Genetic algorithms for partitioning air space

Daniel Delahaye – Jean-Marc Alliot – Marc Schoenauer

Abstract

In this paper, we show how genetic algorithms can be used to compute automatically a balanced sectoring of airspace to increase Air Traffic Control capacity in high density areas.

1 Introduction

The CENA is the organism in charge of studies and research for improving the French ATC systems. Studies on the use of genetic algorithms for conflict resolution have given encouraging results [2], and a new study has been funded to solve the load balancing problem for Air Traffic Control. When joining two airports, an aircraft must follow routes and beacons ; these beacons are necessary for pilots to know their position during navigation and because of the small number of beacons on the ground they often represent crossing points of different airways.

Crossing points may generate conflicts between aircraft when their trajectories converge on it at the same time and induce a risk of collision.

At the dawn of civil aviation, pilots resolved conflicts themselves because they always flew in good weather conditions (good visibility) with low speed aircraft. On the other hand, modern jet aircraft do not enable pilots to resolve conflicts because of their high speed and their ability to fly with bad visibility. Therefore, pilots must be helped by an air traffic controller on the ground who has a global view of the current traffic distribution in the airspace and can give orders to the pilots to avoid collisions.

As there are lot of planes simultaneously present in the sky, a single controller is not able to manage all of them. In France, airspace is partitioned into different sectors, each of them being assigned to a controller.

Sectoring is currently done in an empirical way by some airspace experts who apply rules they have learned with experience. Actually, the sectoring modifications are usually due to traffic evolution on long period and when a sector is regularly overloaded it has to be modified. To reach this aim, an ad hoc commission meets to elaborate new frontiers for the sectors in order to balance the workload. Afterward sectoring is updated (until new problems arise).

This way of making is relevant because it takes into account a lot of practical aspects but has a limited effect on the local zone it treats. One can try to improve and complete this process with an automatic approach in order to give a solution to the sectoring problem in the whole airspace and that could be refined by experts.

This first automatic approach is very new and still may be improved but it gives very encouraging results.

In this paper we show how well Genetic Algorithms manage this problem (after some simplifications and modeling). In the first part we describe more precisely our problem and make some relevant simplification to develop a mathematical model. In the second part we present complete examples of resolution with the different Genetic Algorithms.

2 A simplified model

2.1 Introduction

Before doing a mathematical description of our problem, it is necessary to stress out our framework to introduce some simplifications for our model. Since it is very long to train an air traffic controller on his sector (from 3 to 4 months), we must not investigate a real time sectoring optimization according to the variations of the traffic load. Instead we have to consider a registered maximum load traffic period on the working network. Our problem is then to partition the air space to get a balanced induced control workload.

When examining the physical air traffic network, we notice that airways are superposition of several routes which have the same projection on the floor but different altitudes according to their azimuth (semi circular rule[1]). So an airway can be modeled by a bidirectional link which gathers several individual aircraft route (see figure 1).

Then, our 3 dimensional transportation network will be modeled by a classical 2 dimensions network on a horizontal plan. This modeling is not far from reality because of the small number of routes superposed on the same link (each link roughly contains 4 routes).

When aircraft go from point A to point B, they have to use airways of the air traffic network like drivers do on the road network. As on a road, there are crossing in the sky and aircraft have to safely pass these points. Because of their speed, aircraft can not make anti-collision procedure alone and are helped by controllers who solve the different conflicts that may arise.

But nowadays, there are too many flights in the airspace (for instance an average of 6000 aircraft movements is registered everyday in the French airspace) for a single controller to manage all

[1] Aircraft with heading between 0 and 180 have to fly with odd altitude (in hundred of feet) and even altitude for headings between 180 and 360

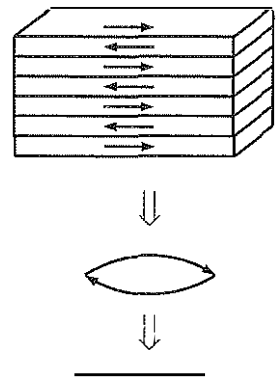

Figure 1 : Airways modeling

this traffic; thus airspace is divided into several sectors, each sector being assigned to a controller. As any human being, a controller has working limits and a sector is said to be saturated when there are 3 conflicts to solve and 15 aircraft in it (this rate must not last too long).

The controller workload has several origins that can be divided into two categories:

1. there are quantitative factors which include the number of flights, the number of conflicts etc..., which can be precisely modeled in a mathematical way and handled by an optimization algorithm;
2. there are psychological factors as stress, concentration etc... which have no evident mathematical formulation but are in direct relationship with the previous ones according to the controllers themselves.

So, we will only take into account quantitative elements in our application on first approximation.

Having now a model, we can define more precisely our goals in the following way:

> one considers an air traffic transportation network in a 2 dimensional space with flows on it inducing a workload distributed over the space. This workload must be partitioned into K equilibrated convex sectors in a way that minimizes coordinations[2].

Figure 2 shows an example of network sectoring with 6 sectors.

This sectoring must take some constraints into account coming from Air Traffic Control system :

[2] When an aircraft crosses a sector frontier, controllers in charge of those sectors have to exchange informations about the flight inducing a workload called coordination

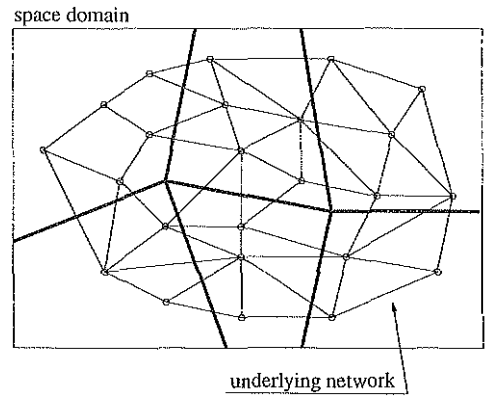

Figure 2 : Example of sectoring

- a pilot must not encounter twice the same controller during his flight to prevent useless coordinations ; this means that an aircraft crossing a sector will encounter 2 and only 2 sector frontiers. To guarantee that our sectors met this constraint we force them to be convex in the topological sense[3]. This constraint gives sectors a polygonal shape.

- a sector frontier has to be at least at a given distance from each network node (security constraint). As a matter of fact, when a controller has to solve a conflict, he needs a minimum amount of time to elaborate a solution. Each controller managing individually his sector, if a sector frontier is too close to a crossing point, he is not able to solve any conflicts because he has not enough time between the coordination step (with the previous sector where the aircraft comes from) and the time the aircraft reaches the crossing point. The minimum delay time is fixed at 7 minutes and can be converted into a distance knowing the aircraft speed.

- an aircraft has to stay at least a given amount of time (a few minutes) in each sector it crosses to give enough time to the controller to manage the flight in good conditions (min stay time constraint). We express this constraint by a minimum distance between two frontiers cutting the same network link.

The last two constraints will be implemented the same way by forcing a minimum length for any link segment between two consecutive frontiers or between a node and a frontier.

[3] this kind of convexity is stronger than the one imposed by our problem (our sectors have to be convex according to the direction of the links of the network and not in all directions) but is easier to implement

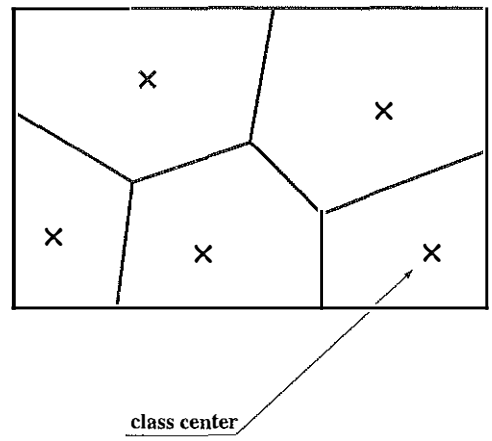

Figure 3 : Example of a 5-partitioning

2.2 Mathematical formulation

2.2.1 The transportation network

We define our transportation network as a doublet (N, L) in which N is the set of nodes (with their positions in a topological space) and L is the set of links each of them transporting a quantity f_{ij} of flow from node i to node j [8].

2.2.2 Construction of sectors

According to the previous section, the sectors we have to build have to be convex (with a polygonal shape induced by the convexity property). To reach this goal we use a Forgy aggregation method [11] coming from dynamic clustering in exploratory statistics which aims at extracting clusters from a set of points randomly distributed in a topological space (see [4, 11]). This method randomly throws K points (the class centers) in the space domain containing the transportation network and aggregates all the domain points to their nearest class center. This method ends up in a K partitioning of our domain into convex sectors with linear frontiers. Figure 3 gives an example of a 5-partitioning of a rectangle.

2.2.3 Workload induced in a control sector

As said before, we just take quantitative criterions into account to compute controller workload (see [13]). According to the controllers themselves, workload can be divided into three parts which correspond respectively to the conflict workload, the coordination workload and the trajectories monitoring workload of the different aircraft which are present in a sector :

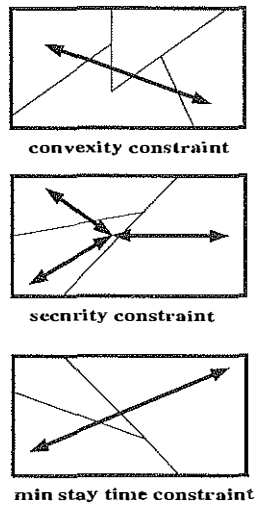

Figure 4 : Constraints examples

- the conflict workload gathers the different actions of the controller to solve conflicts.

- the coordination workload corresponds to the information exchanges between a controller and the controller in charge of the bordering sector or between a controller and the pilots when an aircraft crosses a sector frontier.

- the monitoring aims at checking the different trajectories of the aircraft present in a sector and induces a workload.

2.2.4 Constraints

We handle the different constraints previously introduced in the following way:

sectors convexity: this constraint is already satisfied by the construction method of sectors.

security and Min stay time constraints: those two constraints can be synthesized by an artificial increase of the coordination work load on links.

On the figure 4 we give some examples where the 3 previous constraints are not satisfied.

3 Principle of resolution

3.1 Complexity of our problem

The problem we have to solve can be divided in two different parts corresponding to our two different goals:

1. equilibrium of the different sectors workload according to the number of aircraft and conflicts in each sector;
2. minimization of the coordination workload.

The second criterium is typically a discrete graph partitioning problem with topological constraints and then is NP_HARD [5]. Having chosen a continuous flow representation, the first criterium induces a discrete-continuous problem which is also NP_HARD.

So, according to the size of our network (about 1000 nodes), classical combinatorial optimization is not relevant and stochastic optimization seems to be more suitable.

Moreover this kind of problem may have several optimal solutions (or near optimal) due to the different possible symmetries in the topological space etc..., and one must be able to find all of them because they have to be refined by experts and we do not know at this step which one is really the best. This last point makes us reject classical simulated reannealing optimization which updates only one state variable, even if it might give better results in some cases [7].

On the other hand, Genetic Algorithms (GAs) maintain and improve a numerous population of states variables according to their fitness and will be able to find several optimal (or near optimal) solutions. Then, GAs seem to be relevant to solve our sectoring problem.

3.2 Different kinds of GAs we tried

3.2.1 Generals steps of GAs

First, we generate a population of sectoring, each represented by a set of class centers which are points in our 2-dimensional space. Then each chromosome defines one and only one sectoring. Afterward we select[4] a new population according to the different fitness and apply classical operators as crossover, mutation etc...

Figure 5 gives an example of one GA iteration with 5 sectors.

[4]Selection aims at reproducing better individual according to their fitness. We tried two kinds of selection process, Roulette Wheel Selection" and "Stochastic Remainder Without Replacement Selection", the last one always gave better results.

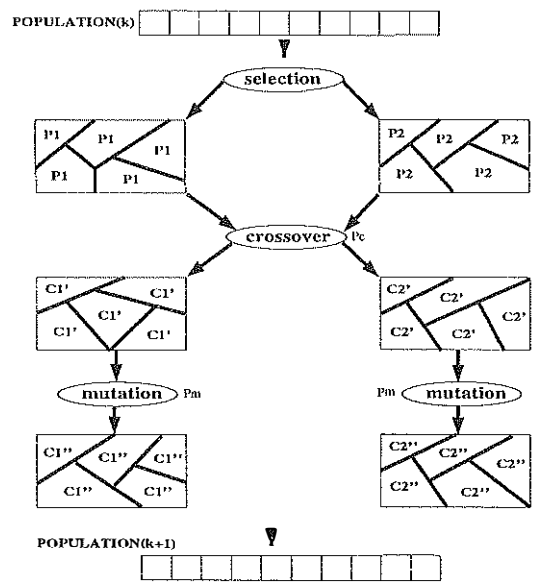

Figure 5 : A GA iteration example, Pc probability of crossover, Pm probability of mutation

3.2.2 GA with binary chromosome

A chromosome must containt all the sectoring information for the GA to be able to evaluate the fitness for each individual. We summerize this information by a set of points in our geographical space called class centers (one can show that for each class center set there is just one sectoring induced see [11])). Having chosen binary strings in this first example, we implement chromosome as a string of bits containing the concatenation of the different class center positions (see figure 6).

This implementation enables us to use classical operators for GAs.

3.2.3 GA with floating point chromosome

In this case, each position (normalized into [0,1]) is directly coded in a chromosome without binary conversion (see [10]). So, the chromosome has the structure showed in figure 7.

This new structure involve some new kind of operators we now describe.

Crossover After selecting two parents in the current population, we randomly chose an allele position (so we select two sectors at the same allele position, one in each sectoring). Afterward, we join by a straight line the associated class centers.Then, we move the class centers on this line according to a uniform random variable. An example of this kind of crossover is given in figure 8 (allele 1 has been selected in this example).

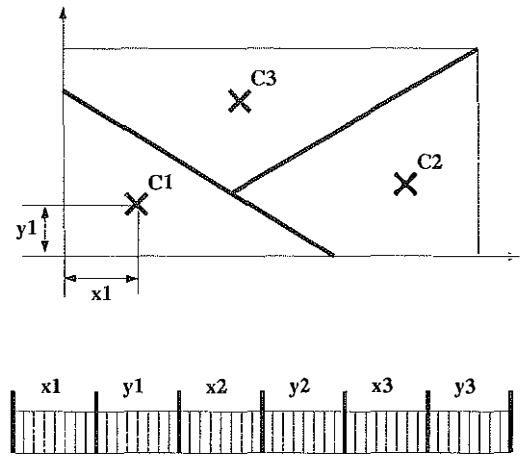

Figure 6 : Construction of binary chromosome

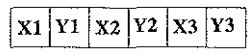

Figure 7 : Floating point chromosome

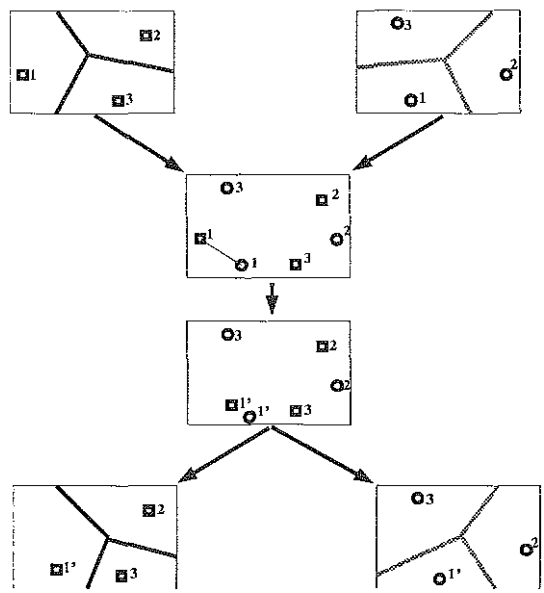

Figure 8 : Example of crossover

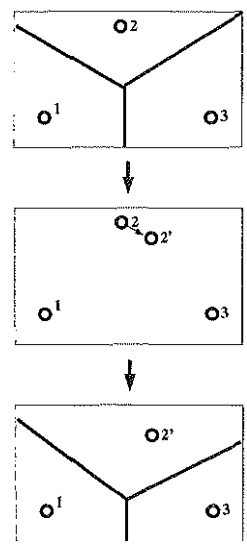

Figure 9 : Example of mutation

Mutation When we mutate a chromosome we randomly select an allele position and we move its associated class center by adding to it a noise[5] (see figure 9 (in this example allele 2 has been selected for mutation)).

The structure of this new GA is exactly the same as the binary GA in the succession of the different steps.

3.3 Evolution GA

This last version of GA is a dynamic variant of the previous one with some analogy with simulated annealing. As for the previous one we select parents from the current population and make them cross and mutate to create a new population.

Crossover After selecting one class center in each parent chromosome (at the same allele position) we move each one in a random way with progressive decreasing range as the generation number increases.

This kind of crossover process induces large moving at the beginning (→ quasi randomly exploration of the state domain) and very small ones at the end (→ these small moving enable the algorithm to "climb hills").

[5] it seems that best results are given with an affine distribution and not with a Gaussian

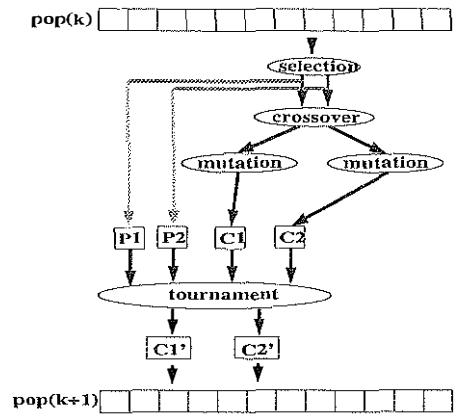

Figure 10 : Evolution GA

Mutation This operator randomly moves one class center in a chromosome with the same law as for crossover. According to the range law the initials moving are very important so space is explored in a quasi randomly way and the moving become smaller as the generation number increases. The objective evaluations are refined as in an "hill climbing" process in which the climbing direction is given by the selection process.

After applying those two operators, we have four individuals (two parents P1,P2 and two children C1,C2) with their respective fitness. Afterward, those four individuals compete in a tournament. The two winners are then inserted in the next generation. The selection process of those winners is the following.

If C1 is better than P1 then C1 is selected. Else C1 will be selected according to a probability which decrease with the generation number.

Then, at the beginning, C1 has a probability 0.5 to be selected although it is worse than P1 and this probability decrease to 0.01 at the end of the process.

A description of this algorithm is given on figure 10

4 Results

4.1 Binary GA

Those evaluations were done with the classical SGA [6, 12] of Goldberg and give good results on very small networks.

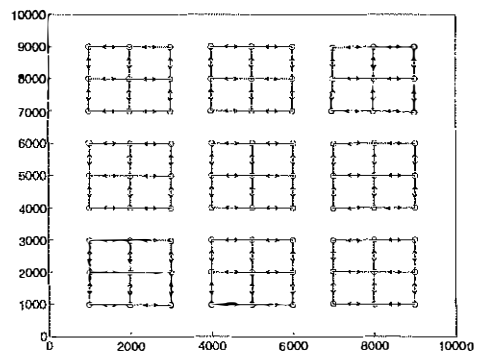

Figure 11 : Square network

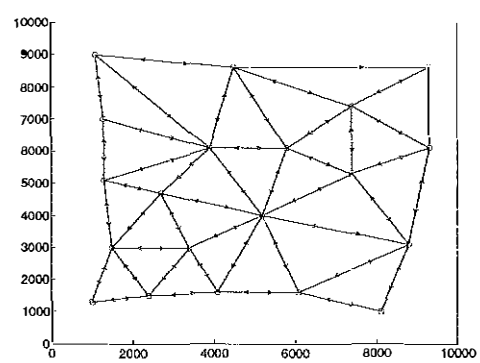

Figure 12 : asymmetric network

When the network size increases, this algorithm becomes inefficient because of the crossover and mutation operators which induce a quasi random moving in the state space. This is due to the fact that those operators do not take into account the space point position in the chromosome and break it very roughly. This last point made us change the structure of our chromosome into a floating string where the crossing position respects each individual floating allele.

4.2 Floating GA (FGA) and Evolution GA (EGA)

The previous algorithm being too limited, we tried and compare two floating point GA (F-GA,EGA). The results of those two algorithms are very encouraging as shown by the following experiments results.

To compare and evaluate these algorithms, we have used an artificial test network (see figure 11 and figure 12)

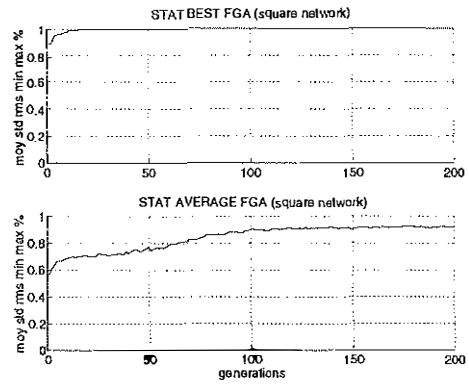

Figure 13 : FGA stat results for the square network

As we can see the first network has trivial solutions with 9 sectors. These solutions seems to be very evident for a human being because of the brain perception ability to investigate the different symmetries but for a computer these problems have no particularity and remain difficult.

The different parameters we have chosen for our experiments are the following :

population size: 400

number of generations: 200

probability of crossover: 0.6

probability of mutation: 0.06

4.2.1 Convergence

To see the convergence of our algorithms we observed the evolution of the population statistics (max and average) over the generations. As we can see on figure 11, the "square network" can be partitioned into 9 sectors in several ways and will be easier to manage as it has not a unique solution but a solution set. Both algorithms find an exact solution very quickly (15 generations for FGA and 8 for EGA, see figure 13 and figure 14), butEGA gives better results on the average stat.

After this first experiment we tried to partition an asymmetric network into 5 sectors (see figure 12). This network has no trivial solutions (our future real air network neither has) and it seems evident that the solution space is much smaller than for the "square network". This last point induces a slower convergence rate (as we can see on figures 15 and 16) with no exact solutions.

According to the balance error results, the given solutions are very close to our objective and the most unbalanced sectors is less than 0.7% distant from the objective.

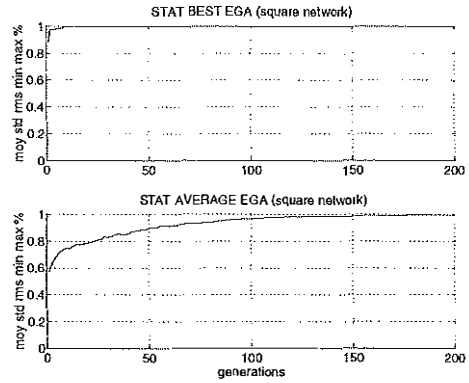

Figure 14 : EGA stat results for the square network

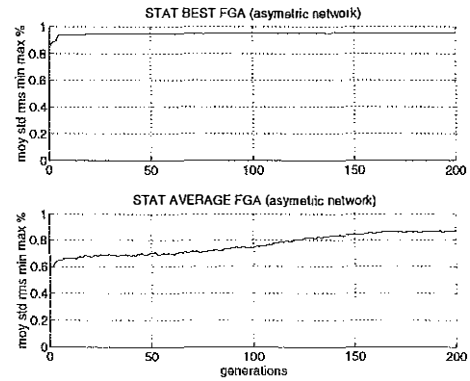

Figure 15 : FGA stat results for the asymmetric network

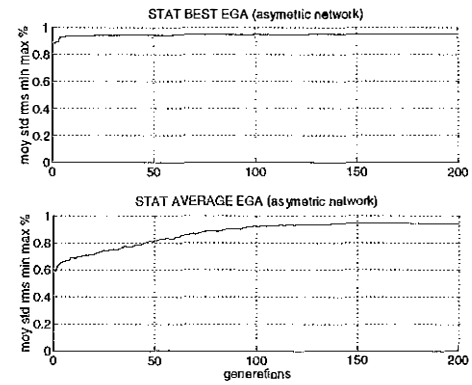

Figure 16 : EGA stat results for the asymmetric network

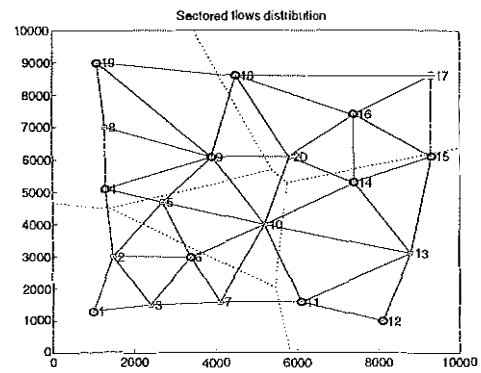

Figure 17 : Physical sectoring result

The physical sectoring result satisfies the topological constraints as we can see an figure 17

5 Conclusion

This study showed us how Genetic Algorithm were suitable to solve the space sectoring problem with very special constraints. To reach this aim we had to extend the chromosome concept to floating strings so that operators do not break the chromosome structure roughly. This modification really improved the algorithm performances regarding the resolution speed and the result accuracy. Afterward we added a tournament operator and used dynamic parameters to improve the space exploration as well as the selection process. This last change brought good improvements to the algorithm convergence rate. Like in every Genetic Algorithm, the key of success lies in the modeling and the operators. Actually, both must be as close as possible to the application problem. In our case, the representation seems to be very close to the physical application but operators can still be improved, though the ones we have used gave very good results.

One possibility to improve this algorithm would be to reinforce the Simulated Annealing concept used in the different operators as Goldberg does in his PRSA [9] algorithm (with binary chromosome). This brings in fact some convergence theorems coming from the Simulated Annealing theory. On the other hand, as for Simulated Annealing, the (stochastic) convergence is ensure only when the fitness probability distribution law is stationary in each state point [1]. We would then have the same drawbacks when this hypothesis is not satisfied.

Finally it would be very interesting to try different acceptance probability laws and different moving probability laws to change the way of exploring the space in our last algorithm.

In this first study, we assumed that traffic flows were correctly assigned (traffic assignment aims at distributing flows over a network to optimize an objective) before the algorithm partitioned the work load spread over space. But after partitioning air space, we are not sure that traffic assignment is still optimal due to the fact that traffic assignment and sectoring interact with each other. The next step of this study is precisely to mix these two optimization problems in a single genetic algorithm which will maximize a global objective function.

Bibliographie

[1] Emile Aarts and Jan Korst. *Simulated annealing and Boltzmann machines.* Wiley and sons, 1989. ISBN: 0-471-92146-7.

[2] Jean-Marc Alliot, Hervé Gruber, and Marc Schoenauer. Using genetic algorithms for solving ATC conflicts. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence Application.* IEEE, 1993.

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

[4] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier, and H. Ralambrondrainy. *Classification automatique des données: environnement statistique et informatique.* Dunod, 1989.

[5] Chung-Kuan Cheng. The optimal partitioning of networks. *Networks*, 22:297–315, 1992.

[6] David Goldberg. *Genetic Algorithms.* Addison Wesley, 1989. ISBN: 0-201-15767-5.

[7] Lester Ingber and Bruce Rosen. Genetic algorithm and very fast simulated reannealing: a comparison. *Mathematical and Computer Modeling*, 16(1):87–100, 1992.

[8] D. Klingman and J. M. Mulvey, editors. *Networks models and associated applications.* North-Holland, 1981. ISBN: 0-444-86203-X.

[9] Samir W. Mahfoud and David E. Goldberg. Parallel recombinative simulated annealing: a genetic algorithm. IlliGAL Report 92002, University of Illinois at Urbana-Champaign, 104 South Mathews Avenue Urbana IL 61801, April 1992.

[10] Zbigniew Michalewicz. *Genetic algorithms+data structures=evolution programs.* Springer-Verlag, 1992. ISBN: 0-387-55387-.

[11] Gilbert Saporta. *Probabilités, analyse des données et statistique.* Technip, 1990.

[12] Robert E. Smith, David E. Goldberg, and Jeff A. Earickson. *SGA-C: A C-language implementation of a Simple Genetic Algorithm*, May 1991. TCGA report No. 91002.

[13] P. L. Tuan, H. S. Procter, and G. J. Couluris. Advanced productivity analysis methods for air traffic control operations. FAA Report RD-76-164, Stanford Research Institute, Menlo Park CA 94025, December 1976.