Genetic Algorithms for solving Air Traffic Control conflicts

Jean-Marc Alliot Hervé Gruber Georges Joly Marc Schoenauer

IRIT* ESE† CENA‡ CMAPX§

Abstract

This paper shows how genetic algorithms methods [5, 7] can be used to solve Air Traffic Control conflicts. We also compare these methods, in order to validate their solutions, to more classical methods such as graph search with A^* and related algorithms, or simulated annealing. We show that genetic algorithms are perfectly suited for ATC conflict problems, as they are able to :

- *give many different solutions for a given problem,*
- *give almost optimal solution in a very short amount of time, with the possibility to enhance the solution later if more time is available, a critical property for ATC systems.*

AI topic: Genetic Algorithm, Optimization
Domain Area: Air Traffic Control
Status: Operational mock-up.
Effort: 1 man/year
Impact: Optimization of planes trajectories inside control sectors.

1 Introduction

Since 1980, air traffic is increasing by 10% each year. The French Air Traffic Control (ATC) system is faced with slightly different problems than the American one. In the USA, problems are mainly related to Airport capacities, but in France (and Europe), the problem is mainly related to En-Route organization and conflict solving inside sectors[1]. Different methods can be taken under consideration: on one hand, we can try to develop cooperative tools that will enhance controllers productivity and keep the man as the central point of the control system [2]. On the other hand, with the development of Data-Link between planes and ground based control centers, the development of GPS and 4D-Flight Management Systems, a partial or complete automation can seriously be considered. There are different approaches that can be undertaken, some based on mainly algorithmic resolution [3], some others relying on expert system methods [4], some planning to use both [13]. Works on these subjects can even require very theoretic work [1] for the development of such systems.

There are roughly two different levels of resolution in an ATC system:

- A strategic level which would organize traffic to minimize the probability of conflicts.
- A tactical level which would solve "in real time" conflicts as soon as they are detected by trajectory predictions, and would optimize trajectories for all planes related to the conflict. This system would act 10 to 15 minutes before the calculated conflict point inside a square sector of 80Nm by 80Nm. The system would have to guarantee that each plane will leave the sector at exactly the point and at exactly the time scheduled, and that each and every conflict inside the sector has been solved. This system will be called the *trajectory planner.*

The goal of the work presented here was to try different approaches for doing trajectory planning for ATC. Trajectory planning has been a classical problem in robotics for the last years [10], but some specificities of ATC (all objects are moving, every object trajectory must be optimized in terms of fuel cost,...) prevent from using directly robotics methods. We decided to use genetic algorithms as the main method for doing trajectory optimization[2], but also to try to compare the results given by genetic algorithms to the results given by classical graph search methods (A, A^* [12] and related algorithms especially iterative deepening A^* [9, 14]). This was necessary to validate the optimality of genetic algorithms results.

An other goal of the work was to give an evaluation of the efficiency of genetic algorithms without devoting too much time to the coding of genetic algorithms methods themselves: so, we decided to use the classi-

*Institut de Recherche en Informatique de Toulouse
†Ecole Supérieure d'Electricité
‡Centre d'Etudes de la Navigation Aérienne
§Centre de Mathématiques appliquées de l'X

[1] The total airspace is divided into control sectors. For the purpose of this study we will consider that sectors are cubic area of 80Nm by 80Nm with an height of approximatively 200000 feet. Two planes in a sector are in conflict if they fly at the same altitude (same flight level) and their horizontal distance is equal or less than 8Nm.

[2] [19] presents also an application of genetic algorithm for bus routing.

cal SGA system developed by David Goldberg[3] [18], and we tried to concentrate on the modeling of the problem itself.

2 A simplified model

2.1 Introduction

The first idea was to encode data with the coordinates of the points of each trajectory. Then, there were three real values (x, y, z) for each point of each plane trajectory. Two planes are not in conflict if they are always at a distance of at least 8Nm from each other. So, to guarantee that two planes are never in conflict, 10 points were required for a 80Nm square sector. Thus, to solve a two planes conflict in a 80Nm square sector, we must minimize a function of $3 \times 10 \times 2 = 60$ real variables. This is definitely out of reach of any genetic algorithm, even when using more elaborate methods (such as Dynamic Parameter Encoding [16]). The few experiments we made confirmed that opinion.

So, the approach to the problem was changed:

- The problem to solve was simplified: each plane would fly with a constant speed and would not be able to change its flight level.
- Instead of coding each point of the trajectory, the actions that a plane could take at each step of the flight were coded. This data encoding is purely symbolic and much simpler than a numeric data encoding for that problem.

To summarize our first model:

- A sector is a 80Nm by 80Nm square;
- each aircraft is flying with a constant speed (430 knots, or 220 m/s);
- the only possible manœuvre is a change of heading, either 30 degrees left or 30 degrees right;
- an entry point and an exit point in the sector are associated to each aircraft; all aircrafts enter and leave the sector at the same time. The exit point is the point that will be reached at exit time if each plane is able to fly straight to it without any heading alteration.
- two planes are in conflict if, at any time, they are closer than 8Nm.

Of course, it is impossible for a plane to leave the sector at exactly the time scheduled if its heading has been modified to solve a conflict, as it is impossible to accelerate or decelerate planes. So, the term to optimize for each plane is the distance of the last point of the computed trajectory to the theoretic exit point.

[3]We also seriously considered using GA-UCSD [17], but one of its main advantage, Dynamic Parameter Encoding, is not directly useful in our example, and its use was slightly more complex.

2.2 Modeling

When a plane enters the sector at the entry point, its position and its current heading are known. Then, the total flying time (approximatively 11 minutes) over the sector is divided in 16 steps of time (around 40 seconds each). During each of these steps, the plane flies with a constant speed and with a constant heading. Before each step, the heading of the plane can be altered, either by turning it left $(+30^o)$ or right (-30^o). As each plane enters and leaves the sector at the same time, each step has exactly the same length for every plane.

In that very simple case, there are only three possibilities for a plane at the beginning of each step: either keeping the same heading, turning left or turning right. This can be coded with only two bits: 00 and 01 means that the plane is keeping its heading, 10 means that it is turning 30^o right and 11 that it is turning 30^o left. As there are 16 steps, all heading alterations for one flight over the sector can be coded with a sequence of $2 \times 16 = 32$ bits. If there are n planes in the sector, a sequence of $32n$ bits is required to code all heading alterations.

To evaluate the fitness of a sequence of bits that represents planes trajectories, the following steps are executed:

- The complete trajectory is computed for each and every plane, according to its entry point, its initial heading and the part of the sequence of bits coding its heading alterations.
- The system checks if there is a conflict between two (or more planes) according to the trajectories calculated (i.e. it checks if two planes are ever closer than 8Nm). If a conflict is found, then the fitness of the sequence is set to 0[4].
- If there is no conflict, the distance d_i from the optimal exit point[5] to the exit point calculated is computed for each plane i. The fitness f is then given by the following formula:

$$f = e^{-\frac{\sum_{i=1}^{n} d_i^2}{K}}$$

There are a few reasons that justify the choice of that formula for calculating fitness: It is clear that if the d_i are all equal to 0, then the trajectory is optimal and fitness is equal to 1; it is better to have three planes 2Nm from their exit point rather than one 6Nm away, and this justify the square for d_i; the function should also decrease quite quickly away from the optimum: that's the

[4]We tried also a different solution: when a conflict was found, fitness was divided by 10. This gave sometimes slightly better results.

[5]Notice again: the optimal exit point is the point that the plane would reach if it was flying straight during each and every step of time. This will not always give a valid trajectory, as conflicts may occur.

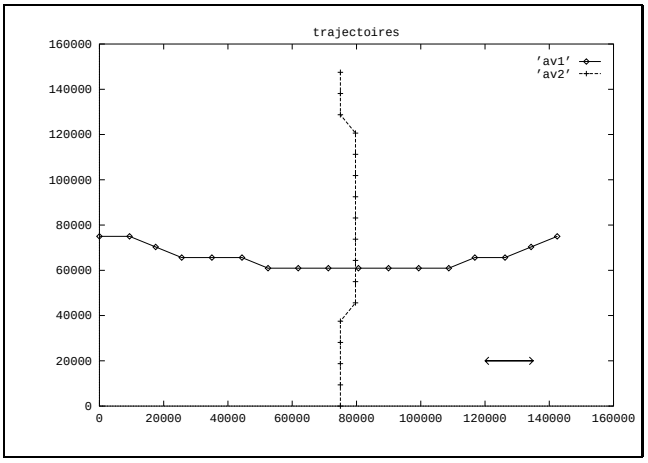

Figure 1:

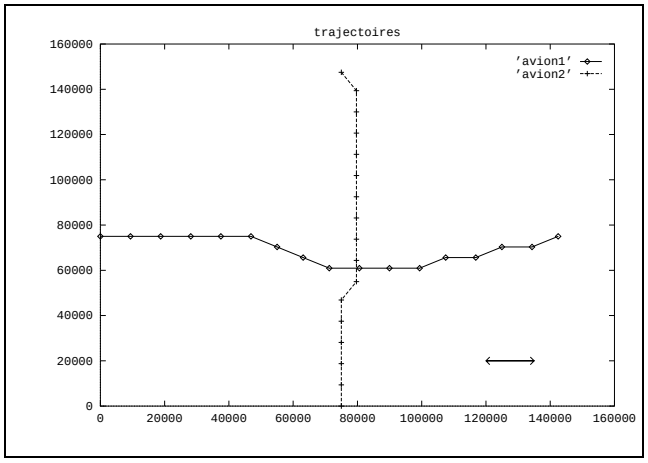

Figure 2: Resolution with 16 steps and constant speed

exponential function; the value of K must be correctly chosen to guarantee a correct behavior of the exponential function : if K is too large, all values will be closed to 1, if K is too small, all values will be closed to 0.

2.3 Two planes conflict

As said above, we used the simple, but very easy to use SGA system developed by David Goldberg.

We first tried to solve a simple problem with only two planes which are in conflict at the center of the sector. 400 initial random sequences were generated of $32 \times 2 = 64$ bits. The crossing parameter was set to 60%, the mutation parameter to 1% and the system evolved for 100 generations. After 54 generations in the first run, the solution represented on figure 1 was calculated and was not enhanced later on with other parameters settings. The run was done on a SparcStation2; it took 30 seconds to complete. In less than 5 seconds a nearly optimal result was computed and was then slightly enhanced. At the end of the run, there was in the population 3 different genomes representing different trajectories which had the same (optimal) fitness. The trajectory represented on figure 1 is one of them.

To check these results, we wanted to use an A^* related algorithm with the f and h functions being our fitness function.

But it was not possible to check the optimality of the solution with the A^* algorithm. The problem was not a matter of time but a matter of space, even with 32Mb of memory and 128Mb of swap space: a classical limitation for A^*. We had to use a related algorithm, close to Iterative Deepening A^* that is too complex to be described here. The solution calculated by this algorithm is shown in figure 2. This solution is slightly different from the solutions calculated by our genetic algorithm, but its cost (fitness) is exactly the same. In that case, the genetic method was optimal and definitely much faster (3 times faster) without any optimizations.

2.4 Three planes conflict

The problem with 3 planes, the 3 planes being in conflict at the same point (a plane crossing the sector from the lower left corner to the upper right cornerwas added to the previous example) was also solved. This is of course the worst possible case for three planes crossing a sector, as each conflict resolution is constrained by each other conflict resolution.

The best results were obtained with 1000 random sequences, a crossing factor set to 40% and a mutation factor set to 5%. The same results were computed with many different parameters setting, but were not enhanced any more with the simple method used. The run was done on a SparcStation2 and took 2 minutes. Here again, a nearly optimal result was calculated in a very short amount of time and then slightly enhanced.

It was impossible to check out the optimality of the solution as it was impossible to solve the problem either in a reasonable amount of time, or with the memory we had even with iterative deepening. However, using simulated annealing [20] with a very slow scheme for decreasing temperature, a slightly better solution was computed after a considerable amount of time (see table 1). We think that the solution calculated by the GA is very close to the optimal solution, but we were not able to prove it.

The problem for a four planes conflict was also solved, with the four planes being in conflict at the same point. This situation is purely theoretic as four planes conflict never occur in En-Route control, but it was an interesting test to check the efficiency of GA. There again, very satisfactory results were obtained in a short amount of time.

	Genetic algorithm	Simulated annealing
δ_1	4.56	3.04
δ_2	3.04	4.56
δ_3	5.29	4.55
Quadratic error	7.62	7.13

Table 1: Comparison of genetic algorithm and simulated annealing for three planes conflict

3 A model including speed modifications

3.1 Introduction

Making a plane climb or descend is very rarely used to solve a conflict as it is a serious penalty in terms of fuel cost and passengers comfort. But the possibility for a plane to accelerate or decelerate had to be included in our model to have a more realistic system. So, we modified our model to include the following:

- planes can increase or decrease their speed by 20m/s at each step.
- speed must stay in the range 170–260m/s (330–510kts).

In that model, planes should leave the sector at the time exactly scheduled (each plane being able to accelerate and decelerate, it is always able to compensate trajectory alteration by speed modifications). The term to optimize for each plane will be fuel cost. We have considered that, around 220m/s, fuel cost would approximatively be proportional to the cube of speed, according to [11].

3.2 Modeling

The simplified model has to be slightly modified to include speed alterations. At each step, a plane can change heading *and* change speed. Speed can either:

- stay stable;
- increase by 20m/s;
- decrease by 20m/s.

2 more bits are used to code speed alterations at each step: 00 and 01 means that speed is constant, 10 that speed will increase by 20m/s and 11 that it will decrease by 20m/s.

Evaluating the fitness of a sequence is less trivial than expected. The first step is of course to compute the complete trajectory along with the speed on each segment for every plane according to the sequence of bits and of the initial positions, speed and headings of each plane.

Then, the simplest idea would be to give a fitness proportional to fuel cost if all planes reach their destination and a fitness of 0 if any of the planes is either in conflict or does not reach its destination. However, this simple evaluation does not work correctly. Too many sequences have a fitness evaluated to 0, especially at the beginning of search, as random sequences do not give trajectories with planes reaching their destination. Under those conditions, a genetic algorithm does not behave properly. Then a slightly different function was used for evaluating fitness:

$$f = e^{-\sum_{i=1}^{n} (\frac{d_i}{K_1})^4 + (\frac{C_i}{K_2})^2}$$

The signification of the variables are:

- d_i is, for plane i, the distance from the plane destination to the last computed point of the trajectory according to the sequence of bits.
- C_i is the fuel cost over the computed trajectory for plane i.
- K_1 and K_2 are constants.

Some of the justifications given for the function used for our simpler model apply also here. In addition, this function is designed to correctly balance :

1. fuel cost
2. distance to the optimal exit point (destination)

The function gives distance a higher power than fuel cost. Thus, the GA is coerced to select first trajectories that satisfy the most important criterion: being very close to the optimal exit point. Then, for planes having similar or almost similar distance cost, fuel cost makes the difference and the solution that has been calculated in the first stage is later enhanced and fuel cost is drastically reduced.

3.3 Results

Figure 3 and figure 4 show results after a run of 100 generations with an initial population of 400 strings and parameters set to 40% for inter-crossing and 5% for mutations. Full details in respect to the evolution of the optimality of the solution in terms of distance reduction and fuel cost reduction are available in [6][6], but the GA performed exactly as expected: after optimizing distance, it then optimized fuel cost.

The same results were obtained with other parameter settings but it was not possible to enhance them. The solution was found in 1 minute on a SPARC2. The solutions found after the 10 first seconds were distance optimal, but not fuel cost optimal. However, after 20 seconds, a nearly optimal solution was computed and was only slightly enhanced later.

These results are definitely good, but we would have liked to check their optimality with a complete and

[6]This report is available by anonymous ftp on gogol.cenatls.cena.dgac.fr (143.196.1.6).

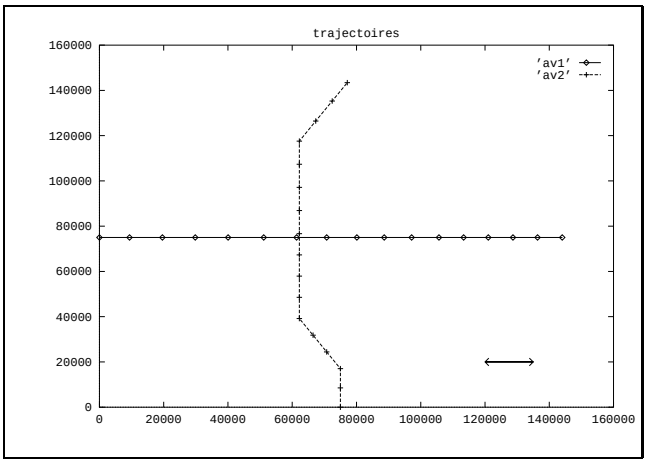

Figure 3: Trajectories calculated by a genetic algorithm ($n = 16$)

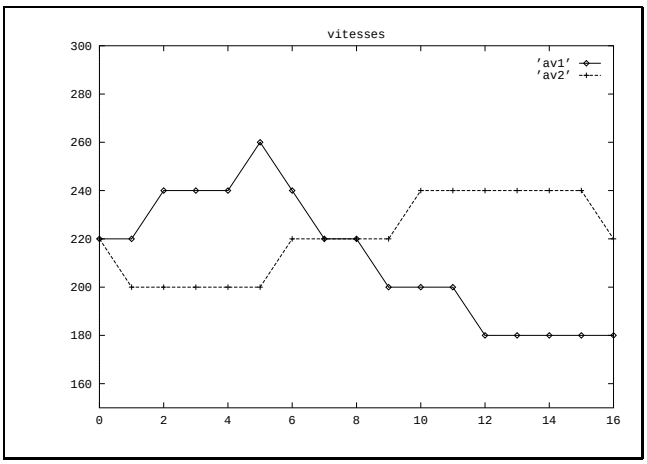

Figure 4: Speed variations ($n = 16$)

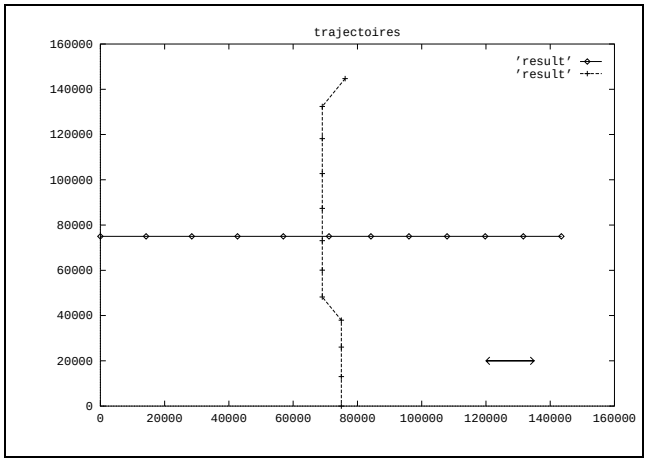

Figure 5: Trajectories calculated by an A related ($n = 11$)

exact search. Here again, we wanted to use an A^* related algorithm with the h function being our fitness function.

It was impossible to optimize enough our A^*-related algorithm to get the exact solution. It was necessary to use fewer steps in the resolution. Instead of 16 steps, it was only possible to use 8 steps. This was definitely two few to be able to compare. In fact, with 8 steps, separation of planes can not even be guaranteed: as we check for separation only before and after each step, two planes flying in opposite direction can cross each over in a step, generating a conflict that would remain unnoticed by the system.

So, instead of using an A^* algorithm, we used an A algorithm by multiplying the h function by 1.1 This guarantees that the cost of the solution we will find with this A algorithm will not be worst than 1.1 times the optimal solution cost. However, even with this method, it was impossible to have a solution with 16 steps and we had to to fall back to 11 steps. Thus, it is not longer possible to compare exactly the results given by the GA and the A algorithm, as there are not the same number of steps.

However, it is instructive to compare trajectories and speed modifications. Results of the A algorithm, with a maximal error of 10% and 11 steps, are shown on figure 5 and figure 6. They exhibit very similar shape for the trajectories, and very similar behaviors for speed alterations.

4 Discussion and future developments

In this study, we have demonstrated that genetic algorithms can be of interest for ATC. There are two interesting points in GA resolution methods:

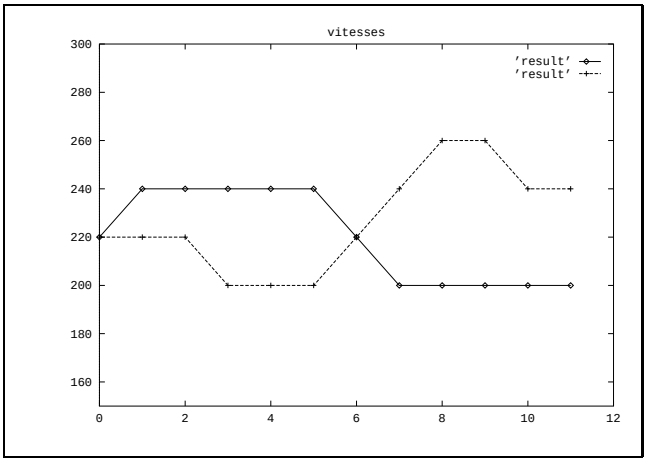

Figure 6: Speed variations with an A related algorithm ($n = 11$)

- Compared to classical graph search such as A^* which can take a very long time to compute a solution, GA are able to calculate in a few seconds a good solution, with the possibility to let the GA enhance thus result later on if more time is available. This is very important for ATC systems which are real time systems and need solutions in a limited amount of time, even if these solutions are not optimal.

- Compared to simulated annealing which gives only one solution, a population of strings holds many different strings close to the better solution found. This is also interesting for an ATC systems where a human supervisor will always be present to optimize and organize the solutions computed by the automated system. Giving a panel of choices to the supervisor is a definite advantage over simulated annealing technics which give only one solution.

We have already began to include some technical improvements:

- The initial population should not be random trajectories, but trajectories already close to the solution. It is clear that a straight line from the entry point in the sector to the exit point, for each plane, is always quite close to the solution. If we add a Gaussian noise to the string coding these trajectories, we will certainly have a much better initial population for solving conflicts.

- We will modify standard crossing and mutation operators to include some knowledge of ATC, instead of doing "blind" crossing and "blind" mutations. A very simple example: it is currently meaningless to cross strings at odd positions as every action is coded with two bits.

- We will change the selection process to include a mechanism based on elitist selection, to keep in the set of strings some elements which code optimal trajectory for each plane.

We will also try a new kind of coding, which will not code the trajectories with bits, but directly in terms of linked lists of points, as suggested by Marc Schoenauer [15]: crossing two strings is just cutting two linked lists at the same position and connecting the two part, as we would do for two strings of bits. A mutation becomes adding Gaussian noise to one point of a list, etc...

From an ATC point of view, we are also planning to include some improvements, mainly for calculating trajectories and fuel cost, using an operational simulator [8] instead of rough approximations. We will also try to enhance the set of available maœuvres to have a more realistic resolution system that could be tested during real time simulations.

The results of the study were considered promising by this contracting party: the grant has been renewed for one more year and a person from the CENA[7] will work full time on this project with us to enhance ATC parts.

[7]The CENA is the state organism in charge of the design of the future French ATC system.

References

[1] Jean-Marc Alliot and Andreas Herzig. Implementing PROLOG extensions: a parallel inference machine. In *Proceedings of Fifth Generation Computer Systems conference (FGCS-92)*, volume 2, pages 833–842. Institute for New Generation Computing Technology, OHMSA, Ltd, June 1992.

[2] Jean-Marc Alliot and Marcel Leroux. 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.

[3] Vanessa Fong. Automated problem resoltuion prototype in automated en route air traffic control. In *Proceedings of the Canadian conference on electrical and computer engineering*, 1989.

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

[5] Hervé Gruber. Comparaison de différentes méthodes d'IA pour la résolution de conflit. Technical Report 92-024, Centre d'Etudes de la Navigation Aérienne, Juin 1992. Responsable de stage : Jean-Marc Alliot.

[6] John Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[7] Georges Joly. Mass, un générateur de trafic avion. Technical Report N92-584, CENA, 1992.

[8] Richard E. Korf. Depth first iterative-deepening: an optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[9] Fred Krella et al. Arc 2000 scenario (version 4.3). Technical report, EUROCONTROL, April 1989.

[10] Jean-Claude Latombe. *Robot motion planning.* Kluwer Academic Publishers, 1991.

[11] Thierry Mandon and Georges Joly. MASK: Aircraft performance model. Technical report, Centre d'Etudes de la Navigation Aérienne — Centre d'Etudes EUROCONTROL, September 1991. CENA: R91-024, EEC: 2581/B3/BP02B.

[12] Judea Pearl. *Heuristics.* Addison-Wesley, 1984. ISBN: 0-201-05594-5.

[13] Tekla S. Perry and John A. Adam. Improving the world's largest, most advanced system! *IEEE Spectrum*, February 1991.

[14] U. K. Sarkar, P. P. Chakrabarti, S. Ghose, and S. C. De Sarkar. Reducing reexpansions in iterative deepening search by controlling cutoff bounds. *Artificial Intelligence*, 50:207–221, 1991.

[15] Marc Schoenauer. Utilisation des algorithmes génétiques pour l'étude du tracé des voies de t-gv. Technical report, Centre de Mathématiques Appliquées de l'École Polytechnique, 1992.

[16] Nicol N. Schraudolf and Richard K. Belew. Dynamic Parameter Encoding for genetic algorithms. Technical Report 90–175, UCSD, September 1991.

[17] Nicol N. Schraudolf and John J. Grefenstette. *A user's guide to GAucsd 1.2.* UCSD, 1991.

[18] 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.

[19] Sam R. Thangiah and Kendall E. Nygard. School bus routing using genetic algorithms. In *Proceedings of the SPIE Conference on the Applications of Artificial Intelligence X: Knowledge Bases Systems*, pages 387–398, 1992.

[20] P. J. M. van Laarhoven and E. H. L. Arts. *Simulated annealing: theory and applications.* Kluwer Academic Publisher, 1987. ISBN: 90-277-2513-6.