Genetic Algorithms for Air Traffic Control Systems

Daniel Delahaye Nicolas Durand Jean-Marc Alliot Marc Schoenauer

CENA* CENA† ENAC‡ CMAPX§

Abstract

The Air Traffic Control system of a country manages all the aircraft that fly in its airspace, designs control sectors, manages the flows between the different airports and beacons, ensures separation between aircraft during their flight, take off and landing. Thus, it operates at different levels, each one of them designed to provide control, ensure safety, and limit the traffic passed to the following level. In this paper we show how Genetic Algorithms can improve some of the tasks manually done by the ATC system. After a brief description of GAs, some of the improvements used (simulated annealing, sharing), we study three applications of GAs to ATC. We first show an application of GAs to en-route conflict resolution. Then we give an example on GAs used to optimize air space sectoring. The last part gives an application of GAs to traffic assignment.

1 Introduction

The Air Traffic Control (ATC) system of a country operates at five different levels :

1. Airspace design (airways, control sectors, ...). When joining two airports, an aircraft must follow routes and beacons; these beacons are necessary for pilots to know their position during navigation and help controllers to visualise the traffic. As there are many aircraft simultaneously present in the sky, a single controller is not able to manage all of them. So, airspace is partitioned into different sectors, each of them being assigned to a controller. This task aims at designing the air network and the associated sectoring.
2. Air Traffic Flow Management (ATFM) (strategic planning, a few hours ahead). With the increasing traffic, many pilots choose the same routes, generating many conflicts on the beacons inducing overloaded sectors. Traffic assignment aims at changing aircraft routes to reduce sector congestions, conflicts and coordinations.
3. Coordination planning (a few ten minutes ahead). This task guarantees that new aircraft entering sectors do not overload the sector.
4. Classical control in ATC centers (up to 20 mn ahead). At this level, controllers solve conflicts between aircraft.
5. Collision avoidance systems (a few minutes ahead). This level is activated only when the previous one has failed.

Each level has to limit and organize the traffic it passes to the next level, so that this one will never be overloaded.

However, the ATC systems in Europe and in the US have now reached their limits, although the reasons are different in these two regions. Therefore, one can reasonably expect that optimizing the system will provide substantial benefits in terms of safety, capacity, and induced costs. In this paper we show how Genetic Algorithms can bring new solutions to improve some of the tasks previously described. The first part shortly describes general principles of GA and introduces some improvements that we found useful. In the second part, we will give a description of the three problems we solve, the coding we choose and the results obtained. This project is performed by the Centre d'Etudes de la Navigation Aerienne (institute in charge of studies and research for improving the french ATC systems) and the Ecole Nationale de l'Aviation Civile. The Ecole Nationale Superieure d'Electronique, d'Electrotechnique, d'Informatique et d'Hydraulique de Toulouse and the Centre de Mathematiques Appliquees de l'Ecole Polytechnique collaborated to this project.

*Centre d'Etudes de la Navigation Aérienne

†Ecole Nationale Supérieure d'Electronique, d'Electrotechnique, d'Informatique et d'Hydraulique de Toulouse

‡Institut de Recherche en Informatique de Toulouse

§Centre de Mathématiques appliquées de l'X

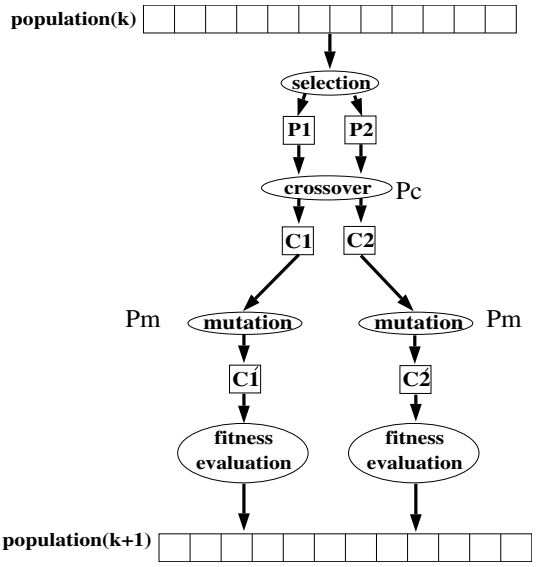

Figure 1: GA principle

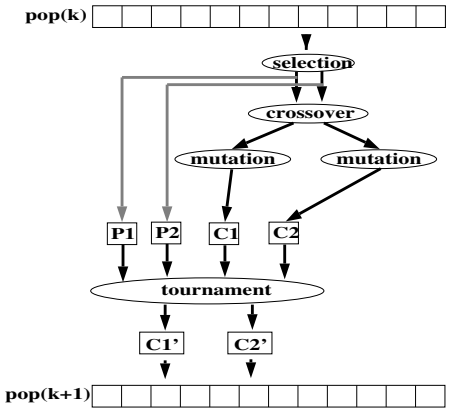

Figure 2: GA and SA mixed up

2 Genetic Algorithms

2.1 Principles

We are using classical Genetic Algorithms and Evolutionary Computation principles such as described in the literature [Gol89, Mic92]; Figure 1 describe the main steps of GAs.

First a population of points in the state space is randomly generated. Then, we compute for each population element the value of the function to optimise, which is called *fitness*. In a second step we select[1] the best individuals in the population according to their fitness. Afterwards, we randomly apply classical operators of crossover and mutation to diversify the population (they are applied with respective probabilities P_c and P_m). At this step a new population has been created and we apply the process again in an iterative way.

2.2 Improvements

2.2.1 Simulated Annealing Tournament

GA can be improved by including a Simulated Annealing process after applying the operators [MG92]. For example, after applying the crossover operator, 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 the winners is the following : if $C1$ is better than $P1$ then $C1$ is selected. Else $C1$ will be selected according to a probability which decreases with the generation number. At the beginning of the simulation, $C1$ has a probability of 0.5 to be selected even if its fitness is worse than the fitness of $P1$ and this probability decreases to 0.01 at the end of the process. A description of this algorithm is given on figure 2.

Tournament selection brings some convergence theorems from the Simulated Annealing theory. On the other hand, as for Simulated Annealing, the (stochastic) convergence is ensured only when the fitness probability distribution law is stationary in each state point [AK89].

2.2.2 Sharing

Most of our problem are very combinatorial and may have many different optimal solutions. In order to find most of these solutions and to avoid local optima, sharing process [YG93] was introduced. Thus, in each problem, we had to define a distance between two chromosomes. This sharing process has the great advantage to grow in $n \log(n)$ if n is the size of the population. Results show that sharing was very useful for combinatorial problems.

Simulated annealing and sharing process have really improved convergence of GAs and were definitely adopted for the three following applications.

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

3 Conflict resolution problem

3.1 Description and Complexity of our problems

Inside a sector, we want to give aircraft conflict free trajectories as close as possible to optimal trajectories. An aircraft is said to be conflict free when it is distant from the other aircraft of a separation norm d at each point of its trajectory.[2].

This conflict resolution problem must respect the following constraints :

1. trajectories must respect both aircraft and pilot performances. Considering the evolution of ATC toward automation, trajectories must remain simple for controllers to describe as well as for pilots to understand and follow. Conflicts will be solved, if possible without changing the aircraft flight levels for comfort and economical reasons.

2. conflict free trajectories must be as close as possible to optimal trajectories. They must remain simple enough to allow conflicts involving many aircraft to be computed in a real time situation. Our main goal is to find out the global optimum and not only a suitable solution.

Optimal Command Theory with State Constraints developed by Bryson and Ho [BH75], supplementary conditions exposed by Kreindler [Kre82], Bryson, Denham and Dreyfus [BDD63] leads to the following conclusions : at the optimum, as long as the norm separation constraint is not saturated, aircraft fly in straight line. When saturating the constraint, aircraft start turning, and as soon as the separation constraint is freed aircraft fly straight again. This leads us to adopt a turning point modelling : a mathematical study of our problem [Dur96] shows that conflict free trajectories can be approximated by turning point trajectories (figure 3).

For a conflict involving two aircraft, when only one aircraft turns, the turning point approximation lengthens the optimal trajectory for less than 1% if distance between the aircraft and the conflict point is greater than two separation norms and the angle of incidence between trajectories is greater than 30 degrees. We can also prove that the offset modelling (see figure 4), which moves an aircraft to put it on a parallel route, is worse. It has the only advantage to

[2]We define two different separation norms in the vertical dimension (2000 feet) and in the horizontal plan (8 nautic miles)

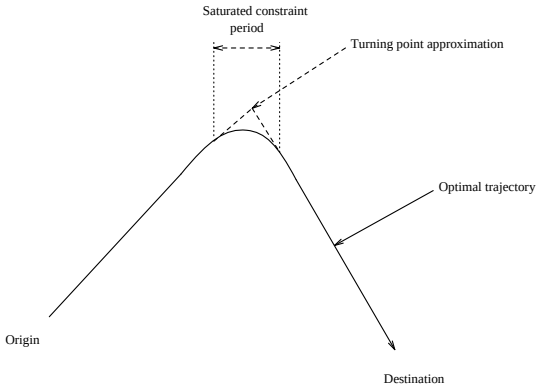

Figure 3: Turning point approximation.

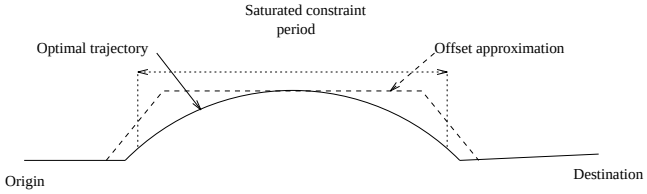

Figure 4: Offset approximation.

linearize the separation constraints. The offset is thus very easy to calculate, but separation constraints must be checked during manœuvers and the complexity of the problem remains. Moreover, the offset modelling requires one more manœuver for the pilot. With the turning point modelling, the whole information of a trajectory can thus be described by the coordinates of the turning point. This modelling reduces to $\Re^2$ the space solution set, which was a set of functions defined over a time interval. The third dimension extension comes from the aircraft piloting constraints. When changing an aircraft level, we can not choose its climbing rate or descending rate for technical reasons. We can olny choose the beginning of vertical evolution, the level reached and the length of the vertical offset. For practical reasons, we may not change the flight level of an aircraft already turned off or turn off an aircraft changing its level. The performance model used by the french ATC gives us the flying parameters for many different types of aircraft.

If we can easily prove that the minimised function is convex, the set of conflict free trajectories is not. It is not even connected. For a conflict involving two aircraft, the set of conflict free trajectories has two connected components. For a conflict involving n aircraft there may be 2^n connected components in the free trajectory space which proves that an algorithm solving 2^n optimum searches is clearly exponential. It

is important to note that this complexity is independent of the modelling chosen. The offset modelling seems to be very attractive, because it linearizes constraints. Nevertheless, each constraint multiplies by two the number of linear programs to solve. Our problem involves $\frac{n(n-1)}{2}$ constraints. Moreover, linearizing the minimised function, multiplies by 2^n the number of linear program to solve (we minimise the sum of each aircraft offset which may be positive or negative). Finally, we will have to solve $2^{\frac{n(n+1)}{2}}$ linear programs, each one involving $\frac{n(n+1)}{2}$ linear constraints. For $n = 5$, we have 32768 linear programs to solve and 15 constraints in each program (this problem is studied in [DMA94] and [Med94]).

Using classical methods, such as gradient methods for example, is useless for our problem, because of the arbitrary choice of the starting point required by these methods. Each connected component may contain one or several local optima, and we can easily understand that the choice of the starting point in one of these components cannot lead by a classical method to an optimum in another component. We can thus expect only a local optimum. Practical attempts done on LANCELOT (*Large And Nonlinear Constrained Extended Lagrangian Optimisation Techniques*) [CGT92] have confirmed this problem and high-lighted others. Convergence is very slow, particularly when we take the speed constraint into account. Solving a five aircraft conflict without vertical offset takes a week on a Sun workstation. This approach was abandoned. The combinatory induced by the offset modelling is so important that we cannot expect to find the global optimum efficiently. Moreover, the separation during manœuvers must be checked afterwards. For these reasons, classical methods were abandoned for GAs, particularly efficient to solve combinatorial problems

3.2 Coding

Our coding contains both floating and discrete value. An example of chromosome is given in figure 5. The parameters are initialised as follows :

- Vertical Evolution
  This parameter can be GOES_DOWN, STAYS or GOES_UP. The value is set at random.

- Heading and Time (for an aircraft whose vertical evolution is STAYS)
  We have the initial heading of the aircraft, we set a new one using a random number generator. We add to the initial value the maximum turning angle multiplied by a random number. The time

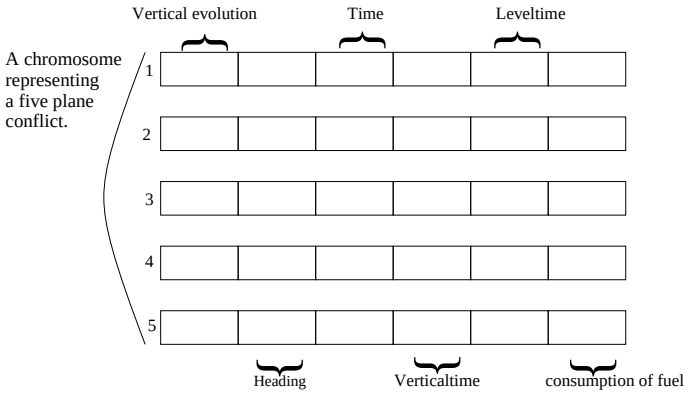

Figure 5: Structure of the chromosome

parameter defines the time the aircraft will fly following the previously specified heading.Then the aircraft will turn and head to its Destination point.

- Verticaltime and Leveltime(for an aircraft which goes up or down)
  The Verticaltime parameter represents the duration of the Vertical Evolution, if there is one. The Leveltime parameter represents the duration the aircraft will fly at the same level, after its vertical evolution. Then the aircraft may have to change once again its flight level, for instance if it has been climbing too much it may need to climb down for a while.

- Consumption of fuel
  This value is updated during the flight.

An aircraft which does not remain at the same level during its flight is not changing its heading, this is easy to understand for practical reasons. A pilot will not accept changing its altitude and also its heading, for the comfort of the passengers.

A problem is given by a set of aircraft entering an airspace in which conflicts are going to occur. We have for each aircraft :

1. Its type : for instance a Boeing 747 (we are using a realistic model to make our aircraft fly : this model is based on the characteristics of the aircraft.

2. Its position (latitude and longitude, in degrees, altitude in feet).

3. Its heading, in degrees.

4. Its destination point altitude.[3]

Other global data given :

1. The duration of our study and time step used during the simulation.
2. The horizontal separation and the vertical separation, in Nm.
3. The percentage error about aircraft speed (most aircraft can not maintain a constant speed).
4. The maximum turning angle, in degrees.

The main issue was to know how we were going to compute the *fitness* of a chromosome. We have a poly criteria problem to solve, in fact the following criteria have to be matched together to give us a single fitness function :

- A difference of flight level at the end of the simulation (for instance an aircraft was bound to reach flight level 300 and it only reached level 290).
- The delay induced by a change of heading.
- The consumption of fuel induced by a modification of the flight level.

The difference of flight level at the end of the simulation has been heavily penalised so that each aircraft reaches its flight level destination. The solutions creating conflicts have not been eliminated, but they are heavily penalised, because if we want the GA to work we have to let it reproduce, cross and mutate chromosomes. Chromosomes creating conflicts are often near a suitable solution so they have to survive, for a while, among the population. The problem which occurred when introducing the third dimension was to evaluate the cost of a vertical evolution. The algorithm must try to keep the aircraft as much as possible at the same level because of the problems caused by a flight level modification (consumption of fuel, discomfort for the passengers, difficulty to know, with accuracy, how the aircraft climbs).

The crossover operator used randomly picks up one part of each parents chromosome, whereas the mutation operator randomly changes one of the parameters of an aircraft of a chromosome.

3.3 Results

To validate this algorithm, we experimented it on conflicts involving five aircraft. This conflict is very hard to solve and would probably never really occur.

[3] We only need to know the altitude of the destination point because the other coordinates are computed by our model using the heading of the aircraft and its characteristics.

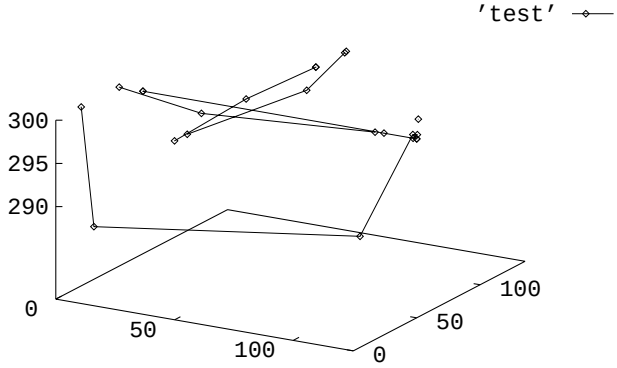

Figure 6: Efficiency of a flight level modification .

Figure 6 gives a 3-dimensional representation showing how an aircraft avoids the conflict by going down 10 levels[4].

We can also see on figure 7 that the 4 aircraft conflict remaining is very well solved in the horizontal plane, each aircraft almost reaches its destination point. This is a good example of the interest of having an aircraft changing level in order to solve the conflict more easily. Moreover the flight level modification is done with efficiency, we can see that the aircraft avoids the conflict as closely as possible. It only modifies its altitude to avoid a conflict.

The parameters for the simulation were :
Population size : 200
Number of generations : 100
Probability of crossover : 0.6
Probability of mutation : 0.1

Figures 8 and 9 give the results of the algorithm.

At the end of the GA, we used a gradient method in order to locally improve the best Chromosome.

This study shows that GA are very efficient for our problem. It has the great advantage not to use heuristics for solving conflicts, to reach a global optimum and not only an available solution. More details can be found in [AGS93, DAAS94].

[4] 10 levels represent 1000 Feet, which was precisely the vertical separation used for the simulation.

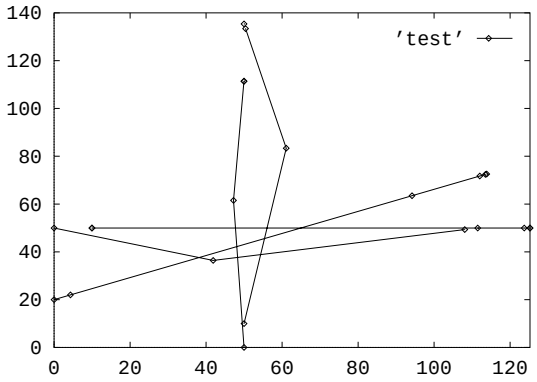

Figure 7: Efficiency of a flight level modification (horizontal view).

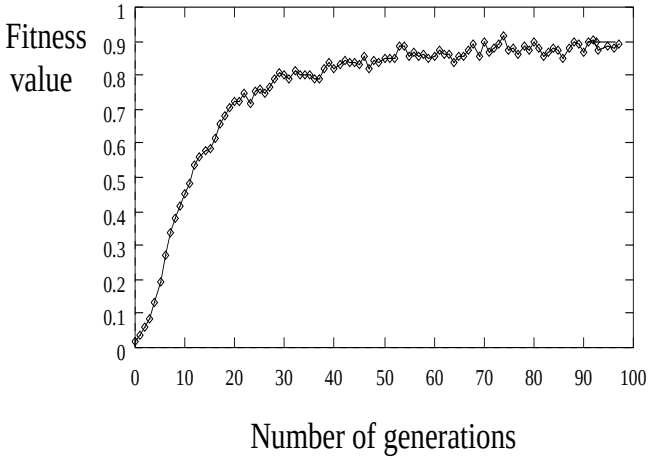

Figure 8: Average fitness evolution.

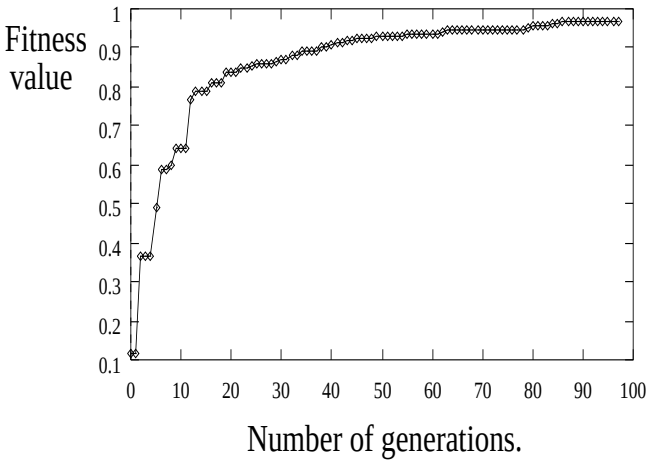

Figure 9: Best fitness evolution.

4 Air space sectoring

4.1 Description and Complexity of the problems

> *we consider 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 minimises coordinations*[5].

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

- a pilot must not encounter the same controller twice during his flight to prevent superfluous coordinations ; this means that an aircraft crossing a sector will encounter 2 and only 2 sector boundaries. To guarantee that sectors meet this constraint we force them to be convex in the topological sense[6]. This constraint gives sectors a polygonal shape.

- a sector frontier has to be at least at a given distance from each network node (safety constraint). When a controller has to solve a conflict, he needs a minimum amount of time to develop a solution. Since each controller manages 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 is fixed at 7 minutes and can be converted into a distance once the aircraft speed is known.

- an aircraft has to stay at least a few minutes in each sector it passes through to give the controller enough time to manage the flight in optimum 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.

We have to build convex sectors (with a polygonal shape induced by the convexity property). To reach this goal we use a Forgy aggregation method [Sap90]

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

[6]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

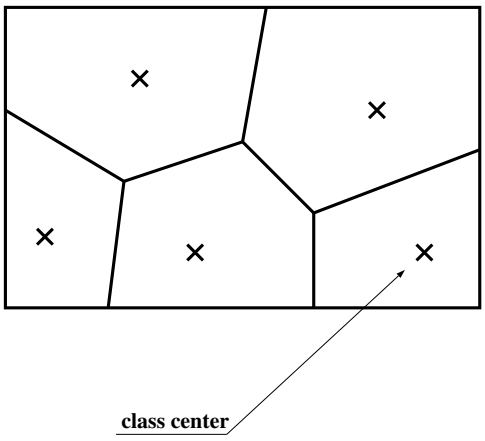

Figure 10: Example of a 5-partitioning

coming from dynamic clustering in exploratory statistics which aims at extracting clusters from a set of points randomly distributed in a topological space (see [CDG+89, Sap90]). 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 10 gives an example of a 5-partitioning of a rectangle.

The problem we have to solve can be divided into two separate parts corresponding to two different goals:

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

2. minimisation of the coordination workload.

The second criterion is typically a discrete graph partitioning problem with topological constraints and then is NP_HARD [Che92]. Having chosen a continuous flow representation, the first criterion induces a discrete-continuous problem which is also NP_HARD. So, according to the size of our network (about 1000 nodes), classical combinatorial optimisation is not relevant and stochastic optimisation 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 we must be able to find all of them because they have to be refined by experts and we do not know at this

X1	Y1	X2	Y2	X3	Y3

Figure 11: Floating point chromosome

step which one is really the best. This last point makes us reject classical simulated reannealing optimisation which updates only one state variable, even if it might give better results in some cases [IR92].

On the other hand, Genetic Algorithms (GAs) maintain and improve a population of numerous state 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.

4.2 Coding

A chromosome must contain all the sectoring information for the GA to be able to evaluate the fitness for each individual. This information is represented by a set of points in our geographical space called class centers (it can be shown that for each class center set there is just one sectoring induced see [Sap90])). Our chromosome will be composed of a string of floats containing the concatenation of the different class center positions (see figure 11).

This structure involves 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). Afterwards, 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 12 (allele 1 has been selected in this example).

Mutation When we mutate a chromosome we randomly select an allele position and we move its associated class center by adding noise to it[7] (see figure 13 (in this example allele 2 has been selected for mutation)).

4.3 Results

The results of the algorithm are very encouraging as shown by the following experiments results.

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

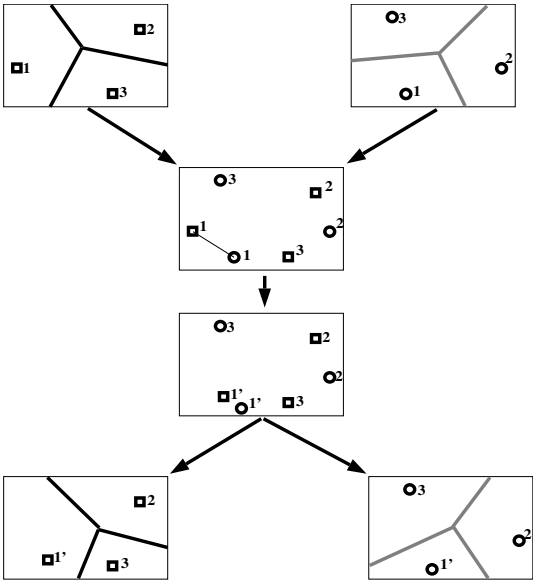

Figure 12: Example of crossover

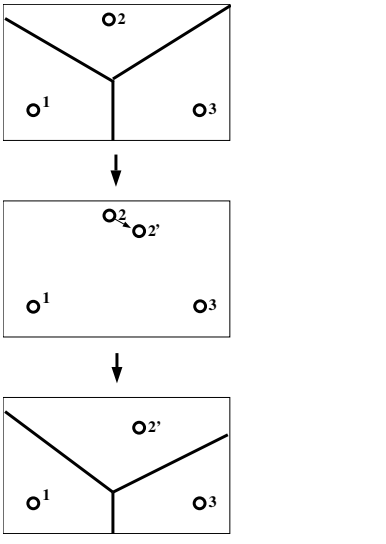

Figure 13: Example of mutation

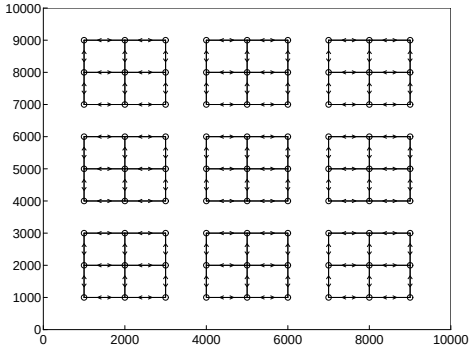

Figure 14: Square network

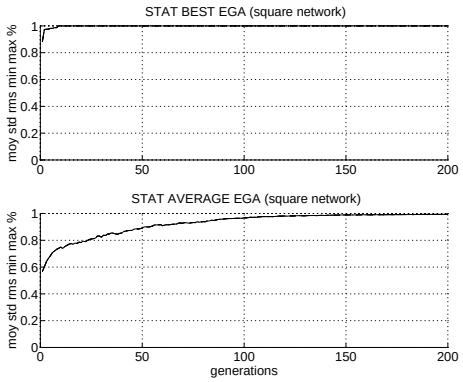

Figure 15: EGA stat results for the square network

To evaluate this algorithm, we have used an artificial test network (see figure 14)

As we can see this network has trivial solutions with 9 sectors. These solutions seem 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 characteristic 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

To see the convergence of our algorithm we observed the evolution of the population statistics (max and average) over the generations. It finds an exact solution very quickly (generation 8, see figure 15).

This study was discussed with more details and re-

sults in [DASF94a].

5 Traffic assignment problem

5.1 Description and Complexity of our problems

> *we consider an air traffic transportation network in a 2 dimensional space sectorised into K sectors for which we want to assign traffic between Origin-Destination pairs in a way that minimises extra route distance and reduces sector workloads.*

This traffic assignment must take the following constraint into account :

1. the flow between each OD pair must not be partitioned.

This constraint is mandatory, as we want to be sure that planes of different airlines on the same Origin-Destination will follow the same route: this is the *equity constraint*.

Because of the sectoring, the cost on one link depends directly on the flow on this link but is also in relationship with the flows of all the links in the same sector. This last point makes traffic assignment dependent on the order : as soon as a path is assigned for an OD pair, all previous assignments must be reconsidered as the new assignment changes the cost function by changing the link costs for all links in its sector.

So the problem induces a high combinatorial complexity for which we must try to find a solution in a discrete space with $n!$ points where n is the number of Origin-Destination pair, a problem known to be NP_HARD.

According to the number of Origin-Destination pair we have to handle (several hundred), classical combinatorial optimisation is not relevant and stochastic optimisation seems to be more suitable. As our goal is not to build the ultimate traffic assignment system, but a tool to help human experts assigning flows, we are definitely interested in all optimal or nearly optimal solutions the algorithm might find. The many constraints and the lack of efficient classical methods solving this problem led us to use GAs.

5.2 Coding

To code our problem, we did not use binary chromosomes. The problem is not well suited for binary coding, and, as it has been advocated already by different experts, a specific coding with specific operators is usually more efficient.

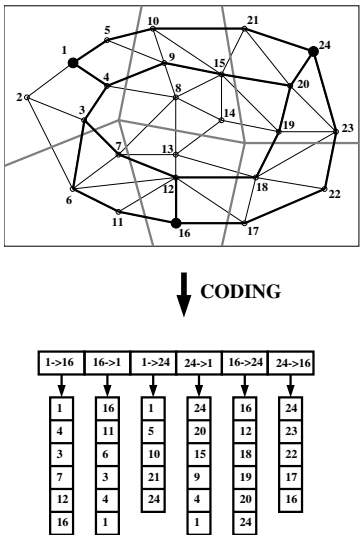

Figure 16: Structure of the chromosome

An example of the coding of a chromosome is given in figure 16. The chromosome is a list of cells ; each cell is the coding of the path for an OD flow. On the example, we see that all planes going from airport 1 to airport 16 will follow the path : airport 1, beacons 4,3,7,12 and airport 16. Planes from 16 to 1 will follow the path 16,11,6,3,4,1 and etc. So, all information necessary is encoded in each chromosome. It enables us to compute for each chromosome the traffic assignment cost giving the GA fitness.

One difficult point is the initialization of the population : to create one chromosome, we take into account distance costs only and we increase them by a random extra cost. Then, we apply a Dijkstra algorithm to find min cost paths for all the Origin-Destination pairs. This generates a list of OD paths which is our chromosome. We repeat those operations till the population size is reached. According to the deviation of the blank noise added as a cost, paths more or less different from the optimal ones are generated (optimal in the sense of the distance criterium only of course). This initialization method avoids the creation of purely random paths and ensures, for instance, that an aircraft coming from Madrid and going to London will not be routed via Moscow.

We had then to create operators for crossover and mutation. The efficiency of the algorithm depends of the ability of those operators to create new individuals that respect the constraints of our problem and that

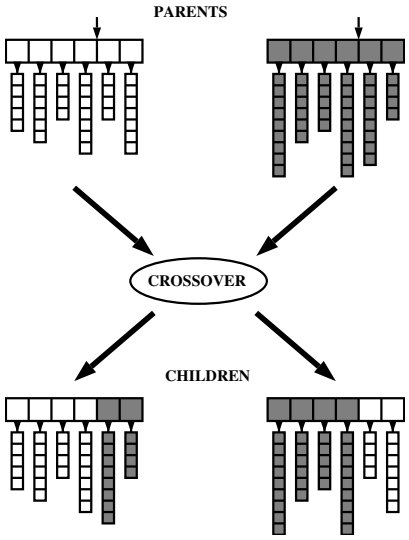

Figure 17: Crossover operator

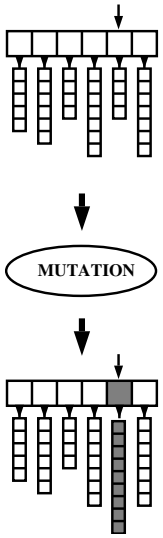

Figure 18: Mutation operator

generate paths not too far from the optimal ones.

The crossover is implemented as a slicing crossover : after selecting two parents in the current population, we randomly chose an allele position creating twice two paths subsets. Then, we just exchange the two last subsets to create two children. An example of crossover is given on figure 17.

To mutate a chromosome, we randomly select an allele position and generate a new path for the Origin-Destination pair selected by the same process as for generating the initial population. An example, of mutation is given on figure 18.

5.3 Results

To validate the algorithm, we used a toy network for which we knew a trivial traffic assignment solution (this network is drawn on figure 19). All the nodes on the first diagonal (upper-left to lower-right) are airports, all nodes on the other diagonal are beacons.

All airports in the upper left corner generate a traffic flow which must be routed to the symmetric airport (relative to the center of the web) in the lower right corner. Respectively, each airport in this corner generates a flow that must be routed to the symmetric airport in the upper left corner. Face to face flows on the same link are forbidden and link capacity is very limited, in order to prevent two flows to be routed on the same link.

The parameters for the simulation were :
Population size : 400
Number of generations : 300

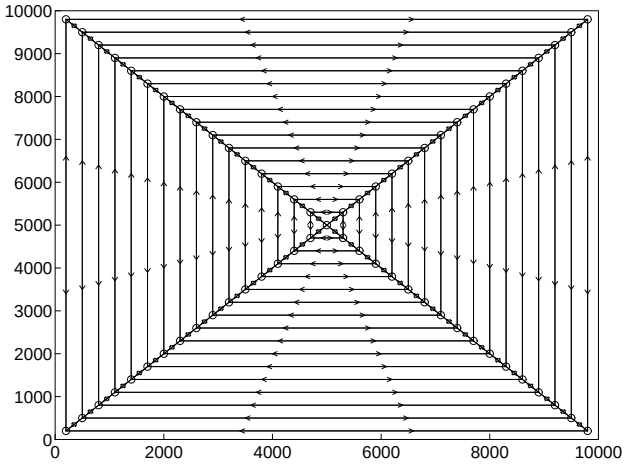

Figure 19: Test network

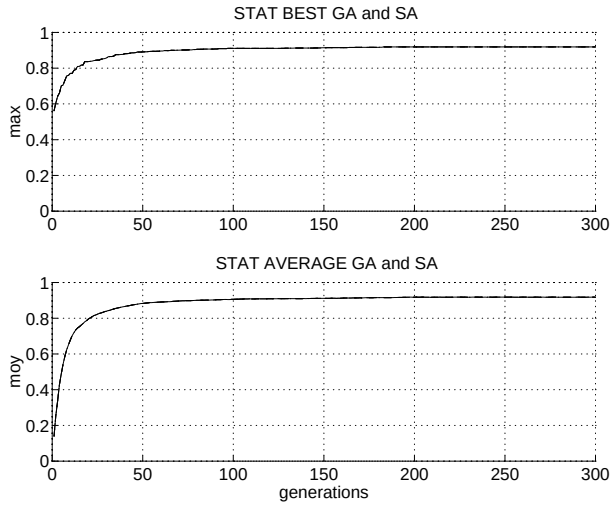

Figure 20: Fitness evolution

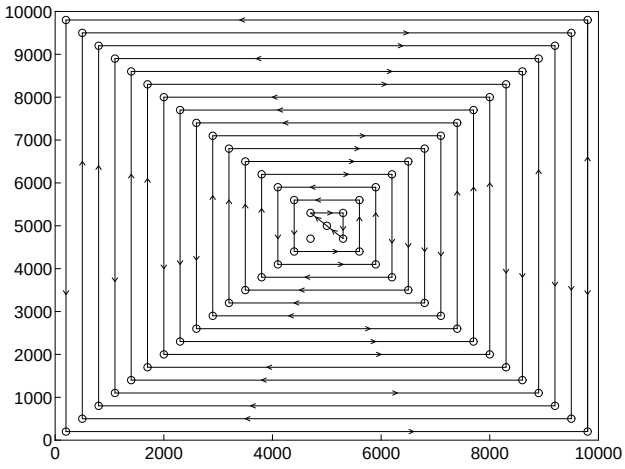

Figure 21: Traffic assignment result

Probability of crossover : 0.6
Probability of mutation : 0.1

The evolution of best-ever chromosome fitness and average chromosome fitness is displayed in figure 20 : an optimal solution is found at generation 180.

The solution is displayed in figure 21. It is clearly a correct solution (there were many other solutions with the same fitness : the direction of planes on each link can be either clockwise or counter clockwise). As for the previous part, it must be noted that, even if this solution is trivial to find for a human being because of the symmetries of the problem, it remains as difficult as any other problem for our algorithm. The algorithm was then tested on more realistic networks (too large to be presented here) and gave also good results [DASF94b] ; moreover, we were not able to find a better traffic affectation by hand, which is a good presumption of a correct behaviour of the algorithm.

6 Conclusion

This study shows how Genetic Algorithm are suitable to solve some ATC problems with very special constraints. To reach this aim we had to extend the classical binary concept to floating strings. This modification really improved the algorithm performances regarding the resolution speed and the and the accuracy of the result. Subsequently we added a tournement operator and used dynamic parameters to improve the space exploration as well as the selection process. We also used a fast sharing process which enabled us to find several different solutions of our problems. The two fisrt problems being dependent (Traffic assignment and Sectoring interact with each other), the next step of this study is to mix the two first problems in a single multi-objective genetic algorithm. For the conflict resolution problem our next purpose is to foresee real conflicts happening during a traffic day, and to solve them in a real time to make the aircraft avoid each other when the conflict happens. A project of automating and optimizing the ATC system will follow this initial study. Through this paper, GAs showed that they were completly adapted to this project.

References

[AGS93] Jean-Marc Alliot, Hervé Gruber, and Marc Schoenauer. Using genetic algorithms for solving ATC conflicts. In *Proceedings of the Ninth Conference on Artificial Intelligence Application*. IEEE, 1993.

[AK89] Emile Aarts and Jan Korst. *Simulated annealing and Boltzmann machines.* Wiley and sons, 1989. ISBN: 0-471-92146-7.

[BDD63] A.E. Bryson, W.F. Denham, and S.E. Dreyfus. Optimal programming problems with inequality constraints: Necessary conditions for extremal solutions. *AIAA Journal*, 1:2544–2550, november 1963.

[BH75] Bryson and Ho. *Applied Optimal Control.* Hemisphere Publishing Corporation, New York, 1975.

[CDG+89] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier, and H. Ralambrondrainy. *Classification automatique des données: environnement statistique et informatique.* Dunod, 1989.

[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.

[Che92] Chung-Kuan Cheng. The optimal partitioning of networks. *Networks*, 22:297–315, 1992.

[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.

[DASF94a] Daniel Delahaye, Jean-Marc Alliot, Marc Schoenauer, and Jean-Loup Farges. Genetic algorithms for partitioning airspace. In *Proceedings of the Tenth Conference on Artificial Intelligence Application.* IEEE, 1994.

[DASF94b] Daniel Delahaye, Jean-Marc Alliot, Marc Schoenauer, and Jean-Loup Farges. Genetic algorithms for air traffic assignment. In *Proceedings of the 1994 European Conference on Artificial Intelligence.* ECAI'94, 1994.

[DMA94] Nicolas Durand, F. Médioni, 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.

[Dur96] Nicolas Durand. *Optimisation de Trajectoires pour la Résolution de Conflits en Route.* PhD thesis, ENSEEIHT, Institut National Polytechnique de Toulouse, 1996.

[Gol89] David Goldberg. *Genetic Algorithms.* Addison Wesley, 1989. ISBN: 0-201-15767-5.

[IR92] Lester Ingber and Bruce Rosen. Genetic algorithm and very fast simulated reannealing: a comparison. *Mathematical and Computer Modeling*, 16(1):87–100, 1992.

[Kre82] E Kreindler. Additional necessary conditions for optimal control with state-variable inequality constaints. *Journal of Optimization theory and applications*, 38(2):241–250, october 1982.

[Med94] Frédéric Medioni. Algorithmes génétiques et programmation linéaire appliqués a la résolution de conflits aériens. Master's thesis, Ecole Nationale de l'Aviation Civile (ENAC), 1994.

[MG92] 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.

[Mic92] Zbigniew Michalewicz. *Genetic algorithms+data structures=evolution programs.* Springer-Verlag, 1992. ISBN: 0-387-55387-.

[Sap90] Gilbert Saporta. *Probabilités, analyse des données et statistique.* Technip, 1990.

[YG93] Xiaodong Yin and Noel Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In C.R. Reeves R.F.Albrecht and N.C. Steele, editors, *In proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference, Insbruck Austria.* Springer-Verlag, 1993.