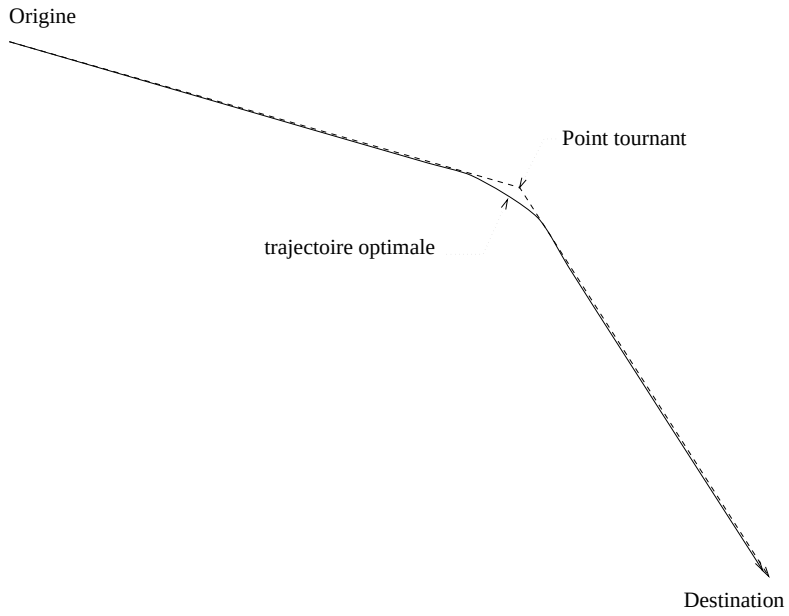


Figure 3.13: Modélisation point tournant.

l'objet du prochain chapitre). Dans le cas où les avions ne sont pas contraints en vitesse, on a montré que l'on peut déterminer géométriquement la solution optimale.

Le second point important résulte de la structure de l'espace des solutions admissibles pouvant contenir pour n avions $2^{\frac{n(n-1)}{2}}$ composantes connexes disjointes. Le problème est de ce fait fortement combinatoire et nous avons pu observer numériquement qu'un algorithme d'optimisation locale tel que LANCELOT ne permet pas de trouver l'optimum global pour des valeurs de n faibles (par exemple $n = 5$).

3.3 Approximation par point tournant et par offset

Les trajectoires optimales étudiées dans la section 3.2.2 étant formées de deux segments rectilignes relativement longs et d'une portion courbe réduite, il apparaît naturel de vouloir les approcher par des trajectoires composées de segments de droite. La modélisation par point tournant ramène la trajectoire à deux segments, alors que la modélisation par offset conserve trois segments.

3.3.1 Point tournant

La trajectoire des avions se ramène donc à deux droites se coupant en un point tournant (figure 3.13).

On peut représenter l'augmentation en pourcentage de l'allongement lorsque l'on remplace la trajectoire d'évitement réelle par un point tournant (voir figure 3.14). On observe que pour la solution optimale, l'augmentation d'allongement due à notre approximation reste inférieure à 0.25% tant que le temps d'anticipation t est supérieur à 2 et que pour l'autre solution, cet écart ne dépasse pas 3%.

L'angle d'incidence a une forte influence sur l'efficacité de ce mode de résolution . On a représenté sur la figure 3.15 l'allongement de la trajectoire en fonction de la distance au point de conflit (t représente le rapport des distances (distance au point de conflit/(norme de séparation)) et de l'angle

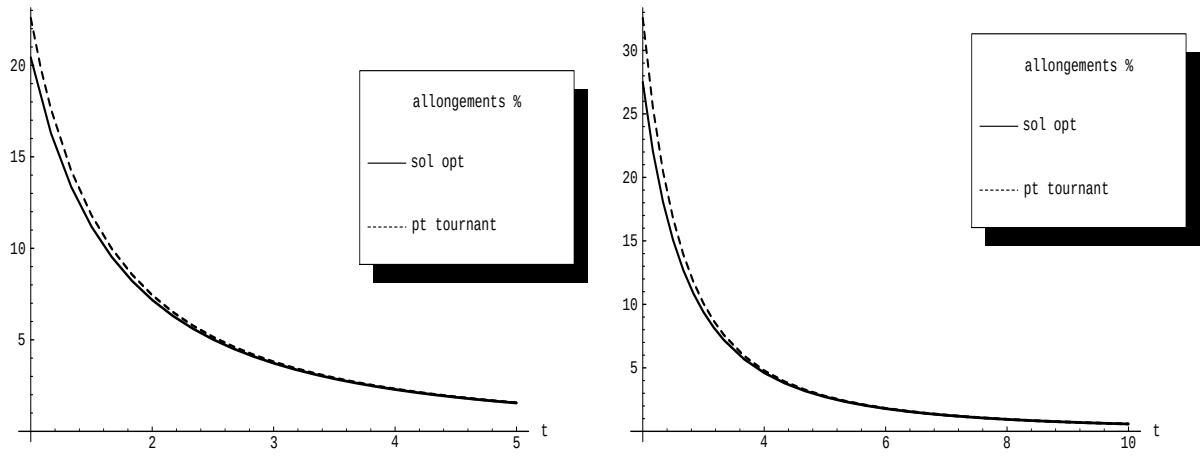


Figure 3.14: Allongements en %, l'avion mobile passant derrière (à gauche) ou devant (à droite) l'avion fixe.

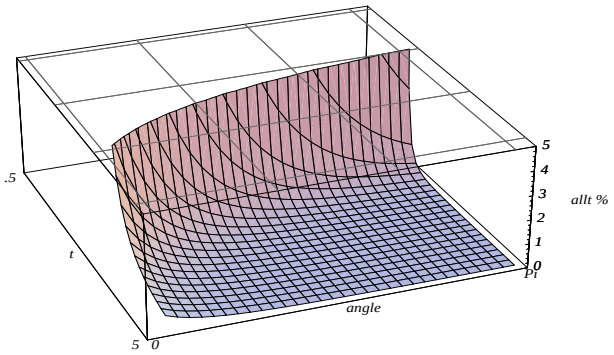


Figure 3.15: Augmentation de l'allongement $x_{t+} - x_+$ en fonction de t et ϕ.

de convergence. On constate que pour des valeurs faibles de l'angle de convergence et pour une faible anticipation, la méthode du point tournant est peu efficace.

3.3.2 Approximation par offset

On appellera résolution par offset, la solution de notre programme de minimisation lorsque l'on met en parallèle l'un des deux avions impliqués dans le conflit. C'est la solution adoptée par Niedringhaus [NFC+83, Nie89b, Nie89a], pour le projet d'automatisation complète AERA 3 dans la fonction "Gentle-Strict".

Nous avons représenté : les pourcentages d'allongement des trajectoires en fonction du temps d'anticipation $tx = ty$ pour des angles d'incidences de trajectoires valant $\frac{\pi}{2}$ (figure 3.16), $\frac{7\pi}{8}$ (figure 3.17) et $\frac{\pi}{8}$ (figure 3.18), et des angles de mise en offset valant $\frac{\pi}{3}$, $\frac{\pi}{4}$, $\frac{\pi}{6}$, $\frac{\pi}{8}$. Pour une incidence de trajectoires valant $\frac{\pi}{8}$, l'avion dévié ne peut pas passer devant l'avion fixe avec ce type d'offset.

Le principal intérêt de cette modélisation est que la fonction à minimiser devient linéaire, de même que les contraintes de séparation des avions une fois l'offset réalisé (à condition que l'angle de déviation soit fixé à priori). C'est ce que nous allons montrer. On s'intéresse donc ici à deux avions, a_i et a_j. On note $\phi_{i,j}$ l'angle formé par les trajectoires non déviées de a_i et a_j, t^i_{ij} (resp. t^j_{ij}) l'instant auquel la trajectoire non déviée de l'avion a_i (resp. a_j) coupe celle de l'avion a_j (resp. a_i), et v_i et v_j les normes des vecteurs vitesse des avions (par hypothèse constantes sur $[t_o, t_f]$).

On considère le repère orthonormé (C_{ij}, E_j, E_i), où C_{ij} est le point d'intersection des trajectoires des deux avions a_i et a_j, et dont l'axe des abscisses (E_j) est dirigé par le vecteur vitesse de a_j (voir figure 3.19).

On obtient les paramétrisations suivantes pour les coordonnées au temps $t \in [t_o, t_f]$ des avions a_i et a_j, dans ce repère :

$$\begin{aligned} x'_i(t) &= v_i(t - t^i_{ij})\cos(\phi_{ij}) \\ y'_i(t) &= v_i(t - t^i_{ij})\cos(\phi_{ij}) \\ \\ x'_j(t) &= v_j(t - t^j_{ij}) \\ y'_j(t) &= 0 \end{aligned}$$

Dans ce repère la contrainte de séparation donne donc pour le couple (a_i, a_j) :

$$\forall t \in [t_o, t_f],\ \ (x'_i(t) - x'_j(t))^2 + (y'_i(t) - y'_j(t))^2 \geq d^2 \tag{3.1}$$

En résolvant, il vient :

$$v_i{}^2 v_j{}^2 \sin(\phi_{ij})^2 (t^i_{ij} - t^j_{ij})^2 \geq d^2(v_i{}^2 + v_j{}^2 - 2v_i v_j \cos(\phi_{ij})) \tag{3.2}$$

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

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

$$(t^i_{ij} - t^j_{ij}) v_i v_j \sin(\phi_{ij}) \geq d\sqrt{v_i{}^2 + v_j{}^2 - 2v_i v_j \cos(\phi_{ij})} \tag{3.3}$$

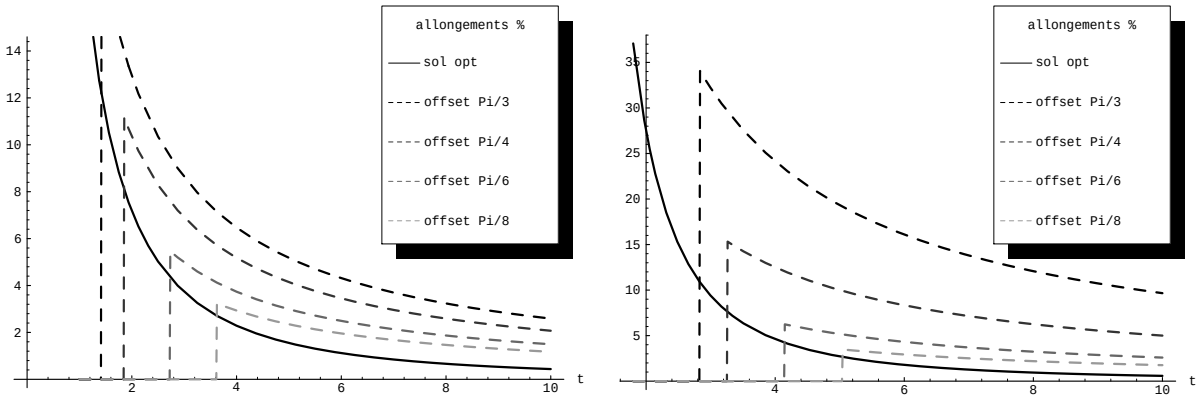


Figure 3.16: Allongements optimaux en %, l'avion mobile passant derrière (à gauche) ou devant (à droite) l'avion fixe, $\phi = \frac{\pi}{2}$.

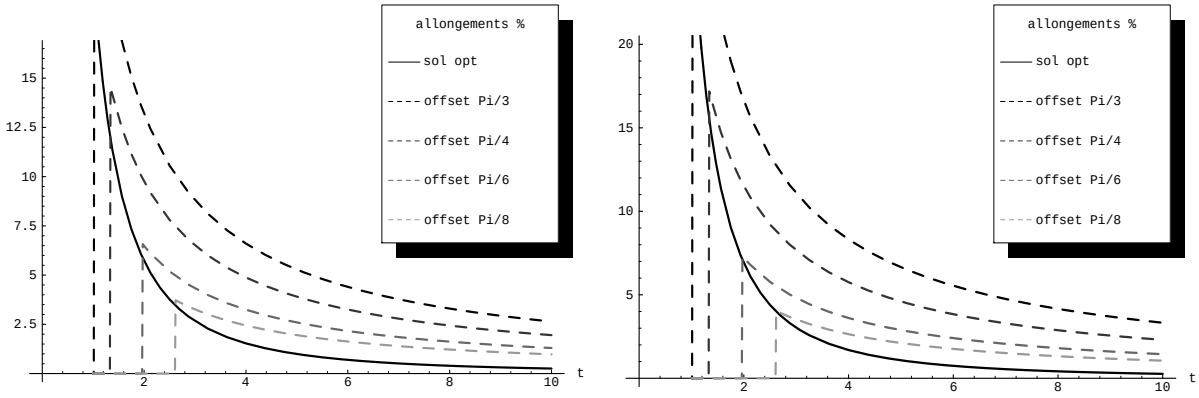


Figure 3.17: Allongements optimaux en %, l'avion mobile passant derrière (à gauche) ou devant (à droite) l'avion fixe, $\phi = \frac{7\,\pi}{8}$.

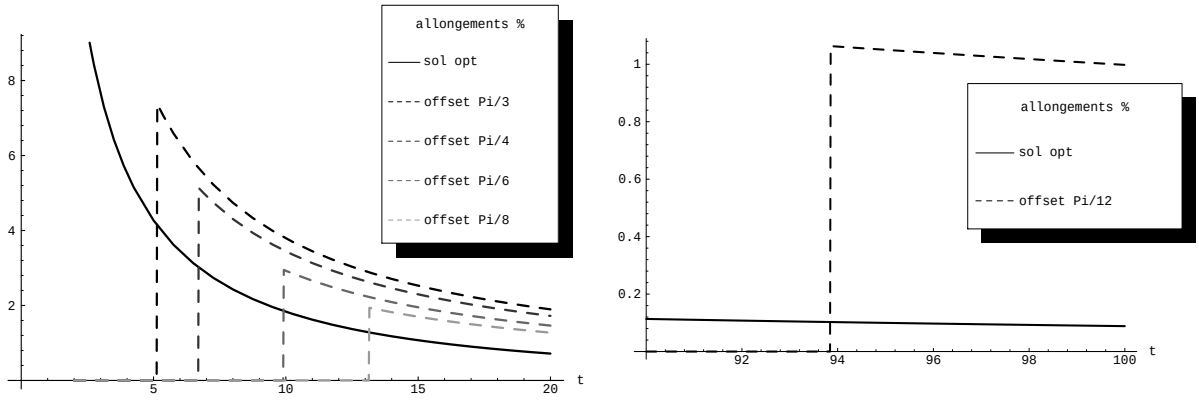


Figure 3.18: Allongements optimaux en %, l'avion mobile passant derrière (à gauche) ou devant (à droite) l'avion fixe, $\phi = \frac{\pi}{8}$.

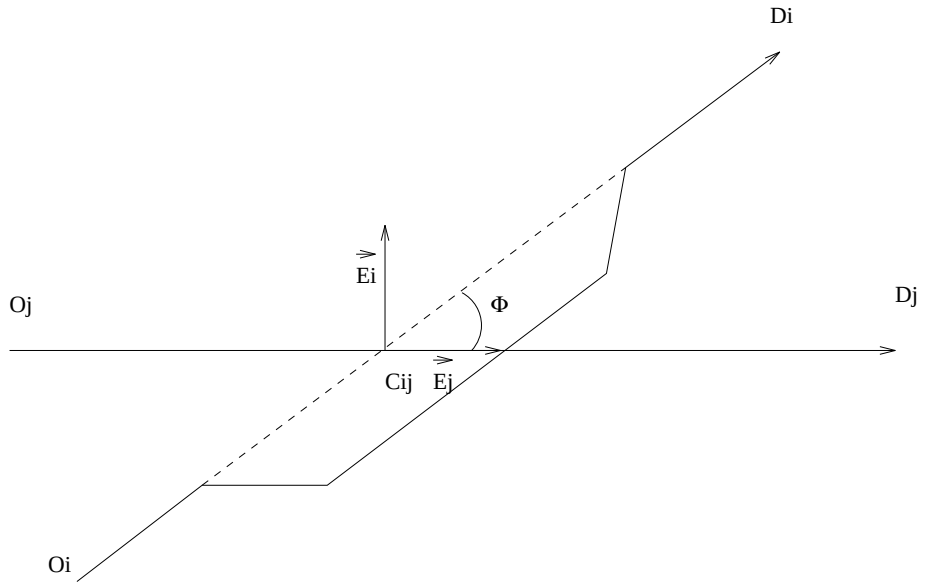


Figure 3.19: Conflit à deux avions, un seul avion étant dévié

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

$$(t^j_{ij} - t^i_{ij}) v_i v_j \sin(\phi_{ij}) \geq d\sqrt{{v_i}^2 + {v_j}^2 - 2 v_i v_j \cos(\phi_{ij})} \tag{3.4}$$

La mise en offset d'un avion (ou des deux avions a_i et a_j) a pour effet d'introduire un décalage d'une trajectoire par rapport à l'autre et d'influer ainsi sur l'instant auquel la trajectoire de chacun des deux avions coupe celle de l'autre. Nous nous intéressons à un couple d'avions, a_i et a_j, tous deux impliqués dans un conflit à n avions. La trajectoire de chacun de ces deux avions peut être déviée.

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

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

Ainsi, quand l'avion a_i passe derrière l'avion a_j, selon le sens de l'offset de chacun des deux avions a_i et a_j on obtient les quatre conditions de séparations suivantes, qui sont linéaires en d_i et en d_j :

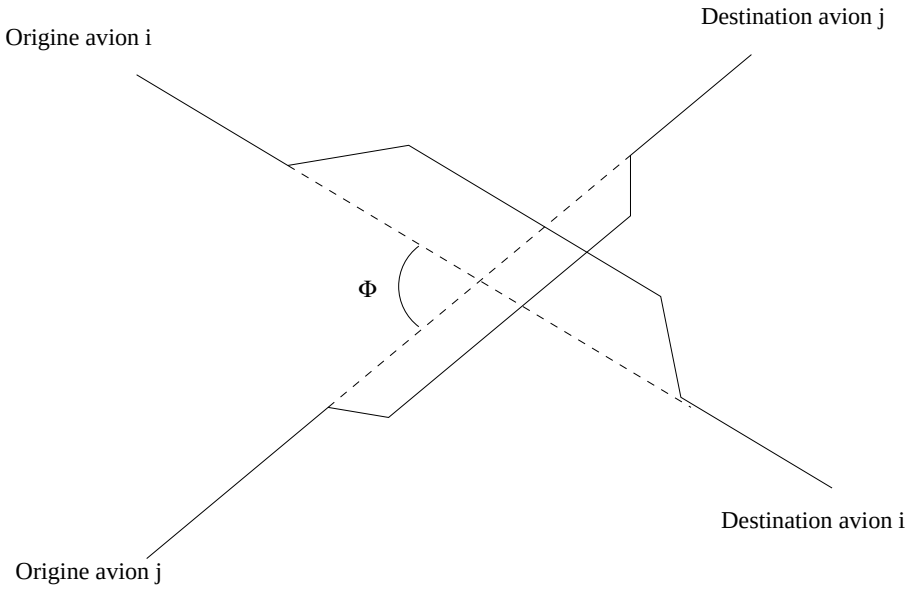


Figure 3.20: Conflit à deux avions, les deux offsets étant à l'extérieur

- Les offsets des deux avions sont à l'extérieur (c'est le cas qui est représenté sur la figure 3.20) :

$$\begin{aligned}(t_{ij}^i + \frac{d_i \tan(\frac{\beta}{2})}{v_i} + \frac{d_i \cot(\phi_{ij})}{v_i} + \frac{d_j}{v_i \sin(\phi_{ij})} \\ -t_{ij}^j - \frac{d_j \tan(\frac{\beta}{2})}{v_j} - \frac{d_j \cot(\phi_{ij})}{v_j} - \frac{d_i}{v_j \sin(\phi_{ij})}) v_i v_j \sin(\phi_{ij}) \\ -d\sqrt{{v_i}^2 + {v_j}^2 - 2 v_i v_j \cos\phi_{ij}} \quad \geq \quad 0 \end{aligned} \tag{3.5}$$

- L'offset de l'avion a_i est à l'extérieur, celui de l'avion a_j est à l'intérieur :

$$\begin{aligned}(t_{i,j}^i + \frac{d_i \tan(\frac{\beta}{2})}{v_i} + \frac{d_i \cot(\phi_{ij})}{v_i} - \frac{d_j}{v_i \sin(\phi_{ij})} \\ -t_{i,j}^j - \frac{d_j \tan(\frac{\beta}{2})}{v_j} + \frac{d_j \cot(\phi_{ij})}{v_j} - \frac{d_i}{v_j \sin(\phi_{ij})}) v_i v_j \sin(\phi_{ij}) \\ -d\sqrt{{v_i}^2 + {v_j}^2 - 2 v_i v_j \cos\phi_{ij}} \quad \geq \quad 0 \end{aligned} \tag{3.6}$$

- L'offset de l'avion a_i est à l'intérieur, celui de l'avion a_j est à l'extérieur :

$$\begin{aligned}(t_{ij}^i + \frac{d_i \tan(\frac{\beta}{2})}{v_i} - \frac{d_i \cot(\phi_{ij})}{v_i} + \frac{d_j}{v_i \sin(\phi_{ij})} \\ -t_{ij}^j - \frac{d_j \tan(\frac{\beta}{2})}{v_j} - \frac{d_j \cot(\phi_{ij})}{v_j} + \frac{d_i}{v_j \sin(\phi_{ij})}) v_i v_j \sin(\phi_{ij}) \\ -d\sqrt{{v_i}^2 + {v_j}^2 - 2 v_i v_j \cos\phi_{ij}} \quad \geq \quad 0 \end{aligned} \tag{3.7}$$

- Les offsets des deux avions sont à l'intérieur :

$$\begin{aligned}(t_{ij}^i + \frac{d_i \tan(\frac{\beta}{2})}{v_i} - \frac{d_i \cot(\phi_{ij})}{v_i} - \frac{d_j}{v_i \sin(\phi_{ij})} \\ -t_{ij}^j - \frac{d_j \tan(\frac{\beta}{2})}{v_j} + \frac{d_j \cot(\phi_{ij})}{v_j} + \frac{d_i}{v_j \sin(\phi_{ij})})v_i v_j \sin(\phi_{ij}) \\ -d\sqrt{{v_i}^2 + {v_j}^2 - 2v_i v_j \cos\phi_{ij}} \quad \geq \quad 0 \end{aligned} \tag{3.8}$$

Et on obtient quatre équations similaires dans le cas où l'avion a_i passe devant l'avion a_j (en intervertissant dans les équations ci-dessus les indices i et j).

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

Cette modélisation étendue au cas de n avions nécessite la résolution de $2^{\frac{n\,(n+1)}{2}}$ programmes linéaires comprenant chacun $\frac{n(n+1)}{2}$ contraintes linéaires. Pour un conflit à 5 avions, cela représente 32768 programmes d'optimisation à résoudre comprenant 15 contraintes linéaires chacun. Pour $n = 7$, plus de 2 million de programmes linéaires doivent être résolus, comprenant 21 contraintes chacun. On se retrouve face à une très forte combinatoire. Il est intéressant de noter que l'on retrouve les composantes connexes observées dans le paragraphe 3.2.3. En effet, chaque programme linéaire détermine un ordre de passage entre une paire d'avions et résout le problème à l'intérieur d'une composante connexe de l'espace des solutions admissibles. Il faut néanmoins choisir le sens de la déviation pour chaque avion pour que le programme devienne linéaire ; c'est pourquoi la complexité passe de $2^{\frac{n\,(n-1)}{2}}$ à $2^{\frac{n\,(n-1)}{2}} \times 2^n = 2^{\frac{n\,(n+1)}{2}}$. F. Médioni [MDA94] s'est intéressé à combiner un algorithme génétique avec un programme d'optimisation linéaire pour résoudre le problème décrit ci-dessus. Nous présentons ces résultats dans la section 3.7.

Nous ne nous sommes intéressés jusqu'à présent qu'à satisfaire la contrainte de séparation lorsque l'avion est établi en offset. Durant la mise en offset, les deux avions doivent rester séparés. La vérification de cette contrainte est longue, ou impose de prendre des marges de manœuvre qui pénalisent la résolution.

3.3.3 Avions en rattrapage

Nous n'avons pas étudié jusqu'à présent les cas de conflits causés par des rattrapages. Dans ce cas, lorsque deux avions sur la même route ont des vitesses proches, l'offset devient plus avantageux que le point tournant. En effet, que l'avion le plus rapide dépasse le plus lent ou que l'avion le plus lent évite le plus rapide (figure 3.21), l'offset permet d'éviter un éloignement de plus d'une séparation standard de la trajectoire initiale. Si l'on note r le rapport des vitesses des deux avions, et X l'allongement due à la technique du point tournant par rapport à l'offset, on peut apprécier sur la figure 3.22 la supériorité de la technique de l'offset par rapport à la technique du point tournant simple.

Les calculs présentés ci-dessus nous conduisent finalement à la conclusion suivante : la modélisation point tournant, très efficace pour résoudre les problèmes de conflits à angle non nuls, s'avère insuffisante dans le cas des rattrapages. Il faut donc conserver les deux modélisations, qui ont des champs d'application différents.

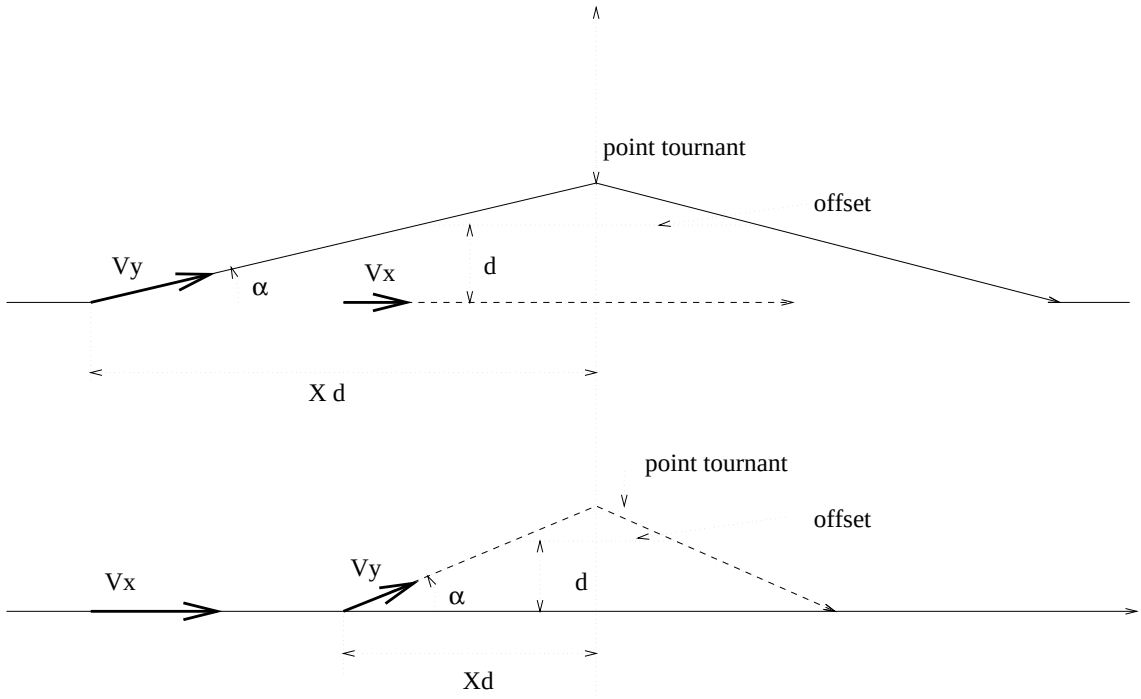


Figure 3.21: Rattrapages

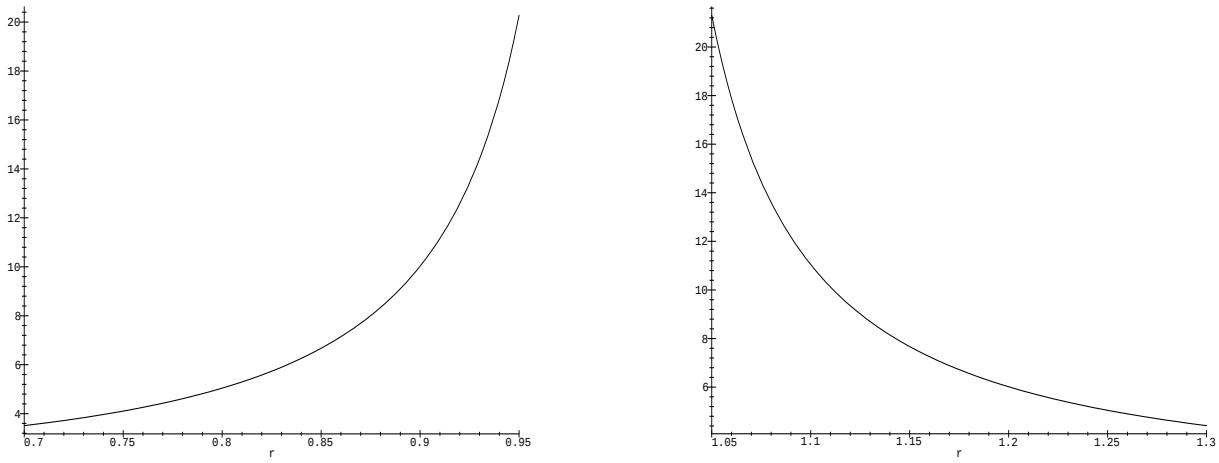


Figure 3.22: A gauche : X en fonction de r pour $r < 1$ - A droite : X en fonction de r pour $r > 1$.

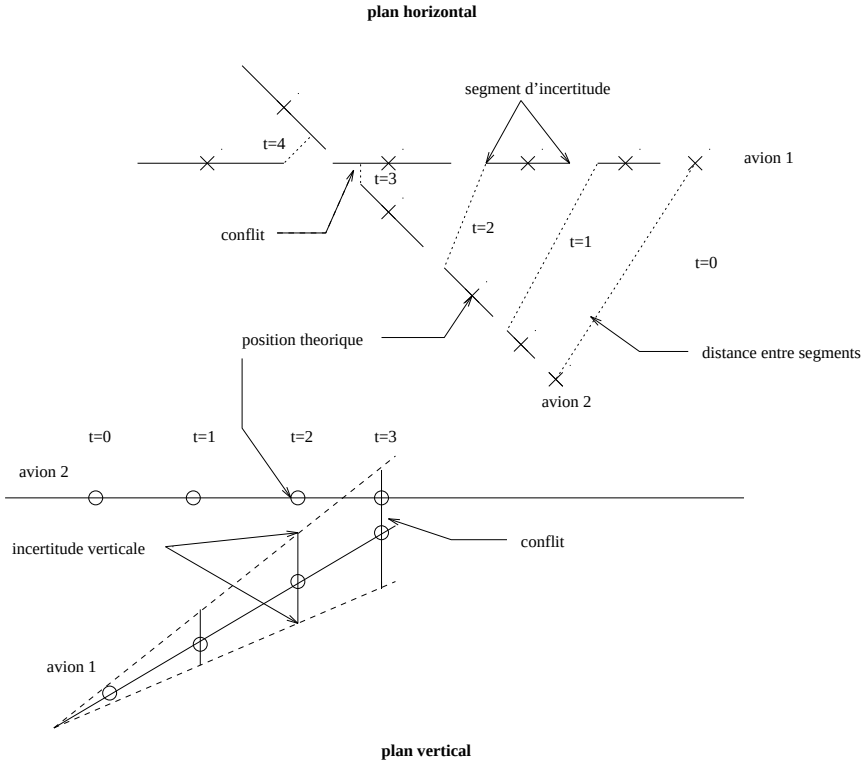


Figure 3.23: Modélisation de l'incertitude.

3.4 Modélisation de l'incertitude

3.4.1 Nécessité de modéliser l'incertitude

Si un avion de ligne est capable de suivre précisément une route et une altitude, maîtriser précisément sa vitesse sol n'est pas possible en raison notamment des vents. Il est d'autre part impossible de corriger en permanence la vitesse car les réacteurs ont un régime de consommation idéal dont il ne faut pas trop s'écarter. Enfin, ils ne doivent pas, pour des raisons mécaniques, être soumis à des changements de régime permanents. Il est donc indispensable de modéliser les avions non par des points matériels mais par des segments de droites. Il suffira alors de vérifier, pour que la contrainte horizontale soit respectée, que les segments de droite qui modélisent les avions soient séparés par la séparation standard (voir figure 3.23). Dans le plan vertical, l'incertitude est beaucoup plus forte. On maîtrise en effet très mal le taux de montée d'un avion qui dépend de paramètres aussi variés que la masse, où la mise en route de la climatisation. En conséquence, le même principe que dans le plan horizontal est repris mais les pourcentage d'incertitude seront beaucoup plus forts.

3.4.2 Évaluation de la probabilité de conflit.

Compte-tenu de l'incertitude sur la vitesse des avions que nous observons, il est évident qu'un conflit ne peut être prévu à l'avance avec certitude. Il se peut en effet que les erreurs sur la vitesse de chacun des avions suffisent à éviter le conflit. Dans ce cas, une manœuvre d'évitement s'avère inopportune. Le but de ce paragraphe est de mesurer la valeur de la probabilité de conflit en fonction de la configuration du conflit et de l'incertitude sur les vitesses.

On ne peut bien évidemment pas étudier toutes les configurations de conflits. Aussi, nous nous limiterons au cas où les deux ont la même vitesse V, convergent sous un angle ϕ vers un point de conflit. Les avions sont distants du point de conflit de $V\,T$. On suppose que chaque avion se déplace à vitesse constante, avec une incertitude en pourcentage sur cette vitesse notée Er. Pour simplifier, on supposera donc que chacun des deux avions a une probabilité uniforme d'avoir sa vitesse comprise entre $(1-Er)\,V$ et $(1+Er)\,V$. On notera d la norme de séparation choisie. On veut mesurer la probabilité de conflit en fonction de l'anticipation T, de l'angle d'incidence ϕ et de l'incertitude Er choisie.

On numérotera 1 et 2 les indices des deux avions. A un instant $t \geq -T$, les avions sont repérés par leur position dans le plan :

$$\begin{array}{l} -T\,\vec{V_1} + t\,(1+Er_1)\,\vec{V_1} \\ -T\,\vec{V_2} + t\,(1+Er_2)\,\vec{V_2} \end{array}$$

avec Er_1 et Er_2 compris entre $-Er$ et Er. A partir des positions des deux avions à tout instant t, on cherche à quelle condition sur Er_1 et Er_2 il y a conflit. Pour cela, on est amené à déterminer le signe du discriminant d'une équation du second degré. La condition, une fois simplifiée s'écrit de la façon suivante.

Il y a conflit si et seulement si :

$$\frac{\frac{V^2\,T^2}{d^2}\,(\sin\phi)^2 - 1}{2\,(1-\cos\phi)} \quad \leq \quad \frac{(1+Er_1)\,(1+Er_2)}{(Er_1-Er_2)^2}$$

Il est clair que si l'angle ϕ vaut 0 ou π, le conflit est certain. On remarquera que la condition précédente ne dépend que du rapport $x = \frac{V\,T}{d}$ qui mesure l'anticipation par rapport au point de conflit. On peut représenter la condition par une fonction $f(x,\phi,Er_1,Er_2)$ qui vaut 0 lorsque la condition n'est pas satisfaite et 1 lorsque la condition est satisfaite. On intègre ensuite cette fonction sur le pavé des erreurs possibles et on obtient la probabilité $p(x,\phi,Er)$ pour que les deux avions entrent en conflit avec une anticipation x, un angle d'incidence ϕ et une erreur sur les vitesses Er.

A titre d'illustration, nous allons observer l'évolution de la probabilité de conflit lorsque :

- l'anticipation $x = \frac{V\,T}{d}$ augmente (voir figure 3.24), on fixera $\phi = \frac{\pi}{2}$, $Er = 5\%$, $V = 400$ kts, $d = 10$ nm et on fera varier T de 0 à 120 minutes. On constate que la probabilité de conflit vaut 1 jusqu'à 20 minutes, après quoi elle décroît. On remarquera qu'elle décroît alors de moins en moins vite avec le temps.

- l'angle ϕ varie de 0 à π (voir figure 3.24), on fixera $V = 400$ kts, $d = 10$ nm, $T = 30$ minutes, et $Er = 5\%$. On observe que l'angle ϕ a une très forte influence sur la probabilité de conflit. En effet, dans cet exemple, celle-ci varie de 0.75 à 1 selon que les avions convergent à 20 degrés ou 120 degrés. On observe donc que le risque de conflit est d'autant plus faible que l'angle d'incidence est faible. Or on sait que le conflit est d'autant plus difficile à résoudre que cet angle est faible. On en déduit que les conflits d'avions convergents sont à la fois moins certains et plus difficiles à résoudre. La prise de risque est donc plus grande dans le choix de l'instant de résolution de conflit.

- le pourcentage d'erreur Er varie de 0 à 20% (voir figure 3.25), on fixera $V = 400$ kts, $d = 10$ nm, $T = 30$ minutes, et $\phi = \frac{\pi}{2}$. La sensibilité de la probabilité de conflit au pourcentage d'erreur est semblable à sa sensibilité au temps d'anticipation.

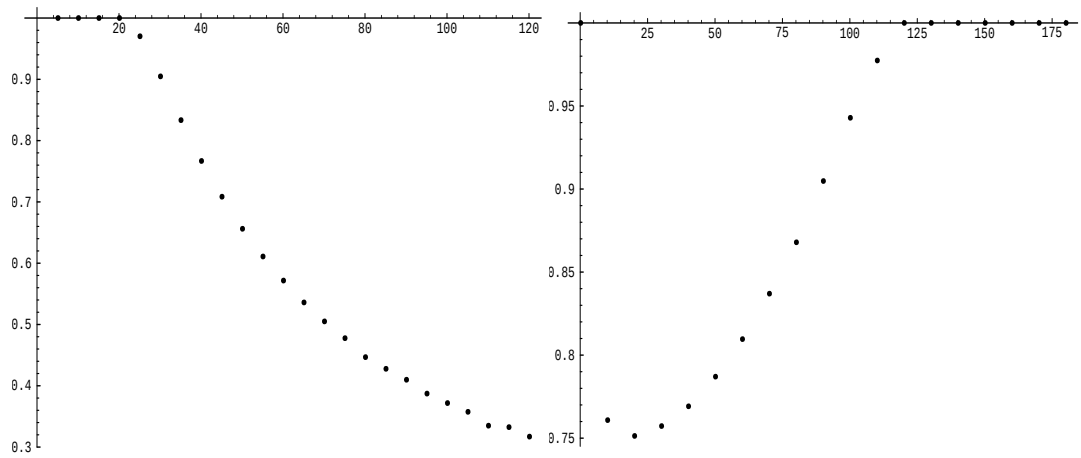


Figure 3.24: Probabilité de conflit en fonction du temps d'anticipation en minutes (à gauche), de l'angle d'incidence en degrés (à droite).

- l'angle ϕ varie de 0 à π et le temps d'anticipation varie de 0 à 120 minutes (voir figure 3.25), on fixera $V = 400$ kts, $d = 10$ nm et $Er = 5\%$. On visualise a lors bien le fait que la probabilité de conflit est fortement liée à l'angle d'incidence. Le choix du moment de résolution est donc fortement dépendant de la géométrie du conflit.

Il apparaît donc au travers de ces résultats qu'une fois de plus, les conflits à faible incidence sont ceux qui posent le plus de problèmes. Moins "certains", ils sont plus difficiles à résoudre.

3.4.3 Espérance de coût de résolution de conflit.

Si l'on veut résoudre un conflit de façon certaine avec un temps d'anticipation T, il faut intégrer l'incertitude sur les vitesses dans la résolution. Ainsi au lieu de séparer des points par séparation standard, il faut séparer de cette séparation standard des segments de droite dont la longueur augmente avec le temps. Pour cela, on peut mesurer l'allongement de la trajectoire en nautiques lorsque l'on bouge un des deux avions en fonction du temps d'anticipation engendré par la résolution d'un conflit. Pour ce calcul, nous garderons les mêmes caractéristiques que précédemment pour les deux avions et choisirons un angle d'incidence de $\frac{\pi}{2}$ et une erreur sur la vitesse de 5%. Le calcul est fait dans les deux modélisations offset et point tournant (voir figure 3.26). On observe qu'avec la modélisation par offset, l'évitement le plus tardif est le plus économique. La courbe est en fait une droite. L'allongement est donc une fonction linéaire du temps d'anticipation. Pour la modélisation point tournant, au phénomène d'accroissement de l'allongement provoqué par l'incertitude s'ajoute un phénomène inverse. Plus l'on anticipe le conflit, plus la déviation de l'avion mobile sera faible, et si l'on s'y prend tard, l'allongement sera fort.

On souhaiterait déterminer le moment optimal pour résoudre un conflit à deux avions en tenant compte de ce phénomène d'incertitude. Pour cela, il faut préciser quel est le critère d'optimalité retenu. N'ayant pas de certitude sur l'existence ou non d'un conflit lorsque le temps d'anticipation augmente, on ne pas parler de coût de résolution. Par contre on peut définir l'espérance du coût de résolution de conflit. Pour tout temps d'anticipation T, on connaît la probabilité pour que le conflit se produise

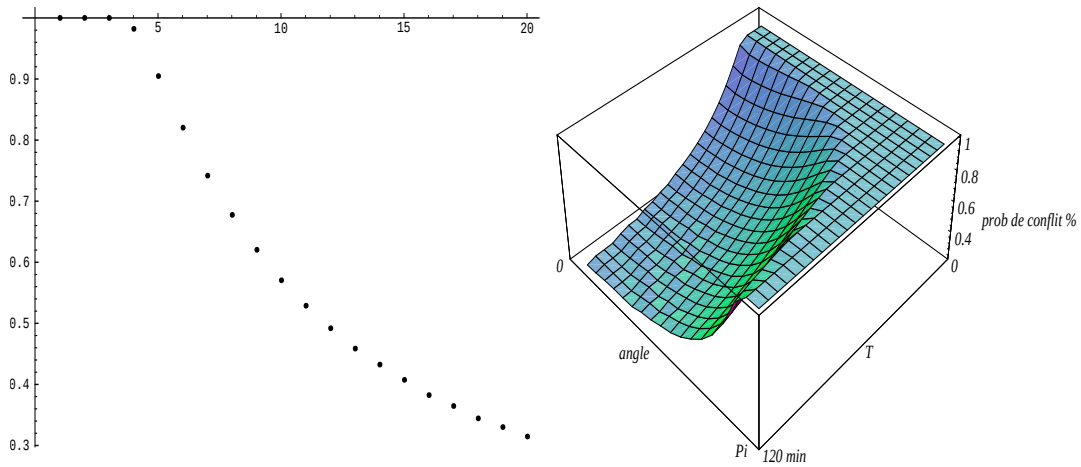


Figure 3.25: Probabilité de conflit en fonction du pourcentage d'erreur sur les vitesses (à gauche), en fonction de l'angle d'incidence et du temps d'anticipation (à droite).

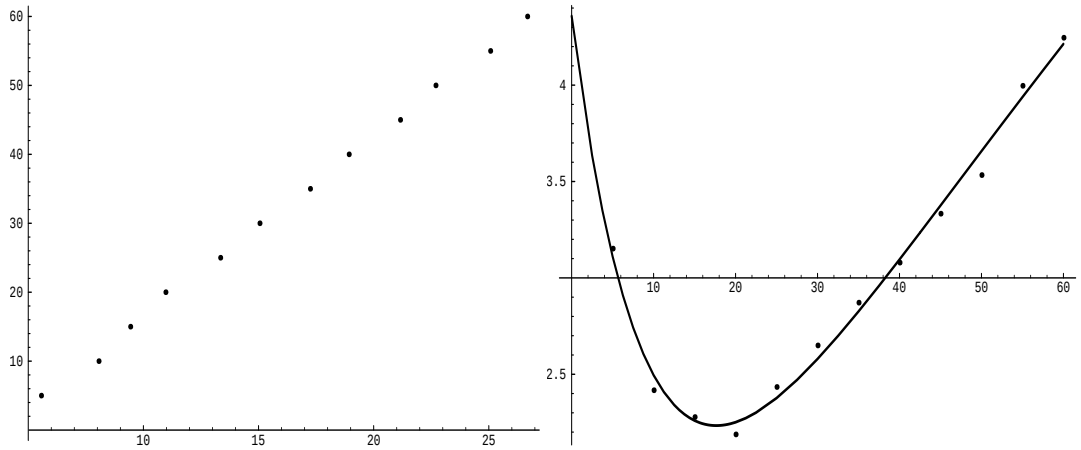


Figure 3.26: Allongement de la trajectoire en nautiques en fonction du temps d'anticipation, modélisation par offset (à gauche), modélisation point tournant (à droite)

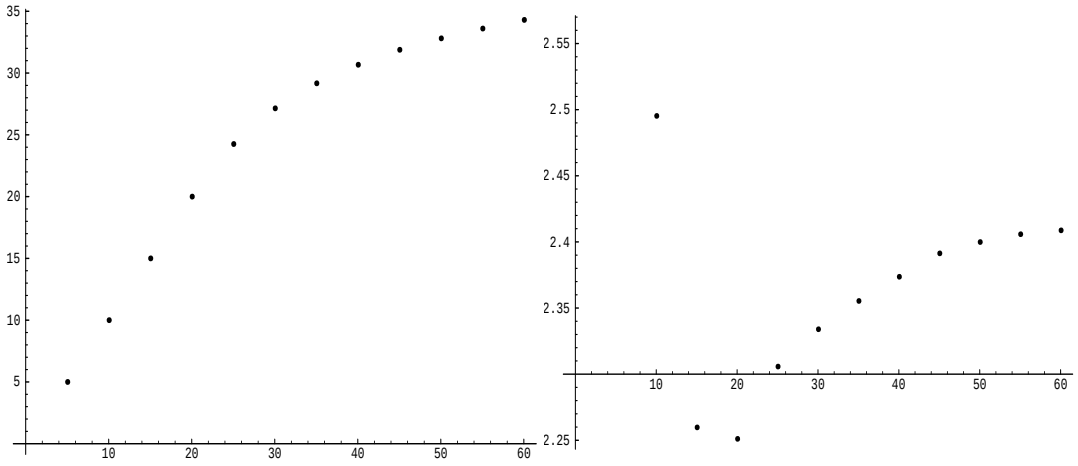


Figure 3.27: Espérance du coût de résolution en fonction du temps d'anticipation, modélisation par offset (à gauche), modélisation point tournant (à droite).

et le coût associé à ce temps. Il suffit de multiplier les deux pour obtenir l'espérance du coût de résolution du conflit sachant que s'il n'y a pas de conflit, le coût de résolution est nul. On observera la figure 3.27 pour les deux modélisation offset et point tournant. Si pour l'offset, l'espérance de coût de résolution est minimale pour un temps d'anticipation le plus faible possible, pour la modélisation point tournant, dans cet exemple, le coût de résolution est minimal pour une anticipation de 20 minutes ou 130 nautiques.

Les calculs que nous venons de présenter permettent de conclure qu'une résolution par offset doit être la plus tardive possible. Pour une résolution par point tournant, il existe un compromis qui minimise l'espérance de coût de résolution du conflit. Ce compromis dépend de l'incertitude sur les vitesses et de la configuration du conflit. Cette étude n'était pas exhaustive mais propose un critère d'évaluation de l'instant optimal de début de résolution de conflit.

3.5 Modélisation pour la résolution de conflits complexes

Les études précédentes ont conduit aux conclusions suivantes :

- On peut modéliser une manœuvre d'évitement horizontale entre deux avions par un point tournant.
- La modélisation par offset reste indispensable pour des cas de rattrapages.
- L'incertitude sur les vitesses des avions et sur leurs taux de montée doit être prise en compte et fait apparaître dans le cas de la modélisation point tournant un temps "idéal" de début de manœuvre.

D'autres contraintes pratiques sont maintenant introduites dans le modèle afin de pouvoir résoudre des conflits complexes mettent en jeu plusieurs avions :

- Les angles de déviation pour le point tournant et l'offset seront discrétisés et pourront donc prendre pour valeur : -30, -20, -10, 0, 10, 20 et 30 degrés.
- Un manœuvre commencée ne pourra pas être remise en cause, on pourra néanmoins modifier son ampleur sans en modifier la structure.
- Le système ne proposera pas de régulation de vitesse (le lecteur trouvera partie 3.2.1 des arguments justifiant ce choix.
- Les manœuvres d'évitement proposées sont horizontales. Le système ne pourra donc pas modifier la trajectoire verticale d'un avion même s'il peut faire tourner des avions évolutifs (en montée ou en descente). Une telle hypothèse est relativement restrictive : un certain nombre de résolutions ne seront plus optimales, en particulier, lorsqu'un des avions impliqué dans un conflit est évolutif. Les contrôleurs le savent bien et essayent d'employer ce genre de résolution le plus fréquemment possible pour les avions qui ne sont pas en palier. Ces manœuvres sont souvent optimales par rapport à une déviation horizontale lorsque les avions sont sur une route identique ou opposée. Se restreindre au plan horizontal n'est pas un objectif et la dimension verticale devra nécessairement être intégrée dans les évolutions futures. Néanmoins l'ajout de telles manœuvre ne complexifie pas de façon significative le modèle.

Nous obtenons donc le modèle suivant (figure 3.28) où une manœuvre est déterminée par les 4 paramètres suivants :

- la date de début du virage d'éloignement t_0 ;
- la date de fin du virage d'éloignement t_1 ;
- la date de début du virage de retour sur la trajectoire initiale t_2.
- l'angle de déviation de la manœuvre : α ;

Les différents t_i représentent des dates relatives. L'instant $t = 0$ correspond au début de la prévision. A cette date, la position de l'avion est connue sans incertitude. On la considère comme *la position initiale* de l'avion.

Le virage de retour a évidemment le même angle α mais dans le sens opposé. La durée de la phase de retour est donc égale à la durée de la phase d'éloignement. La date de fin de manœuvre, notée t_3 n'est donc pas un paramètre de la manœuvre et on a : $t_3 = t_2 + t_1 - t_0$

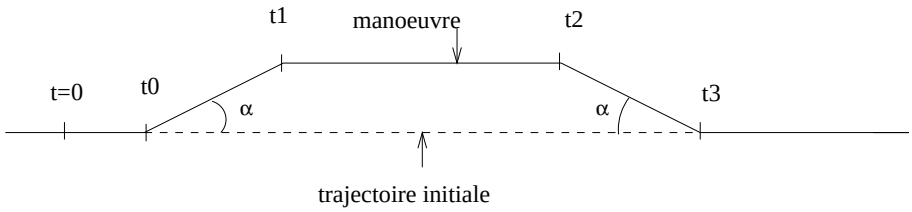
Si on est en présence d'un point tournant, on aura $t_2 = t_1$.

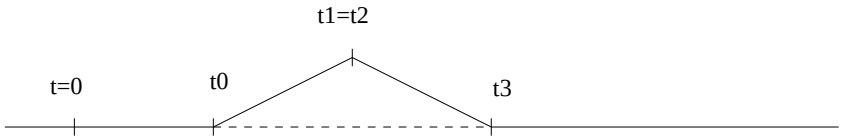
Un tel modèle réduit donc la taille du problème de façon significative. Pour un conflit impliquant n avions, la dimension de l'espace de recherche sera de $4n$. Ceci nous permettra de résoudre des gros conflits sans avoir à explorer un domaine de solutions admissibles trop vaste.

3.5.1 Modèle de cluster

Nous nous proposons donc de résoudre non pas uniquement des conflits élémentaires mais des situations plus complexes. Nous introduisons ici la notion de *clusters*. Un cluster est le résultat d'une fermeture transitive dans l'ensemble des conflits sur un horizon temporel fixé H. Si un avion A est en conflit avec B à l'instant t et B est en conflit avec C à l'instant $t + \Delta t$ avec $\Delta t < H$ alors A, B et C appartiennent au même cluster. Ainsi dans la figure 3.29, les avions A et B ne sont jamais en conflit mais ils appartiennent au même cluster. Une déviation de C pour éviter B peut lui permettre d'éviter A. L'optimisation doit donc se faire de façon globale.

Modele d’offset





Modele de point tournant

Figure 3.28: Modèles de manœuvre.

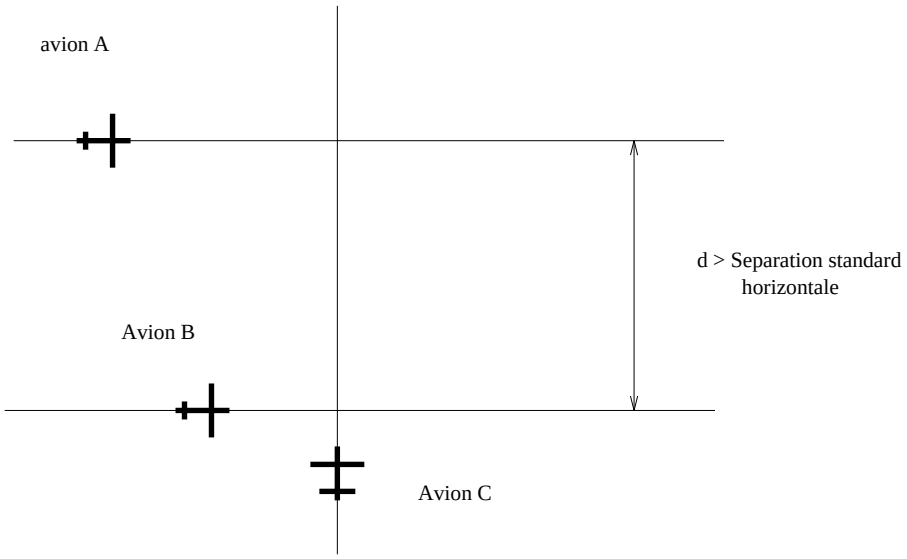


Figure 3.29: Exemple de cluster.

3.5.2 Déroulement en temps réel

Un des objectifs retenus est de pouvoir détecter et résoudre les conflits en temps réel. Fixons nous un délai Δ. La situation doit être revue toutes les Δ secondes à partir de nouvelles données. Le système dispose donc d'un temps inférieur à Δ pour analyser la nouvelle situation et trouver les trajectoires optimales sans conflit pour l'ensemble des avions impliqués dans la fenêtre d'anticipation F_a. Le reste de l'intervalle Δ est consacré à la transmission des manœuvres du sol vers le bord. Durant l'intervalle Δ, les avions volent toujours. Le système ne peut donc changer la trajectoire prévue de chacun des avions pendant les Δ premières secondes de F_a. Ces trajectoires correspondent donc à la prévision effectuée au tour précédent. Par contre le système reste libre de modifier toutes les trajectoires entre Δ et F_a à l'exception des branches de retour en point tournant ou en offset. En effet, cette partie de la manœuvre est complètement déterminée par le début de la manœuvre.

Appelons t le temps relatif utilisé par le système de résolution, T l'échelle de temps absolu et T_0 la date de début de la première itération de notre processus de résolution. A chaque itération i, le processus analyse une nouvelle situation entre $t = 0$ et $t = F_a$. La situation prévue à l'instant $t = 0$ correspond à la situation réelle existant à $T = T_0 + i\Delta$ en considérant que $i = 0$ pour la première itération. La figure 3.30 peut alors se commenter de la façon suivante :

itération 1 ($T = T_0$) : Le système prévoit un offset pour $t > 2\Delta$.

itération 2 ($T = T_0 + \Delta$) : Le système a réduit l'offset en un point tournant prévu pour $t > \Delta$.

itération 3 ($T = T_0 + 2\Delta$) : Le système a réduit le point tournant prévu. La date de début de manœuvre est prévue pour $t < \Delta$. Ce début de manœuvre sera donc communiqué au pilote. Par rapport au modèle présenté, on a fixé les variables t_0 et $sens$.

itération 4 ($T = T_0 + 3\Delta$) : L'avion a effectué son début de manœuvre et le système prévoit toujours le même point tournant. Le reste de la manœuvre est donc à son tour figées : t_1 et t_2 (en l'occurrence $t_1 = t_2$ car il s'agit d'un point tournant).

itération 5 ($t = T_0 + 4\Delta$) : L'avion est sur sa branche de retour.

3.6 Résolution par algorithmes génétiques

La modélisation simplifiée par point tournant ou offset permet dans le cas de conflits à deux avions d'approcher avec une perte acceptable les trajectoires optimales. Dans la mesure où il parait peu envisageable de donner à un avion plus d'un ordre de manœuvre à la fois, cette modélisation pourra être conservée pour la résolution de conflits à n avions avec $n > 2$. L'intérêt principal d'une telle modélisation est qu'elle permet de réduire fortement la taille de l'espace de recherche. On peut désormais définir une trajectoire d'évitement par la donnée d'au plus trois temps et d'un angle de déviation.

On a pu également observer que résoudre un conflit en modifiant seulement la vitesse des avions requiert une prise de décision fortement anticipée compte tenu des faibles marges de manœuvre des avions (en route) et des incertitudes sur leurs tenues de vitesses. Une modification de vitesse pourrait donc être efficace à condition de pouvoir la combiner avec des manœuvres de type point tournant ou offset, hypothèse que nous avons pour l'instant écartée.

Enfin, afin de pouvoir rendre le modèle réaliste, il est essentiel de tenir compte de l'incertitude sur les vitesses, taux de montée et de descente dans le modèle. La notion de séparation standard est

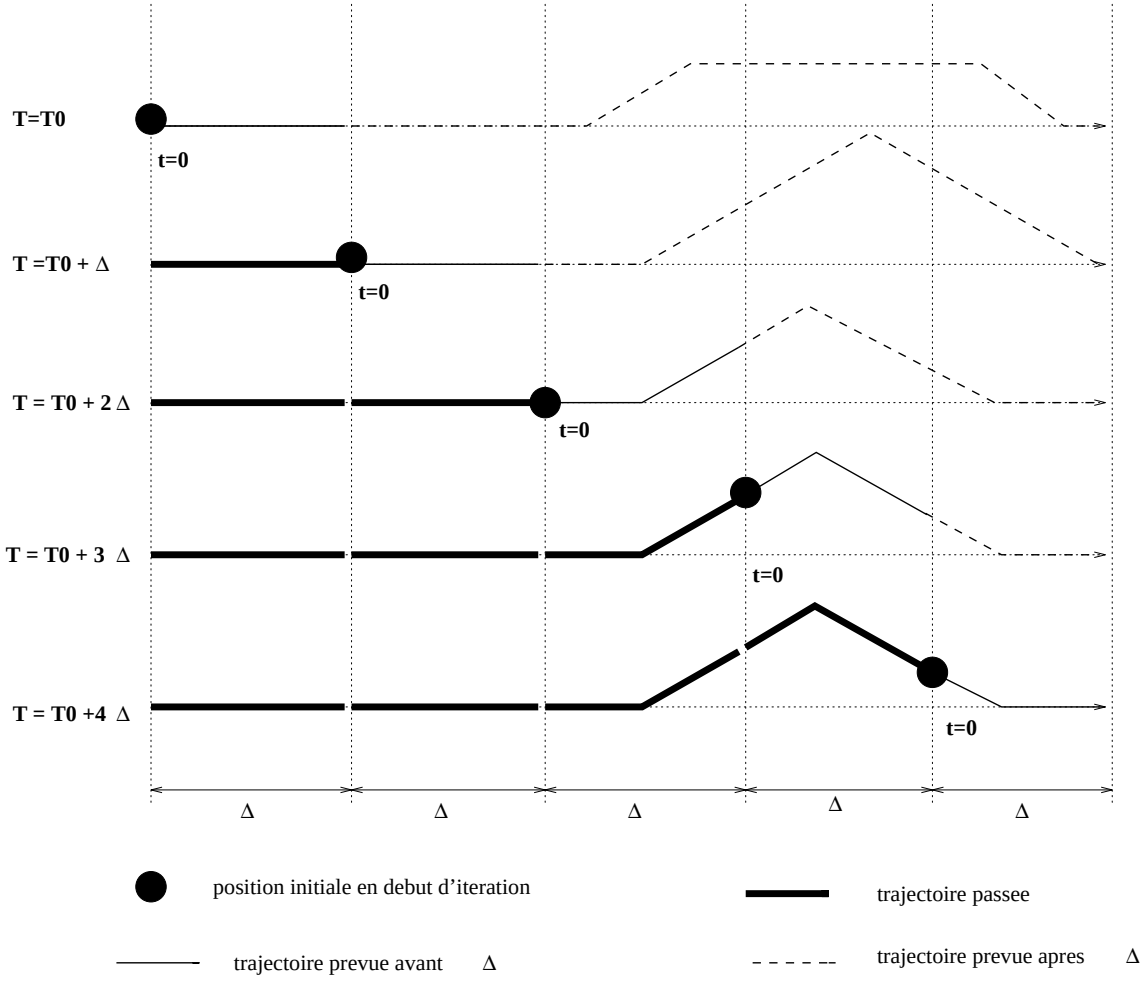


Figure 3.30: Séquencement en temps réel

ainsi remplacée par des volumes d'occupation de l'espace, se déformant avec le temps, représentant les positions potentielles futures des avions.

Tous ces éléments, associés à la forte complexité du problème, nous ont amenés à utiliser les algorithmes génétiques pour résoudre les conflits à plusieurs avions.

3.6.1 Codage

Pour le problème de résolution de conflits, les variables t_0, t_1, t_2 et α de chaque avion sont disposées dans une matrice (voir figure 3.31) qui constitue un individu de la population. On appellera gènes les variables t_0, t_1, t_2 et α de chaque avion. Un élément de population contient donc $4\,n$ gènes.

3.6.2 Génération de la population initiale

Dans le modèle que nous avons défini partie 3.28, l'algorithme de résolution de conflits est ré-initialisé pour chaque cluster toutes les Δ minutes. Il se peut donc que l'algorithme ait à prendre en compte des avions déjà engagés dans certaines manœuvres de résolution. Ces avions sont alors soumis à

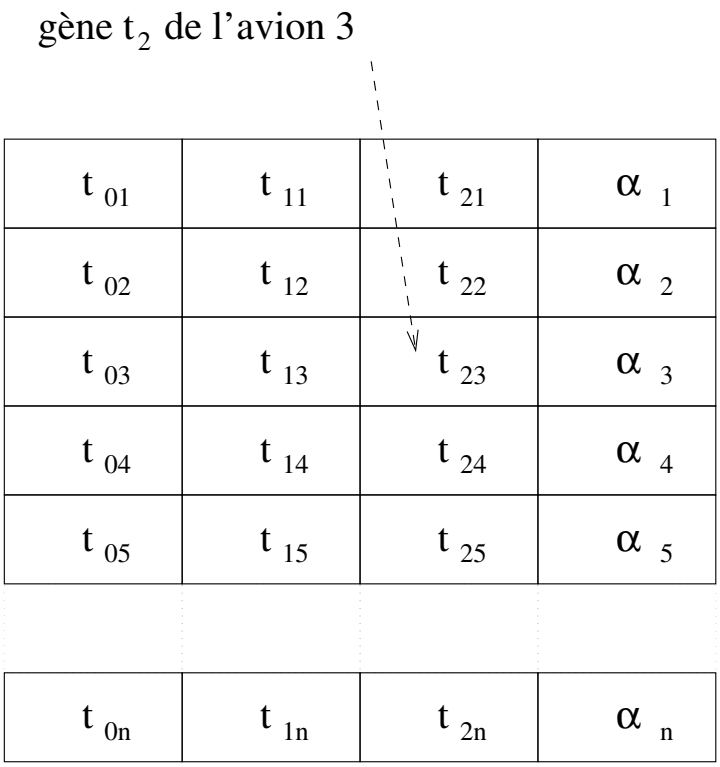


Figure 3.31: Codage d'un individu à n avions.

certaines contraintes gérées à priori par une procédure chargée de vérifier leur respect à toute étape de l'algorithme. Ces contraintes sont les suivantes :

- Lorsqu'un conflit à n avions est fourni à l'algorithme génétique, il se peut que certains gènes correspondant à des manœuvres déjà entamées ou qui doivent être entamées avant Δ soient imposés à l'algorithme génétique. Ainsi, tous les gènes t_0, t_1, t_2 dont la valeur est inférieure à Δ sont imposés dans l'algorithme et ne pourront plus être modifiés par la suite. De même si $t_0 < \Delta$ alors l'angle α correspondant est fixé et ne pourra plus être modifié. Toutes ces variables ne seront donc pas initialisées de façon aléatoire.

- On doit vérifier que $t_0 \leq t_1 \leq t_2$. Pour des avions arrivant à destination, il certaines manœuvres doivent être terminées avant une certaine échéance t_{3max} de sorte qu'il faut vérifier que $t_3 = t_2 + t_1 - t_0 \leq t_{3max}$.

Enfin, pour que l'algorithme génétique puisse détecter rapidement les solutions sans conflit, il sera fait en sorte que chaque avion de chaque chromosome de la population ait initialement une chance sur 3 de ne pas être dévié.

Les contraintes liées à la séparation des avions sont par contre directement introduites dans la fitness et ne sont pas gérées à priori mais à posteriori.

3.6.3 Fitness

La fonction fitness doit prendre en compte à la fois les contraintes de séparation des avions et la qualité de la résolution lorsque celle-ci est réalisée. Réduire à une seule valeur fitness toutes les calculs

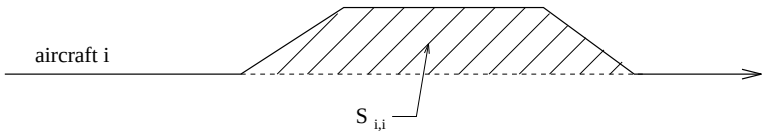


Figure 3.32: Surface occupé par la manœuvre

effectués pour l'évaluation d'un individu est très réducteur et une grande partie de l'information utile est perdue dans des opérations d'additions (voir [DAN94]). Aussi, la fonction fitness unique sera remplacée par une matrice triangulaire inférieure F. Si le cluster à résoudre comprend n avions alors n sera la taille de la matrice F. Si $i \neq j$, alors $F_{i,j}$ mesure la gravité du conflit entre l'avion i et l'avion j. S'il n'y a pas de conflit, $F_{i,j} = 0$, $F_{i,j}$ croit avec la gravité du conflit. $F_{i,i}$ mesure la pénalisation de la trajectoire de l'avion i provoqué par ses manœuvres. Cette matrice fitness d'où on pourra extraire une fitness scalaire contient beaucoup plus d'information qu'une fitness scalaire et permettra de définir les opérateurs de croisement et de mutation adaptés au problème de résolution de conflits.

Nous sommes en face d'un problème d'optimisation multi-critère. Nous avons retenu les critères suivants :

- Le retard induit pour chaque avion doit être aussi faible que possible.
- Le nombre d'avions déviés et le nombre total de manœuvres doit être aussi faible que possible, il se peut donc que les retards ne soient pas partagés par tous les avions.
- La durée d'une manœuvre doit être aussi faible que possible pour pouvoir rendre le plus tôt possible un avion disponible pour une autre manœuvre.
- Les trajectoires doivent respecter les normes de séparation.

Détaillons le calcul des éléments de la matrice F.

Calcul des éléments de la diagonale

Pour prendre en compte à la fois le retard dû à la manœuvre et la durée de la manœuvre elle-même, on peut mesurer la surface S occupée par la résolution (figure 3.32). Cette surface augmente à la fois avec le retard induit et la durée de la manœuvre. Il est intéressant de rappeler que cette notion de surface occupée est un des critères utilisés par le contrôleur pour décider d'une manœuvre. Pour minimiser le nombre de manœuvres, on ajoute à S le nombre de manœuvres multiplié par un certain coefficient k. Pour un point tournant, il faut donc rajouter $2\,k$ à S, pour un offset, il faut rajouter $3\,k$.

$$F_{i,i} \quad = \quad S_i + k\,N_i$$

Évaluation des termes non diagonaux

A chaque pas de temps t, on évalue la différence (si elle est positive) $C_{t,i,j}$ entre la séparation standard et la distance entre les segments i et j représentant les positions des avions i et j à t. Ces valeurs sont additionnées pour tous les temps t pour donner $F_{i,j}$, mesure du conflit entre i et j.

$$F_{i,j} \quad = \quad \sum_{t=0}^{total\,time} (C_{t,i,j})$$

A partir de la matrice F, et pour les besoins de l'opérateur de sélection, on est en mesure de calculer une fitness f scalaire de la façon suivante :

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

On remarquera que cette fitness permet de distinguer les individus pour lesquels il reste des conflits (leurs fitness sont inférieures à 0.5) des individus où il n'y a plus de conflit (leurs fitness sont supérieures à 0.5).

3.6.4 Opérateurs de croisement et de mutation adaptés

Pour les clusters de grande taille, il est intéressant d'utiliser la séparabilité partielle du problème et d'introduire l' opérateur de croisement adaptés. Pour cela, à partir de la matrice fitness F, on peut définir pour l'avion ou le chromosome i sa fitness locale :

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

L'opérateur de croisement est décrit par la figure 3.33. Après avoir choisi deux parents A et B, on compare les fitness locales de leurs avions (ces fitness sont notées A_i et B_i sur la figure 3.33). Pour l'avion i, si $A_i \leq B_i - \delta$ (δ est une valeur permettant de moduler le déterminisme de l'opérateur) les deux enfants héritent de l'avion i du père A. Si $B_i \leq A_i - \delta$, les deux enfants héritent de l'avion i du père B. Si aucune de ces deux conditions n'est respectée, les avions i des fils 1 et 2 sont deux combinaisons aléatoires des avions i des deux parents.

De même, on pourra définir un opérateur de mutation adapté décrit par la figure 3.34. Après avoir choisi un individu à muter, un avion est choisi (sur la figure 3.34 l'avion 4 est choisi). Un avion i ayant une fitness F_i faible (c'est à dire inférieure à ϵ où ϵ module le déterminisme de l'opérateur) ne sera choisi que si toutes les fitness locales sont faibles.

Ces opérateurs ont l'avantage d'être assez déterministes en début de convergence de sorte qu'une solution sans conflit (dont la fitness est supérieure à 0.5) peut être rapidement dégagée. Quand les solutions sans conflit deviennent suffisamment nombreuses, ces opérateurs deviennent moins déterministes et la recherche dans l'espace d'état devient plus large. L'association de ces opérateurs avec une méthode de sharing devient indispensable pour atténuer l'effet du déterminisme introduit.

3.6.5 Sharing simplifié

On se propose d'utiliser la méthode de sharing simplifiée suivante qui a l'avantage de distinguer de façon discrète deux solutions qui n'ont pas les mêmes caractéristiques. Ainsi, définissons la distance discrète suivante ($\alpha_i(A)$ est l'angle de déviation de l'avion i du chromosome A) :

- deux individus A et B sont séparés par une distance nulle si :

$$\forall i \in [1,n]\,, \alpha_i(A) = \alpha_i(B)$$

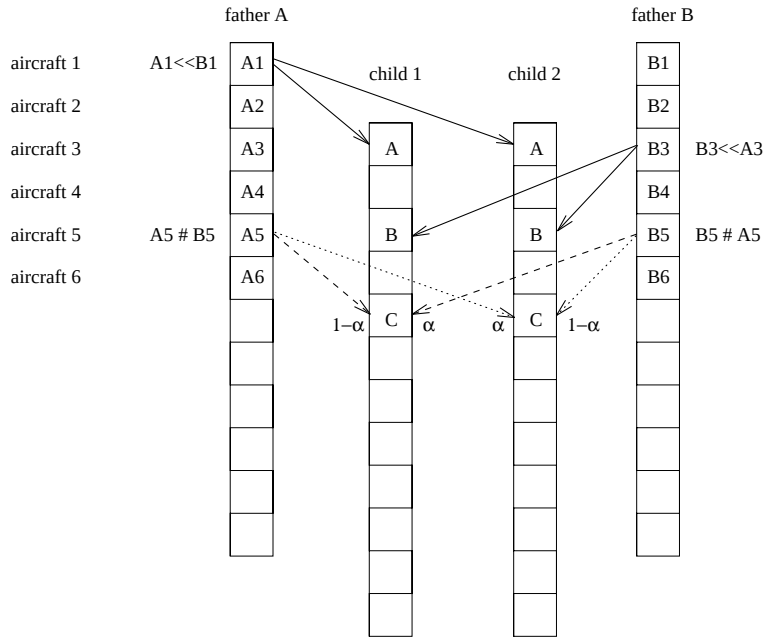


Figure 3.33: Opérateur de croisement adapté

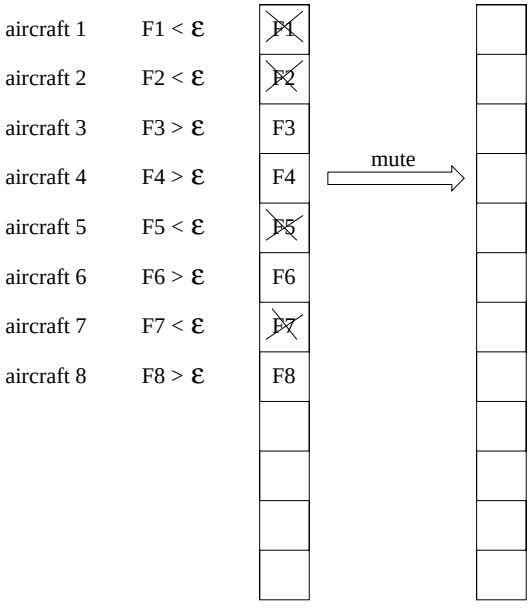


Figure 3.34: Opérateur de mutation adapté

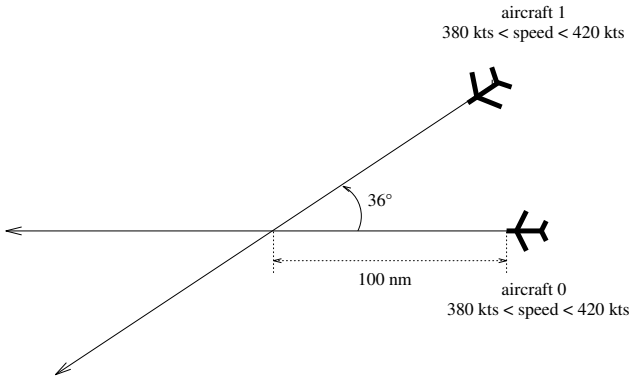


Figure 3.35: Conflit à deux avions

- deux individus A et B sont séparés d'une distance 1 si :

$$\exists i \in [1, n], \alpha_i(A) \neq \alpha_i(B)$$

Deux individus séparés par une distance nulle seront dits appartenir à la même classe. Si l'on applique la méthode de sharing clusterisé présentée précédemment, on remarque que cela revient à diviser la fitness de chaque individus par le nombre de représentants de sa classe.

3.6.6 Méthode locale en fin de convergence

Les algorithmes génétiques sont très efficaces pour résoudre des problèmes d'optimisation combinatoire de grande taille mais leur efficacité est moindre pour une recherche locale d'une bonne précision. En conséquence, il est très utile, après la dernière génération de l'algorithme génétique d'appliquer une méthode d'optimisation locale (de type gradient) sur la meilleure solution de chaque classe d'individus définie précédemment. La méthode locale utilisée est très simple : tous les gènes de tous les chromosomes de l'individu qu'on optimise sont successivement modifiés de façon à améliorer la fitness globale sans générer de nouveau conflits.

3.6.7 Applications numériques

Justification du modèle

Dans ce premier exemple, on considère un conflit très simple entre 2 avions (voir figure 3.35).

Le temps d'anticipation Δ est fixé à 2 minutes et la vitesse des avions à 400 nœuds avec une incertitude de 5 pour cent. Dans ce cas, si l'avion 0 a une vitesse sol de 380 nœuds et l'avion 1 une vitesse réelle de 420 nœuds, il n'y aura finalement pas de conflit (la séparation standard est de 4 nm). Au cas où les avions ne respecteraient pas leur trajectoire, l'algorithme calcule des solutions dans le cas le plus défavorable. La figure 3.36 donne le résultat de l'algorithme génétique à $t = 3$ minutes, 5 minutes, 7 minutes, 9 minutes, 11 minutes and 13 minutes. Les traits épais représentent les deux minutes à venir qui ne peuvent être modifiées. Les traits fins décrivent les trajectoires optimales provisoires calculées par l'algorithme génétique.

En raison de l'incertitude, la première optimisation donne des trajectoires robustes mais très pénalisantes. Au fur et à mesure que le temps passe, les trajectoires s'affinent car l'incertitude décroît. Finalement, à $t = 12$ le conflit disparaît.

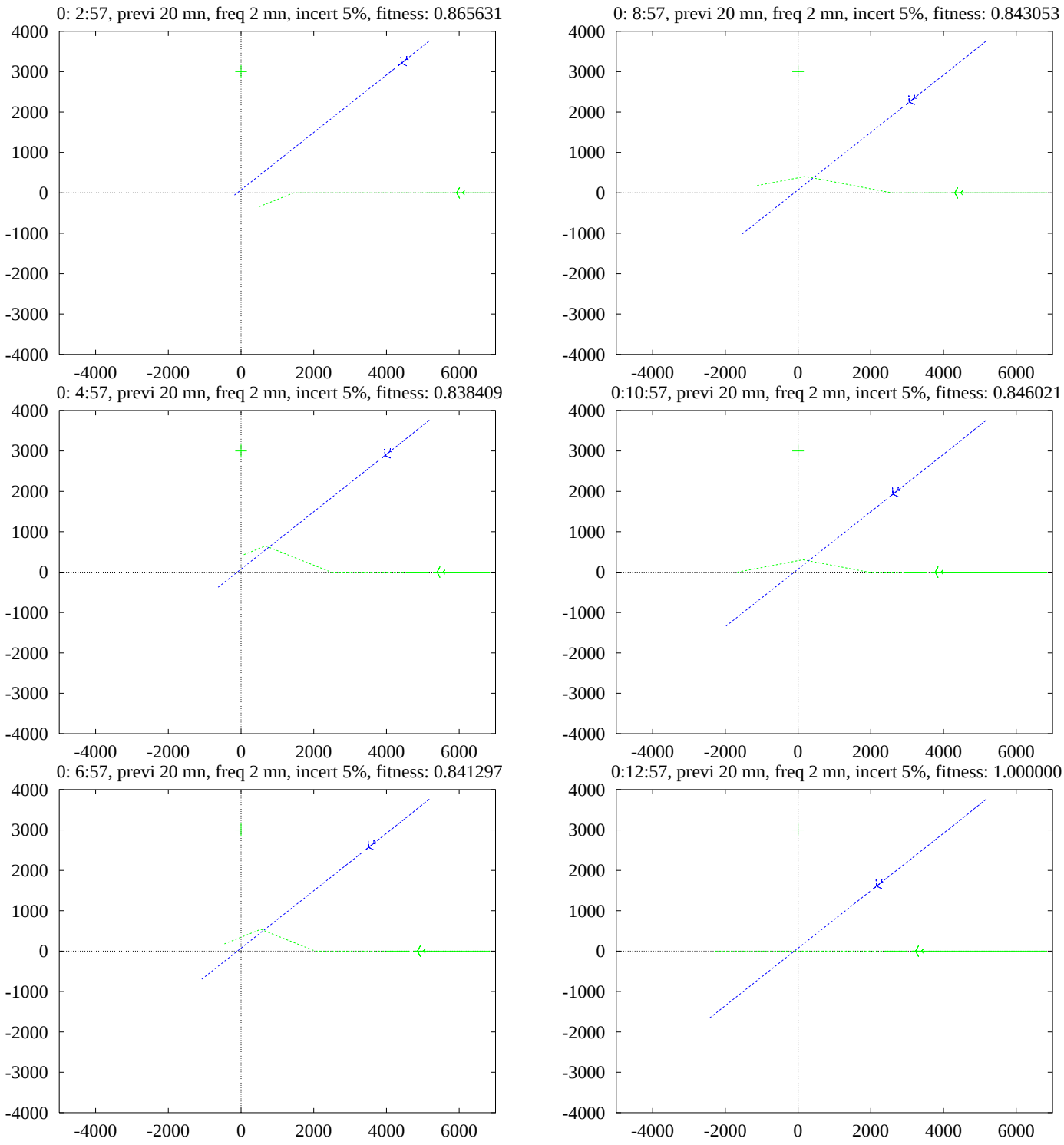


Figure 3.36: t=3, 5, 7, 9, 11, 13 minutes

incertitude	0%	1%	2%	3%	4%	5%
1 mn	4.00	4.24	4.30	4.55	4.65	4.70
2 mn	4.03	4.35	4.52	5.00	5.30	5.77
3 mn	4.01	4.44	4.89	5.16	5.61	5.97
4 mn	4.02	4.50	5.02	5.63	6.10	6.68
5 mn	4.03	4.57	4.89	6.01	6.61	7.32

Tableau 3.5: Distance minimale entre les avions pour différents temps d'anticipation.

Un problème combinatoire

Dans cet exemple, 5 avions convergent vers le même point à une vitesse de 400 nœuds avec 3 pour cent d'erreur. Δ vaut ici 5 minutes. Les autres paramètres de l'algorithme génétique n'ont pas changé. La figure 3.37 donne les résultats. La distance minimale d'espacement des avions est de 6.01 malgré une forte incertitude sur les vitesses (3%).

Le tableau 3.5 donne la distance minimale observée entre les avions pour différents pourcentages d'erreur et différents temps d'anticipation. Il est clair à la vue de ce tableau que lorsque l'on a simultanément une forte incertitude et une grande anticipation la qualité de la résolution décroît fortement.

Les résultats représentés ne donnent que la meilleure solution de ce problème à 5 avions. L'algorithme génétique propose grâce au sharing d'autres solutions proches de la solution optimale (notamment la solution symétrique dans le cas ci-dessus).

Plusieurs solutions quasi-optimales

Le but de cet exemple est de montrer que l'algorithme génétique est capable de générer plusieurs solutions proches de l'optimum lorsqu'elles existent. On a, après la première optimisation conservé toutes les solutions admissibles ayant une fitness au moins égale à 95% de la meilleur fitness.

On peut voir figure 3.38 les solutions quasi-optimales proposées pour un conflit à trois avions.

Efficacité des opérateurs adaptés

Pour tester l'efficacité des opérateurs de croisement et de mutation adaptés introduit en 3.33, nous proposons de résoudre un conflit impliquant chacun 20 avions. Ce conflit implique 20 avions convergeant simultanément sur un même point. Ce conflit a très peu de solutions car tous les avions sont en conflit deux à deux. En fait, deux solutions existent : dans la première, tous les avions tournent à droite, dans la seconde, tous les avions tournent à gauche. La figure 3.39 donne le résultat de l'optimisation.

Pour étudier l'effet des opérateurs de croisement adaptés combinés à l'utilisation du sharing, 4 différents tests ont été effectués sur les deux conflits précédents en utilisant : les opérateurs adaptés sans sharing, les opérateurs adaptés avec sharing, les opérateurs classiques sans sharing, les opérateurs classiques avec sharing. Pour ces quatre tests, nous avons mesuré pour chaque exemple la valeur du meilleur élément de population (figure 3.40) pour les 20 générations de l'algorithme génétique. On peut d'abord constater que les opérateurs adaptés sont très efficaces pour trouver des solutions admissibles . En effet, avec les opérateurs classiques, aucun des deux conflits n'est résolu avant la génération 20 (en fait, des tests montrent que la première solution admissible est obtenue après la génération 200). Avec les opérateurs adaptés, une solution admissible est toujours trouvée avant la génération 10 (Une solution est admissible si sa fitness est supérieure à 0.5). La figure 3.40 montre que le sharing a tendance a freiner la croissance de la fitness du meilleur élément. Cependant la fitness

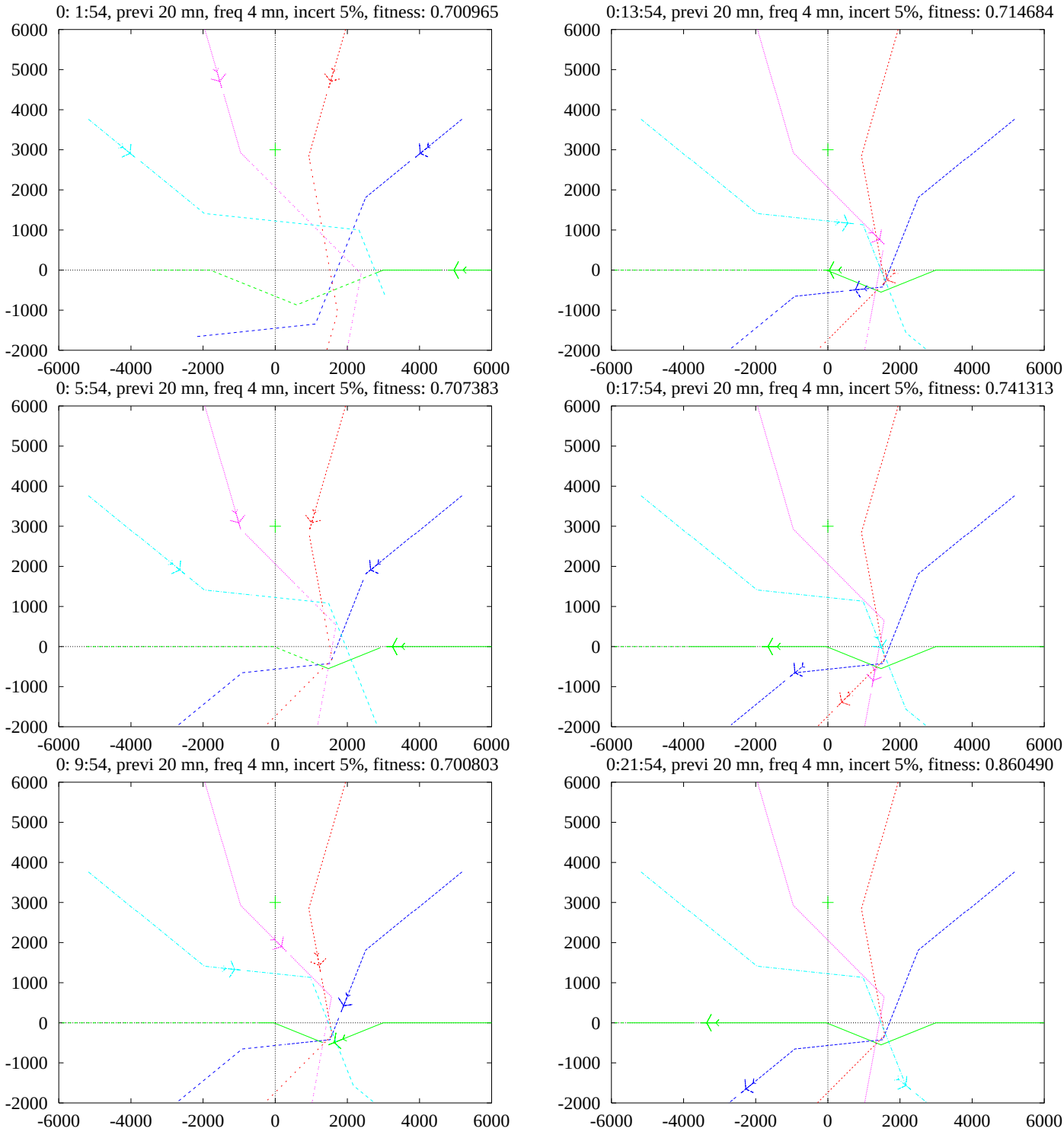


Figure 3.37: t=2, 6, 10, 14, 18, 22 minutes

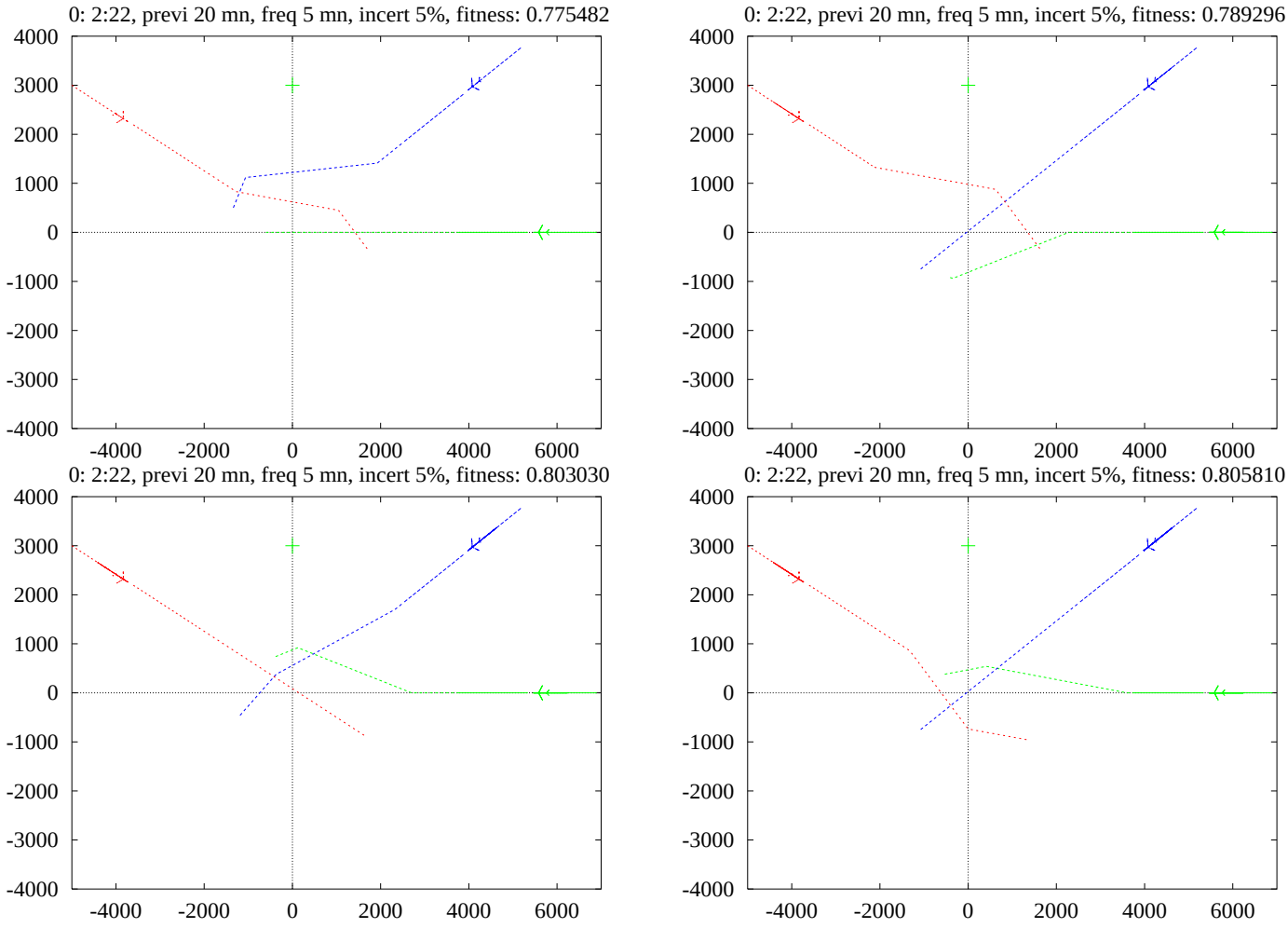


Figure 3.38: Conflit à 3 avions, plusieurs solutions quasi-optimales

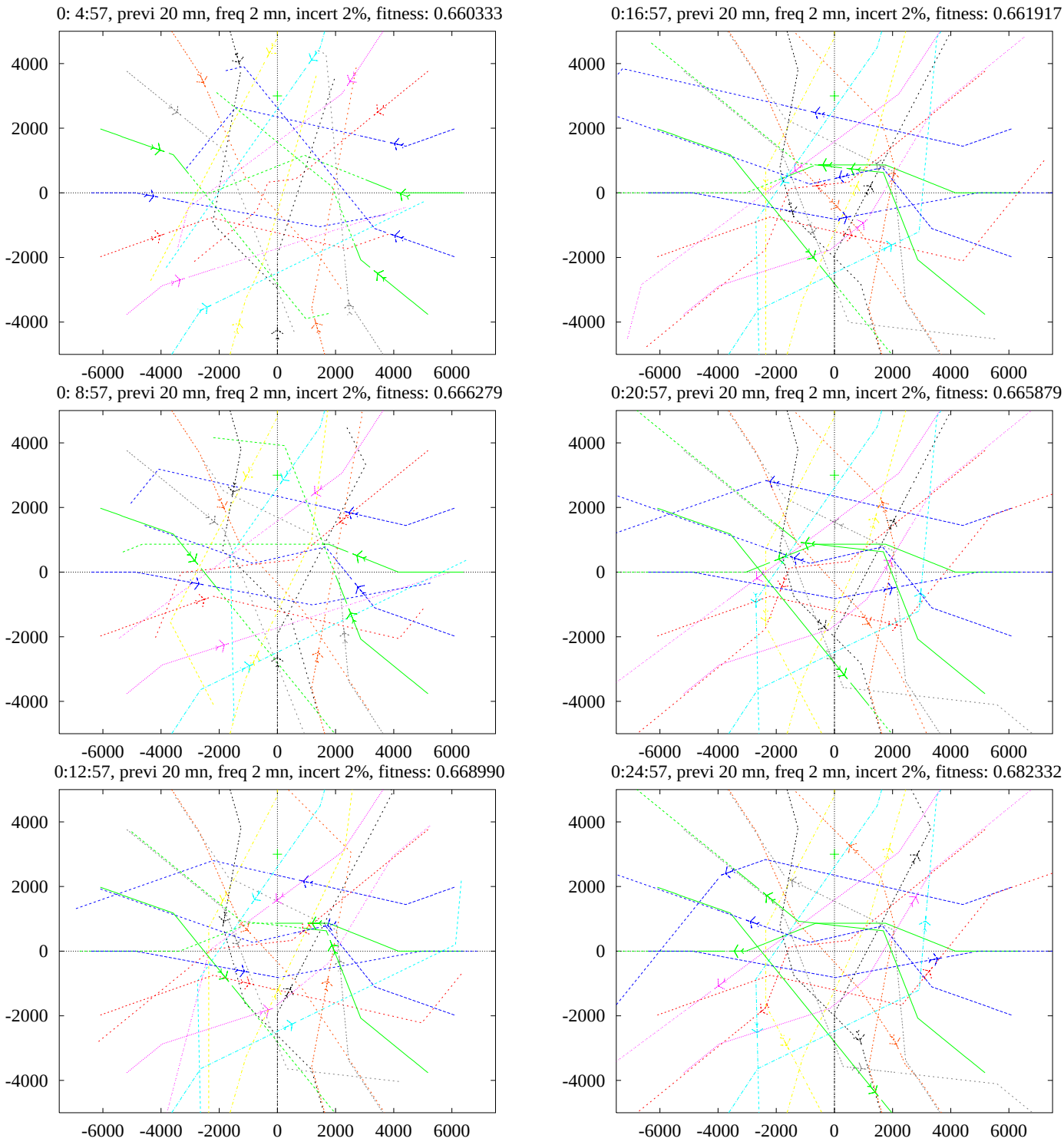


Figure 3.39: 20 avions sans conflit

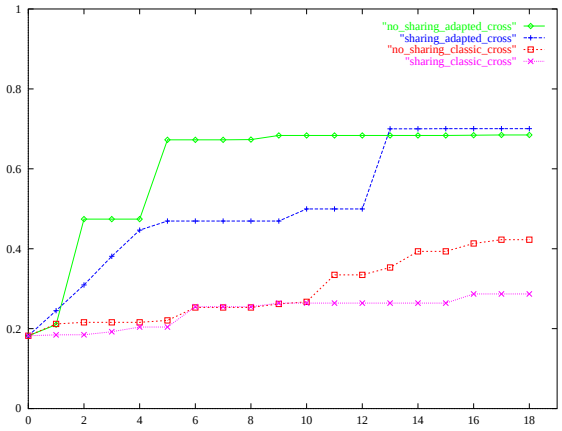


Figure 3.40: Meilleure fitness (avions en cercle

finale est aussi bonne lorsque l'on utilise le sharing que lorsque l'on ne l'utilise pas. Elle est même meilleure en utilisant le sharing lorsque le conflit est difficile à résoudre. Le sharing permet d'ailleurs d'obtenir dans ce dernier cas les deux solutions symétriques (tous les avions tournant à droite et tous les avions tournant à gauche).

Un conflit de rattrapage.

Le but de cet exemple est de montrer l'intérêt de la modélisation par offset dans le cadre d'un conflit en rattrapage. Dans l'exemple présenté 5 avions se suivent sur la même route. L'avion de tête est le plus lent (350 nœuds), l'avion de queue est le plus rapide (550 nœuds).

Les résultats sont représentés figure 3.41. On observe que l'offset est utilisé pour les 4 avions les plus rapides, le plus lent n'étant pas dévié. La déviation est est d'autant plus importante que l'avion est rapide, ce qui reste en accord avec les conclusions de la partie 3.3.3.

Une résolution globale des clusters

Dans cet exemple, deux conflits indépendants à deux avions face à face sont regroupés dans le même cluster par l'intermédiaire d'un avion coupant leurs axes. La figure 3.42 donne le résultat de l'optimisation. On observe qu' une seule déviation de l'avion coupant les axes résout 4 conflits.

3.6.8 Architecture générale du simulateur de trafic

Les tests effectués précédemment montrent que l'algorithme génétique utilisé est très performant sur des cas d'école. Il convenait donc de tester l'algorithme dans un environnement réel.

L'architecture du simulateur de trafic (figure 3.43) est la suivante :

- Le processus $P1$ est le simulateur de trafic proprement dit décrit dans la section 3.1.1.
- Le processus $P2$ est chargé de détecter les paires d'avions en conflit, de fabriquer les clusters d'avions par fermeture transitive des paires d'avions en conflits. Il vérifie également les trajectoires nouvelles proposées par $P3$.
- Le processus $P3$ est l'algorithme de résolution de conflits décrit dans le chapitre précédent.

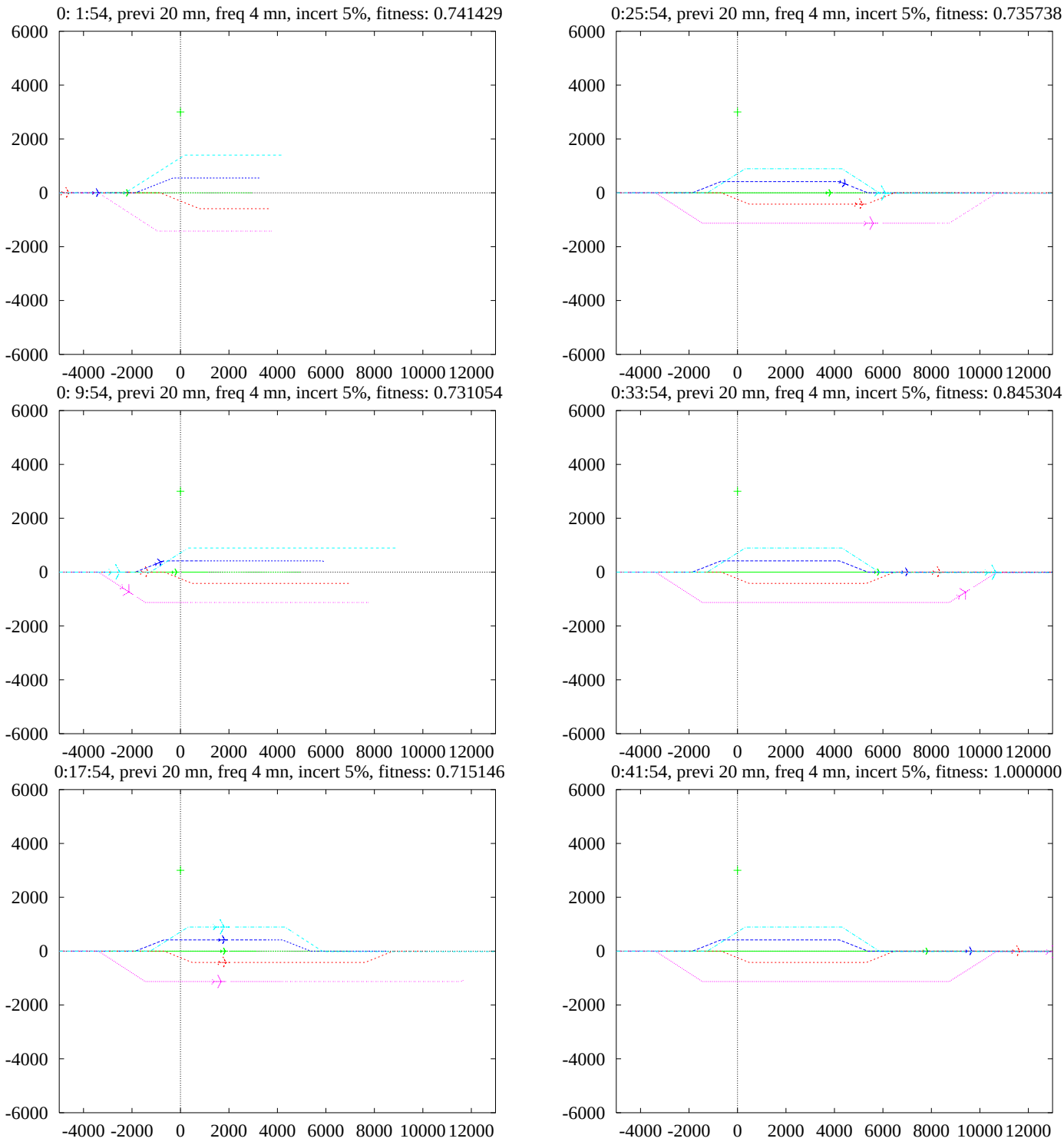


Figure 3.41: 5 avions sans conflit

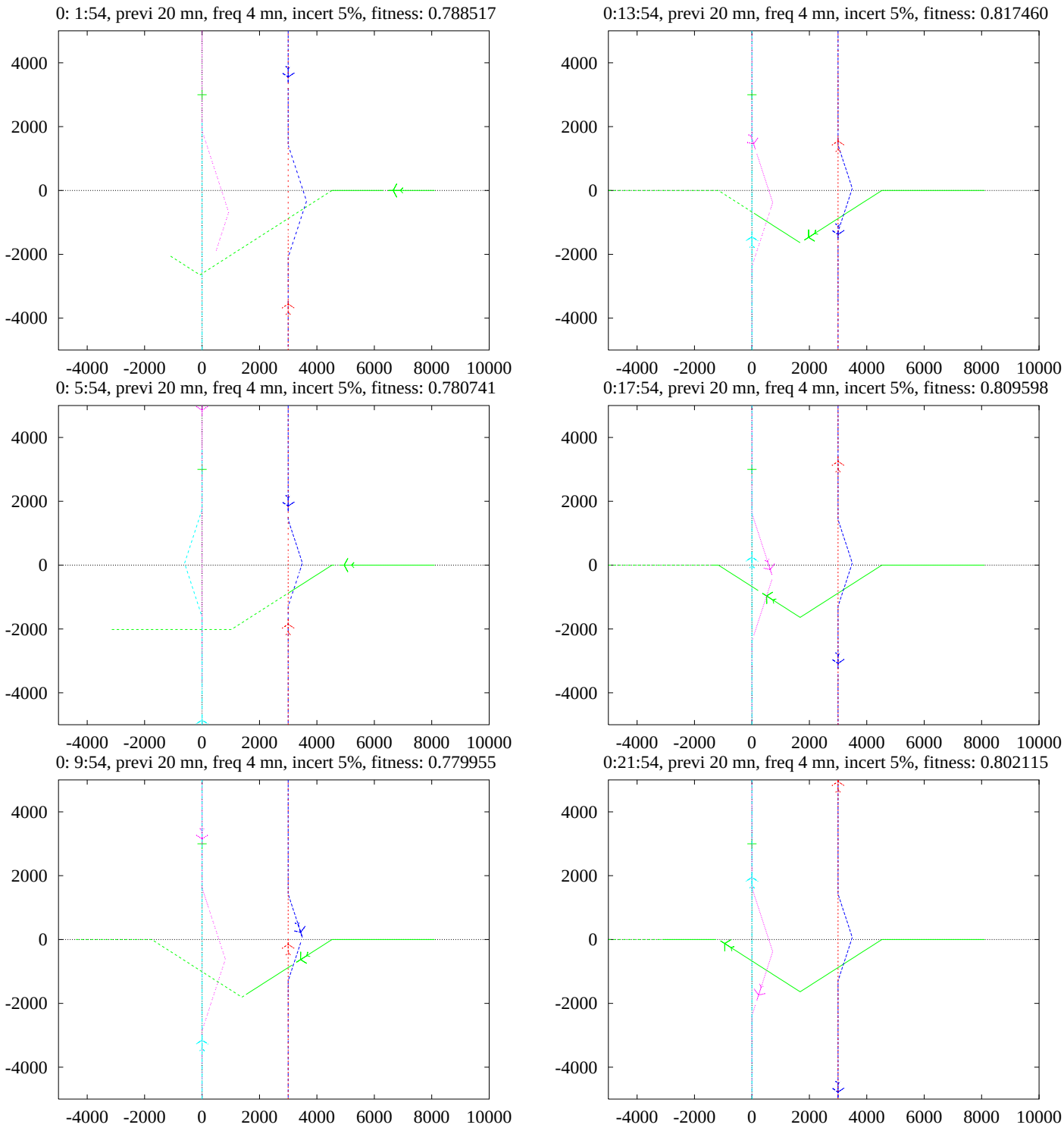


Figure 3.42: 5 avions sans conflit

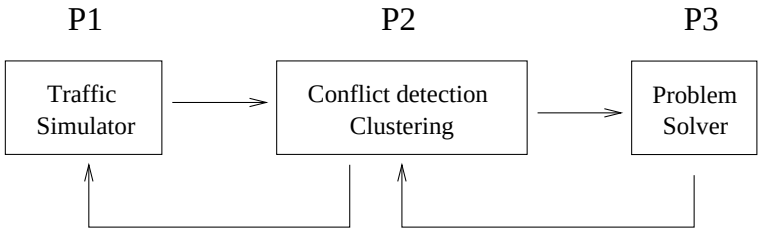


Figure 3.43: Architecture générale

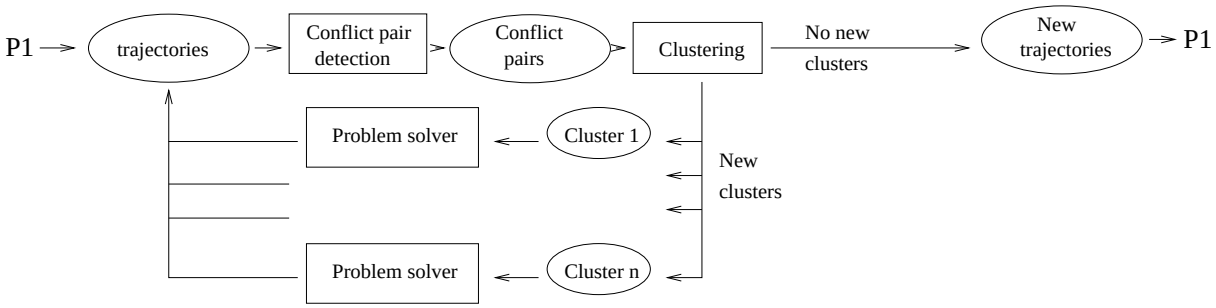


Figure 3.44: Architecture détaillée du simulateur de trafic

L'architecture du simulateur de trafic est détaillée figure 3.44.

Le processus $P1$ envoie les positions courantes des avions et les plans de vols au processus $P2$. $P2$ construit alors les trajectoires prévues pour les 20 minutes à venir et détecte les paires de conflits. $P2$ prend en compte les incertitudes sur les vitesses (voir la partie 3.4). Après avoir détecté les paires d'avions, $P2$ construit les clusters d'avions par fermeture transitive des paires d'avions sur les 20 minutes à venir. Les clusters sont en fait les classes d'équivalence de la relation "est en conflit avec". Tous les clusters sont ensuite résolus par le résolveur de conflits de façon indépendante. Le résolveur de conflits retourne à $P2$ les trajectoires modifiées. $P2$ vérifie alors que les trajectoires modifiées d'un cluster ne génèrent pas de nouveaux conflits avec les avions d'un autre cluster. Si tel est le cas, les ordres de résolutions pour les 5 minutes à venir sont envoyés à $P1$. Si le résolveur a généré de nouveaux conflits entre des avions de deux clusters différents, ces deux clusters sont regroupés, formant un nouveau cluster sur lequel on applique le résolveur de conflits. Ce processus est répété jusqu'à ce que tous les conflits disparaissent. Ce processus converge nécessairement. Dans le pire des cas, tous les avions en conflits dans les 20 prochaines minutes se retrouvent dans le même cluster. Cependant, le processus est d'autant plus efficace qu'il est en présence de nombreux clusters que l'on résoudra en parallèle[16].

3.6.9 Résultats sur une journée de trafic réelle

Les expérimentations pratiques sur le simulateur de trafic sont toujours en cours mais certains résultats sont déjà disponibles [Cha95].

Nous décrivons ici un test effectué avec les plans de vols du 12 novembre 1992. 4835 vols ont été enregistrés ce jour-là. Le pourcentage d'incertitude sur les taux de montée et sur la vitesse ont été fixés à 1% et les séparations standards fixées à 6 nm et 1000 pieds. Les routes directes ont été utilisées

[16]Dans la pratique, on utilise un réseau de stations de travail reliées par Ethernet et PVM [GBD+94] pour passer des messages entre les différentes applications

(les avions ne transitant pas par des balises, mais rejoignant directement leur destination). Seuls les conflits “en route” (au dessus de 6000 pieds) ont été considérés.

Lorsque cette journée est simulée avec un algorithme de détection ne prenant pas en compte l'incertitude, 665 conflits au dessus de 6000 pieds sont détectés.

Lorsque une simulation complète est exécutée, le processus $P2$ détecte 4408 paires d'avions en conflits ne représentant en fait que 908 conflits différents (un certain nombre de conflits durent plus de 5 minutes et sont donc détectés plusieurs fois). 1888 clusters ont étés proposés au résolveur de conflits. Il en a résolu 1541.260 ont été presque résolus (le conflit prévu n'a finalement pas eu lieu en raison de l'incertitude adoptée). Le résolveur a échoué sur 80 conflits représentant en fait 23 différents conflits (répétés dans le temps). En observant ces conflits, on a pu remarquer qu'ils étaient tous dus à des décollages, atterrissages ou entrées dans l'espace aérien français simultanés (ces conflits apparaissent car des données plan de vol non régulées ont été utilisées). Dans ce cas, le résolveur de conflit ne peut bien évidemment pas séparer les avions. Ne considérer que les conflits au dessus de 6000 pieds ne suffit pas à s'affranchir de ce problème. Un algorithme de pré-régulation des plans de vol est en cours d'élaboration pour résoudre ce problème.

3.6.10 Conclusion

Les Algorithmes Génétiques se sont avérés très efficaces pour résoudre des conflits pouvant impliquer jusqu'à une vingtaine d'avions. L'introduction d'un opérateur de croisement adapté aux fonctions partiellement séparables combiné avec une méthode de sharing s'est toutefois révélée très utile. On notera que l'efficacité de cet opérateur peut également se constater sur des fonctions tests classiques partiellement séparables. La fonction d'évaluation utilisée est très simple mais pourrait être affinée autant qu'on le souhaite. Les résultats que nous avons obtenus sont très encourageants. Ils le sont d'autant plus que le résolveur de conflit ne donne pour l'instant que des résolutions dans le plan vertical.

3.7 Programmation linéaire et AG

3.7.1 Introduction

Nous avons présenté dans la section précédente la modélisation par offset qui linéarise les contraintes de séparation. Rappelons que, si l'on a fixé le sens de l'offset et l'ordre de passage des n avions, on a à résoudre, pour obtenir les valeurs des offsets, un problème d'optimisation linéaire à n inconnues (les valeurs des offsets des n avions), et à $\frac{n(n+1)}{2}$ contraintes (ou à $\frac{n(n+2)}{2}$ contraintes, si on borne les valeurs des offsets).

Notons que l'on a tout de même $2^{\frac{n(n+1)}{2}}$ façons possibles de fixer les sens des offsets et les ordres de passage. En effet, pour chaque avion, il y a deux sens de déviation possibles : à gauche, ou à droite. Et pour chaque couple d'avions (a_i, a_j) (il y en a $\frac{n(n-1)}{2}$), il y a deux ordres de passage possibles au point d'intersection C_{ij} de leurs trajectoires : soit a_i passe après a_j, soit a_i passe avant a_j. Dès que le nombre d'avions impliqués dans le conflit augmente, il devient beaucoup trop long d'utiliser une technique d'énumération exhaustive de toutes les combinaisons possibles : pour 5 avions, il y en a 32768, et pour 6 avions, il y en a 2097152.

Un problème linéaire peut en revanche se résoudre facilement, à l'aide par exemple d'un simplex, mais on n'obtient alors qu'un optimum local, et l'on voit bien que la forte combinatoire du problème empêche de tester l'ensemble des configurations possibles dès que le nombre d'avions devient un peu important.

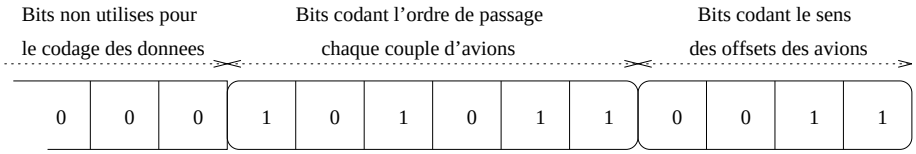


Figure 3.45: Codage d'une configuration par un entier

Notre idée est donc d'utiliser un algorithme génétique qui générera des "configurations" : une *configuration* représente la donnée des sens de déviation et des ordres de passage, nécessaire pour établir les conditions linéaires de séparation des avions. Une fois la configuration choisie, la résolution du problème linéaire donnera la valeur de l'offset à faire subir à chaque avion de manière à minimiser la somme des retards.

3.7.2 Codage des données du problème

Le sens de déviation de chaque avion ne peut prendre que deux valeurs différentes (offset à droite ou offset à gauche). De même, si l'on considère le couple d'avions (a_i,a_j), et que l'on considère que les trajectoires des deux avions sont portées par des droites sécantes, il n'y a que deux ordres de passage des avions au point C_{ij}, point d'intersection de leurs trajectoires qui soient possibles sans conflit : soit a_i passe en C_{ij} après a_j, soit a_i passe avant. Ces données peuvent, de même que les sens des offset, se coder de manière binaire On utilise donc le mode de codage le plus classique pour les chromosomes dans le cadre des algorithmes génétiques : une séquence de bits, dans laquelle chaque bit traduit la valeur, ou l'état d'une donnée du problème.

Pour un problème à n avions, les n premiers bits (à partir de la droite), codent les sens des offset de chaque avion : si le i^{eme} bit est un 1, l'avion a_i sera dévié vers la gauche ; il sera dévié vers la droite si ce bit est un 0. Les $\frac{n(n-1)}{2}$ bits suivants codent le sens de passage des avions pour chaque couple d'avions. On numérote les couples d'avions selon l'ordre suivant : (a_1, a_2), $(a_1, a_3), \ldots (a_1, a_n)$, $(a_2, a_3), \ldots (a_{n-1}, a$ Si pour le couple d'avions (a_i, a_j), où $i < j$, a_i passe derrière a_j, le bit correspondant est un 1; ce bit est un 0 si a_i passe devant a_j. La figure 3.45 montre un exemple du codage d'une configuration pour un problème d'évitement impliquant 4 avions. Le chromosome correspondant est codé par l'entier 691; l'avion a_1 est dévié vers la gauche, l'avion a_3 est dévié vers la droite, l'avion a_1 passe derrière l'avion a_2 (5^{eme} bit de 691), et l'avion a_2 passe devant l'avion a_4 (9^{eme} bit de 691).

3.7.3 Croisement et mutation

Le principe de la technique de croisement que nous avons utilisé est de choisir aléatoirement les bits dont l'enfant E_1 hérite du parent P_1. Ses autres bits lui viendront du parent P_2, et l'héritage de l'enfant E_2 sera déterminé comme étant le complémentaire de celui de E_1.

Pour la mutation, on choisit aléatoirement un des bits de la séquence utilisée pour le codage d'une configuration, et on le modifie.

3.7.4 Évaluation de la fitness

Nous nous sommes proposé de prendre comme fitness le retard minimal associé à une configuration, retard obtenu au moyen du programme d'optimisation linéaire, comme critère d'adaptation de l'élément de la population correspondant à cette configuration.

Cette méthode présente un inconvénient majeur : certaines configurations mènent à des problèmes d'optimisation *infaisables*, c'est à dire pour lesquels les contraintes ne peuvent pas être toutes satisfaites simultanément. Prendre une *fitness* nulle pour les éléments de la population correspondant à de telles configurations ne serait pas une solution satisfaisante : une configuration menant à un problème infaisable peut être mieux *adaptée* qu'une autre, en ce sens qu'elle peut être plus proche qu'elle d'une configuration menant à un problème faisable. Si une configuration mène à un problème d'optimisation infaisable, on essaiera de supprimer une contrainte, si les contraintes restantes ne peuvent toujours pas être satisfaites simultanément, on en supprimera une seconde, et ainsi de suite, jusqu'à ce que les contraintes restantes soient simultanément satisfiables. Cette méthode se base sur l'idée intuitive selon laquelle un problème d'optimisation infaisable est d'autant plus "proche" d'un problème faisable qu'il y a peu de contraintes à supprimer pour que les contraintes restantes puissent être toutes simultanément satisfiables. La fitness d'une configuration "infaisable" sera pénalisée en fonction du nombre de contraintes qu'il a fallu supprimer pour trouver une configuration satisfiable, mais ne sera donc pas nulle.

3.7.5 Résultats

L'évaluation des résultats obtenus pose le problème suivant : il est difficile d'obtenir par une autre méthode une solution dont on sache qu'elle est optimale, pour la comparer à la solution obtenue grâce à l'algorithme ci-dessus. De plus l'algorithme génétique est un algorithme stochastique : pour évaluer son efficacité, il faut pouvoir le faire tourner un grand nombre de fois, et considérer la moyenne et la distribution des résultats obtenus.

Conflit à cinq avions

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

Nous considérons pour cela l'exemple suivant : les avions vont à la même vitesse (400 kts), ils sont au temps t_o régulièrement répartis sur un demi cercle de centre C (et de rayon 100 Nm), et leurs trajectoires se coupent en C (voir figure 3.46). La norme de séparation est de 7 Nm. On obtient alors deux solutions optimales équivalentes : soit les quatre premiers avions sont déviés vers la droite, tandis que le cinquième n'est pas dévié, et si $i < j$, a_i passe derrière a_j, soit le premier avion n'est pas dévié, et les quatre suivants sont déviés vers la gauche, et si $i < j$, a_i passe devant a_j. C'est cette dernière solution qui est représentée sur la figure 3.46.

Pour un conflit à 5 avions, le programme a été lancé 50 fois sur l'exemple présenté ci-dessus, avec 150 éléments de population et pendant 100 générations.

La figure 3.46 montre la répartition du nombre d'appels au programme d'optimisation linéaire nécessaire à l'obtention d'une solution optimale. Ces résultats sont à comparer au fait que pour parcourir toutes les configurations, il faut effectuer 32 768 appels au programme linéaire (ce qui correspond à un temps de calcul de 108 secondes). Le gain en nombre d'appels au programme linéaire est beaucoup plus sensible dans le cas d'un conflit à 6 avions, dont la combinatoire est beaucoup plus importante. Nous allons étudier plus en détail les résultats obtenus dans ce cas.

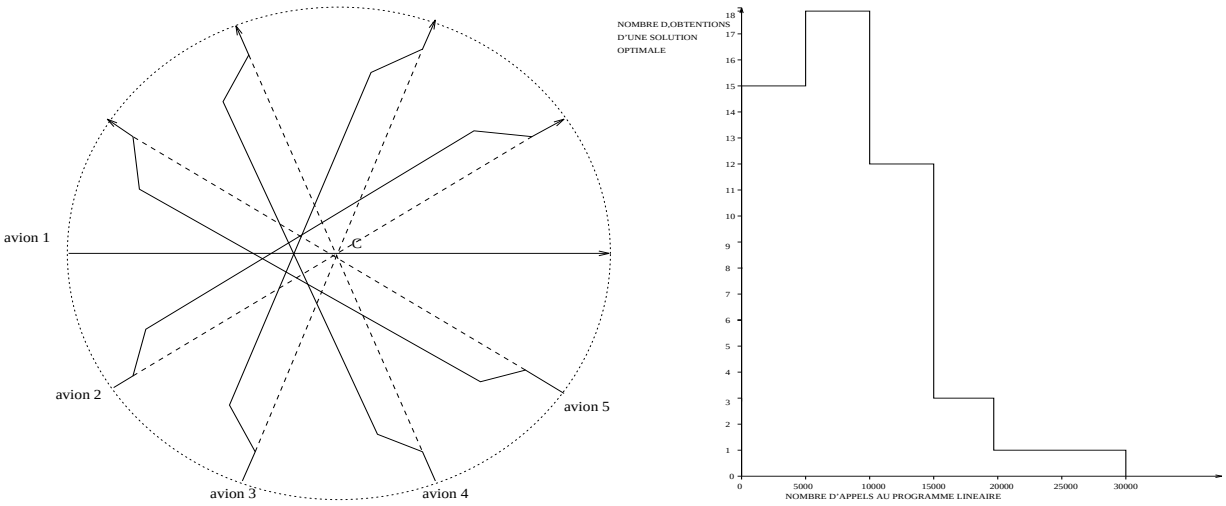


Figure 3.46: Cas à 5 avions

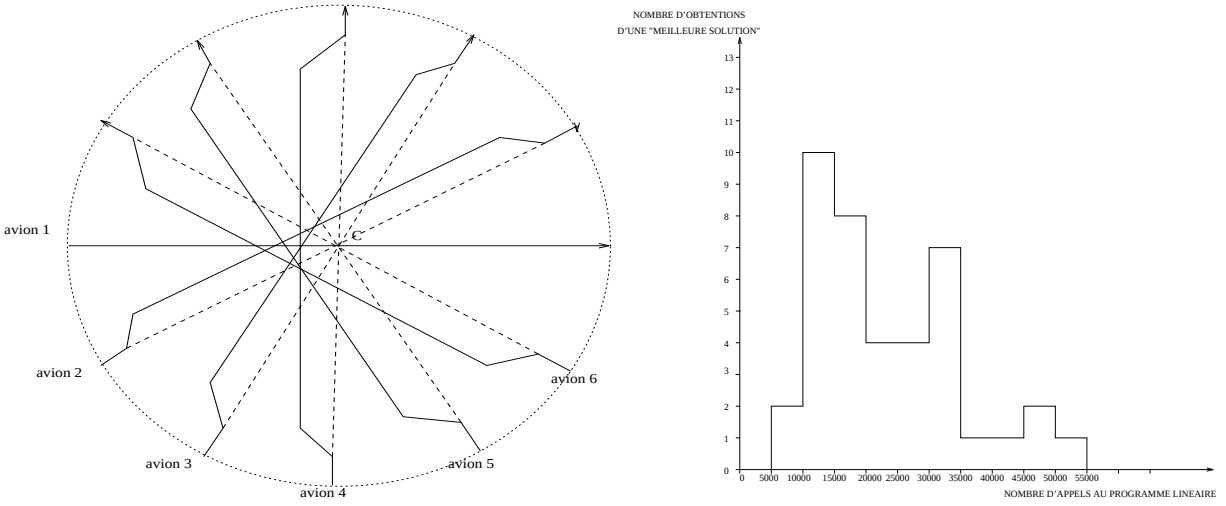


Figure 3.47: Cas à 6 avions

Conflit à six avions

Les solutions obtenues dans le cas d'un conflit à 6 avions sont plus délicates à vérifier. Nous avons appliqué la méthode décrite ci-dessus à un conflit à 6 avions comparable au conflit à 5 avions présenté dans la section précédente : les avions vont à la même vitesse (400 kts), ils sont au temps t_o régulièrement répartis sur un demi cercle de centre C (et de rayon 100 Nm), et leurs trajectoires se coupent en C (voir figure 3.47). La norme de séparation est toujours de 7 Nm.

Le programme a été utilisé avec les mêmes paramètres que dans le cas à 5 avions. Sur les 50 fois essais, nous avons obtenu une des deux *meilleures solutions* 39 fois. Il a fallu en moyenne 23000 appels au programme d'optimisation linéaire pour obtenir une de ces solutions. La figure 3.47 montre la répartition de ce nombre d'appels.

Nous rappelons qu'un parcours exhaustif de tous les cas de figure en nécessite 2097152.

3.7.6 Conclusion

Beaucoup de paramètres ayant permis d'atteindre ces résultats pourraient encore être ajustés de manière à améliorer l'efficacité de la méthode présentée ici (notamment les formes exactes des fonctions utilisées pour le calcul de la *fitness* des éléments). Cependant les résultats obtenus, surtout avec l'exemple à 6 avions, montrent que cette méthode peut s'avérer intéressante sur le plan théorique. Pourtant, sur le plan pratique, cette méthode présente un inconvénient majeur : la trajectoire des avions doit nécessairement être rectiligne uniforme. Or, la vitesse d'un avion varie ne serait-ce qu'en raison d'un changement d'altitude. D'autre part, cette méthode ne permet pas de modéliser simplement l'incertitude sur les vitesses. En conclusion, la résolution par techniques linéaires ne nous semble pas utilisable pour résoudre des conflits "réels".

3.8 Techniques de parcours de graphe

Les essais faits avec les techniques classiques de parcours de graphe sont restés à un stade préliminaire et mériteraient d'être repris en profondeur. Les travaux avaient été effectués par Hervé Gruber [Gru92].

3.8.1 Algorithme de type A^*

Les algorithmes de type A, dont l'algorithme A^* ([Pea90, Pea84, FG87, Far96], sont classiquement utilisés en Intelligence Artificielle et en Robotique depuis de nombreuses années. Le coût à minimiser est la somme des distances parcourues par les avions pour rejoindre leur destination, et l'heuristique est simplement la somme des distances restant à parcourir, en supposant que les avions peuvent rejoindre directement leur destination. Cette heuristique est clairement minorante, garantissant ainsi l'optimalité de la solution.

La modélisation choisie par Hervé Gruber ne permettait pas de résoudre des conflits à plus de deux avions, que ce soit de façon optimale ou même sous-optimale. Nous avons modifié légèrement cette modélisation : ici, un avion est dévié à partir d'un instant t_0 de 30 degrés à gauche ou à droite pendant un temps t_1, puis regagne directement sa destination.

Il est alors possible de résoudre de façon optimale des conflits à deux avions en un temps très court (de l'ordre de quelques secondes). Cependant, il reste impossible de résoudre des conflits ne serait-ce qu'à trois avions. La raison en est simple. En utilisant l'heuristique présentée ci-dessus, et notre modélisation, l'algorithme A^* tend à reprendre la génération des états après un conflit dès le début de la trajectoire. C'est une mauvaise stratégie de résolution. En effet, les avions ont alors de fortes chances de se retrouver à nouveau en conflit au même point. Il vaut en fait mieux essayer de reprendre la génération des états le plus près possible du point de conflit, car une déviation à ce moment là à de plus fortes chances de résoudre. On peut favoriser ce comportement en multipliant la valeur de la fonction heuristique par un facteur de l'ordre de 1.2 (cela se comprend aisément). On obtient alors des résolutions sous-optimales, mais en un temps beaucoup plus réduit. Par exemple, la résolution d'un conflit à deux avions ne prend plus que quelques centièmes de seconde, et l'on parvient même à résoudre des conflits à cinq avions (en un temps important cependant, de l'ordre de plusieurs minutes). Cependant, le conflit présenté figure 3.48 est simple dans sa structure (avions en face à face). Dans le cas de conflits plus complexes, nous ne sommes pas parvenus à obtenir des résultats dans des temps raisonnables (moins d'une heure).

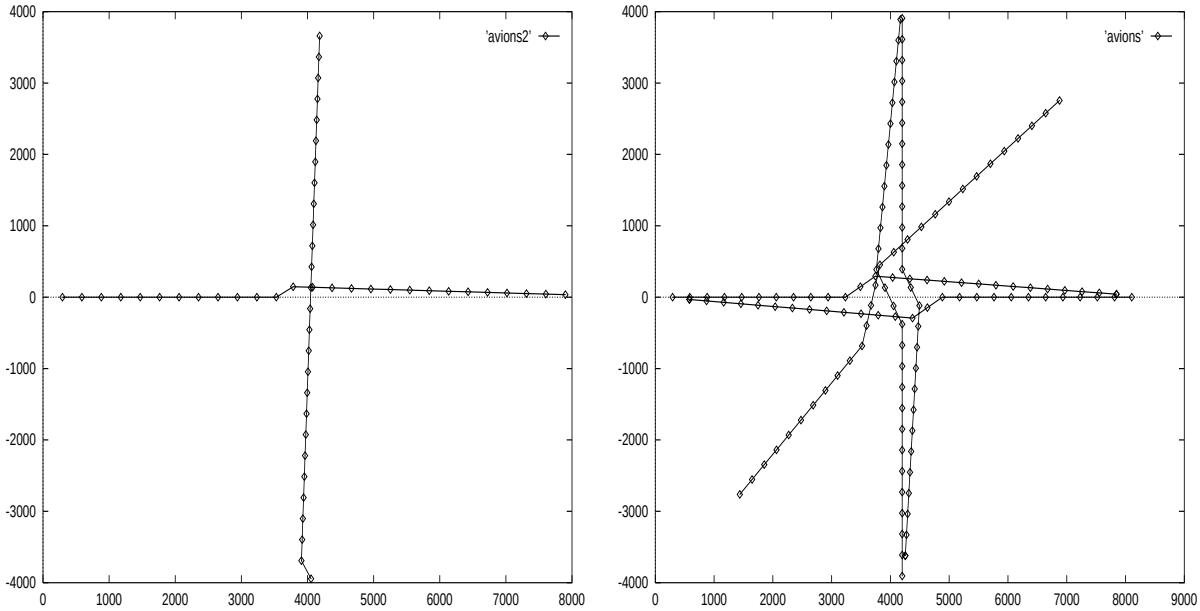


Figure 3.48: Résolution d'un conflit à deux avions de façon optimale et d'un conflit à cinq avions de façon sous optimale

3.8.2 Méthode de plus forte pente

Elle consiste à développer le fils de plus faible coût parmi ceux qui ont été créés à la dernière génération. Au lieu de trier tous les états, on ne trie en fait que les états créés à chaque génération. On s'attend donc à trouver très vite une solution qui n'est pas nécessairement optimale. Les résultats sont en fait très surprenants. Ce comportement s'explique par le fait que la recherche "s'égare" dans des trajectoires inattendues : les deux avions volant parallèlement tout en maintenant la distance de séparation et réduisant la distance au but.

Le résultat obtenu lorsque l'on fixe la durée des pas de calcul sans limiter leur nombre, montre de façon comique, ce à quoi peut aboutir cette méthode trop simpliste (figure 3.49).

3.8.3 Conclusion

Les deux méthodes précédemment évoquées montrent les deux écueils duaux, presque caricaturaux, auxquels on se trouve confronté lorsque l'on travaille sur des algorithmes de parcours de graphe. D'un côté un algorithme garantissant l'optimalité pour la résolution, mais lent et coûteux en mémoire, de l'autre un algorithme extrêmement rapide mais fournissant des solutions inutilisables.

Il est clair qu'un algorithme aussi primitif que la méthode de plus forte pente est inapplicable. En revanche, les résultats connus sur la complexité A^* ne permettent guère d'espérer résoudre des conflits impliquant trop d'avions avec cet algorithme. En effet, le nombre de nœuds de A^* peut-être considéré comme proportionnel au nombre de nœuds du graphe au carré. Or, dans le cas qui nous intéresse, le nombre de nœuds du graphe croît de façon exponentielle avec le nombre d'avions (on peut en fait montrer que dans le cas où notre heuristique est minorante, la complexité du problème croît comme s^{4n} où s est le nombre de pas de temps et n le nombre d'avions).

Nous pensons cependant fermement que cette direction de recherche doit être reprise sérieusement. Il faut en particulier s'intéresser à des algorithmes de "moyen terme" (A_ϵ, développement partiel, etc)

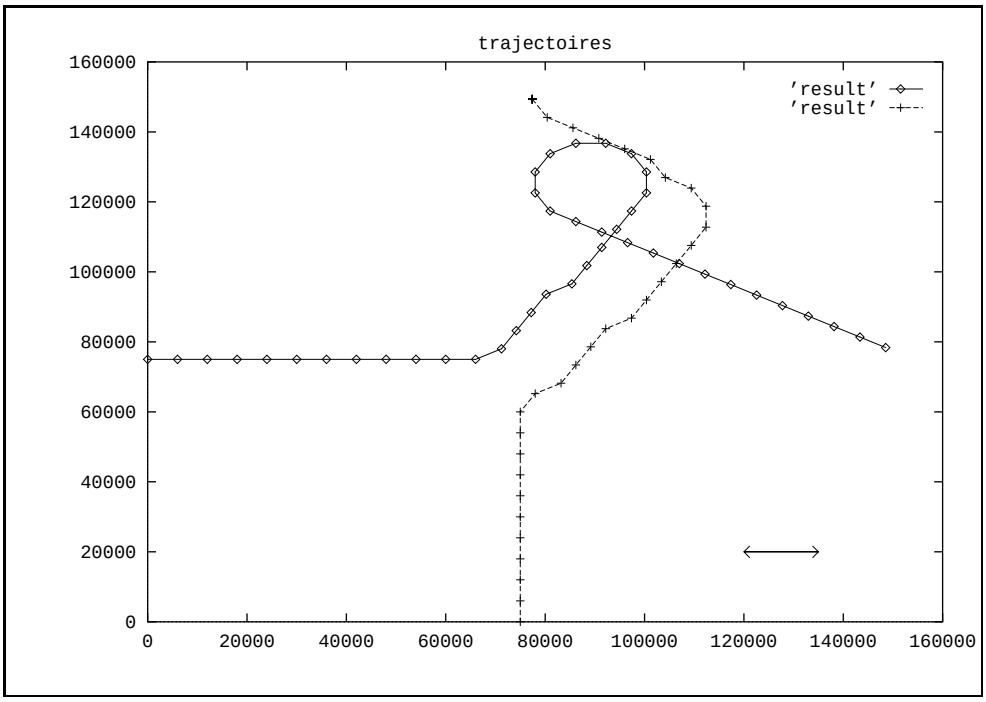


Figure 3.49: Trajectoires fantaisistes obtenues par la méthode du gradient (n=11)

qui nous permettraient de mieux tirer partir de notre connaissance du problème dans le développement des états. Il s'agit là d'une direction de recherche à reprendre.

3.9 Résolution réactive et autonome

3.9.1 Méthodes réactives à modèle physique

Présentation

Les méthodes de résolution décrites dans ce paragraphe sont radicalement différentes de celles évoquées précédemment. Le projet ATLAS est le premier projet à avoir envisagé l'hypothèse d'avions autonomes [DA93] au plan européen. Cette hypothèse a été également étudiée en détail par Karim Zeghal [Zeg93, Zeg94] dans sa thèse. Il introduit la notion de coordination d'actions grâce à différentes forces qui s'exercent sur les agents, dans notre cas, les avions. Il définit ainsi trois types de forces qui devront agir suivant l'urgence :

- Les forces attractives qui permettent aux avions d'atteindre leur objectif (une balise ou leur destination finale par exemple).

- Les forces répulsives qui permettent aux avions d'éviter un obstacle proche donc dangereux. Cet obstacle peut être un avion ou une zone interdite.

- Les forces de glissement qui permettent de contourner les obstacles. Les avions ont alors une action coordonnée (voir figure 3.50). Un force de glissement est définie de la manière suivante :

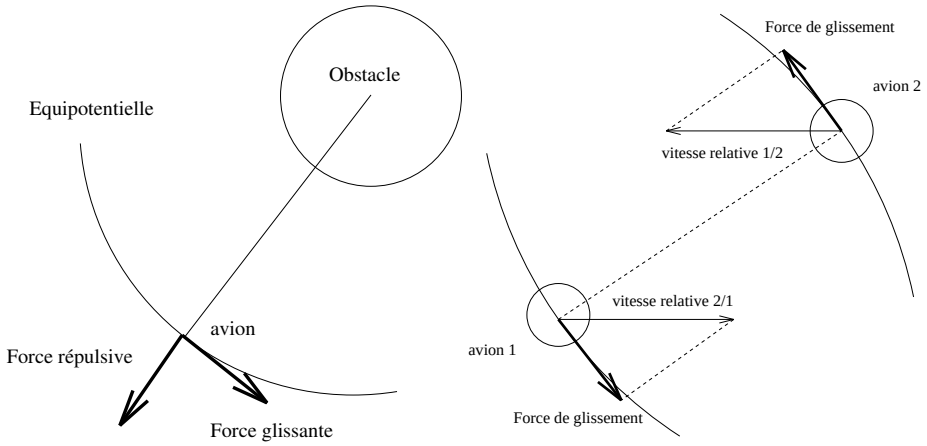


Figure 3.50: Force répulsive et force glissante, forces de glissements coordonnées.

si l'on observe l'équipotentielle de danger passant par l'avion, une force de glissement est tangente à cette équipotentielle alors que la force répulsive est normale à cette équipotentielle. Il y a donc plusieurs forces de glissement possibles. Si l'on reste dans le plan horizontal, la figure 3.50 montre que l'on peut définir deux sens de glissement (à droite ou à gauche). Le sens optimal (il s'agit ici d'optimalité locale, pour l'avion concerné) est celui qui favorise le rapprochement vers l'objectif. Dans le cas d'un obstacle mobile, Karim Zeghal définit une action coordonnée d'évitement. Si deux avions arrivant au même point suivent des forces de glissement complémentaires, alors l'évitement peut se faire de façon très efficace. Dans le cas simple de deux avions (voir figure 3.50), on projette par exemple la vitesse relative sur l'équipotentielle de danger pour chaque avion et on obtient ainsi deux forces de glissements coordonnées.

Il s'agit ensuite, pour limiter les changements brutaux de direction, de gérer l'intensité des différentes forces entre elles. En effet, les avions étant limités par leurs taux de virage, de montée et de descente, ils ne peuvent pas effectuer de manœuvre trop rapide. Les différentes forces s'exerçant sur les avions s'additionnent donc avec des coefficients variables suivant l'imminence du danger.

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

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

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

- Dans la mesure où le processus individuel ne nécessite que des informations locales à l'appareil, il peut fonctionner de façon autonome à partir d'une perception de son environnement. Karim Zeghal propose donc un système d'avion autonome "hybride"[17]. L'intérêt premier de ce système est sa robustesse. La panne d'un avion ne remet pas en cause la sécurité du système. Il apparaît une forme de redondance dans le système qui renforce sa robustesse. Ce système devrait gérer des conflits entre 4 et 10 minutes à l'avenir, ce qui suppose qu'un système supérieur continue à gérer la densité locale de trafic afin d'éviter une saturation.

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

- Karim Zeghal propose également un système d'aide au contrôleur. Il suffit pour cela d'utiliser le processus individuel sous forme de processus centralisé et de simuler les trajectoires futures. Le résultat d'une simulation est alors un ensemble de trajectoires assurant l'évitement sur l'intervalle de temps considéré pour l'ensemble des appareils. Ceci pourrait selon lui avoir un grand intérêt pour planifier les trajectoires et faire des propositions au contrôleur.

  Les résolutions proposées par ces techniques étant des commandes continues, il semble peu probables que celle-ci puissent être mises en pratique dans un cadre où le pilote et les contrôleurs sont maintenus dans le système.

- Enfin on peut imaginer un système double "intermédiaire". Dans ce cas, certains avions sont autonomes et d'autres pas. Le contrôleur a alors une disponibilité supplémentaire et la capacité de son secteur augmente.

La robustesse des solutions est un des atouts des travaux de Karim Zeghal. Il reste néanmoins de nombreux problèmes à résoudre. Dans la mesure où l'on souhaite continuer à confier à l'homme son rôle de pilote dans l'avion, il est indispensable de pouvoir lui transmettre des manœuvres exécutables et donc simples. Par exemple, on peut demander à un pilote de changer de cap pendant un certain temps ou de prolonger une montée ou de modifier une heure de début de descente. On ne peut pas lui demander de modifier en permanence son cap sa vitesse et son taux de montée ou descente. Il serait donc nécessaire de revoir la modélisation des trajectoires pour les simplifier et les rendre accessibles au pilote. De plus, si l'on doit pouvoir prouver que pour deux avions ces techniques peuvent résoudre tous les conflits, la généralisation à n avions (qui consiste à additionner les forces générées par chaque avion) n'a été vérifiée que sur des exemples. Enfin, aucune recherche d'optimalité globale n'est envisagée.

Évaluation

Jean-François Bosc a implanté la méthode précédente en y apportant quelques amélioration. Il a d'autre part imposé un certain nombre de contraintes :

- tous les avions ont le même comportement
- on utilise seulement des routes directes
- on ne modifie pas la vitesse des avions, car on suppose que les marges de maœuvre sur la vitesse sont trop faibles
- on limite le taux de virage a 3 degrés par seconde

Les résultats obtenus sont intéressants à plus d'un titre (voir figure 3.51). Pour des valeurs élevées du volume de trafic et de la séparation horizontale (trafic multiplié par 2,5 et 10 NM de séparation), la résolution finit par générer plus de conflits qu'il n'y en avait initialement. Ceci est dû à la fois à l'accroissement des temps de vol qui devient très important et aux "oscillations" des avions soumis aux force issues d'intrus multiples. Une des limitations des méthodes réactives est qu'on est incapable de tenir compte du fait qu'un intrus en rapprochement pourra passer à distance suffisante sans modification de trajectoire. Une manœuvre est effectuée, ce qui perturbe inutilement la trajectoire et les autres évitements en cours. On a également remarqué qu'en général les variantes de la méthode de résolution qui étaient moins performantes sur un trafic modéré se dégradaient moins nettement lorsque la complexité augmente.

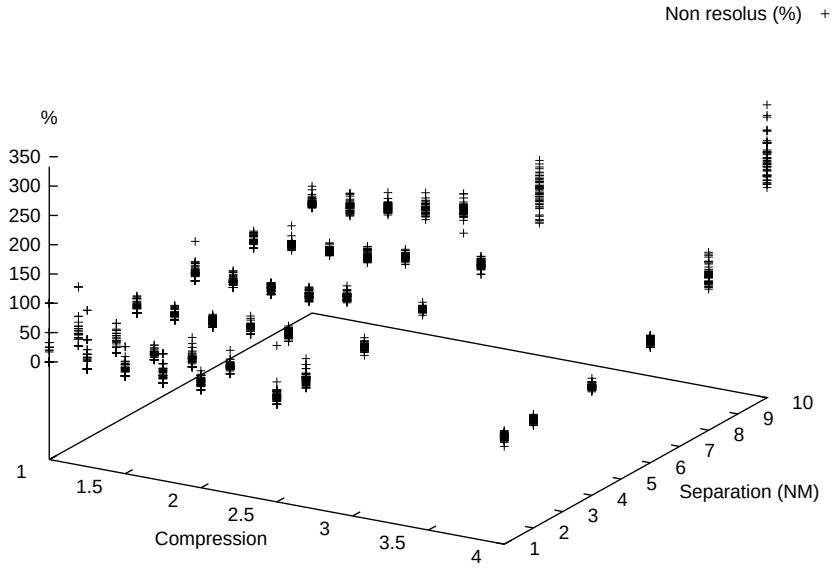


Figure 3.51: Pourcentages de conflits non résolus

Il est donc probable que la saturation observée est une saturation de la méthode de résolution plutôt qu'une saturation de l'espace aérien. Au niveau de complexité d'ARC2000 (trafic multiplié par 2.5, et des séparations horizontales allant de 5 à 10 NM selon la géométrie du conflit), la méthode Zeghal implantée sous cette forme fournit des résultats (en termes de pourcentage de résolution et de délais) nettement moins bons (25% de conflits non résolus et 6.7% de délai moyen avec une séparation (constante) de 6 NM).

3.9.2 Résolution par réseaux de neurones

Principe de résolution

La technique développée ici consiste à utiliser un réseau de neurones pour piloter chaque avion[18]. A chaque instant, le réseau de neurones donne une commande cap à l'avion en fonction des données recueillies. On peut voir sur la figure 3.52 quelle est la structure du réseau, ainsi que ses entrées :

- Le cap α pour rejoindre la destination, et sa valeur absolue $|\alpha|$ (in degrés).
- La distance λ à l'autre avion et son gradient $\frac{d\lambda}{dt}$.
- Le cap γ de l'autre avion
- L'angle β de convergence des trajectoires
- Une classique entrée de biais dont la valeur est 1.

[18]On pourra se reporter à [DAN96a] pour plus de précisions.

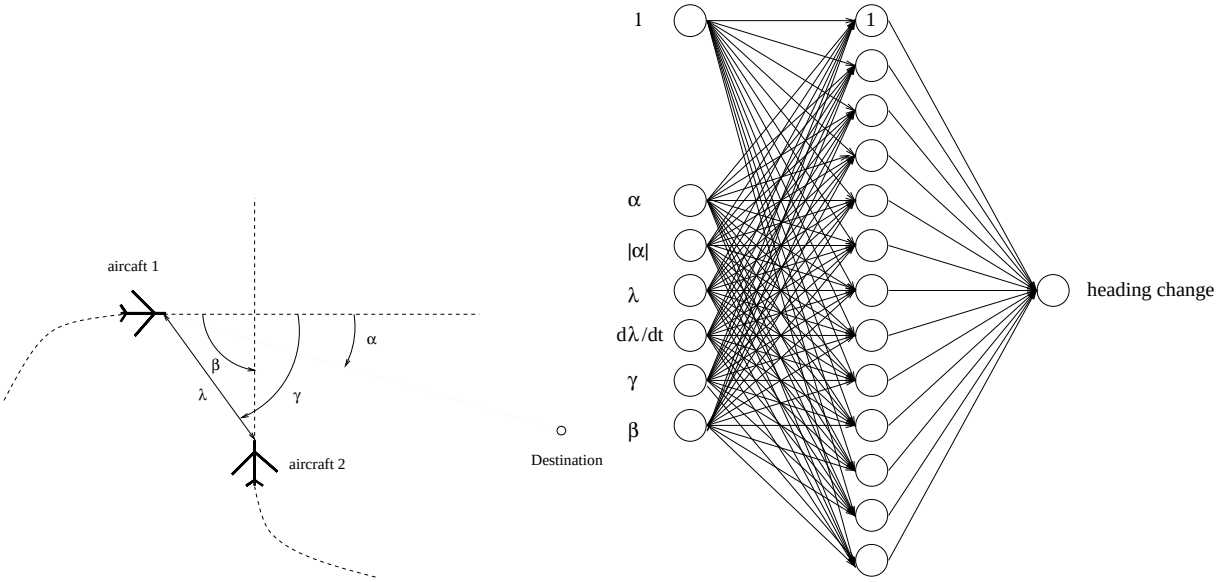


Figure 3.52: Les entrées du réseau de neurones et sa structure

Le réseau quant à lui est simplement un classique réseau à 3 couches. La fonction d'activation est la classique fonction logistique $act(s) = \frac{1}{1+e^{-s}}$

Le problème est qu'il est impossible d'utiliser des techniques d'apprentissage supervisé pour réaliser l'apprentissage du réseau. En effet, il est long et coûteux de construire une résolution optimale de conflits à deux avions, et impossible de le faire pour un nombre d'avions supérieur à deux. Pour construire le réseau de neurones, nous utilisons donc un algorithme génétique (cette technique a déjà été employé, en particulier pour traiter le problème du parcage d'une voiture [SRD93]).

Chaque élément de population de l'algorithme génétique est un réseau de neurones, représenté par la matrice des poids de ses connexions. On utilise un croisement barycentrique et la mutation ajoute un bruit gaussien à un certain nombre de poids du réseau, pris au hasard.

La fitness de chaque réseau est calculée de la façon suivante : on fait voler les deux avions sur un ensemble de configurations considérées comme représentatives. On dit alors que la fitness vaut :

$$F \quad = \quad \frac{1}{D}\, e^{-V}$$

D est la moyenne des délais induits par les déviations de l'avion et V est le nombre moyen de conflits résiduels[19].

La base d'apprentissage

On a utilisé 12 configurations différentes. Dans chacune de ces configurations, les avions sont distants de 20 miles à $t = 0$.

- dans 4 configurations, les avions ont la même vitesse et convergent avec des angles différents (20, 60, 120, 150 degrés, voir figure 3.53).

[19]On retrouve ici la classique méthode consistant à pénaliser la fitness d'un élément violant les contraintes, sans toutefois l'annuler

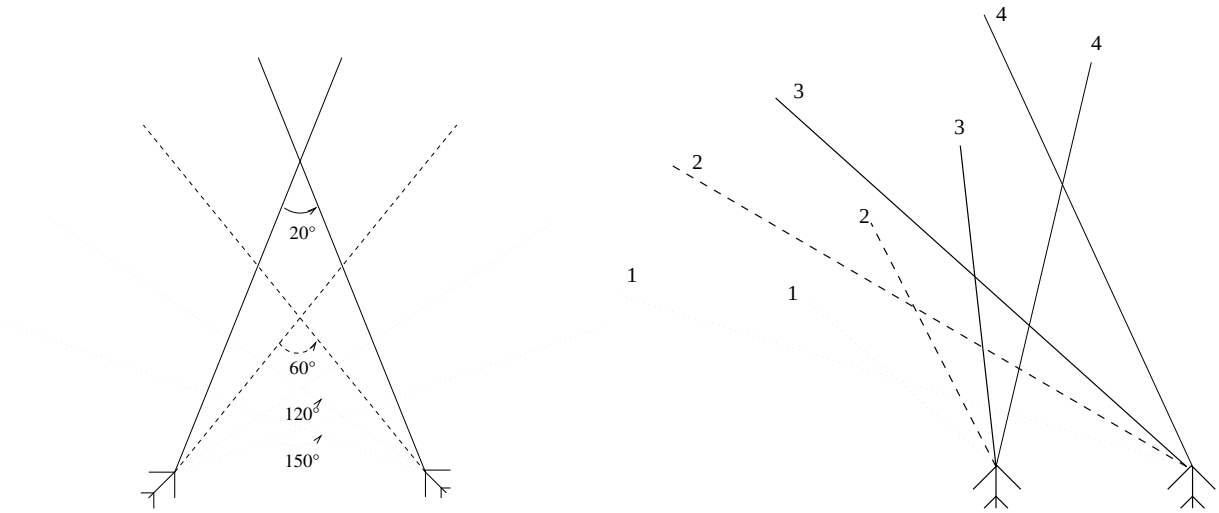


Figure 3.53: 4 configurations à vitesse égale et 4 configurations à vitesse différentes

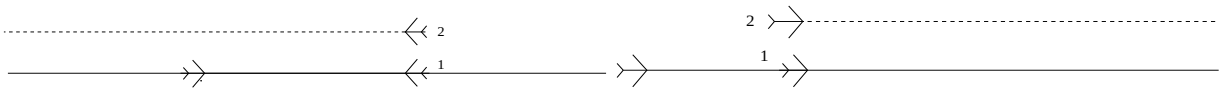


Figure 3.54: 2 configurations en face à face et 2 configurations en rattrapage

- dans 4 configurations, les avions ont des vitesses différentes et leurs caps sont calculés de façon à générer des conflits. (les vitesses sont de 500, 300, 350, 400, et 450 nœuds, voir figure 3.53).
- dans 2 configurations, les avions sont face à face, avec la même vitesse (voir figure 3.54).
- dans 2 configurations, les avions sont en rattrapage (voir figure 3.54).

En raison des symétries, ces 12 configurations sont représentatives de tous les cas possibles. Nous appelons “configuration positive” une configuration dans laquelle l’angle entre l’avion le plus lent et l’avion le plus rapide est positif. Quand on rencontre une configuration négative, on utilise la configuration positive symétrique, en donnant des valeurs négatives aux entrées du réseau.

Résultats numériques

Pour valider les résultats, nous avons testé notre réseau sur des configurations non apprises, afin de vérifier sa capacité à généraliser. Les solutions optimales ont été calculées avec LANCELOT [CGT92] et comparées avec les résolutions fournies par le réseau de neurone. Sur chacune des figures, la résolution du réseau de neurones est à gauche, et la solution optimale trouvée par Lancelot à droite.

- La figure 3.55 montre un exemple de conflit à 90 degrés, les deux avions ayant la même vitesse. Les deux solutions sont identiques.
- La figure 3.55 donne un exemple de conflit à 15 degrés (les avions ont la même vitesse). Il s’agit d’un conflit très difficile à résoudre. Les solutions sont différentes, mais les coûts sont proches et la solution donnée par le réseau de neurones est robuste.
- La figure 3.56 donne un exemple de résolution de face à face. Les solutions sont identiques.

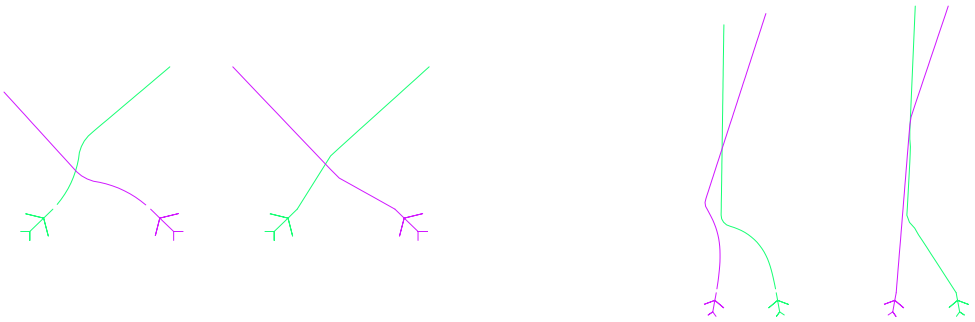


Figure 3.55: Conflit à 90 degrés et à 15 degrés

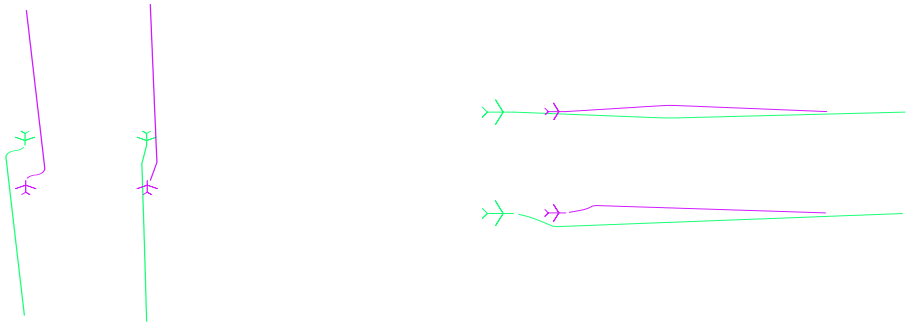


Figure 3.56: Face à face et rattrapage.

- La figure 3.56 présente un exemple de résolution de rattrapage. Les solutions sont identiques.

Conclusion

La résolution réactive de conflits à deux avions par réseaux de neurones donne d'excellents résultats. Mais le problème de cette méthode est l'extension à un nombre plus élevé d'avions. Nous sommes parvenus à la faire fonctionner avec trois avions, mais le nombre d'entrées du réseau augmente et le nombre de configurations d'apprentissage augmente également. L'extension à des cas de quatre ou cinq avions paraît difficile, du moins dans ce cadre.

3.10 Conclusion

Nous nous sommes efforcés de présenter dans ces quelques pages un panorama assez complet des techniques qui peuvent s'appliquer pour résoudre des conflits aériens. Le travail qui reste à effectuer est encore trés important :

- Il faut examiner à nouveau les techniques classiques de recherche de type A. Ces algorithmes devraient se révéler extrèmement utiles lorsque le nombre d'avions n'est pas trop important.
- Il faut reprendre les expérimentations et réaliser une validation sur plusieurs journées de trafic.
- Il faut que nous disposions d'un algorithme susceptible d'affecter des créneaux d'arrivée aux avions dans l'espace que nous contrôlons, de façon à éliminer les conflits aux frontières.
- Enfin, il nous faudrait connecter l'ensemble des filtres dont nous disposons (affectation de créneaux, résolution à 20 minutes et évitement court terme), de façon à faire des estimations de sensibilité de chacun de ces filtres à ceux qui le suivent ou le précèdent dans la chaîne de contrôle.

Nous poursuivons activement ces quatre directions, dans le cadre de DEA ou de thèses.

Chapitre 4

Sectorisation de l'espace et répartition des flux

4.1 Introduction

Le contrôle[1] de la circulation aérienne organise les flux aériens afin d'assurer la sécurité des vols (en terme de risque de collision) et d'améliorer la capacité du réseau de routes sur lequel les avions se déplacent. Suivant la nature du trafic, on distingue les trois types de contrôle suivants :

- contrôle d'aérodrome : gestion des phases de roulage, de décollage et d'atterrissage;
- contrôle d'approche : gestion du trafic en étape préparatoire à l'atterrissage ou post-décollage dans une zone proche d'un aérodrome;
- contrôle en route : il concerne essentiellement le trafic en croisière entre les aérodromes.

Dans le cadre de ce travail, on s'est plus particulièrement intéressé aux problèmes du contrôle en route.

Actuellement on enregistre sur le territoire français environ 4000 mouvements par jour, ce qui représente une charge de contrôle impossible à gérer par un seul contrôleur. On répartit alors cette charge de travail en divisant l'espace aérien en plusieurs secteurs pour chacun desquels on affecte une équipe de contrôleurs. Le nombre de secteurs est alors déterminé par la capacité d'un contrôleur à gérer N avions simultanément (dans la pratique la moyenne semble être de 10 à 15 avions; lorsque cette limite est atteinte on dit que le secteur est saturé). Un des objectifs principaux de la sectorisation est de fournir des secteurs équilibrés en terme de charge de contrôle afin que chaque équipe de contrôleurs travaille de la même façon.

Au cours de ces dernières décennies, au fur et à mesure de l'augmentation du trafic, l'espace aérien a été divisé en secteurs de plus en plus petits afin d'éviter la saturation de ces derniers. Malheureusement, ce principe de resectorisation présente une limite dans la mesure où l'on doit ménager un temps suffisant au contrôleur pour gérer son trafic (élaboration des stratégies de résolution des conflits entre aéronefs) et donc générer des secteurs dont la taille permet de satisfaire cette contrainte. De plus, le contrôleur ne connaît que le trafic lié à son secteur et lorsqu'un avion passe d'un secteur à un autre, il s'opère un dialogue entre les contrôleurs et le pilote afin d'assurer la sécurité du vol lorsqu'il pénètre dans le nouveau secteur (ce dialogue induit une charge de travail supplémentaire pour les contrôleurs

[1] Ce chapitre présente les résultats obtenus par Daniel Delahaye dans le cadre de sa thèse.

appelée *charge de coordination*). Ainsi, en augmentant le nombre des secteurs, on augmente aussi les coordinations et donc la charge de contrôle globale.

Dans le cadre opérationnel, le principe de resectorisation d'un domaine d'espace aérien est le suivant. Lorsque l'on détecte un secteur qui se trouve souvent en limite de saturation, une équipe d'experts se réunit et propose une nouvelle sectorisation de la zone incriminée. Pour valider cette proposition on procède à deux types de simulations : simulation arithmétique et simulation temps réel. La première consiste à évaluer la sectorisation à l'aide de logiciels de simulation sur trafic enregistré (ou simulé). Si les résultats de ces premières évaluations sont satisfaisants, la sectorisation est ensuite testée en temps réel à l'aide d'un environnement de contrôle simulé faisant intervenir des contrôleurs et des pseudo-pilotes (sur poste de pilotage reconstitué en salle (version simplifiée)). Après cette dernière étape, on modifie les frontières des secteurs sur site et on contrôle que ces changements permettent effectivement de résoudre les problèmes initiaux. On gardera cette sectorisation jusqu'à ce que de nouveaux problèmes apparaissent, liés principalement à la croissance du trafic.

Cette approche est essentiellement locale et aboutit à une sectorisation globale qui est la juxtaposition de régions de contrôle optimisées individuellement. Pour mieux comprendre les méthodes utilisées dans le contrôle du trafic aérien, on pourra consulter les références [Mai91, Vil84, CDV92].

Le travail de Daniel Delahaye se décomposait en trois parties principales :

Sectorisation d'un réseau à flux affectés : il s'agit de réaliser une sectorisation équilibrée de l'espace en fonction des flux affectés sur le réseau de routes.

Affectation des flux sur un réseau sectorisé : résolution du problème dual : l'affectation des avions sur les routes aériennes, la sectorisation étant fixée.

Optimisation simultanée de la sectorisation et de l'affectation de trafic : Partant simplement des demandes de trafic entre les paires Origine-Destination, il s'agit de construire simultanément une sectorisation et une affectation des avions sur les routes.

4.2 Modélisation

4.2.1 Routes aériennes

Lorsqu'un avion relie deux villes du continent européen, on pourrait s'attendre à ce que la route directe soit utilisée systématiquement pour des questions d'économie de carburant. En réalité, les avions suivent des routes aériennes constituées d'une succession de tronçons orientés différemment dont les extrémités correspondent à des balises qui matérialisent souvent les croisements de routes. En préparant sa navigation, le pilote jalonne sa route de points de report (balises) sur lesquels il devra faire un passage à la verticale afin de confirmer sa position. Le nombre de balises au sol étant limité, la route réellement suivie s'écartera plus ou moins de la route idéale en fonction de la disposition locale de ces dernières.

Ce principe de navigation par jalonnement sur balises est pénalisant en terme de consommation car il induit des rallongements de route systématiques ; actuellement en Europe, le réseau de routes aériennes provoque des rallongements moyens de 9 %, soit 420000 heures de vol supplémentaires par an ou 1.2 million de tonnes de kérosène. Les avions modernes sont maintenant équipés de calculateur de bord qui, lorsqu'ils sont couplés au pilote automatique, sont capables de suivre avec une grande précision n'importe quelle route définie par un point origine et un point destination, dont les coordonnées sont introduites par l'équipage ou sont extraites d'une base de données du calculateur de bord. Malheureusement, ce système de progression de point à point n'est pas encore possible pour

deux raisons. D'une part, il existe dans beaucoup de pays un assez grand nombre de zones réservées aux militaires qui doivent être contournées par les routes aériennes civiles, d'autre part, le contrôle de la circulation aérienne n'est pas capable d'assurer un écoulement sûr et ordonné du trafic lorsque chacun va directement de son point de départ à son point de destination. L'aiguilleur du ciel a besoin pour travailler de points de report par rapport auxquels il peut situer son trafic.

Une route aérienne pouvant être utilisée dans les deux sens, il a été élaboré une règle de séparation verticale (règle semi-circulaire) imposant des altitudes de vol aux aéronefs afin d'assurer des croisements en toute sécurité. Dans un soucis de simplicité, une route aérienne sera donc simplement modélisé par un arc bidirectionnel joignant deux balises.

Le réseau aérien sera donc modélisé par un réseau de transport qui aura les propriétés suivantes :

- Il est pilotable car il nous est possible d'imposer des routes aux avions par l'intermédiaire des contrôleurs

- Les coûts d'arcs sont non séparables : en effet, le nombre moyen de conflits entre aéronefs à la verticale d'une balise est proportionnel aux flux sur les deux routes aériennes. Il faut bien se rappeler que la résolution d'un conflit induit une charge de contrôle importante et provoque parfois des modifications de navigation pénalisant la consommation. Le coût sur un arc dépend donc des coûts sur les autres arcs, d'où la non-séparabilité.

- Les coûts d'arcs sont asymétriques : en effet, si l'on route des avions vers des secteurs déjà chargés, on peut provoquer la saturation du secteur et donc le refus du contrôleur en charge de ce secteur d'accepter ces nouveaux avions et donc des coûts de déroutement assez élevés quand bien même il n'y a pas beaucoup de trafic (donc de congestion) sur la route initialement choisie.

Les demandes de trafic des usagers sont ensuite distribuées sur le réseau (à l'initiative des utilisateurs ou par l'intermédiaire d'un processus d'affectation de trafic ou les deux). En supposant que les avions se déplacent à des vitesses moyennes quasi-identiques (homogénéité du trafic), la répartition de flux induite fait apparaître une charge de contrôle répartie dans l'espace que nous nous proposons maintenant de modéliser avant de décrire les problèmes qu'il nous faut résoudre.

4.2.2 Charge de contrôle dans un secteur

Introduction

Un secteur de contrôle est un domaine limité de l'espace traversé par des routes aériennes, pour lequel une équipe de contrôleurs assure la sécurité des vols qui y transitent en séparant les aéronefs entre eux. Plus le nombre d'avions dans un secteur est important, plus la charge de contrôle induite augmente (de façon non linéaire). Il existe une limite au delà de laquelle le contrôleur en charge du secteur ne peut plus accepter de nouveaux avions et oblige ces derniers à contourner le secteur en traversant des secteurs voisins moins chargés. On dit alors que le secteur est saturé. Cet état critique doit être évité car il provoque un phénomène cumulatif de surcharge sur les secteurs amonts pouvant remonter jusqu'aux aéroports de départ. En effet, lorsque le trafic ne peut être dévié, il est mis en attente dans les secteurs amonts faisant augmenter progressivement la charge de contrôle de ces derniers jusqu'à ce qu'ils soient saturés. Le seuil au delà duquel le secteur est saturé est très difficile à estimer car il dépend de la géométrie des routes qui le traversent, de la géométrie du secteur lui-même, de la répartition des avions sur les routes, des performances de l'équipe de contrôle etc. Un seuil généralement admis est de 3 conflits et 15 avions dans un secteur donné. Cette charge maximum ne doit pas perdurer plus

de 10 minutes car elle provoque un fort stress des contrôleurs qui risquent alors de ne plus pouvoir assurer la gestion du trafic dans des conditions optimales de sécurité.

Après une enquête auprès des contrôleurs on remarque que la charge de travail dans un secteur dépend des critères qualitatifs et quantitatifs. Les critères qualitatifs regroupent essentiellement les facteurs humains dont le principal est le stress. Tous les contrôleurs ne réagissent pas de la même façon face à une situation de trafic difficile et il est donc délicat de fournir un modèle "mathématique" de stress applicable à tous les contrôleurs. On peut seulement préciser que le stress est directement lié aux critères quantitatifs suivants :

- charge de conflit;
- charge de coordination;
- charge de monitoring.

Il existe d'autres charges de contrôle [TPC76] facilement quantifiables mais leur impact sur l'ensemble de la charge secteur est négligeable par rapport aux trois précédentes.

Nous allons dans un premier temps, préciser et quantifier chacune de ces trois charges de contrôle avant de fournir un modèle mathématique de la charge globale dans un secteur.

Charge de résolution des conflits

On dit que deux avions sont en conflit lorsque la distance qui les sépare risque de devenir inférieure à une valeur particulière appelée norme de séparation. Lorsque deux avions sont en conflit, le contrôleur doit modifier la route des avions afin d'assurer le respect des normes de séparation.

Suivant l'angle de croisement des routes aériennes les conflits sont plus ou moins faciles à résoudre car les avions sont plus ou moins longtemps en situation conflictuelle.

En supposant que la charge de contrôle à la verticale du croisement est proportionnelle au nombre de conflits générés, on a $C_{re} = \alpha(\theta_{ijl}) f_{ij} f_{lj}$ dans le cas d'un croisement à deux routes, où $\alpha(\theta_{ijl})$ est un coefficient de proportionnalité dépendant de l'angle de croisement entre les routes, et f_{ij} et f_{lj} les flux sur les arcs (i, j) et (l, j). Lorsque le croisement comporte plus de deux routes incidentes, la charge de conflit est la somme des charges induites par les arcs pris deux à deux. La charge de conflit dans un secteur S est alors la somme des charges de conflits sur chacun des nœuds contenus dans ce secteur.

Charge de coordination

Tous les avions qui sont dans un même secteur communiquent au moyen de la même fréquence avec le contrôleur en charge du secteur. Lorsqu'ils changent de secteur, ils doivent changer de fréquence et il s'opère alors un transfert de contrôle. Ce transfert doit avoir fait l'objet au préalable d'une négociation entre le contrôleur qui transfère et le contrôleur qui reçoit, pour assurer que celui-ci peut accepter l'avion et pour définir les modalités (niveau de vol, etc) selon lesquelles l'opération a lieu. Un transfert nécessite un travail relativement important de la part des deux contrôleurs; de plus c'est une opération au cours de laquelle des incompréhensions ou des erreurs peuvent se produire causant des pertes accidentelles de séparation. Les charges de contrôle induites par ces transferts sont regroupées dans une charge unique appelée coordination. Dans un réseau de transport sectorisé la charge de coordination est proportionnelle aux flux coupés par les frontières des secteurs. En étudiant la charge de coordination générée par un arc (i, j) de route aérienne dont une partie ou la totalité appartient à un secteur S_k, on peut identifier trois cas de figure :

1. Les deux extrémités de l'arc appartiennent au secteur S_k auquel cas la charge de coordination est nulle.

2. Une seule extrémité de l'arc appartient au secteur S_k Il existe donc une intersection entre l'arc (i, j) et la frontière du secteur S_k. On peut alors représenter la charge de coordination par : $C_{co} = \beta_{ij} f_{ij}$ où β_{ij} est un coefficient de proportionnalité permettant de pondérer l'influence de la coordination par rapport aux autres charges de contrôle.

3. Les deux extrémités de l'arc sont extérieures au secteur S_k ; dans ce cas le flux est coupé deux fois. On peut alors modéliser la charge par : $C_{co} = 2\beta_{ij} f_{ij}$

Charge de monitoring

Dans un secteur de contrôle les avions qui ne sont pas en conflit ou en transfert nécessitent une surveillance de la part du contrôleur qui vérifie le bon déroulement des plans de vol sur l'image radar et qui essaye de déterminer les risques potentiels de conflits futurs induits par ces avions. Le monitoring est en fait la tâche de fond du contrôleur et représente une source importante de stress pour ce dernier. Cette charge de contrôle est directement liée au nombre d'avions présents dans le secteur de contrôle. Pour un secteur S_k on peut la modéliser par : $C_{mo}(S_k) = \eta \sum_{(i,j)\in L_k} p_{ij}(k) f_{ij}$; avec :

- $p_{ij}(k)$ proportion de l'arc (i, j) contenue dans le secteur S_k;
- η : coefficient de proportionnalité.

La charge de contrôle dans un secteur est donc la somme de la charge de conflit, de la charge de coordination et de la charge de monitoring.

Disposant maintenant d'un modèle mathématique du réseau aérien ainsi que de la charge de contrôle (qu'il nous faudra confronter à la réalité), nous pouvons formaliser la description des deux problèmes que nous nous sommes attachés à résoudre.

4.3 Problèmes à résoudre

4.3.1 Problème de sectorisation

A partir de la connaissance de la répartition surfacique (réseau à deux dimensions) de la charge de contrôle, on se propose de trouver une sectorisation équilibrée aboutissant à un rendement homogène des équipes de contrôle en charge de l'ensemble de l'espace aérien. En examinant cette charge de contrôle et en faisant abstraction, dans un premier temps, de la charge de coordination, on remarque qu'elle comporte deux composantes spatiales distinctes :

- une composante discrète localisée au niveau des nœuds (conflits);
- une composante continue répartie sur les arcs (monitoring).

Lorsque l'on met en place des frontières de secteur, ces dernières coupent des arcs de routes aériennes générant a posteriori une nouvelle charge de contrôle (coordination) pouvant remettre en cause l'équilibre obtenu pour les deux critères précédents (conflit et monitoring). Un petit exemple

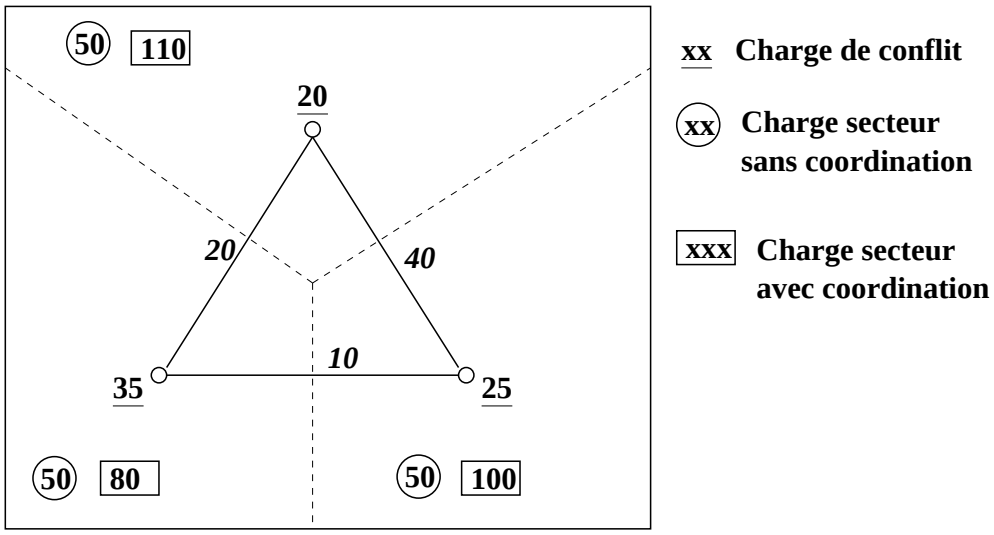


Figure 4.1: Influence de la coordination sur l'équilibrage a posteriori

simple nous permet de mieux comprendre ce problème. Sur la figure 4.1 la sectorisation est initialement équilibrée (à 50) pour les charges de conflit et de monitoring. Si l'on tient compte des coordinations dans l'équilibrage, on constate un déséquilibre net des nouvelles charges de contrôle dans les secteurs. De fait, ce n'est qu'après avoir pris la décision de sectoriser que l'on sait a posteriori si les secteurs sont équilibrés ou non. Une analogie avec la mécanique nous permet de mieux sentir ce problème. Si l'on assimile la charge de contrôle à une masse, notre réseau de transport peut être modélisé par un ensemble de boules denses réparties aux nœuds, reliées entre elles par des barres pesantes. Notre objectif initial est donc de rechercher une sectorisation pour laquelle chaque secteur a la même masse. Le problème est que l'on ajoute des "masses de coordination" lorsque l'on coupe des barres modifiant ainsi l'équilibre calculé. Ce premier objectif d'équilibrage est important mais doit être complété par un objectif connexe de minimisation des coordinations. En effet, si l'on se contente seulement de rechercher l'équilibrage, on risque d'obtenir des aberrations en terme de contrôle dont un exemple extrême est donné figure 4.2. On remarque sans difficulté que le cas 2 est nettement meilleur en terme de contrôle quand bien même il est moins équilibré.

Enfin, la sectorisation construire doit prendre en compte certaines contraintes :

Convexité de routes : Lorsque l'on examine la sectorisation actuelle ainsi que le réseau de routes associé, on remarque que pour chacun des trajets possibles reliant une paire Origine-Destination, les secteurs rencontrés ne sont traversés qu'une seule fois. Ainsi un pilote ne contacte qu'une seule fois le contrôleur en charge du secteur qu'il traverse.

Temps minimal de résolution : Lorsqu'un contrôleur détecte un conflit entre deux avions, il doit se ménager un délai suffisant afin d'élaborer un processus de résolution adapté. Or, comme nous l'avons vu précédemment, les conflits entre aéronefs sont localisés aux croisements des routes aériennes et donc sur les nœuds du réseau de transport. De plus, le trafic géré par un contrôleur est limité au secteur dont il a la charge et il ne dispose pas d'informations sur le trafic présent dans les secteurs voisins. Si au moment où un avion lui est transféré ce dernier se trouve en conflit, le contrôleur n'a pas le temps de construire un processus de résolution. Pour éviter ce problème, il faut éloigner suffisamment les frontières des secteurs des points de croisement afin

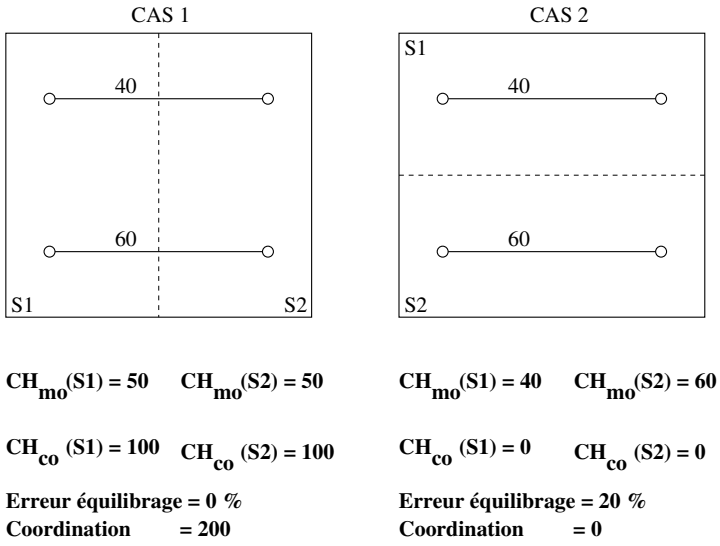


Figure 4.2: Importance de la minimisation des coordinations

de ménager un temps minimum de résolution au contrôleur recevant les avions. Ceci induit une seconde contrainte, dite contrainte de sécurité : la distance entre un nœud et une frontière de secteur sera au moins égale à une distance de sécurité de l'ordre de 50NM.

Temps minimal de séjour : pour qu'un contrôleur puisse agir facilement sur les vols présents dans son secteur, ces derniers doivent y séjourner un temps suffisant égal au temps de sécurité. Un avion devra rester un temps minimum dans chacun des secteurs qu'il traverse.

Le problème de sectorisation peut donc se formuler de la façon suivante : **Soit un réseau de transport dans un espace à deux dimensions sur lequel une répartition de flux produit une charge de contrôle répartie dans l'ensemble de l'espace aérien. On se propose de sectoriser cet espace en K secteurs équilibrés (en terme de charge de contrôle) en minimisant les coordinations. Cette sectorisation devra respecter les contraintes de convexité de route, de sécurité aux balises et de temps de séjour minimum.**

4.3.2 Problème d'affectation

L'affectation de trafic a pour but de définir les routes des flux aériens afin de réduire la charge de contrôle et donc d'augmenter la capacité du système même si cela doit se faire au détriment des performances individuelles. L'amplitude de ces modifications de route doit être suffisamment faible pour que la pénalisation sur les coûts de transport soit comparable au gain apporté sur les performances ATC. Il y a donc un compromis à trouver entre performances ATC et pénalisation des routes, sachant qu'il existe des intérêts communs au système ATC et aux compagnies aériennes. Pour des raisons d'équité, on ne peut faire suivre des routes différentes à des avions joignant la même paire Origine-Destination. En effet, l'avion qui serait routé sur une route plus longue en distance, trouverait ce choix injuste, même s'il permet de diminuer fortement le coût global de transport. Ainsi, si l'on veut que la libre concurrence s'exerce entre les compagnies, sans pour autant laisser toute la liberté à ces dernières pour choisir leurs routes, provoquant des surcharges secteurs anarchiques, il faut nous astreindre à placer les avions sur la même route. Dans la suite de ce rapport cette contrainte sera appelée : *contrainte d'équité*. Notre objectif final étant l'optimisation de la sectorisation (dont les

modifications se font à longue échéance), on ne prendra pas en compte l'aspect dynamique du trafic et nous nous attacherons à travailler sur des flux d'avions macroscopiques. De plus, on se placera dans le cas le plus défavorable consistant à prendre les demandes de trafic maximum entre chaque paire Origine-Destination.

Le problème d'affectation peut donc s'énoncer ainsi : **Soit un réseau de transport dans un espace à deux dimensions, divisé en K secteurs. On désire affecter le trafic entre chaque paire Origine-Destination, en respectant la contrainte d'équité, afin de minimiser la charge de contrôle tout en réduisant les rallongements induits.**

4.4 Complexité associée

4.4.1 Problème de sectorisation

Le problème de sectorisation peut être décomposé en deux sous-problèmes presque indépendants, correspondant aux deux objectifs fixés :

- équilibrage des charges secteur;
- minimisation des coordinations.

Le premier sous-problème est discret-continu NP_complet (discret : équilibrage des charges de conflit et de coordination; continu : équilibrage des charges de monitoring). En effet, en ne considérant que la partie discrète liée aux charges de conflit et de coordination, le problème est équivalent à un K_Partitionnement de graphe en composantes connexes qui est connu pour être NP_complet [Che92]. Le sous-problème de minimisation des coordinations est aussi un K_partitionnement de graphe à minimisation de flux coupés et est donc NP_complet. On peut donc raisonnablement penser que la combinaison de ces deux problèmes est aussi NP_complet.

4.4.2 Problème d'affectation

En considérant, le réseau sans trafic, notre problème consiste à affecter les demandes de chaque paire Origine-Destination sur le chemin qui minimise le critère global C_s. Ainsi pour chaque paire Origine-Destination, on associera un chemin d'affectation de la demande qui produira globalement une répartition de flux sur chacun des arcs du réseau. Les points de notre espace d'état sont donc constitués de l'ensemble des paires Origine-Destination auxquelles on associe la liste des nœuds constituant le chemin d'affectation.

La complexité de notre problème est liée à la non séparabilité des coûts qui rend l'affectation dépendante de l'ordre dans lequel on traite les diverses paires Origine-Destination. Ainsi, s'il n'existe pas d'autre principe de résolution que l'algorithme trivial d'énumération des points de l'espace d'état, on peut supposer que notre problème est NP_complet.

4.5 Sectorisation d'un réseau à flux affectés

4.5.1 Introduction

Nous allons ici résoudre le problème de sectorisation d'espace en secteurs convexes équilibrés[2]. Pour cela on considère un réseau de transport à deux dimensions sur lequel sont affectées des demandes

[2]Ce travail a donné lieu à deux publication[DASF94a, DASF94c]

de trafic (Origine-Destination) entre les divers centroïdes induisant une répartition spatiale des flux. Cette répartition de flux génère une charge de contrôle dans l'espace qu'il nous faut maintenant sectoriser. Chacun des secteurs ainsi créés sera ensuite attribué à une équipe de contrôleurs qui aura la responsabilité du trafic qui y transitera.

Lorsqu'un contrôleur arrive sur une nouvelle position de contrôle, il lui faut entre trois et quatre mois d'apprentissage pour acquérir les réflexes de gestion du trafic associés à ce nouveau secteur. Ce long délai de formation impose une certaine stabilité de la sectorisation dans un centre de contrôle quand bien même une sectorisation dynamique serait parfois plus adaptée. En effet, pour chaque répartition de trafic sur le réseau, il existe une sectorisation optimale. Mais celle-ci serait inefficace si elle est changée toutes les heures car aucune équipe de contrôleurs ne peut la gérer. La sectorisation résulte donc d'un compromis. Elle est calculée pour accepter la charge de contrôle maximale aux heures de pointe mais elle devient sous-optimale pendant les heures creuses. Les flux sur le réseau qui sont considérés pour la sectorisation sont des flux moyennés correspondant aux périodes de fort trafic.

4.5.2 Modélisation

Charge de contrôle

Comme nous l'avons vu au chapitre 2 la charge de contrôle est la somme de trois termes :

- la charge de conflit $C_{cf}(S_k)$;
- la charge de coordination $C_{co}(S_k)$;
- la charge de monitoring $C_{mo}(S_k)$;

$$\Rightarrow C(S_k) = C_{cf}(S_k) + C_{co}(S_k) + C_{mo}(S_k).$$

Si l'on considère la charge globale de contrôle C_{gl}, la répartition optimale de charge dans une sectorisation à K secteurs est donnée par :

$$C(S_k)_{Opt} = \frac{C_{gl}}{K}.$$

A partir de cet optimal, on peut élaborer un critère d'équilibrage relatif C_E :

$$C_E = \sum_{k=1}^{K} \frac{|C(S_k) - \frac{C_{gl}}{K}|}{\frac{C_{gl}}{K}} \quad 0 \leq C_E \leq (2K-1)$$

Si l'on veut se rapprocher d'un équilibre sur chacune des charges de contrôle, il faut appliquer cette formule aux conflits, à la coordination ainsi qu'au monitoring et le critère d'équilibrage relatif devient :

$$C_E = \sum_{k=1}^{K} \frac{|C_{(cf)(S_k)} - \frac{C_{gl(cf)}}{K}|}{\frac{C_{gl(cf)}}{K}} + \sum_{k=1}^{K} \frac{|C_{(co)(S_k)} - \frac{C_{gl(co)}}{K}|}{\frac{C_{gl(co)}}{K}} + \sum_{k=1}^{K} \frac{|C_{(mo)(S_k)} - \frac{C_{gl(mo)}}{K}|}{\frac{C_{gl(mo)}}{K}}$$

Avec cette deuxième forme, on conçoit très bien qu'il sera plus difficile et même parfois impossible d'annuler le critère car certains réseaux ne peuvent pas être équilibrés pour les trois types de charges.

La minimisation de la coordination sera établie en réduisant la part relative de coordination par rapport aux charges globales de conflit et de monitoring :

$$C_C = \frac{\sum_{k=1}^{K} C_{co}(S_k)}{C_{gl(cf)} + C_{gl(mo)}}.$$

On voit bien ici que notre problème est un problème multi-objectifs mais dans un premier temps et dans un souci de simplification nous nous ramenons à un problème mono-objectif en élaborant un critère global C_G qui est une pondération des deux précédents : $C_G = \alpha C_E + (1-\alpha)C_C$; avec $0 \leq \alpha \leq 1$.

Principe de construction des secteurs

Il nous faut tout d'abord une méthode permettant de générer des populations d'individus dans l'espace d'état. Pour atteindre cet objectif, on utilise le principe d'agrégation de Forgy [Sap90] utilisé dans les nuées dynamiques en statistique exploratoire et qui permet d'extraire des clusters d'un ensemble de points répartis dans un espace muni d'une distance.

Une méthode simple pour obtenir un K-partitionnement d'un domaine D consiste à "jeter" aléatoirement dans le domaine D un ensemble de K points (appelés centres de classe) et de regrouper chacun des points contenus dans D à son centre de classe le plus proche (principe d'agrégation). Pour éviter d'avoir des secteurs vides, il faut interdire à deux centres de classes d'occuper la même position dans D. Il est facile de prouver que les secteurs ainsi générés sont géométriquement convexes. Cette propriété de convexité géométrique est plus forte que la convexité de route et permet donc de respecter la contrainte associée (pour un arc de route aérienne)[3]. On peut également aisément vérifier qu'à chaque ensemble de centres de classe C correspond une sectorisation unique en K secteurs convexes du domaine D (la réciproque est fausse).

Ce principe de synthèse de sectorisation est simple à mettre en œuvre car il nécessite peu d'informations (les positions géographiques des centres de classe) pour définir complètement l'ensemble des secteurs et satisfait la contrainte de convexité.

Prise en compte des autres contraintes

Les contraintes de sécurité et de temps de séjour minimum sont prises en compte à l'aide d'un coefficient de pénalité qui majore artificiellement la coordination sur les arcs qui violent les contraintes. La nouvelle charge de coordination sera exprimée de la façon suivante :

$$C_{co}(S_k) = \sum_{\substack{i \in N_k \\ \oplus j \in N_k}} \delta\beta_{ij} f_{ij} + \sum_{\substack{i \notin N_k \\ j \notin N_k \\ (i,j) \in L_k}} 2\delta\beta_{ij} f_{ij}$$

$\oplus$: ou exclusif ; le coefficient de pénalité δ modélisant le dépassement des contraintes est calculé de la façon suivante ; Soit p_{ij}^k la longueur relative du segment de l'arc (i,j) contenu dans le secteur S_k et L_{ij} la longueur totale de l'arc (i,j). Si la portion d'arc contenu dans le secteur S_k est inférieure à la distance de sécurité, il faut majorer la coordination associée :

[3]Il faut préciser que ce type de convexité (géométrique) restreint notre espace d'état en nous interdisant de générer des secteurs polygonaux non convexes.

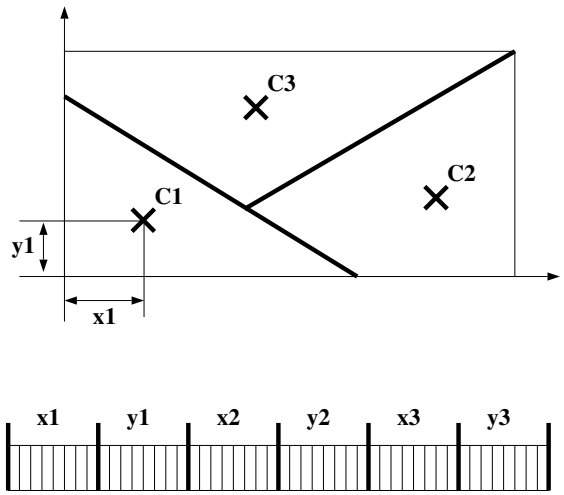


Figure 4.3: Codage binaire

$$\begin{cases} \text{Si } (p_{ij}^k.L_{ij}) < d_{secu} \text{ alors } \delta = e^{-[2(\frac{p_{ij}^k.L_{ij}}{d_{secu}}-1)]}; \\ \text{Sinon } \delta = 1 \end{cases}$$

On traite ainsi de la même façon les deux contraintes.

A partir de cette formalisation, il nous est possible de développer un principe d'optimisation utilisant les algorithmes génétiques.

4.5.3 Optimisation du problème par Algorithmes Génétiques

Introduction

Dans un premier temps, les algorithmes binaires ont été utilisés pour résoudre ce problème. Le codage du chromosome associé à ce type d'algorithme est constitué de la concaténation du codage binaire de la position de chacun des centres de classe définissant une sectorisation (voir figure 4.3).

Ce codage permet ensuite d'utiliser les opérateurs de croisement et de mutation standard. Après évaluation, on remarque que ce type d'algorithme génétique se comporte correctement pour des petits réseaux ($<$ 20 nœuds) mais lorsque la taille du réseau augmente, il se produit un phénomène de marche aléatoire, lié principalement à la brutalité des opérateurs de croisement et de mutation qui font abstraction de la notion de centres de classe mais travaillent seulement au niveau des bits. Il y a perte de la structure du problème et ceci nous a conduit à changer le codage de l'algorithme en utilisant des algorithmes génétiques opérant sur des données plus complexes. Nous présentons maintenant ce nouveau codage ainsi que les opérateurs associés.

Codage du chromosome

Le chromosome doit contenir l'ensemble des informations nécessaires à l'évaluation de la fitness. Comme nous l'avons dit précédemment la connaissance de la position des centres de classe dans l'espace géographique est suffisante pour définir la sectorisation et donc évaluer ses performances. Ces informations sont codées dans le chromosome à l'aide d'une matrice (de dimensions K x 2) contenant

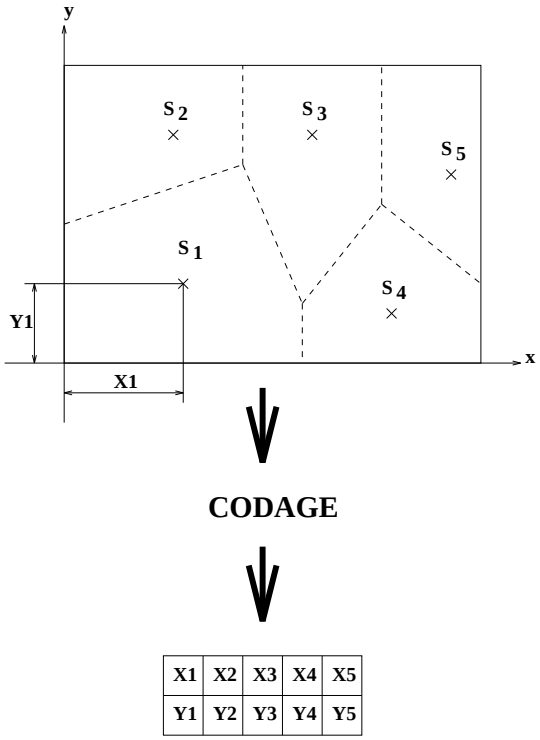


Figure 4.4: Codage du chromosome

les abscisses et les ordonnées des centres de classe. Un exemple de codage d'un chromosome est donné en figure 4.4.

Comme il n'y a pas bijection entre le codage du chromosome et la sectorisation associée, une même sectorisation peut être générée par plusieurs ensembles de centres de classe différents. Toutefois lorsque le nombre de secteurs est important, ce qui est notre cas, il est quasiment impossible qu'un tel événement se produise au sein d'une population. En excluant ce cas de similitude très rare, où deux ensembles de centres de classe génèrent la même sectorisation, il faut remarquer qu'un même ensemble de centres de classe peut être codé par une multitude de chromosomes différents. En effet, toute permutation des centres de classe effectuée à l'intérieur d'un même chromosome, génère le même ensemble, donc la même sectorisation. Ainsi, le nombre de chromosomes synthétisant la même sectorisation est égale à $K!$. Pour une sectorisation à trois secteurs il y aura donc $3! = 6$ chromosomes possibles ayant les mêmes centres de classe :

$$(C_1, C_2, C_3); (C_1, C_3, C_2); (C_2, C_1, C_3); (C_2, C_3, C_1); (C_3, C_1, C_2); (C_3, C_2, C_1).$$

Ce dernier point aura des répercussions dans le calcul de la distance inter-chromosomes pour le sharing d'état.

Principe de génération de la population initiale

Le processus de génération de la population initiale est immédiat. En effet, pour chaque individu contenu dans la population, on tire aléatoirement et de manière uniforme K couples de coordonnées géographiques représentant les K centres de classe définissant chaque sectorisation.

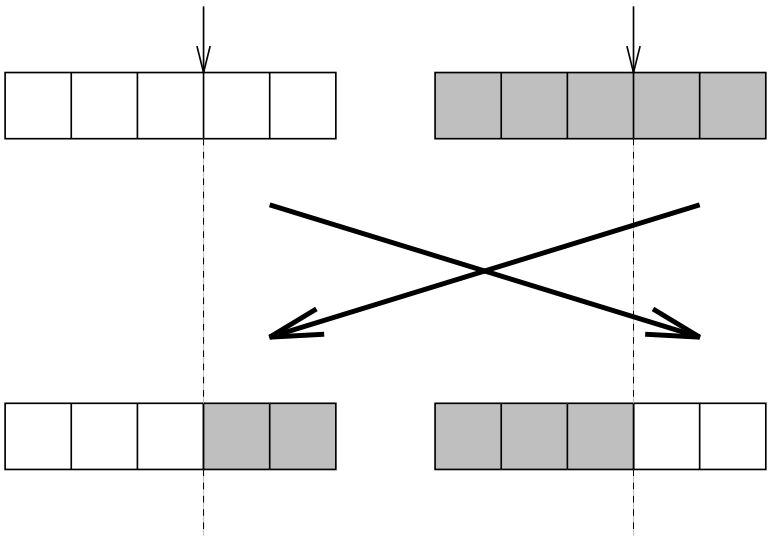


Figure 4.5: Slicing crossover

Opérateur de croisement

Pour ce type de problème nous avons testé deux types de croisement :

- “slicing crossover”;
- croisement barycentrique.

Le premier correspond au croisement classique utilisé dans les algorithmes génétiques binaires mais implanté ici avec des chaînes de flottants. Ce croisement consiste à tirer aléatoirement une position dans le chromosome et à échanger les deux sous-chaînes ainsi produites (voir figure 4.5).

Le second consiste à tirer aléatoirement un centre de classe dans chacun des chromosomes puis à joindre artificiellement ces derniers par une ligne droite ; ensuite on déplace aléatoirement (distribution uniforme) les centres sur cette ligne afin de générer deux nouveaux individus. La position des nouveaux centres de classe est alors donnée par :

$$C_1 = \alpha P_1 + (1-\alpha) P_2$$
$$C_2 = (1-\alpha) P_1 + \alpha P_2$$

(voir figure 4.6 (dans cet exemple le gène 2 a été sélectionnée)). Ce dernier type de croisement est plus adapté pour les espaces continus et fournit de meilleurs résultats sur notre problème qui comporte une composante continue.

Opérateur de mutation

Pour muter un chromosome, on tire aléatoirement une position dans le chromosome puis on déplace le centre de classe associé en ajoutant du bruit à chacune de ses coordonnées suivant une loi de distribution particulière. Un exemple de mutation est donné en figure 4.6.

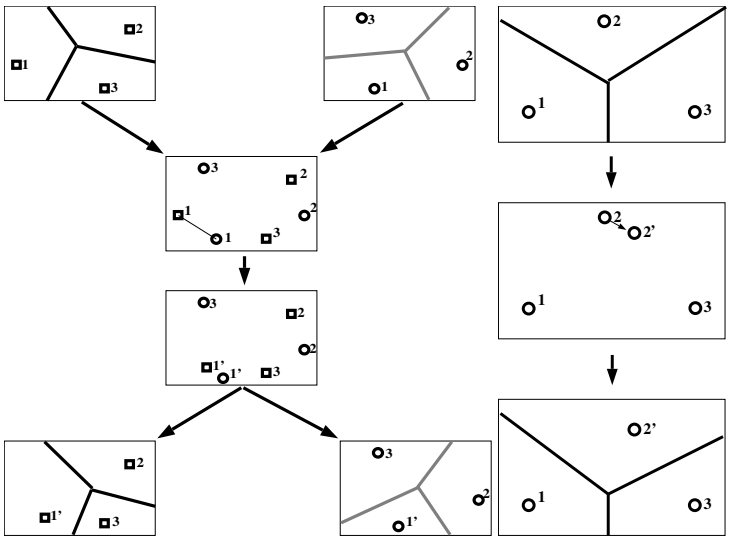


Figure 4.6: Croisement et mutation

Introduction de la distance inter-chromosomes pour le sharing

Pour faire du sharing, il faut être capable d'estimer la distance entre deux chromosomes dans l'espace d'état afin de déterminer le niveau d'agrégation des individus. Les chromosomes étant des concaténations de vecteurs en dimension deux, la méthode la plus simple pour calculer cette distance consiste à faire la somme des distances euclidiennes inter-centre de classe pris deux à deux dans l'ordre où ils apparaissent dans le chromosome. Cette méthode conduit à des erreurs importantes car comme nous l'avons dit dans la section précédente, un même ensemble de centres de classe peut être codé de multiples façons, induisant des valeurs de distance non nulles pour deux codages différents d'une même sectorisation, ce qui est contraire à l'objectif recherché dans le sharing.

Exemple :

$$\text{soient deux individus} \quad \begin{array}{l} I_1 = \{(10,10);(20,20);(30,30)\} \\ I_2 = \{(30,30),(20,20),(10,10)\} \end{array}$$

$$\text{ici } d(I_1, I_2) = \sqrt{20^2+20^2} + \sqrt{20^2+20^2} = 57$$

Pour éviter ce problème, il faut calculer la somme des distances des centres de classe les plus proches contenus dans chacun des chromosomes. Ainsi, on commence par rechercher dans I_2 le centre le plus proche d'un centre de I_1. On calcule la distance associée puis on élimine ces deux centres de I_1 et I_2 respectivement en itérant ce processus sur les centres restants.

Calcul du barycentre entre deux chromosomes

Le chromosome étant défini dans un espace vectoriel, le calcul du barycentre entre deux individus est direct. En effet, soient deux individus I_1 et I_2 auxquels on associe les coefficients barycentriques n_1 et n_2. Pour chaque centre de classe de I_1 on recherche dans I_2 le centre le plus proche. Soient $\vec{C}_1$ et $\vec{C}_2$ les positions géographiques de ces centres, alors le centre barycentrique est donné par :

$$\vec{C}_B = \frac{n_1\vec{C}_1 + n_2\vec{C}_2}{n_1+n_2};$$

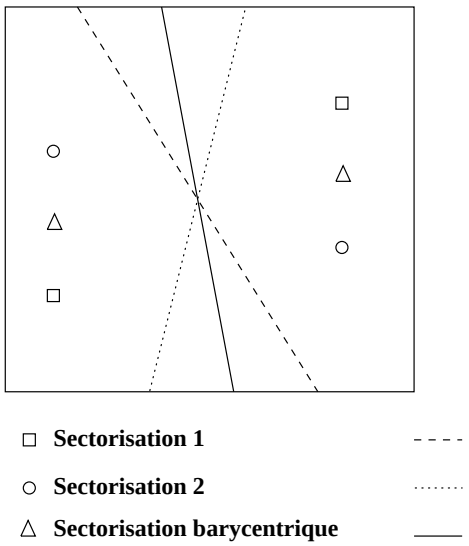


Figure 4.7: Exemple de sectorisation barycentrique

On élimine ensuite ces deux centres des individus I_1 et I_2 avant d'appliquer le même processus sur les centres restants. Un exemple graphique d'une sectorisation barycentrique à coefficients unitaires ($n_1 = 1$ et $n_2 = 1$) est donné sur la figure 4.7.

Pour chaque individu il nous faut maintenant déterminer son adaptation qui sera utilisée dans le processus de sélection des algorithmes génétiques.

Calcul et normalisation de la fitness

Pour chacun des secteurs synthétisés par le chromosome, on évalue les charges de conflit, de coordination et de monitoring associées. La charge de conflit aux nœuds étant indépendante de la sectorisation, elle est calculée une seule fois au début du processus de résolution ; seule la classification des nœuds dans les différents secteurs sera effectuée à chaque évaluation de fitness. Pour les deux autres charges, il nous faut déterminer pour chaque arc :

- la répartition sectorielle du flux sur l'arc (c'est-à-dire la répartition du flux dans les différents secteurs) ;
- le nombre de frontières de secteur qui coupent cet arc.

Ces deux quantités sont recherchées par dichotomie pour chacun des arcs constituant le réseau de transport. On considère pour cela les positions géographiques des nœuds extrémités de l'arc (segment $[N_o, N_d]$) que l'on traite, ainsi que la position des centres de classe permettant de synthétiser la sectorisation (voir figure 4.8).

On effectue ensuite une recherche dichotomique en parcourant le segment $[N_o, N_d]$, afin de déterminer les centres de classe les plus proches de chacun des points de l'arc et donc la répartition sectorielle associée (on ne calcule pas l'équation des droites constituant les frontières des secteurs). A partir des charges de conflit aux nœuds et de la répartition sectorielle de flux, on déduit les trois charges associées à chacun des secteurs et ceci nous permet ensuite de déterminer les critères d'équilibrage et de coordination utilisés dans l'évaluation finale de la fitness.

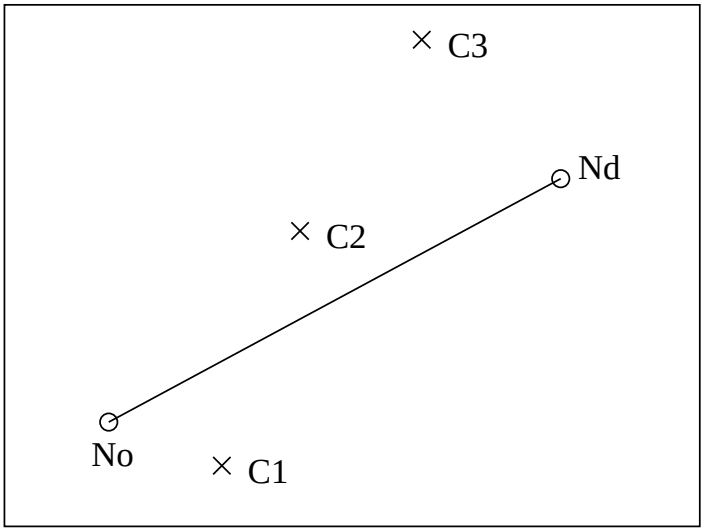


Figure 4.8: Répartition sectorielle d'un arc

Mixage de l'AG avec une méthode déterministe

Le principe d'évaluation décrit précédemment correspond à la version de base de l'algorithme dont nous donnons maintenant une amélioration, basée sur le mixage de l'AG avec une technique déterministe d'équilibrage.

Connaissant les charges de contrôle associées à chacun des secteurs, il nous est facile de déterminer les secteurs les plus chargés et d'en déduire une tendance de déplacement des centres de classe permettant de diminuer l'écart de masse entre les secteurs. En effet, si l'on considère une répartition surfacique homogène de masse dans un rectangle et en examinant la situation initiale décrite sur partie gauche de la figure 4.9, on conclut facilement qu'il faut déplacer les centres à droite pour se rapprocher de l'équilibre. Ce principe d'équilibrage a été implanté au niveau de l'évaluation des fitness de l'algorithme génétique. En effet, c'est à ce niveau que l'on calcule les diverses charges de contrôle des secteurs et que l'on peut effectivement déduire le déséquilibre associé. A partir de la connaissance des écarts de "masse" entre les secteurs, on déduit les directions de déplacement des centres de classe qui permettent de nous rapprocher de l'équilibre.

Pour comprendre le principe de rééquilibrage déterministe utilisé, nous allons prendre le cas simple d'une sectorisation à deux secteurs sachant que nous avons étendu cette méthode aux sectorisations à "K" secteurs. Nous considérons alors une densité surfacique de masse $\rho(x, y)$ répartie sur un rectangle de dimension $X.Y$ (voir figure 4.9). Ce rectangle est ensuite sectorisé à l'aide de centres de classe jetés aléatoirement dans cet espace. Pour simplifier la démarche (sans perte de généralité) nous supposons que les centres de classe se trouvent sur une droite horizontale parallèle à la médiane horizontale "d". Les masses associées à ces deux secteurs sont alors données par :

$$m_1 = \int_0^Y \int_0^{X_M} \rho(x,y)dxdy;\ m_2 = \int_0^Y \int_{X_M}^X \rho(x,y)dxdy;$$

où M est le point médian du segment $[C_1, C_2]$:

$$X_M = \frac{x_1 + x_2}{2};\ Y_M = \frac{Y}{2}.$$

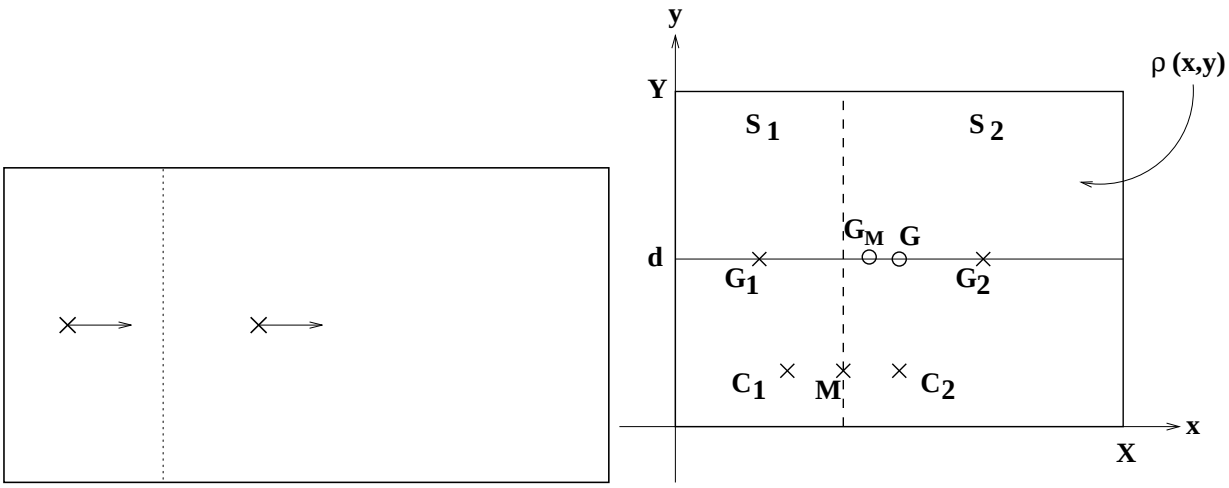


Figure 4.9: Équilibrage et densité surfacique de masse

A l'aide de ces masses, on peut calculer les positions des centres de gravité associés à chacun des secteurs S_1, S_2 ainsi que le centre de gravité total : $\vec{G_1}, \vec{G_2}, \vec{G}$.

On appelle centre géométrique :

$$\vec{G}_M = \left(\frac{\vec{G_1}+\vec{G_2}}{2}\right)$$

Le fonctionnement de l'algorithme est alors basé sur la remarque suivante :

Les deux secteurs sont équilibrés lorsque le centre de gravité est confondu avec le centre géométrique[4].

On cherchera donc à induire des déplacements des centres de classes tendant à rapprocher ces deux points. Plus précisément il faut déplacer le centre géométrique vers le centre de gravité. La difficulté vient du fait que l'on ne peut pas changer l'un sans modifier l'autre.

Nous allons alors envisager deux approches :

1. Approche analytique :

   En revenant à notre exemple et en considérant $\rho(x, y)$ constant ($= \rho_0 = 1$), on a :

   $$\begin{cases} m_1 = Y G_{Mx} \\ m_2 = Y(X - G_{Mx}) \end{cases}$$

   et

   $$G_x = \frac{m_1 G_{1x} + m_2 G_{2x}}{m_1 + m_2} = \frac{G_{Mx} G_{1x} + (X - G_{Mx}) G_{2x}}{X}$$

   On se propose de trouver le déplacement Δ à ajouter à x_1 et à x_2 qui permettent d'annuler l'écart entre $\vec{G}_M$ et $\vec{G}$. On pose :

   $$\begin{cases} x'_1 = x_1 + \Delta; \\ x'_2 = x_2 + \Delta \end{cases}$$

[4]Cet objectif est bien différent de celui proposé dans les méthodes de partitionnement (voir chapitre 3) pour lesquelles on recherche une minimisation de l'inertie intra-classe.

Après déplacement on a :

$$G'_x = \frac{G'_{Mx}G'_{1x} + (X - G'_{Mx})G'_{2x}}{X};$$

On montre sans difficulté que l'écart $\vec{G'} - \vec{G'}_M$ s'annule pour :

$$\Delta = \frac{X}{2} - G_{Mx}.$$

2. Approche itérative : Soit la série $U^n = |m_1^n - m_2^n|$ où n représente le numéro de l'itération courante. Soit :

$$\vec{\Delta}^n = \vec{G}^n - \vec{G}_M^n = \frac{m_1^n - m_2^n}{2(m_1^n + m_2^n)}\left[\vec{G}_1^n - \vec{G}_2^n\right]$$

Posons :

$$\begin{cases} x_1^{n+1} = x_1^n + \frac{\Delta_x^n}{\alpha} \\ x_2^{n+1} = x_2^n + \frac{\Delta_x^n}{\alpha} \end{cases}$$

avec

$$\Delta_x^n = \frac{X - (x_1^n + x_2^n)}{4}$$

On a :

$$m_1^n = Y\left(\frac{x_1^n + x_2^n}{2}\right) \qquad m_2^n = Y\left[X - \left(\frac{x_1^n + x_2^n}{2}\right)\right]$$

$$m_1^{n+1} = m_1^n + Y\frac{\Delta_x^n}{\alpha} \qquad m_2^{n+1} = m_2^n - Y\frac{\Delta_x^n}{\alpha}$$

or

$$\Delta_x^n = \frac{X - (x_1^n + x_2^n)}{4} = \left(\frac{X - \frac{2m_1^n}{Y}}{4}\right) = \left(\frac{X - 2\left[X - \frac{m_2^n}{Y}\right]}{4}\right)$$

$$m_1^{n+1} = m_1^n + \frac{Y}{\alpha}\left[\frac{X - \frac{2m_1^n}{Y}}{4}\right] \quad m_2^{n+1} = m_2^n + \frac{Y}{\alpha}\left[\frac{X - \frac{2m_2^n}{Y}}{4}\right]$$

d'où

$$m_1^{n+1} = m_1^n + \frac{1}{\alpha}\left[\frac{XY - 2m_1^n}{4}\right] \quad m_2^{n+1} = m_2^n + \frac{1}{\alpha}\left[\frac{XY - 2m_2^n}{4}\right]$$

soit

$$m_1^{n+1} = m_1^n + \frac{1}{\alpha}\left[\frac{m_2 - m_1^n}{4}\right] \qquad m_2^{n+1} = m_2^n + \frac{1}{\alpha}\left[\frac{m_1 - m_2^n}{4}\right]$$

Etude de U^n

A partir des expressions précédentes on déduit :

$$\begin{aligned} U^{n+1} &= U^n - \tfrac{1}{2\alpha} U^n \\ \Rightarrow U^{n+1} &= \left(1 - \tfrac{1}{2\alpha}\right) U^n = \beta U^n \\ \Rightarrow U^n &= \beta^n U^0 \end{aligned}$$

On obtient ainsi une série géométrique dont la convergence est assurée pour $\beta < 1$ ($\Rightarrow \frac{1}{4} < \alpha < \infty$).

Dans le cas général où $\rho(x, y)$ est quelconque, on induit des déplacements :

$$\vec{\Delta_n} = \alpha_n(\vec{G}_n - \vec{G}_M).$$

pour lesquels la valeur du coefficient α_n dépend du problème et sera ajustée lors des expérimentations (en règle générale on prend $0 < \alpha < \frac{1}{2}$).

Si on augmente le nombre de secteurs, on applique ce principe à chaque paire de secteurs possibles puis on effectue la somme vectorielle des écarts ainsi obtenus en s'assurant que les centres de classe restent dans le domaine géographique (si tel n'est pas le cas on fait des troncatures).

Par rapport à notre problème, ce processus d'équilibrage ne peut être mené à terme pour deux raisons essentielles ;

- la répartition surfacique de masse est homogène et non continue en certains points faussant ainsi les calculs de déplacement ;
- quand bien même il serait exact, ce processus ne permet pas de satisfaire le critère de minimisation des coordinations.

La difficulté consiste alors à déterminer le nombre d'itérations d'équilibrage. Après expérimentation, et pour deux à cinq itérations, on s'aperçoit que l'algorithme se confine dans des zones d'espace d'état desquelles il ne peut sortir facilement par mutation car le processus d'équilibrage l'y ramène systématiquement. A l'inverse, si l'on ne fait aucun équilibrage l'algorithme est plus lent à converger. D'après les évaluations, il semble qu'un seul équilibrage soit nécessaire. Ainsi après chaque calcul des charges de secteur, on effectue une itération d'équilibrage qui modifie le chromosome avant d'évaluer le critère.

4.5.4 Évaluation

Pour évaluer et comparer les différentes versions d'algorithmes, on utilise un réseau de test pour lequel une solution évidente est connue (voir figure 4.10). Sans nuire au principe général d'évaluation, on fait l'hypothèse que les flux sur les arcs de ce réseau sont constants. Les paramètres utilisés pour l'AG furent les suivants : 400 éléments de population, 200 générations, 60% de probabilité de croisement , 6% de probabilité de mutation.

Pour observer la convergence de l'algorithme, la meilleure fitness ainsi que la moyenne des fitness sur l'ensemble de la population, sont mémorisées à chaque génération. Dans le cas présent, l'algorithme travaille sur une fitness normalisée qui est bornée à "1" lorsque les critères d'équilibrage et de coordination s'annulent simultanément. L'évolution de ces deux paramètres en fonction de la

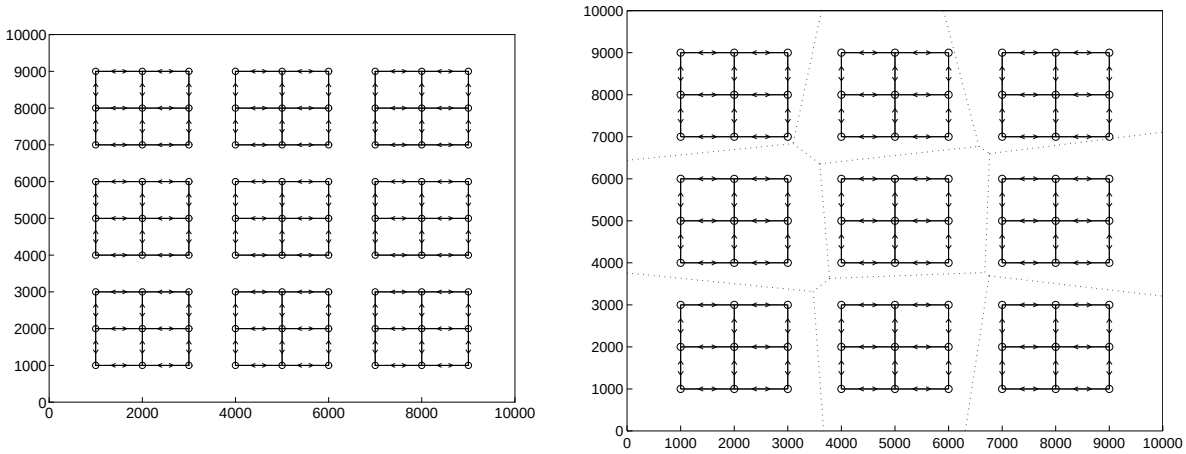


Figure 4.10: Réseau test et sectorisation construite par l'AG

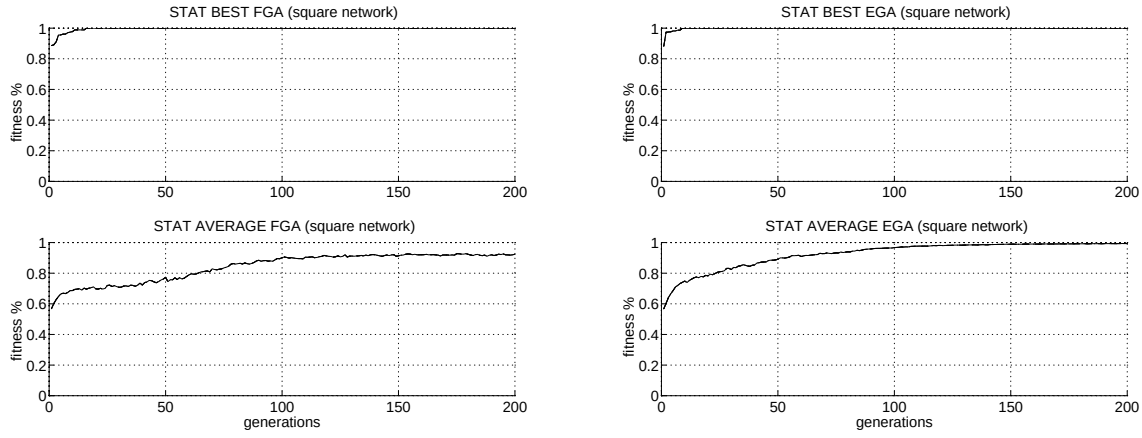


Figure 4.11: Convergence de l'AG sans et avec RS sur le réseau symétrique

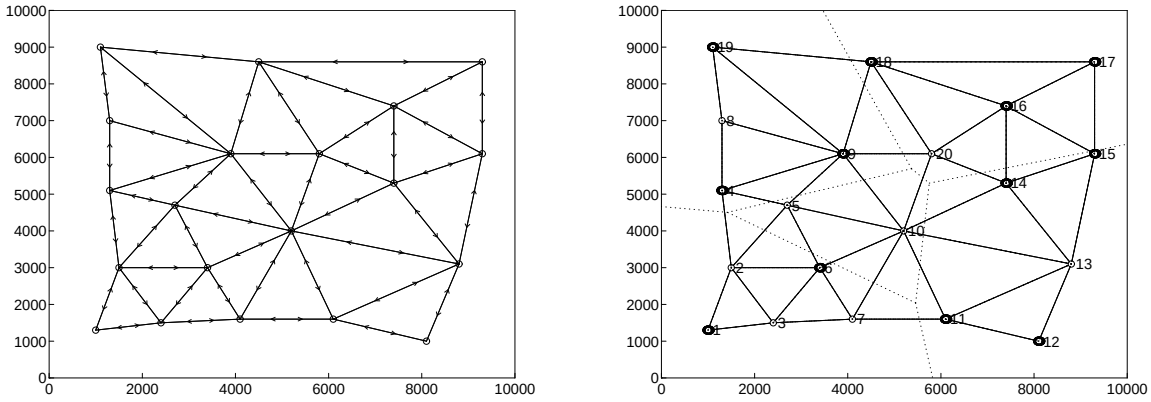


Figure 4.12: Réseau asymétrique et sectorisation construite par l'AG

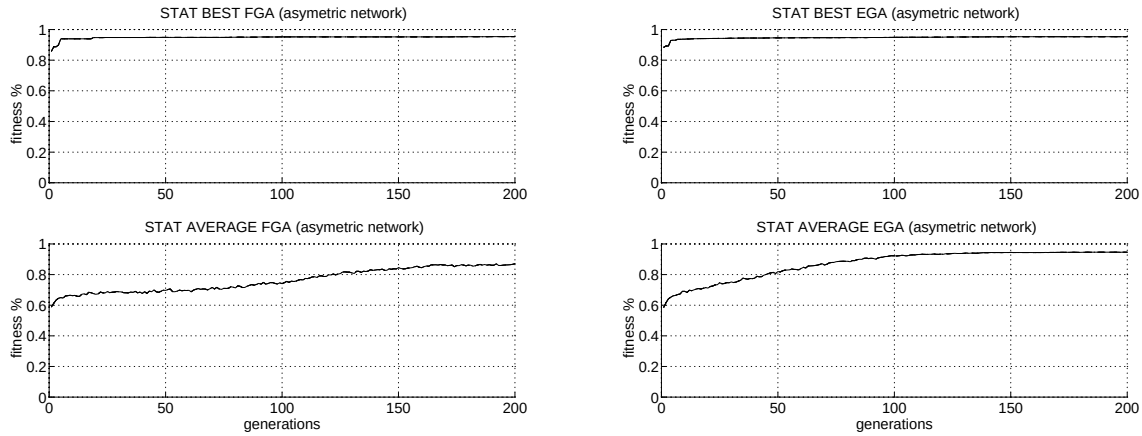


Figure 4.13: Convergence de l'AG sans et avec RS sur le réseau asymétrique

génération est donnée sur la figure 4.11 (sans recuit simulé (FGA)) et sur la figure 4.11 (avec recuit simulé dans le croisement (EGA)).

Les deux algorithmes trouvent une solution exacte très rapidement (15 générations pour FGA et 8 générations pour EGA). Un exemple de solution fourni par l'algorithme est donné en figure 4.10.

Le temps de résolution pour ce type de problème est de cinq minutes sur station "Sparc 10" (pour les deux cents générations).

Après ce premier réseau, nous avons testé l'algorithme sur un réseau quelconque pour lequel il n'y a pas de solution évidente (voir figure 4.12).

On se propose de partitionner ce réseau en cinq secteurs. Comme on le constate sur les courbes de convergence, l'algorithme a un très bon comportement et fournit une solution quasi-exacte en moins de 10 générations (voir figure 4.13 et figure 4.13 pour cette solution le secteur le moins équilibré est à 0.7 pour cent de l'équilibre exact)

A l'inverse du cas précédent, ici la fitness ne peut atteindre 1 car il est impossible d'annuler la coordination. La sectorisation physique générée par l'algorithme est donnée en figure 4.12 et le temps de résolution associé est de trois minutes. Comme on peut le constater, cette sectorisation respecte les

contraintes et présente un aspect tout à fait satisfaisant.

4.5.5 Conclusion

Cette première étude a apporté des résultats encourageants concernant l'utilisation des algorithmes génétiques dans le problème de sectorisation des réseaux de transport. Le codage du chromosome a une grande importance dans l'efficacité de l'algorithme car il permet la synthèse de sectorisation avec un grande simplicité d'implantation. Cette simplicité se retrouve dans les opérateurs de croisement et de mutation en favorisant les performances de l'algorithme. L'utilisation du recuit simulé dans l'opérateur de croisement améliore la convergence au niveau de la statistique des moyennes.

Dans la description précédente, la valeur de K (nombre de secteurs) était fixe mais il serait envisageable de la coder directement dans le chromosome pour que l'algorithme génétique ajuste sa valeur de façon adaptative. Il faudrait alors modifier le critère pour pénaliser fortement les individus dans lesquels il y a des secteurs qui ont une charge supérieure à la charge maximum que peut gérer un contrôleur.

4.6 Affectation des flux sur un réseau sectorisé

Le problème à résoudre est un problème d'affectation statique dans un réseau à coûts d'arcs non séparables asymétriques[5]. Nous allons tout d'abord présenter les méthodes classiques et leurs limitations, puis décrire un principe d'affectation de trafic respectant la contrainte d'équité, basé sur les algorithmes génétiques.

4.6.1 Méthodes classiques d'affectation statique

Algorithme d'affectation de Dijsktra

Initialement cet algorithme a été développé pour simuler le premier principe de Wardrop et converge donc vers un équilibre utilisateur. Nous avons vu précédemment qu'il y a équivalence entre équilibre utilisateur et équilibre système lorsque l'on remplace les coûts réels par les coûts marginaux associés, à condition de vérifier la condition de séparabilité. Notre objectif étant d'obtenir un équilibre système, on se propose d'utiliser un algorithme de Dijsktra sur un réseau pondéré par les coûts marginaux. Le principe de cet algorithme est assez intuitif et consiste à rechercher, pour chaque paire OD, le chemin le plus court sur lequel on affecte un quantum de trafic en réactualisant les coûts d'arcs associés.

Dans le cadre des réseaux à coûts d'arcs non séparables, on peut continuer à utiliser cet algorithme sur les coût marginaux (en vue de déterminer un coût système) à condition que la non séparabilité soit localisée aux nœuds. De fait, on peut remplacer les coût réels par les coûts marginaux pour obtenir l'équilibre système à l'aide d'un algorithme de Dijsktra. Si les coûts sont non séparables mais si l'interaction n'a lieu qu'au niveau des nœuds, l'algorithme de Dijsktra continue également à fonctionner. Dans le cas d'un réseau quelconque, pour lequel la dépendance inter-arcs n'a pas de particularité, les termes dérivés ne s'annulent plus et il faut alors utiliser des algorithmes plus élaborés. C'est le cas du problème qui nous intéresse ici.

[5] Ce travail a donné lieu à une publication[DASF94b]

Algorithme de Dafermos

Initialement cet algorithme [Daf72] a été développé pour des applications routières afin de tenir compte des interactions des flux croisés lors des dépassements. Ce modèle mathématique s'applique essentiellement à des réseaux à coûts d'arcs non séparables asymétriques pour lesquels l'hypothèse suivante est vérifiée : la fonction coût que l'on cherche à minimiser (coût total) doit être strictement convexe par rapport à $\underline{f}$. On suppose de plus qu'il y a une répartition de trafic sur le réseau qui satisfait les lois de Kirchoff. Cette répartition pourrait être, par exemple, le résultat d'un algorithme d'affectation de Dijsktra produisant un équilibre utilisateur.

A partir de la distribution de trafic initiale, l'algorithme recherche pour chaque paire Origine-Destination, le chemin utilisé de coût marginal maximum et le chemin de coût marginal minimum (utilisé ou non). Soient ch_{max} et ch_{min} ces deux chemins. On retire ensuite une quantité du flux transitant sur ch_{max} et on le place sur ch_{min} afin de minimiser le critère global. On reproduit ce processus sur toutes les OD en itérant par permutation circulaire, jusqu'à ce que l'on ne puisse plus diminuer le critère global. On montre alors que l'on a atteint l'équilibre système.

Dans notre cas, l'algorithme de Dafermos ne peut être utilisé car il ne prend pas en compte la contrainte d'équité susmentionné.

4.6.2 Optimisation du problème par AG

Aucun des algorithmes classiques ne donnant de résultats, il a été nécessaire de recourir à des techniques d'optimisation stochastique. Les algorithmes génétiques ont été préférés au recuit simulé, car ils permettent de fournir plusieurs solutions de coût proche. Or, nous souhaitons avant tout proposer à des experts humains différentes solutions parmi lesquelles ils pourront choisir celle qui leur paraît la plus adapté. L'AG est donc plus adapté que le recuit simulé.

Codage du chromosome

Le codage du chromosome est une des phases les plus importantes car elle conditionne en grande partie la réussite du processus de résolution. Le chromosome doit contenir l'information nécessaire pour évaluer la fitness associée en modélisant un point de l'espace d'état. Dans le cas présent, chacun des points de l'espace d'état est représenté par un tableau dont chaque colonne décrit le chemin de la paire Origine-Destination associée (voir figure 4.14). Comme on peut le constater, l'espace de recherche est discret.

L'appartenance des nœuds à un chemin implique que ces derniers soient connexes, ce qui interdit certaines combinaisons qui violent cette contrainte. De plus, pour des questions d'économie, on ne code dans le chromosome que les nœuds qui appartiennent à un chemin sans tenir compte des autres nœuds du réseau. Ce codage rend la dimension du chromosome variable en fonction des longueurs des chemins qu'il code comme on peut le constater sur l'exemple de codage donné en figure 4.15. Dans cet exemple les avions reliant l'aéroport 1 à l'aéroport 16 sont routés sur le chemin $\{1, 4, 3, 7, 12, 16\}$. Inversement les avions faisant le trajet inverse sont routés sur le chemin $\{16, 11, 6, 3, 4, 1\}$ etc.

Principe de génération de la population initiale

Pour utiliser correctement les algorithmes génétiques, il nous faut disposer d'une population initiale d'individus regroupant un ensemble de chemins initialisés aléatoirement. Pour chaque individu, on considère le graphe initial dont les coûts d'arcs sont initialisés à l'aide de la distance géographique associée que l'on bruite ensuite à l'aide d'une distribution gaussienne en veillant à ce que les coûts

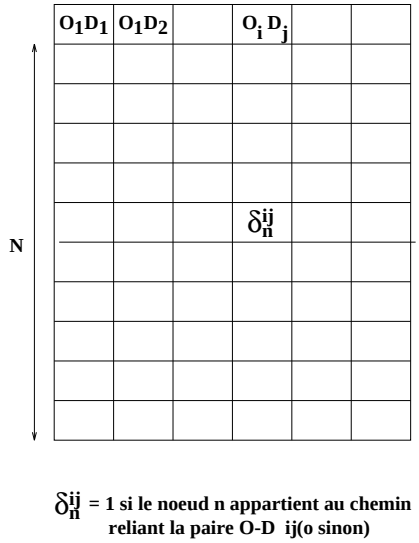


Figure 4.14: Structure d'un point de l'espace d'état

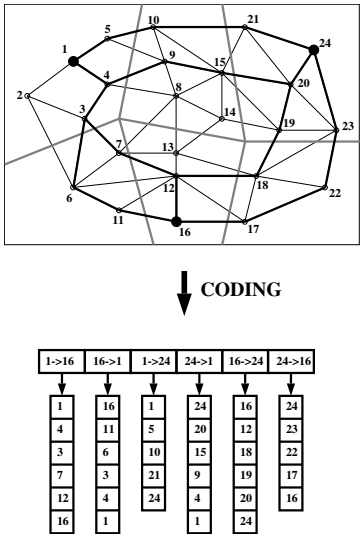


Figure 4.15: Codage du chromosome

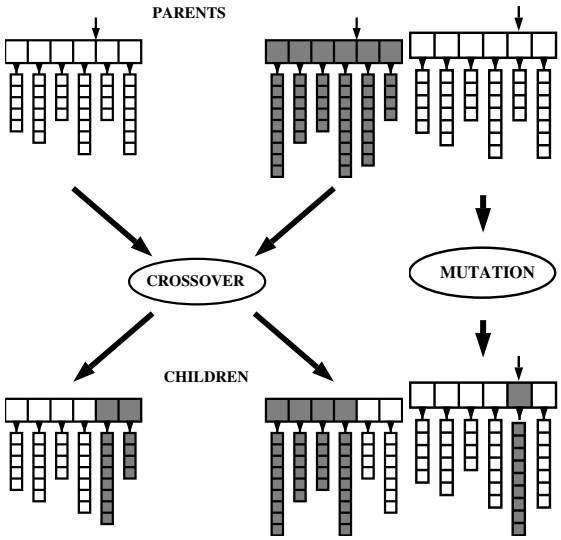


Figure 4.16: Opérateur de croisement et de mutation

restent positifs. Pour chacune des paires Origine-destination contenues dans le chromosome, on recherche le chemin le plus court à l'aide d'un algorithme de Dijkstra (complexité $O(N^2)$ où N est le nombre de nœuds du réseau).

Suivant la déviation de la distribution gaussienne, les chemins générés seront plus ou moins différents du chemin géographique le plus court dans le réseau non bruité. Ce principe d'initialisation évite la génération purement aléatoire des chemins dans le réseau qui conduirait à des aberrations en terme de navigation (par exemple, un chemin qui passerait par Moscou pour une paire Origine-Destination reliant Madrid à Londres). Ce type d'événement doit rester rare mais ne doit pas être complètement inhibé car il permet d'enrichir la diversité des gènes. En effet, après croisement un mauvais chemin peut produire des solutions tout à fait acceptables.

Ayant maintenant une population non homogène, il nous faut définir les opérateurs de croisement et de mutation associés qui vont permettre la génération de nouveaux éléments au sein de la population.

Opérateur de croisement

Notre espace de recherche étant discret, on ne peut envisager d'utiliser un croisement barycentrique et seul le croisement à segmentation de chromosome a été utilisé (Slicing Crossover). Pour effectuer un croisement, on tire aléatoirement une position dans le chromosome puis on échange les deux ensembles de chemins terminaux (voir figure 4.16).

Opérateur de mutation

L'opérateur de mutation nous permet d'enrichir la diversité de la population en créant de nouveaux chemins. En effet, l'opérateur de croisement décrit précédemment génère des nouveaux chromosomes mais ne modifie pas la population de chemins générés aléatoirement lors de l'initialisation du processus. Pour créer de nouveaux chemins, l'opérateur de mutation tire aléatoirement une paire Origine-Destination dans le chromosome et applique le même principe de génération aléatoire de chemin que celui utilisé lors de la création de la population initiale mais ici avec un écart-type plus fort. Un

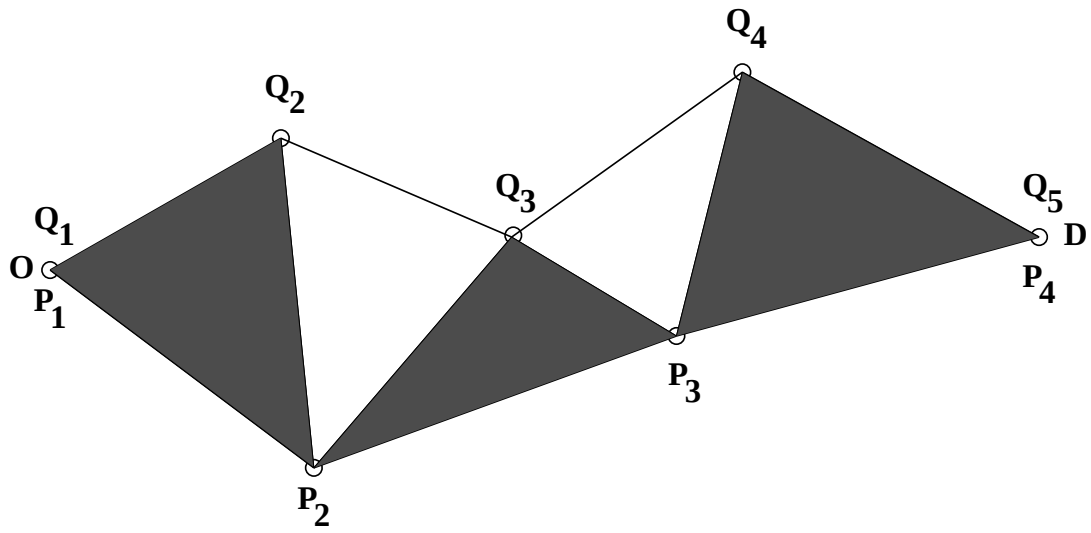


Figure 4.17: Surface délimitée par deux chemins

exemple d'opérateur de mutation est donné dans figure 4.16.

Introduction de la distance inter-chromosomes pour le sharing

Pour effectuer un sharing, on doit disposer d'une distance inter-chromosomes afin d'évaluer le taux d'agrégation des individus dans l'espace d'état. Si l'on considère la description stricte du chromosome, on ne peut pas définir une distance réaliste entre individus car on ne dispose pas d'une structure d'espace vectoriel. En effet, chaque chemin étant codé par la liste des indices des nœuds qui le composent, il est très difficile de déterminer une notion de distance représentative de la différence physique induite dans l'espace géographique sous-jacent. Pour déterminer cette distance entre chemins, on se place directement dans cet espace géographique en considérant non plus les indices des nœuds mais leurs coordonnées. On définit alors la distance entre deux chemins par la surface fermée qu'ils entourent (voir figure 4.17).

Cette surface est alors calculée en faisant la somme des surfaces des triangles construits à l'aide des nœuds de chacun des chemins comme le montre la figure 4.17. L'expression analytique de cette surface est donnée par la formule suivante:

$$S = A(P_1, P_2, Q_2) + \sum_{i=2}^{m-1} \{A(P_i, Q_i, Q_{i+1}) + A(P_i, P_{i+1}, Q_{i+1})\} + \sum_{i=m}^{n-2} \{A(P_m, Q_i, Q_{i+1})\};$$

avec

$$Ch_1 = \{P_1, P_2, ..., P_m\}; Ch_2 = \{Q_1, Q_2, ..., Q_n\}\, m < n\, et\, n \geq 3;$$

Que se passe-t-il lorsque les chemins se croisent? Dans le cas général, cette pseudo-distance majore l'aire comprise entre les deux chemins ; dans le cas particulier de notre réseau de transport planaire dans lequel les croisements de routes se font au niveau des nœuds, la pseudo-distance est toujours égale à l'aire comprise entre les deux chemins.

Disposant d'une distance entre deux chemins, on calcule la distance entre deux chromosomes en faisant la somme des distances respectives associées à chacune des paires de chemins contenues dans chaque chromosome.

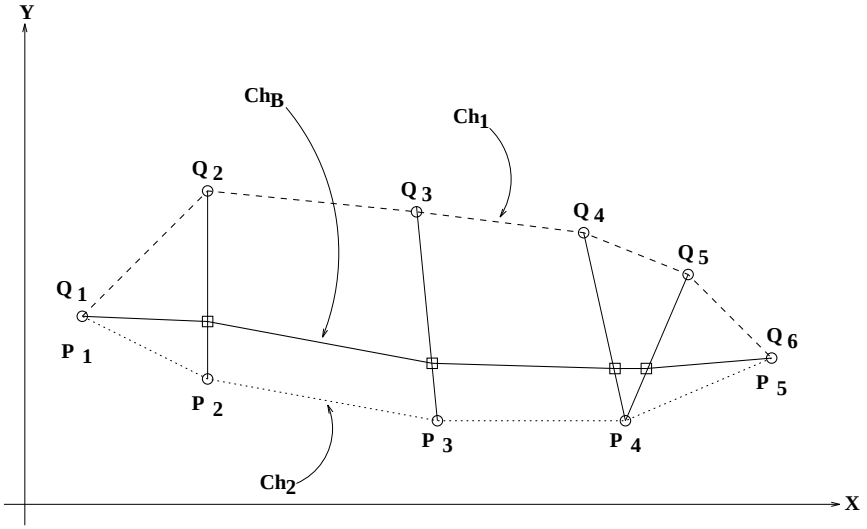


Figure 4.18: Exemple de chemin barycentrique

Calcul du barycentre entre deux chromosomes

Comme pour le calcul de la distance inter-chromosomes, il nous faut travailler dans l'espace géographique sous-jacent au réseau de transport pour calculer le barycentre entre deux chemins. Soient deux chemins :

$$Ch_1 = \{P_1, P_2, ..., P_m\}; Ch_2 = \{Q_1, Q_2, ..., Q_n\}\ n \geq 3;\ m < n$$

Soient n_1 et n_2 les coefficients barycentriques associés respectivement aux chemins Ch_1 et Ch_2. Le chemin barycentrique est alors donné par :

$$Ch_B = (\vec{P}_1, \frac{n_1\vec{P}_2 + n_2\vec{Q}_2}{n_1 + n_2}, \frac{n_1\vec{P}_3 + n_2\vec{Q}_3}{n_1 + n_2}, ...,$$

$$\frac{n_1\vec{P}_{m-1} + n_2\vec{Q}_{m-1}}{n_1 + n_2}, \frac{n_1\vec{P}_{m-1} + n_2\vec{Q}_m}{n_1 + n_2}, ..., \frac{n_1\vec{P}_{m-1} + n_2\vec{Q}_{n-1}}{n_1 + n_2}, \vec{Q}_n);$$

Dans cette expression $\vec{P}$ représente la position géographique du nœud P.

Un exemple de chemin barycentrique est donné sur la figure 4.18 avec $n_1 = 2$ et $n_2 = 1$.

Pour déterminer le barycentre entre deux chromosomes, il suffit d'appliquer ce processus de calcul à chacun des chemins contenus dans le chromosome.

Calcul et normalisation de la fitness

Dans un premier temps, on considère le réseau de transport sans trafic. Pour chaque paire Origine-Destination contenue dans le chromosome, on affecte la demande correspondante sur le chemin décrit par la liste des nœuds associée. On dispose alors de la répartition de flux sur chacun des arcs du réseau, ce qui nous permet de calculer le sous-critère d'affectation C_1 ainsi que la charge de contrôle globale (et donc le critère C).

Le nombre d'opérations nécessaires à l'évaluation d'une fitness dépend de la taille du réseau et varie en $O(N_{od} + L + K(N + L))$ (N_{od} : nombre de paires Origine-Destination, L : nombre d'arcs du réseau, K : nombre de secteurs, N : nombre de nœuds).

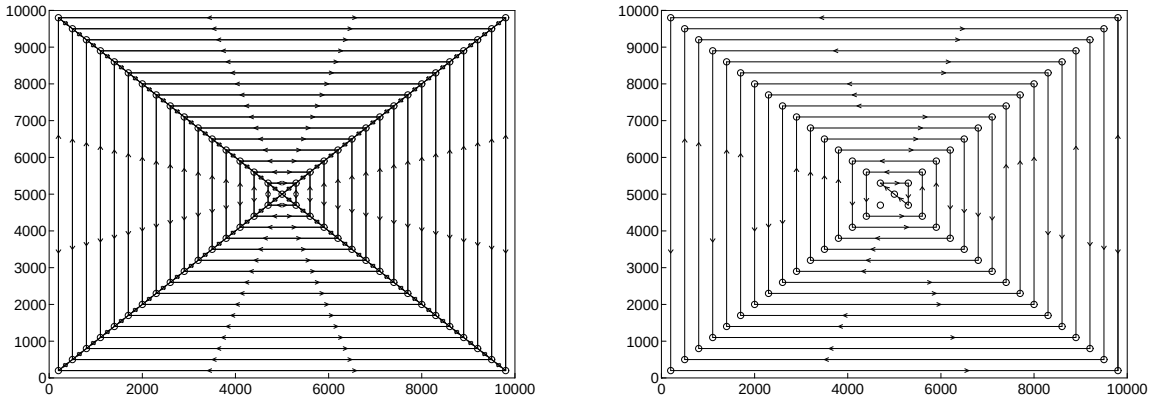


Figure 4.19: Réseau test et affectation résultante

Évaluation

Pour évaluer notre algorithme, on utilise un réseau test pour lequel on connaît une solution triviale d'affectation (voir figure 4.19).

Sur ce réseau, les nœuds sur la première diagonale représentent les aéroports et ceux de la deuxième diagonale sont les balises. On se propose d'affecter une demande de trafic constante entre chaque paire d'aéroports de façon symétrique (chaque aéroport génère une demande à destination de l'aéroport symétrique, ce dernier lui envoyant la même quantité de trafic).

Pour cet exemple, on règle le coefficient de congestion de telle sorte que deux flots de trafic ne peuvent emprunter le même arc ; on interdit de même que deux flots se retrouvent face à face sur un arc. Le critère se présente alors sous la forme suivante :

$$C = \alpha \sum_{(i,j)\in L} C_{ij}(f_{ij}) + (1-\alpha) \sum_{(i,j)\in L} \gamma f_{ij} f_{ji}$$

Comme on peut le constater, il n'y a pas de sectorisation mais la forme du critère est similaire à celle décrite précédemment et ne change en rien la généralité de l'algorithme.

Les paramètres utilisés furent les suivants : 400 éléments de population, 300 générations, 60% de probabilité de croisement , 6% de probabilité de mutation.

Pour ce test nous avons utilisé un algorithme génétique avec du recuit simulé dans l'opérateur de croisement afin d'améliorer les performances de convergence. L'évolution de la fitness (normalisée entre 0 et 1) du meilleur individu ainsi que la moyenne des fitness sur l'ensemble de la population, en fonction de la génération, sont données sur la figure 4.20.

Comme on peut le constater, une solution optimale est trouvée à partir de la génération 180 pour laquelle intervient seulement la partie du critère liée à la distance de parcours. Cette solution est obtenue en six minutes sur une station "Sparc 10". L'affectation physique correspondant à cette solution est donnée sur la figure 4.19.

Au cours de ces évaluations, on s'aperçoit que les performances de cet algorithme sont très sensibles à l'amplitude de l'écart-type de la distribution utilisée pour bruiter les coûts d'arcs dans le processus de génération des chemins aléatoires. Pour chaque problème, il faut régler ce paramètre afin d'obtenir des chemins aléatoires différents du chemin optimal (en terme de distance) mais qui à l'inverse ne soient pas trop pénalisants par rapport au rallongement induit. Une solution consisterait

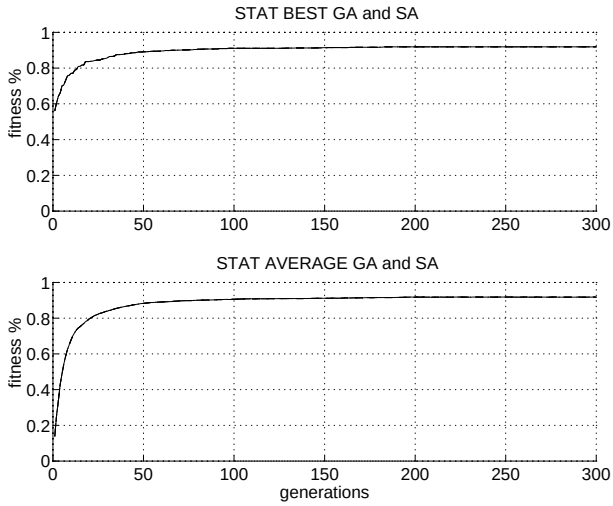


Figure 4.20: Évolution de la fitness normalisée

à coder ce paramètre dans le chromosome et à le faire ajuster par l'algorithme génétique de façon adaptative.

4.6.3 Conclusion

La contrainte d'équité nous empêche d'utiliser les méthodes classiques d'affectation de trafic sur les réseaux à coûts d'arcs non séparables et induit une forte complexité nous obligeant à nous orienter vers des méthodes d'optimisation stochastique. Après avoir développé un codage du chromosome ainsi que des opérateurs adaptés, on s'aperçoit que les algorithmes génétiques traitent ce problème de façon efficace à condition que le générateur de chemins aléatoires soit adapté au réseau de transport utilisé.

La contrainte d'équité, qui permet de conserver la libre concurrence des compagnies aériennes, a pour inconvénient de réduire notre espace de recherche et nous interdit donc d'accéder à des zones d'espace où le critère serait assurément meilleur. En effet, on montre que lorsque le critère à optimiser n'est pas linéaire par rapport aux flux sur les chemins, mais plutôt quadratique, il est plus avantageux de segmenter les flux afin de rendre le système plus capacitif.

4.7 Sectorisation et affectation de trafic simultanées

La dépendance très forte des problèmes d'affectation et de sectorisation incite à penser que ces deux problèmes doivent être résolus simultanément. Dans le cas de la sectorisation, l'optimisation visait essentiellement à réduire les coordinations et à équilibrer les charges de contrôle. De même, l'optimisation de l'affectation avait pour but la minimisation de la charge globale de contrôle tout en minimisant les rallongements de route induits.

La diversité des critères liés à ces deux problèmes, quand bien même certains sont communs, révèle la nature multi-objectifs du problème global. Dans un premier temps, une approche itérative avec deux algorithmes génétiques bouclés s'est révélée inefficace de par l'instabilité des solutions fournies. Cet échec a amené Daniel Delahaye à proposer de traiter le problème dans sa globalité à

l'aide d'un algorithme génétique unique en codant les deux sous-problèmes dans un diploïde[6]. Cette approche n'a pas encore été testée.

4.8 Regroupement de secteurs

4.8.1 Introduction

Au cours d'une journée de trafic ordinaire, on remarque que la charge de contrôle fluctue dans le temps en fonction des demandes de trafic entre les diverses paires Origine-Destination. Par exemple, en France, il y a une pointe de trafic le matin et une pointe le soir.

Les algorithmes présentés jusqu'à présent ont toujours travaillé avec des flux importants sur le réseau afin d'assurer l'absorption des pointes de trafic, ce qui correspond au cas le plus défavorable. Ainsi nos problèmes sont optimisés pour ce genre de trafic et les solutions apportées deviennent sous-optimales lorsque les flux sur le réseau diminuent.

Dans le système opérationnel actuel, le nombre de contrôleurs varie en fonction des demandes de trafic. Ainsi, la nuit, le nombre d'équipes de contrôle est réduit car il y a beaucoup moins de trafic. Si tel n'était pas le cas, on se retrouverait avec des positions de contrôle traversées par une dizaine d'avions par heure ; ce qui générerait un coût de contrôle surdimensionné par rapport au service rendu aux usagers. Les secteurs sont donc regroupés la nuit en groupe de trois à quatre ; chaque groupe ainsi constitué est ensuite attribué à une équipes de contrôleurs. Actuellement ce regroupement est fait de façon empirique au niveau de chaque centre de contrôle régional par des experts gérant l'espace aérien, qui groupent et dégroupent les secteurs par anticipation des fluctuations de trafic. Ainsi, le matin, les secteurs sont dégroupés un peu avant la pointe de trafic liée à l'ouverture des aéroports.

Dans ce chapitre, nous décrivons une méthode automatique de regroupement[7] de secteurs fournissant une solution globale sur l'ensemble de l'espace aérien.

4.8.2 Modélisation

Nous faisons l'hypothèse que les secteurs initiaux sont convexes au sens géométrique du terme et pourraient être le résultat de l'algorithme de sectorisation décrit dans les chapitres précédents. Ceci ne change en rien les propriétés de l'algorithme de regroupement proposé dans ce chapitre et ce dernier pourrait très bien s'appliquer à la sectorisation actuelle qui est constituée par des secteurs convexes au sens des routes aériennes. Les propriétés d'un bon regroupement sont en fait les mêmes que celles demandées à la sectorisation : on cherche à synthétiser des groupes de secteurs dont les charges de contrôle résultantes sont équilibrées et qui minimisent les coordinations restantes (coordination inter-groupes). Un exemple de regroupement de secteurs de contrôle est fourni sur la figure 4.21.

Comme on peut le constater, le regroupement $\{(S_1, S_2, S_3); (S_4, S_5, S_6)\}$ fait disparaître les charges de coordination liées aux anciennes frontières de secteur : $(S_1, S_2); (S_2, S_3); (S_4, S_5); (S_5, S_6)$.

D'autre part, tous les secteurs appartenant à un même groupe doivent être connexes. Cette contrainte évite par exemple qu'un même contrôleur gère une portion de trafic à Lille et une autre à Marseille ; ce qui est une aberration en terme de contrôle du trafic aérien dans la mesure où l'on perd l'homogénéité du trafic dans les secteurs gérés par une même équipe de contrôleurs et produit de plus des coordinations inutiles.

Comme on peut le remarquer, ce problème est très proche du problème de sectorisation. La différence essentielle réside dans la description de l'espace de recherche associé. En effet, dans le

[6] Un "diploïde" est un chromosome à deux branches possédant des gènes bien distincts.

[7] Ce travail a donné lieu à une publication[DASF95].

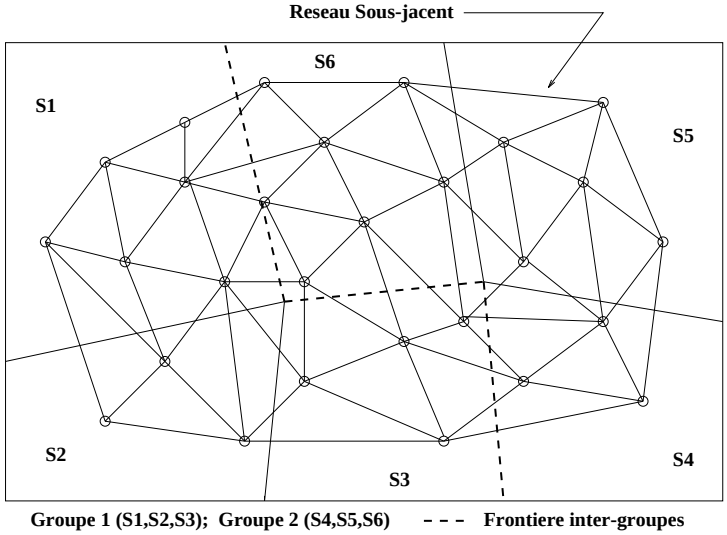


Figure 4.21: Exemple de regroupement pour N=4 et K=2

problème de sectorisation l'espace était continu et les frontières des secteurs générés pouvaient se trouver à n'importe quel endroit dans l'espace géographique. A l'inverse, l'espace de recherche lié au problème de regroupement est un espace discret pour lequel les frontières des groupes constitués doivent coïncider avec les frontières des secteurs existants (chaque groupe associe un ensemble de secteurs sans en créer de nouveaux). Notre problème de regroupement s'apparente donc à un problème de classification.

4.8.3 Complexité du principe de regroupement

Dans un premier temps nous examinons le nombre de points constituant notre espace de recherche en négligeant la contrainte de connexité. Il s'agit donc de savoir quel est le nombre de façon possible de construire K sous-ensembles d'un ensemble de N éléments où N représente le nombre total de secteurs et K le nombre de groupes constitués. Le nombre de cas possibles est donné par le second nombre de Stirling,soit pour N=600 et K=200 (groupement moyen de trois secteurs) :

$$S_{600}^{200} = 0.8\ 10^{1001}$$

Si l'on tient compte maintenant de la contrainte de connexité l'espace de recherche se réduit. Pour étudier la complexité induite, le problème est modélisé de la façon suivante. On construit un graphe dans un espace euclidien dont chaque nœud représente le barycentre de chacun des secteurs constituant la sectorisation initiale et dont chaque arc traduit la relation *ont une frontière commune*.

Il s'agit alors de la recherche d'un K-partitionnement optimal d'un graphe à N nœuds en composantes connexes, problème connu pour être NP_complet. Le problème traité étant de forte dimension (N=600,K=200), seules les techniques d'optimisation stochastique ont été envisagées pour le résoudre.

4.8.4 Optimisation par algorithme génétique

La ressemblance des problèmes de sectorisation et de regroupement conduit à des similitudes dans l'utilisation des algorithmes génétiques pour les résoudre. En effet, le problème de regroupement

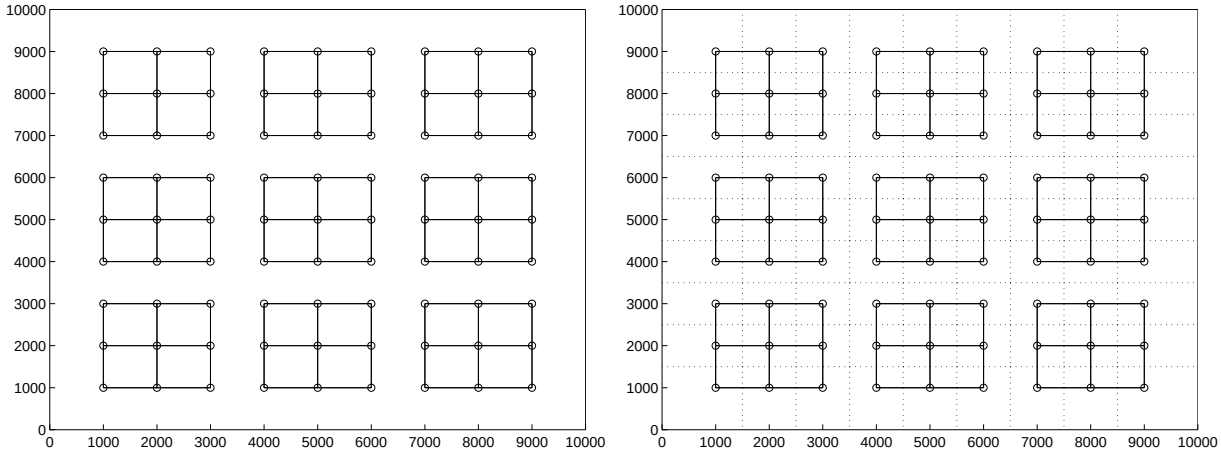


Figure 4.22: Réseau test utilisé et son découpage

étant une version discrète du problème de sectorisation, il est tout à fait logique d'utiliser un codage et des opérateurs similaires.

Ainsi, disposant d'une sectorisation initiale, on calcule pour chaque secteur le barycentre géométrique associé auquel on affecte la charge de contrôle qu'il renferme. Pour générer aléatoirement N groupes de secteurs, on jette N points (centres de groupe) dans l'espace géographique, dont les coordonnées suivent une loi de distribution uniforme, puis on agrège chaque barycentre à son centre de groupe le plus proche. On synthétise ainsi des groupes qui respectent toujours la contrainte de connexité car le polygone construit à l'aide des barycentres des secteurs contenus dans un groupe est convexe, les secteurs associés sont donc obligatoirement connexes. On retrouve alors un codage du chromosome identique à celui utilisé pour le problème de sectorisation dans lequel sont regroupés ici les coordonnées des centres de groupe.

Ayant un codage identique à celui utilisé dans la résolution du problème de sectorisation on utilise les mêmes opérateurs de croisement et de mutation. Le calcul des distances inter-chromosomes et du barycentre entre deux individus est effectué de la même façon.

A l'initialisation du processus, on calcule la charge de contrôle de chacun des secteurs de la sectorisation initiale. Pour chaque individu, on détermine la charge de monitoring et de conflit dans chacun des groupes qu'il renferme en faisant la somme des charges de contrôle contenues dans les divers secteurs ainsi que la charge résiduelle de coordination. Ces quantités nous permettent ensuite de calculer les sous-critères d'équilibrage et de coordination et donc la fitness associée à un regroupement.

De la même manière que pour les problèmes de sectorisation, on valide les performances de l'algorithme à l'aide d'un réseau artificiel pour lequel on connaît une solution évidente. Dans le cas présent, on reprend le réseau test utilisé dans le problème de sectorisation (voir figure 4.22) que l'on découpe initialement en 81 secteurs (voir figure 4.22) et pour lequel on recherche neuf groupes équilibrés qui minimisent les coordinations. La solution évidente de ce problème est constituée des neuf sous-réseaux regroupant chacun neuf secteurs. Une solution optimale est trouvée à la génération 50 (après une minute de temps de calcul sur "Sparc 10").

Le résultat obtenu peut surprendre au premier abord (figure 4.23), mais il faut se rappeler que l'algorithme travaille avec les barycentres des secteurs et non avec les secteurs eux-mêmes. En revenant à la définition initiale des secteurs, on obtient le regroupement décrit sur la figure 4.23 qui correspond bien au résultat attendu.

Les résultats fournis par ces évaluations, nous incitent à penser que les algorithmes génétiques

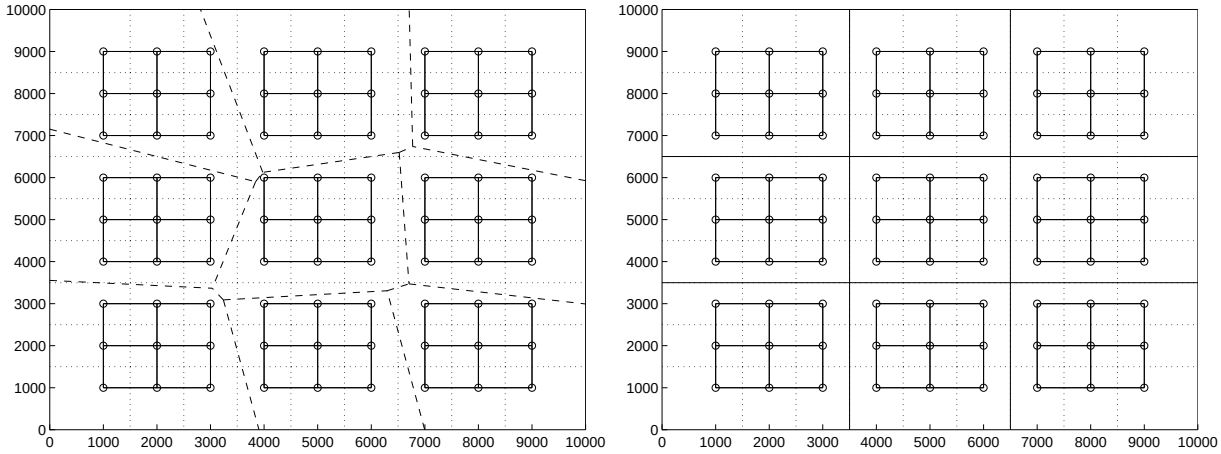


Figure 4.23: Solution et regroupement résultant

sont bien adaptés pour résoudre le problème de regroupement. Malgré la restriction de l'espace d'état liée au codage du chromosome, les solutions apportées par l'algorithme sont de très bonne qualité dans la plupart des cas.

4.9 Limitations du modèle

4.9.1 Modélisation de la charge de contrôle

Les paramètres liés à la description de cette charge ont été choisis empiriquement et pourraient être affinés à l'aide d'expérimentations. Cette modélisation est très délicate car beaucoup de facteurs entrent en jeu dans la résultante de la charge de contrôle. EUROCONTROL (Brétigny projet RAMS) a développé un modèle assez réaliste de cette charge mais au prix d'une complexité élevée. En effet, ce dernier intègre 200 tâches élémentaires de contrôle dans le cadre des simulations arithmétiques qu'on ne peut utiliser dans nos algorithmes génétiques pour des questions de temps de calcul.

Une autre approche consisterait à faire de l'apprentissage sur les paramètres utilisés dans la description simplifiée de la charge de contrôle que l'on a développée. Dans un premier temps, l'algorithme génétique utiliserait des paramètres grossiers, et fournirait tout de même une sectorisation. Ce résultat serait ensuite présenté à des experts qui corrigeraient ce dernier en modifiant les frontières des secteurs à l'aide d'une interface graphique. On calculerait alors un écart (Δ) entre les deux sectorisations qui servirait de base à la redéfinition des coefficients du modèle. On itèrerait ce procédé jusqu'à ce que les experts soient satisfaits du résultat produit par l'algorithme. Si l'on ne peut atteindre cet objectif, il faudrait revoir le modèle lui-même et non le réglage de ses coefficients. Pour que cette méthode fonctionne correctement il faut que les objectifs de sectorisation utilisés par les algorithmes génétiques soient les mêmes que ceux des experts, ce qui n'est pas toujours le cas.

Enfin, il serait possible de faire un ajustement statistique des paramètres utilisés dans notre modèle de charge de contrôle à l'aide de la connaissance précise de la charge de travail dans un secteur (connaissance issue d'un modèle intégrant un grand nombre de tâches de contrôle (simulation RAMS par exemple) que l'on ne peut utiliser dans nos algorithmes à cause de la complexité associée).

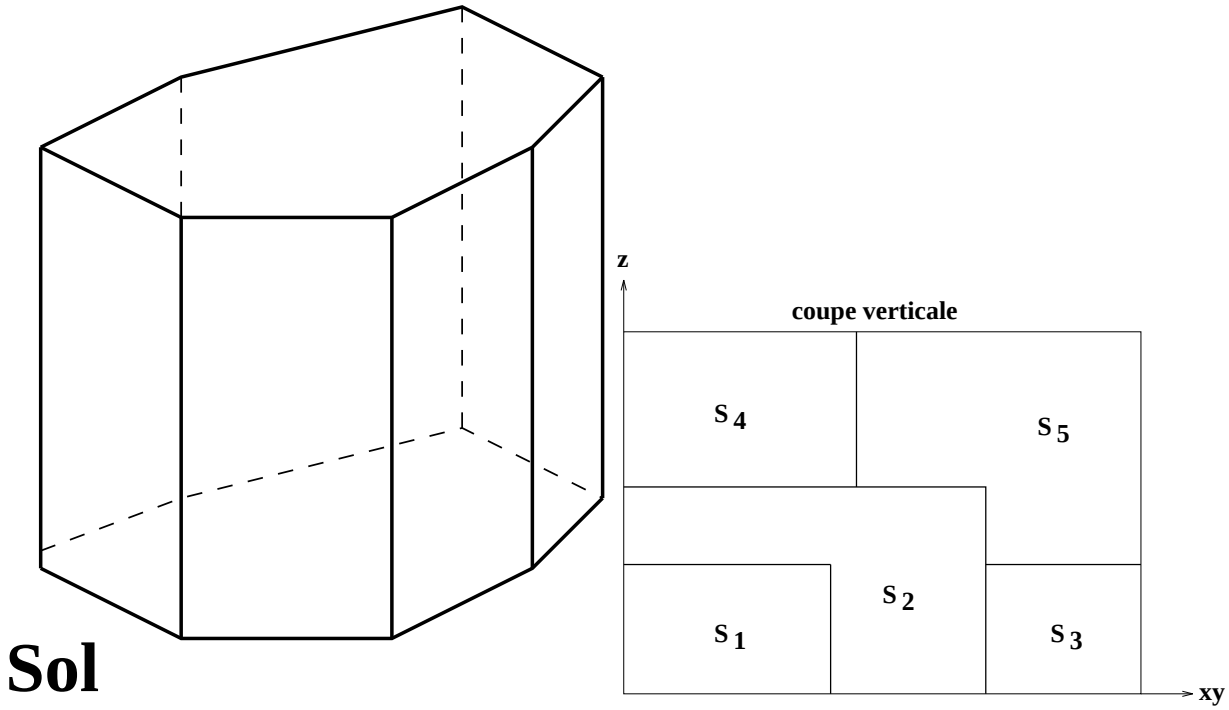


Figure 4.24: Structure des secteurs actuels

4.9.2 Prise en compte de la troisième dimension

Jusqu'à présent, seul le trafic en route a été pris en compte pour synthétiser la charge de contrôle. Cette hypothèse sur la nature du trafic nous a permis de modéliser le réseau de transport aérien à l'aide d'un graphe dans un espace de dimension deux.

Malheureusement, ce type de représentation n'est pas adapté pour le trafic évolutif pour lequel les avions changent très souvent de niveau de vol. Il faut alors utiliser un réseau de transport à trois dimensions dans lequel les arcs représentent des "routes aériennes 3D".

On pourrait essayer de se rapprocher du principe actuel de sectorisation 3D qui utilise des secteurs de forme polygonale à frontières verticales planaires (voir figure 4.24). Ces secteurs ont de plus des extensions verticales différentes qui produit une structure imbriquée dont un exemple est donné sur la figure 4.24. Cependant cette structure de sectorisation demande de modifier en profondeur la représentation du chromosome et donc l'algorithme génétique dans son ensemble.

4.9.3 Limitation de l'exploration de l'espace d'état

La synthèse des secteurs de contrôle à l'aide des centres de classe (voir chapitre traitant de la sectorisation) est très intéressante pour la description du chromosome utilisé par les algorithmes génétiques car très peu d'informations sont nécessaires pour construire une sectorisation complète qui de plus, satisfait la propriété de convexité géométrique. On rappelle que seules les positions géographiques des centres de classe sont codées dans le chromosome. En examinant la sectorisation actuelle, on constate que les secteurs ont des formes polygonales mais pas nécessairement convexes d'un point de vue géométrique. Ce type de représentation est plus riche en terme d'exploration spatiale et permet d'accéder à de nouveaux points de l'espace d'état, que la propriété de convexité nous interdit d'adres-

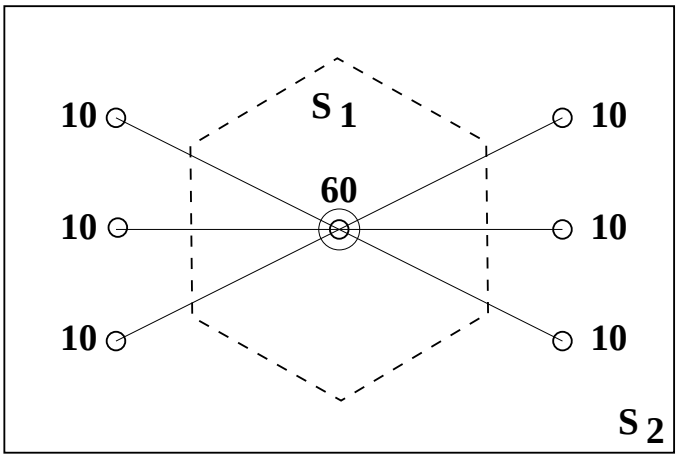


Figure 4.25: Exemple de limitation liée à la convexité géométrique

ser. Ainsi, sur l'exemple de la figure 4.25, une sectorisation convexe ne peut pas équilibrer les charges de contrôle alors qu'en supprimant la convexité géométrique il est possible d'atteindre cet objectif. De la même façon que pour la limitation des algorithmes génétiques par rapport à l'extension tridimensionnelle, le codage de chromosomes permettant de synthétiser des sectorisations polygonales non convexes s'avère assez compliqué. Daniel Delahaye propose diverses solutions pour remédier à ce problème [Del95].

4.9.4 Prise en compte des zones militaires

Une partie de l'espace aérien est réservé aux autorités militaires. L'activation de ces zones, qui est faite à l'initiative des instances de la défense, provoque des perturbations plus ou moins importantes dans le trafic aérien civil suivant la configuration du moment. De plus, les secteurs qui contiennent ces zones voient leur espace se réduire brutalement limitant ainsi les possibilités d'action des contrôleurs. Cette restriction d'espace provoque une augmentation de la charge de contrôle induite pour la même quantité de trafic.

Nous n'avons pas pris en compte ce type d'événement dans nos algorithmes car leur modélisation est très délicate. Un moyen détourné d'intégrer ces zones dans le processus de résolution consisterait à augmenter artificiellement la charge de contrôle des secteurs qui les contiennent.

4.9.5 Problèmes politiques liés à la souveraineté de l'espace aérien

En observant la sectorisation actuelle, on constate que les limites des secteurs épousent les frontières géographiques des pays qui les contiennent. Ceci se justifie dans la mesure où le contrôle du trafic aérien est généralement confié aux pays survolés par les aéronefs. Il est clair que cette restriction réduit les performances de la sectorisation actuelle car elle provoque des coordinations superflues.

4.9.6 Prise en compte de la propagation des flux

Le fait d'utiliser des flux statiques sur le réseau produit une charge de contrôle invariante dans le temps, que l'on a majorée en considérant le flux maximum enregistré sur une journée sur chacun des arcs du réseau. Or, on sait très bien que ces valeurs extrêmes ne sont jamais présentes simultanément, car les heures de pointe de trafic sont locales aux fuseaux dans lesquels se trouvent les aéroports. Ainsi, les pointes sont décalées dans le temps. De plus, lorsqu'il y a une augmentation de la demande de trafic sur une paire Origine-Destination donnée, elle va induire un accroissement du flux sur la route associée avec un délai de propagation lié à la vitesse des aéronefs. Les algorithmes (de sectorisation et surtout d'affectation) devraient tenir compte de cette propagation pour déterminer la charge réellement présente sur le réseau de transport.

4.10 Conclusion

En examinant le contexte opérationnel, on remarque que ces algorithmes ont besoin d'adaptation pour envisager leur utilisation sur le réseau aérien réel. La principale amélioration se situe au niveau de la description de la charge de contrôle. Le problème à résoudre est complexe et il n'existe pas aujourd'hui de modèle fiable facile à implanter.

La seconde limitation se trouve au niveau du codage du chromosome de sectorisation qui, dans son état actuel, synthétise des secteurs convexes au sens géométrique du terme. Ce type de convexité masque des domaines de l'espace d'état où se trouve parfois l'optimum global.

Enfin, la prise en compte de la troisième dimension permettrait de traiter le trafic évolutif dans les zones terminales, mais il faut pour cela synthétiser des secteurs à frontières latérales verticales, ce qui complique sensiblement la description du chromosome.

En amont de la sectorisation, il est clair que le réseau de routes aériennes actuel nécessite lui-même une optimisation qui permettrait ensuite de développer une sectorisation plus performante. En effet, ce réseau a été "construit" en regroupant un ensemble de sous-réseaux (propres à chaque pays européen), optimisés individuellement. Il est possible que l'augmentation de la capacité des secteurs ATC, passe d'abord par une redéfinition du réseau de routes : ces dernières ont été construites pour une demande de trafic donnée qui ne cesse aujourd'hui de se modifier tant en volume qu'en orientation.

Conclusion

Les problèmes liés au trafic aérien sont à la fois intéressants, difficiles, et surtout largement inexplorés. Il s'agit d'une mine extraordinaire pour le chercheur, et on est même parfois surpris de la rareté du travail scientifique sur des domaines aussi fondamentaux pour l'Aviation Civile que la résolution de conflits, la sectorisation ou la prévision de trajectoires.

Pour notre part, nous nous sommes efforcés d'appliquer une méthodologie scientifique à certains problèmes d'optimisation qui se posent dans le cadre du trafic aérien. Nous pouvons résumer notre approche de la façon suivante :

- modélisation mathématique des problèmes rencontrés
- calcul de leur complexité
- recherche d'algorithmes adaptés
- validation des résultats obtenus, sur un plan expérimental par la simulation, sur un plan scientifique par la publication.

Il serait bien difficile, et fort prétentieux, d'estimer la portée de nos travaux. Pourtant, il nous semble indispensable de poursuivre ce type de recherches au sein de l'Aviation Civile. En quelques années, les cockpits d'avion, et les systèmes de commande et de contrôle du vol, sont devenus de véritables centres informatiques, et l'automatisation a été poussée au maximum, peut-être d'ailleurs trop vite. On peut en revanche se préoccuper du retard pris en matière de modernisation du contrôle aérien, et les centres de contrôle donnent par comparaison l'impression d'en être encore à l'âge de pierre. Alors que les constructeurs aériens font des efforts démesurés pour réduire les coûts d'opération des appareils de quelques dixièmes de pourcent, les compagnies estiment actuellement que l'influence du contrôle sur leurs coûts d'opérations est de l'ordre de 10%.

Avec l'augmentation régulière de la densité du trafic, il est probable que la seule augmentation du nombre de contrôleurs ne permettra pas d'obtenir un écoulement fluide. Le problème se pose de façon d'autant plus aiguë pour la France, qui se trouve à la croisée des grands flux de trafic Nord-Sud et Est-Ouest passant au dessus de l'Europe. Elle sera donc un des pays les plus touchés par le problème du contrôle en route, et des délais et coûts induits[8]. C'est pour cette raison qu'il faut mettre en place des structures et des équipes susceptibles de répondre aux préoccupations des compagnies en matière d'optimisation des capacités et des techniques de contrôle. Enfin, pour obtenir une véritable crédibilité, il faut absolument encourager la publication des travaux d'études et de recherches dans des conférences et des revues à comité de lecture *extérieurs* à la DGAC. Ceci peut certes être ressenti comme une contrainte. Pourtant, comme la recherche scientifique l'enseigne, le progrès passe par une

[8]Les problèmes du contrôle américain sont bien différents : il s'agit surtout de problèmes de régulation de flux au niveau des aéroports, qui sont aux Etats-Unis l'élément limitant du système.

permanente remise en cause, et le bénéfice que l'on en retire vaut bien plus qu'un auto-satisfecit trop facilement délivré.

Bibliographie

[AK89] E Aarts and J Korst. *Simulated Annealing and Boltzmann Machine*. John Wiley and Sons, 1989.

[Ale70] Ben Alexander. Aircraft density and midair collision. *Proceedings of the IEEE*, 58(3), March 1970.

[All95] Jean-Marc Alliot. A genetic algorithm to improve an othello program. In *Proceedings of EA95*. Springer Verlag, 1995.

[AS92] Jean-Marc Alliot and Thomas Schiex. *Intelligence Artificielle et Informatique Théorique*. Cepadues, 1992. ISBN: 2-85428-324-4.

[AVAD52] Adelson-Velsky, Arlazarov, and Donskoy. *Algorithms for games*. Springer-Verlag, 1952. ISBN: 0-387-96629-3.

[BD93] A. Bertoni and M. Dorigo. Implicit parallelism in genetic algorithms. *Artificial Intelligence*, 61(2): 307–314, 1993.

[BG87] C.L Bridges and D.E Goldberg. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In *Proceedings of the Second International Conference on Genetic Algorithm*. ICGA, 1987.

[BG91] C.L Bridges and D.E Goldberg. An analysis of multipoint crossover. In *Proceedings of the Foundation Of Genetic Algorithms*. FOGA, 1991.

[BH91] John T. Betts and William P. Huffman. Trajectory optimization on a parallem processor. *Journal of Guidance, Control and Dynamics*, 14(2): 431–439, 1991.

[BM93] T.N Bui and B.R Moon. Hyperplane synthesis for genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithm*. ICGA, 1993.

[BM94] T.N Bui and B.R Moon. Analysing hyperplane synthesis in genetic algorithms using clustered schemata. Technical Report Cse-94-026, Dep. of Comp. Sci. and Engineering, Penn. State University, March 1994.

[Bos94a] J. F. Bosc. Rapport préliminaire 1.1, Ecole Nationale de l'Aviation Civile, Mai 1994.

[Bos94b] J. F. Bosc. Rapport intermédiaire 2.2, Ecole Nationale de l'Aviation Civile, Août 1994.

[Bos94c] J. F. Bosc. Rapport intermédiaire 3.2, Ecole Nationale de l'Aviation Civile, Novembre 1994.

[Bos95a] J. F. Bosc. Rapport intermédiaire 4.3, Ecole Nationale de l'Aviation Civile, Février 1995.

[Bos95b] J. F. Bosc. Rapport intermédiaire 5.3, Ecole Nationale de l'Aviation Civile, Juin 1995.

[Bos95c] J. F. Bosc. Rapport intermédiaire 6.2, Ecole Nationale de l'Aviation Civile, Août 1995.

[Bos96a] J. F. Bosc. Rapport intermédiaire 7.2, Ecole Nationale de l'Aviation Civile, Janvier 1996.

[Bos96b] J. F. Bosc. Rapport intermédiaire 8.3, Ecole Nationale de l'Aviation Civile, Avril 1996.

[Bra90] H. Braun. On traveling salesman problems by genetic algorithms. In *1st Workshop on Parallel Problem Solving from Nature*, October 1990.

[Bro69] C.G. Broyden. A new double-rank minimization algorith. *AMS notices*, 16: 670, 1969.

[Bur94] Michael Buro. *Techniken fur die bewertung von Speilsituationen anhand von beispielen*. PhD thesis, Universitat GH Paderborn, 1994.

[Cat90] O. Catoni. *Large deviations for Annealing*. PhD thesis, Université de Paris XI, 1990.

[CDV92] D Colin De Verdière. Le système français de contrôle du trafic aérien : Le CAUTRA. Technical report, CENA, Toulouse France, Aout 1992.

[Cel90] Joseph C. Celio. Controller perspective of AERA2. Technical report, MITRE, February 1990. MP-88W00015.

[Cer94] R Cerf. *Une Théorie Asymptotique des Algorithmes Génétiques*. PhD thesis, Université Montpellier II (France), 1994.

[CGT92] A.R. Conn, Nick Gould, and Ph. L. Toint. A comprehensive description of LANCELOT. Technical report, IBM T.J. Watson research center, 1992. Report 91/10.

[Cha95] Olivier Chansou. Résolution automatisée de conflits en route. Master's thesis, Ecole Nationale de l'Aviation Civile (ENAC), 1995.

[Che92] C.K Cheng. The optimal partitioning of networks. *Networks*, 22: 297–315, 1992.

[CJ91] R.J. Collins and D.R. Jefferson. Selection in massively parallel genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[Cle90] John C. Clements. Minimum-time turn trajectories to fly-to points. *Optimal Control Applications and Methods*, 11: 39–50, 1990.

[CMMR87] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal unctions of continuous variables with the "simulated annealing" algorithm. In *Proceedings of the ACM Transaction and Mathematical Software*. ACM, 1987.

[CS88] R.A Caruana and J.D Schaffer. Representation and hidden bias : Gray versus binary coding for genetic algorithms. In *Proceedings of the Fifth International Conference on Machine Learning*, 1988.

[DA93] Nicolas Durand and Jean-Marc Alliot. Existing algorithms for collision avoidance, and what the future might hold. Technical report, CENA, 1993.

[DA96] N. Durand and J.M. Alliot. Genetic algorithms for partially separable functions. In *Submitted to the Fourth International Conference on Parallel Problem Solving from Nature, PPSN96, Berlin*, 1996.

[DAAS94] Nicolas Durand, Nicolas Alech, Jean-Marc Alliot, and Marc Schoenauer. Genetic algorithms for optimal air traffic conflict resolution. In *Proceedings of the Second Singapore Conference on Intelligent Systems*. SPICIS, 1994.

[Daf72] S.C Dafermos. The traffic assignment problem for multiclass-user transportation networks. *Transportation Science*, 6: 73–87, 1972.

[DAM93] Patrick Dujardin, Jean-Marc Alliot, and Paul-Henri Mourlon. Different paths to automation. In *IFAC'93*, 1993.

[DAN94] Nicolas Durand, Jean-Marc Alliot, and Joseph Noailles. Algorithmes genetiques : un croisement pour les problemes partiellement separables. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 1994.

[DAN96a] N. Durand, J.M. Alliot, and J. Noailles. Collision avoidance using neural networks learned by genetic algorithms. In *Ninth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Fukuoka*, 1996.

[DAN96b] Nicolas Durand, Jean-Marc Alliot, and Joseph Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.

[DASF94a] D Delahaye, J.M Alliot, M Schoenauer, and J.L Farges. Genetic algorithms for partitioning airspace. In *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Application*. CAIA, 1994.

[DASF94b] D Delahaye, J.M Alliot, M Schoenauer, and J.L Farges. Genetic algorithms for air traffic assignment. In *Proceedings of the European Conference on Artificial Intelligence*. ECAI, 1994.

[DASF94c] Daniel Delahaye, Jean-Marc Alliot, Marc Schoenauer, and Jean-Loup Farges. Genetic algorithms for air traffic. In *Proceedings of the Conference on Artificial Intelligence Application*. CAIA, 1994.

[DASF95] D Delahaye, J.M Alliot, M Schoenauer, and J.L Farges. Genetic algorithms automatic regrouping of air traffic sectors. In *Proceedings of the Fourth International Conference on Evolutionary Programming*. Natural Selection inc., 1995.

[Dav82] Randall Davis. Expert Systems: Where Are We? And Where Do We Go From Here? *The AI Magazine*, Printemps 1982.

[Daw86] Richard Dawkins. *The blind watchmaker*. Norton, New-York, 1986.

[Daw89] Richard Dawkins. *L'horloger aveugle*. Robert Laffont, 1989. Edition orignale [Daw86].

[DCA96] N. Durand, O. Chansou, and J.M. Alliot. An optimizing conflict solver for atc. *ATC Quarterly*, 1996.

[DeJ75] K.A DeJong. *An Analysis of the Behavior of a Class of genetic Adaptative Systems*. PhD thesis, University of Michigan, 1975.

[Del95] Daniel Delahaye. *Optimisation de la sectorisation de l'espace aérien par algorithmes génétiques*. PhD thesis, ENSAE, 1995.

[DFMN95] G. Dean, X. Fron, W. Miller, and J.P. Nicolaon. Arc2000 : An investigation into the feasibility of automatic conflict. Technical report, Centre Expérimental Eurocontrol, 1995.

[DM92] D Dasgupta and D.R McGregor. A structured genetic algorithm. Technical Report IKBS-8-92, Dep. of Computer Science. University og Strathclyde, Clasgow. UK, 1992.

[Dre79] Hubert Dreyfus. *What computers can't do: the limits of artificial intelligence*. Harper and Row, 1979. ISBN: 0-06-090624-3.

[Dre84] Hubert Dreyfus. *Intelligence Artificielle, mythes et limites*. Flammarion, 1984. La traduction française correspond à la deuxième édition américaine [Dre79].

[Dre87] Hubert Dreyfus. Mind over machine. In Rainer Born, editor, *Artificial Intelligence, the case against*. Croom Helm, 1987. ISBN: 0-312-00439-7.

[DS83] J.E. Denis and R.B. Schnabel. *Numerical methods for unconstrained optimization and non linear equation*. Prentice Hall, 1983.

[DS89] K. A. DeJong and W. M. Spears. Using genetic algorithms to solve np-complete problems. In *Proceedings of the International Conference on Genetic Algorithms*, 1989.

[Dur96] Nicolas Durand. *Optimisation de trajectoires pour la résolution de conflits en route*. PhD thesis, INPT, 1996.

[Egl92] R.W Eglese. Simulated annealing : A tool for operational research. *European Journal of Operational Research*, 46: 271–973, 1992.

[EO83] Shinsuke Endoh and Amedeo R. Odoni. A Generalized Model for Predicting the Frequency of Air Conflicts. September 1983.

[Far96] Henri Farreny. *Recherche heuristiquement ordonnee – Algorithmes et propriete*. Masson, 1996.

[FG87] Henri Farreny and Malik Ghallab. *Éléments d'intelligence artificielle*. Hermès, 1987.

[Fle70] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13: 371, 1970.

[FMT93] Xavier Fron, Bernard Maudry, and Jean-Claude Tumelin. Arc 2000 : Automatic radar control. Technical report, Eurocontrol, 1993.

[Fog95] David B. Fogel. *Evolutionnary Computation*. IEEE Press, 1995.

[FOW66] L.J Fogel, A.J Owens, and M.J Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley and sons. NY, 1966.

[Fox90] Mark Fox. AI and Expert Systems : Myths, Legends and Facts. *IEEE Expert*, Fevrier 1990.

[FW83] M.I Freidlin and A.D Wentzell. *Random Perturbations of Dynamical Systems*. Springer-verlag, New-York, 1983.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. Pvm 3 user's guide and reference manual. Technical report, Oak Ridge National Laboratory, 1994.

[GGRG85] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Gucht. Genetic algorithms for the traveling salesman problem. In *1st International Conference on Genetic Algorithms and their Applications*, 1985.

[GL85] D. Goldberg and R. Lingle. Alleles, loci, and the travelling salesman problem. In *1st International Conference on Genetic Algorithms and their Applications*, 1985.

[Gol70] D. Goldfarb. A family of variable metric algorithms. *Mathematical Computations*, 24: 23–26, 1970.

[Gol89a] David Goldberg. *Genetic Algorithms*. Addison Wesley, 1989. ISBN: 0-201-15767-5.

[Gol89b] D.E Goldberg. Genetic algorithms and walsh functions. part 1 and 2. *Complex Systems*, 3: 129–171, 1989.

[Gol89c] D.E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading MA Addison Wesley, 1989.

[Gol91] D.E Goldberg. Real-coded genetic algorithms, virtual alphabets and blocking. *Complex Systems*, 5: 139–167, 1991.

[Gru92] H. Gruber. Comparaison de diverses méthodes d'intelligence artificielle pour la résolution de conflit en contrôle de trafic aérien. Rapport de stage, Centre d'Etudes de la Navigation Aérienne, 1992.

[GT82] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312, London and New York, 1982. Academic Press.

[Han92] E. Hansen. *Global optimization using interval analysis*. Dekker, New-York, 1992.

[HD94] Jin-Kao Hao and Raphael Dorne. Nouveaux opérateurs génétiques appliqués à sat. In *Proceedings de EA'94*. Cepadues, 1994.

[HGL93] A. Homaifar, S. Guan, and G. Liepins. A new genetic approach on the traveling salesman problem. In *Fifth International Conference on Genetic Algorithms*, July 1993.

[HN93] J. Horn and N. Nafpliotis. Multiobjective optimization using the nitched pareto genetic algorithm. Illigal Report 93005, University of Illinois at Urbana, 1993.

[Hol62] John Holland. Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machinery*, 3, 1962.

[Hue94] Denis Huet. Nouvelles méthodespour la résolution de problèmes SAT et CSP. Master's thesis, Ecole Nationale de l'Aviation Civile (ENAC), 1994.

[Ing89] L Ingber. Very fast simulated annealing. *J. Math. Comp. Modeling*, 12(8): 967–973, 1989.

[IR92a] L Ingber and B Rosen. Genetic algorithms and very fast simulated re-annealing. *Mathematical Computer Modeling*, 16(11): 87–100, 1992.

[IR92b] Lester Ingber and Bruce Rosen. Genetic algorithm and very fast simulated reannealing: a comparison. *Mathematical and Computer Modeling*, 16(1): 87–100, 1992.

[JKP72] S.L.S. Jacoby, J.S. Kowalik, and J.T. Pizzo. *Iterative methods for non linear optimization problems*. Prentice Hall, 1972.

[K+89] Fred Krella et al. Arc 2000 scenario (version 4.3). Technical report, Eurocontrol, April 1989.

[LA91] Marcel Leroux and Jean-Marc Alliot. En route air traffic organizer, an expert system for air traffic control. In *Proceedings of the International Conference on Expert systems and their applications*, Avignon, May 1991.

[LeF95] Yann LeFablec. Optimisation par algorithmes génétiques parallèles et multi-objectifs. Master's thesis, Ecole Nationale de l'Aviation Civile (ENAC), 1995.

[LM90] Kai-Fu Lee and Sanjoy Mahajan. The development of a world class othello program. *Artificial Intelligence*, 43, 1990.

[Mai91] G Maignan. *Le Contrôle de la Circulation Aérienne*. Presse Universitaire de France, 1991.

[MDA94] F. Medioni, Nicolas Durand, and J.M. Alliot. Algorithmes génétiques et programmation linéaire appliqués a la résolution de conflits aériens. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 1994.

[ME88] Magdi S. Mahmoud and Shawki Z. Eid. Optimization of freeway traffic control problems. *Optimal Control Applications and Methods*, 9: 37–49, 1988.

[MG92] S.W Mahfoud and D.E Goldberg. Parallel recombinative simulated annealing : A genetic algorithm. Illigal report 92002, University of Illinois, Urbana, IL 61801-2996, April 1992.

[MG94] Dr. C. Meckiff and Dr. P. Gibbs. PHARE : Highly interactive problem solver. Technical report, Eurocontrol, 1994.

[Mic92] Z Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs*. Springer-verlag, 1992.

[MJ91] Z Michalewicz and C.Z Janikov. Handling constraints in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithm*. ICGA, 1991.

[MM93] David Moriarty and Risto Miikulainen. Evolving complex othello startegies using marker-based genetic encoding of neural networks. Technical Report AI93-206, University of Texas, Austin, TX 78712-1188, September 1993.

[MM94] David Moriarty and Risto Miikulainen. Evolving neural networks to focus minimax search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence AAAI-94*, Seattle, WA, 1994.

[Muh89] H. Muhlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[MZ88] L. Mikhailov and J. Zaprianov. Multilevel hierarchical optimization of aircraft trajectories. In *IFAC/IMACS Symposium*, July 1988.

[NFC+83] W.P. Niedringhaus, I. Frolow, J.C. Corbin, A.H. Gisch, N.J. Taber, and F.H. Leiber. Automated En Route Air Traffic Control Algorithmic Specifications: Flight Plan Conflict Probe. Technical report, FAA, 1983. DOT/FAA/ES-83/6.

[Nie89a] W.P. Niedringhaus. Automated planning function for AERA3: Manoeuver Option Manager. Technical report, FAA, 1989. DOT/FAA/DS-89/21.

[Nie89b] W.P. Niedringhaus. A mathematical formulation for planning automated aircraft separation for AERA3. Technical report, FAA, 1989. DOT/FAA/DS-89/20.

[NM65] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7: 308–313, 1965.

[OSH89] I. Oliver, D. Smith, and J. Holland. Permutation crossover operators on the travelling salesman problem. In *Second International Conference on Genetic Algorithms*, July 1989.

[PA91] Tekla S. Perry and John A. Adam. Improving the world's largest, most advanced system! *IEEE Spectrum*, February 1991.

[Pea84] Judea Pearl. *Heuristics*. Addison-Wesley, 1984. ISBN: 0-201-05594-5.

[Pea90] Judea Pearl. *Heuristique*. Cepadues, 1990. Version française de [Pea84].

[Pla93] Pascal Planchon. Use of advanced technologies in ATM domain. In *AGARD conference*, Berlin, 1993. DLR.

[PLG87] C. Pettey, M. Leuze, and J. Grefenstette. A parallel genetic algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, 1987.

[Rei91] G. Reinelt. Tsplib - a traveling salesman problem library. *Journal on Computing 3*, pages 376–384, 1991.

[RR95] Helmut Ratschek and Jon Rokne. Interval methods. In Reiner Horst and Panos Pardalos, editors, *Handbook of global Optimization*. Kluwer, 1995.

[Sap90] G Saporta. *Probabilités, analyse des données et statistiques*. Technip, 1990.

[Sch90] Robert L. Schultz. Three-dimensional trajectory optimization for aircraft. *Journal of Guidance, Control and Dynamics*, 13(6): 936–943, 1990.

[SG91] Nicol N. Schraudolf and John J. Grefenstette. A user's guide to GAucsd 1.2. Technical report, UCSD, 1991.

[SG94] Robert Smith and Brian Gray. Co-adaptive genetic algorithms, an example in othello strategy. In *Proceedings of the Florida Artifical Intelligence Symposium*, 1994.

[SGE91] R.E Smith, D.E Goldberg, and J.A Earickson. *SGA-C: A C-language implementation of a Simple Genetic Algorithm*, May 1991. TCGA report No. 91002.

[SPF93] R.E. Smith, A.S. Perelson, and S. Forrest. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2): 127–149, 1993.

[SPSS83] E. M. Schuster, F. R. Petroski, R. K. Sciambi, and M. MC Stokrp. AERA 2 functional design and performance description. Technical report, MITRE, September 1983. MtR-83W136.

[SRD93] M Schoenauer, E Ronald, and S Damour. Evolving nets for control. In *Proceeding of the Sixth International Conference on Neural Networks and their Industrial and Cognitive Applications*. AFIA, 1993.

[TPC76] P.L Tuan, H.S Procter, and G.J Couluris. Advanced productivity analysis methods for air traffic control operation. Technical report, Stanford Research institute, Menlo Park CA 94025, December 1976.

[Tro93] A. Trouvé. *Parallélisation massive du recuit simulé*. PhD thesis, Université de Paris XI, 1993.

[Vil84] J Villiers. Contribution à une théorie du système de contrôle en route et ses perspectives d'évolution. Technical report, DGAC, Paris France, Aout 1984.

[Vos91] M.D Vose. Generalizing the notion of schema in genetic algorithms. *Artificial Intelligence*, 50: 385–396, 1991.

[Wri91] A.H Wright. Genetic algorithms for real parameter optimization. In *Proceeding of the Foundation Of Genetic Algorithms*. FOGA, 1991.

[WSF89] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In *Third International Conference on Genetic Algorithms*, 1989.

[YG93] X Yin and N Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In *Proceedings of the Artificial Neural Nets and Genetic Algorithms*, 1993.

[Zeg93] Karim Zeghal. Champs de forces symétriques : La logique d'un système anticollision coordonné. Technical report, ONERA, 1993.

[Zeg94] Karim Zeghal. *Vers une théorie de la coordination d'actions, application à la navigation aérienne*. PhD thesis, Universite Paris VI, 1994.

[Zhi91] Anatoly A. Zhigljavsky. *Theory of Global Random Search*. Kluwer Academic Plubishers, 1991.