Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR TP 96655

Application of genetic algorithms in the aerospace domain

H.H. Hesselink, H. Kuiper and J.M. van den Akker

DOCUMENT CONTROL SHEET

	ORIGINATOR'S REF.		SECURITY CLASS.
	NLR TP 96655 U		**Unclassified**

ORIGINATOR
National Aerospace Laboratory NLR, Amsterdam, The Netherlands

TITLE
Application of genetic algorithms in the aerospace domain

PRESENTED AT
EUFIT '96, The Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September 2-5, 1966

AUTHORS	DATE	pp	ref
H.H. Hesselink, H. Kuiper and J.M. van den Akker	**960719**	**10**	**5**

DESCRIPTORS
Aerodynamic characteristics
Air traffic control
Aircraft guidance
Airfoils
Applications of mathematics
Design analysis
Genetic algorithms
Heuristic methods
Optimization
Software tools
Trajectory planning

ABSTRACT
This report contains the paper that was accepted for presentation at EUFIT '96, the Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September 2-5, 1996. In the paper, two case studies, performed at NLR and investigating the applicability of genetic algorithms in complex optimisation problems, are presented.
The first case study concerns the use of algorithms to support planning in Air Traffic Management (ATM). Based on given starting conditions of aircraft, they must reach their exit conditions through flying trajectories such that no conflicts occur. Trajectories are modelled by means of a Mode Of Flight based representation, i.e. trajectories are specified using straight lines, curves, accelerations and altitudes. Alternative trajectories are investigated from the first conflict on. Problem specific knowledge has been incorporated in the mutation and recombination operators. The importance of recombination increases when scaling up to larger problems. Problem specific genetic operators were also used in the second case, which concerns parameter optimalisation for examining airfoil pressure distribution on aircraft wings. The objective function was taken from an earlier experiment with non-linear optimisation methods which did not always find a solution. Although the genetic algorithm was able to find a solution, it was found that the initial population had a great influence on the quality of the final solution. Conclusions that can be drawn from these case studies are that the large amount of variables is difficult to manage and the necessary computing power can be quite large. On the other hand, when the initial problem is investigated and some domain dependent knowledge is added, promising results are obtained.

APPLICATION OF GENETIC ALGORITHMS IN THE AEROSPACE DOMAIN

H.H. Hesselink, H. Kuiper, J.M. van den Akker
Informatics Division
National Aerospace Laboratory NLR, The Netherlands
email: {hessel,kuiperh,vdakker}@nlr.nl

Introduction

The National Aerospace Laboratory NLR is the central institute for aerospace research in the Netherlands. NLR has been involved in various research and development programs where complex search and planning methods have to be applied. In some of those projects, the applicability of genetic algorithms has been investigated.

In this paper we will focus on two case studies, which exemplify the usage of genetic algorithms within an engineering environment. The first case study concerns the application of genetic algorithms to the planning of aircraft trajectories in Air Traffic Management (ATM). The second case study describes how genetic algorithms have been used to help in the design of airfoil pressure distributions.

The approach used for these cases differ in as much that for the first custom software has been developed, whilst for the other an off-the-shelf simulator has been used. For both cases a number of simulations have been performed, investigating the influence of various parameters.

The paper ends with lessons learned during the two case studies.

Case 1: Planning in ATM

Air Traffic Management (ATM) concerns the safe and efficient handling of aircraft. Currently, ATM planning is mainly a human activity. Due to the increasing volume of air traffic, new automated tools to assist the air traffic controller in making planning decisions become necessary.

EXPERIENCE IN CONFLICT FREE PLANNING
In the study (see Ref. [1]), planning without route structures has been investigated. Aircraft trajectories have been determined without constraining them to fly over specific way points. Aircraft trajectories interact by means of separation standards, where the separation standards define the minimum distance that has to be maintained between two aircraft. Apart from the separation standards, there are many other constraints, for example concerning the shape of the trajectory. Characteristics of the aircraft determine bounds upon velocity, acceleration, and total distance travelled.

Finding an optimal conflict free solution, in which the total distance of the aircraft covered is minimised, is a constrained optimisation problem. An additional complexity is the dynamic environment in which the aircraft fly. Parts of this problem can be solved using conventional methods. For example, once a conflict between two aircraft is solved, both the aircraft need to be routed to their exit points. This can be achieved by assigning a turn and a straight line to both aircraft. Finding a route for conflicting aircraft is done by the genetic algorithm. Because of this hybrid approach, it was difficult to find a suitable off-the-shelf product. Therefore, special purpose software has been developed.

For the problem to be solved with genetic algorithms, a suitable representation needs to be chosen. A trajectory is represented as a Mode Of Flight based sequence of parameterised manoeuvres. A straight line, a curve, an acceleration, and an altitude are defined:

$$M = \{\ straight(dt),\ curve(d\alpha),\ accelerate(dv),\ altitude(dh)\ \}$$

A trajectory is represented as a list of these manoeuvres, together with an appropriate value for the parameter R_m of each manoeuvre. A complete planning for all aircraft is represented by a set of these trajectories. One individual in the solution set of the genetic algorithm is considered to consist of one complete planning.

The mutation operator used is conflict driven, instead of only randomly driven. The starting point for mutation is the first conflict; the preceding part of the trajectory remains intact. The operator can be described as follows:

1. Determine the time of the first conflict t_{conf}
2. Select at random one of the aircraft involved in this conflict and extract its trajectory
3. Select at random the following parameters:
   - $t_{resolve} \in [0, t_{conf}]$
   - $m \in M$
   - $x_m \in R_m$
   - $t_{straight} \in [0,300]$
4. Construct a trajectory according to the rule:

   $$head(r,0,t_{resolve}) \cdot m(x_m) \cdot straight(t_{straight}) \cdot m'$$

   (where *head(r,0,t)* denotes the part of trajectory *r* corresponding to interval [0,*t*], the operator · is a concatenation operator, and *m'* is a list of manoeuvres guiding the aircraft to its destination).

The quality of the solution that was found is expressed in the fitness function. The fitness of a single trajectory has two parts: one represents the occurrence of conflicts in the plans, the other results in a penalty when a detour is taken. This penalty is zero in case of a straight line from entry to exit point. A scaling is performed, so that a solution where a number of aircraft make a small detour is preferred over a solution where a single aircraft makes a large detour. Both components of the fitness function are weighted but one must keep in mind that a conflict between two or more aircraft is never allowed in a final solution of the algorithm.

Now, the fitness of one individual is taken equal to the weighted sum over the fitness of all trajectories within the individual:

$$f(j) = \Sigma_{i \in j}\, \omega_i \cdot f_{traj}(i)$$

In a ranking scheme the individuals are sorted and ranked on fitness. The probability of being selected for reproduction is coupled to the rank. In this way, nearby conflicts have a larger chance of being selected for mutation than conflicts that appear later on. It is no use to solve later problems when earlier ones are not resolved yet.

In the experiments, one air traffic control sector of 200x200 km has been investigated. Planning problems with up to 20 aircraft at the same time have been presented to the genetic algorithm. Three different fitness functions, with different weighting values for ω_i, have been compared on a set of 24 random problems. One experiment sets $\omega_i = 1$ for the least fit trajectory and all others to zero. Only the worst fit trajectory is represented in the fitness of an individual. Another experiment takes the average over all trajectories by setting $\omega_i = 1 / \#aircraft$. Finally, a fitness based weighting is used, where bad performing trajectories have a relatively large influence on the fitness of a planning. For each test problem, 40 independent runs were performed for each fitness function, resulting in a total of 960 runs for each point in the graph.

A run is considered to be successful if a planning is constructed such that:

- the planning is free of conflicts, and
- all aircraft leave the sector at their exit position.

The algorithm terminates if it does not enhance its best solution for more than τ iterations. In our experiments we used $\tau = 100$.
From the three compared fitness functions, the function with all equal weights is consistently the worst. This low performance is a result of the low selective pressure enforced by the method. The algorithm taking only the first conflict in consideration performs much better. Unfortunately the rate of success drops fast as the number of aircraft increases. The lack of differentiation between different plans having the same conflict is assumed to be the reason for the fast drop of rate of success. Further evidence for this is the low number of iterations used in this case. Fast convergence may be an indication of bad exploration of the search space.

Putting a high emphasis on the worst trajectory, but also including the others has generally the highest rate of success. The advantage is that this does differentiate between different plans that have the same first conflict, because all trajectories influence the fitness value. Preference should be given to this fitness function, when the number of aircraft increases.

CURRENT WORK IN CONFLICT FREE PLANNING
Recently, an another evolutionary tool using genitic algorithms, for ATM-planning problems within a 3-dimensional airspace involving a few hundred aircraft in a sector of 2000 by 2000 kilometer has been developed. This tool can handle problems of the same order of magnitude as the ATM-planning problem for the current traffic in Western Europe. A paper on this algorithm will appear (see Ref. [5]). Here the algorithm is shortly described.

When scaling up to larger problems and a larger sector, recombination becomes more important, because problems may be solved locally in a part of the sector. Several local solutions may be combined in a new individual. The algorithm described in the first part of this case is mutation based, since it turns out to be hard to define a good recombination operator for a smaller sector; almost each trajectory influences the others. For the algorithm in our current project, a recombination operator which incorporates problem-specific knowledge has been developed. It is the most important operator in the algorithm.

There is a strong preference for trajectories that do not contain many maneuvers. Therefore, the initial population contains only straight-line trajectories on a fixed flight-level, i.e. on a fixed altitude. The fitness function includes the number of conflicts, the numbers of maneuvers and the sum of the distances travelled by the aircraft. Of course, a run of the algorithm is only successful if it generates a plan without conflicts. The only operator that generates new trajectories is the mutation operator. Two kind of mutations are applied: level-mutation, i.e. changing the flight level for a part of the trajectory, and detour mutation, i.e. change the heading of a trajectory and after some time change it again towards the destination.

An important conclusion from the work is that for large scale ATM planning problems it is required to incorporate knowledge regarding the problem domain in a evolutionary tool.

Case 2: Airfoil pressures

Many airfoil and wing design methods are based on the solution of a so-called inverse problem. An inverse problem involves the determination of the shape of an airfoil or wing such that for this contour an a priori prescribed pressure distribution exists at the flow condition considered. In airfoil design, using these type of methods, the basic idea is that the designer is able to translate his design requirements in terms of aerodynamic quantities such as lift, drag, and pitching moment into a properly defined target pressure distribution, and therefore the shape of the airfoil or wing.

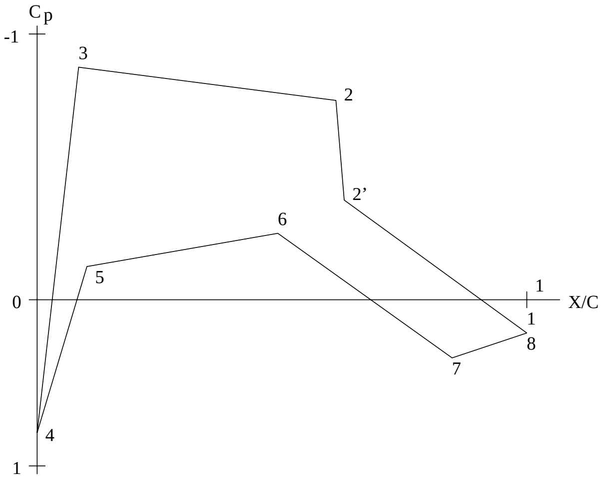

Fig 1: Schematic representation of pressure distributions

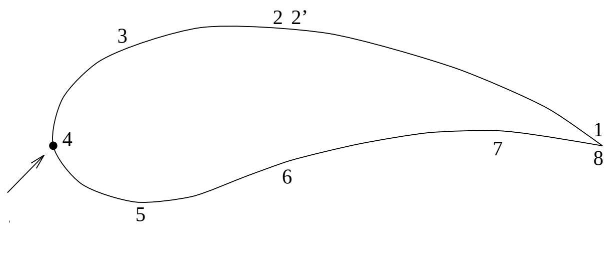

Fig 2: Relation between airfoil and pressure distributions

A characteristic pressure distribution is defined by eight co-ordinates, (x_i,y_i), where linear interpolation is chosen between these co-ordinates to simplify the approach (see figure 1). Of these eight co-ordinates, three are taken to be fixed, which leaves five co-ordinates (each consisting of level and position, therefore ten design variables) free to represent a large class of (simplified) pressure distributions. Figure 2 shows the actual airfoil and the identification of the coordinates from figure 1.

The resulting parameter optimisation problem has the following properties:

- the objective function and some constraints are highly non-linear;
- the objective function is continuous, but may locally not be differentiable;
- the objective function may suffer from numerical irregularities due to the numerical discretization and solution errors which are inherent to the computational analysis program used in the application;
- existence of local and global minima is unknown; usually an optimisation process is started from a initial solution and improved step-by-step.

The state-of-the-art non-linear optimisation method MINOS, employing a reduced gradient method, was compared with the Simplex method for the above-mentioned optimisation problem (see Refs. [3] and [4]). It appeared that MINOS, making use of derivatives in the optimisation process, could not find a feasible solution, contrary to the Simplex method, which does not make use of derivatives of the objective function. Apparently, the large effort to compute derivatives did not pay off, and it was speculated that it was questionable whether gradient information at all was valuable, since the surface described by the objective function is very complex and irregular. Due to the complexity and irregularity of the problem, solutions found were also sensitive to the choice of starting point.

Because of these results, the same problem has also been tried with a genetic algorithm (see also Ref. [2]). For the experiments with the genetic algorithm, the same objective function has been used. This objective function combines (using different weights) the drag coefficient and penalties on thickness, lift coefficient, and separation of the flow at the upper and lower trailing edge of the airfoil.

Using an existing objective function when implementing a genetic algorithm has the advantage that it is not necessary to define a fitness function and to implement it; it also makes comparison of the results with previous studies easier. A (major) drawback could be that the objective function was specifically designed for another optimisation method, which could negatively influence the performance of the genetic algorithm. This could result in a biased comparison of methods.

In the experiments, the ten design variables were encoded as floating point numbers and concatenated into a chromosome. As genetic operators, were chosen: proportional random selection for selection, and random intermediate recombination for operator. The latter operator is based on the principle of linear combinations:

$$x_i^a := \quad x_i^a + \lambda \cdot (x_i^b - x_i^a) \qquad y_i^a := \quad y_i^a + \lambda \cdot (y_i^b - y_i^a)$$

Where λ is random and in [0,1], and *a* and *b* are the parents for the operation. This operator was chosen because it was thought that randomly exchanging co-ordinates in the chromosomes would too often lead to an invalid solution.

For the mutation operator, it was opted to mutate a design variable x_i with a certain probability by: $x_i := x_i + \xi_i$. Where $\xi_i \in [-\varepsilon_I, \varepsilon_i]$, and ε_i small.

The following settings were used during all simulations:

- Population size: 100
- $\varepsilon_i = 0.1$, and a mutation probability of 0.005 (which means for each *variable* there is a probability of 0.005 that it will be mutated)
- a crossover application rate of 0.6
- scaling of fitness values
- maximum of 100000 function evaluations per simulation
- elitist strategy: the best performing solution will always survive into the next generation, therefore ensuring the best solution so far will be saved.

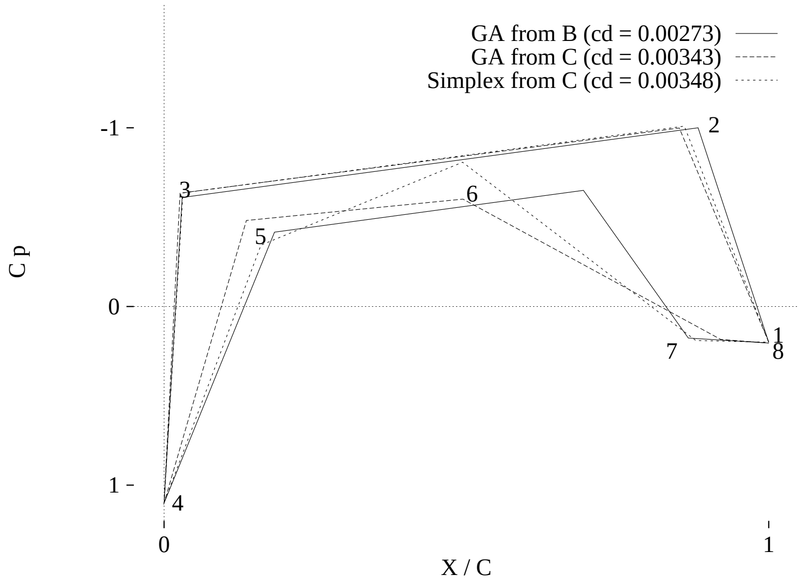

Fig 3: Results of the experiments

A number of experiments was then performed, only changing the initial population for each experiment. Two starting points were taken, both non-optimal solutions. These were used to create initial populations containing a mix of random and non-random members, or a mix of both starting points. From these experiments, the following could be concluded (see also figure 3):

- different minima were generated according to different initial populations; these minima do satisfy the constraints to good approximation. The Simplex method could not find feasible solutions from all starting points. In figure 3 the two results from the genetic algorithm are shown, as well as the one from the Simplex method.
- the genetic algorithm used about 100 times more function evaluations than the Simplex method. Probably the genetic algorithm can be improved by using an objective function better suited for genetic algorithms (currently it is necessary to adjust more than one parameter simultaneously at one time in order to get a new solution that is still within the constraints), by using more appropriate initial populations and by using more problem-specific genetic operators.

Lessons learned

During the two projects described, a number of observations were made which have an influence on the applicability of genetic algorithms in the aerospace domain.

“Genetic algorithms - isn't that just random search?” It can be difficult at times to explain to how genetic algorithms work, and why they work. For funding of a project, getting the right idea across is important. Also, often as soon as the word “random” is used, this is misunderstood for genetic algorithms to be equivalent to a random search, and therefore not worth further investigation. Explaining why the random element is needed, is often difficult.

Use of problem specific operators. For both cases, specific genetic operators had to be developed, as the “standard” operators were either not applicable (due to encoding used) or not effective. It is not clear how much problem-specific information should be encoded into the operators. On the hand this will increase effectiveness, on the other hand the operators will become less portable to other (even related) problem areas.

Number of function evaluations. It is very difficult to predict the number of function evaluations needed for a minimal acceptable solution to be reached. This prediction is important for two reasons: if genetic algorithms are used in a real-time situation, it is necessary to know whether a solution can be reached in the available time. Also, within commercial projects often computing resources have to be charged to the customer. By constraining the maximum number of iterations, a maximum bound can be established, however, the quality of the final solution is difficult to determine.

The initial population. Especially with the second case, the solutions obtained are very dependent on the initial population. It is not yet clear how much effort should be spent on finding "acceptable" starting solutions, using other methods (if possible), or if the genetic algorithm should be allowed to work from a completely random initial population.

“All those dials and buttons” During the simulations, most parameters had to be determined empirically. Especially when computer resources are limited, the “tuning” of the genetic algorithm is difficult. Also, the relationship between the different parameters is not always clear. Note that is probably the case for more search and planning methods.

Validation of results. Within the aerospace industry, for a large number of software systems, the software and results have to be validated before they are accepted by the user community. Currently with genetic algorithms it is very difficult to understand how a certain solution has been reached (e.g. going from unacceptable solutions to acceptable solutions). It is important that only validated solutions are offered to the users. Moreover, they should understand a given solution, otherwise they will not accept it.

Off-the shelf simulators vs. own code. For the second case study, use was made of a simulator from the public domain, as opposed to the first case. Although no simulator software had to be written this way, the packages found were not really suitable for use outside a research environment. For application in a production environment, probably more user friendly interfaces have to be developed in which input parameters can be specified, etc. Note that for both cases, the development of the fitness functions and problem encoding was the most time consuming.

Conclusions and future work

In the aerospace domain, there is a need for methods to solve complex search and planning problems. These problems often are in a complex domain, which is also often ill-structured. From the two case studies it can be concluded that using genetic algorithms for a number of those problems can be successful. Although we now have a feeling for which type of problems genetic algorithms can be useful, the exact characteristics are not yet clear. Genetic algorithms provide a extensive population of solutions, allowing to cope with changing circumstances.

Further research will be devoted to real-time re-planning. Due to changing weather conditions, or delays on airports, it is likely that some aircraft do not fly according to their planning. Under such circumstances, it is important to have rapid re-planning tools. Within this context both genetic algorithms as well as other methods will be investigated.

Acknowledgements

The authors wish to thank C.H.M. van Kemenade of the department of software technology of the CWI, the Netherlands. The first case study is based on his graduation work done at NLR and on his current collaboraton with NLR. The authors also wish to thank A.J. van der Wees of the Informatics Division of NLR for his work in the second case.

References

[1] C.H.M. van Kemenade, C.F.W. Hendriks, H.H. Hesselink and J.N. Kok, “Evolutionary computation in Air Traffic Control planning”, in: *Proceedings of the sixth international conference on Genetic Algorithms*, L.J. Eshelman (ed.), pp. 611-616, Morgan Kaufmann Publishers, 1995 (also published as NLR TP 94565)

[2] H. Kuiper, A.J. van der Wees, C.F.W. Hendriks and Th.E. Labrujère, *Application of genetic algorithms to the design of airfoil pressure distributions*, NLR TP 95342 L, 1995

[3] J.A. van Egmond, *Numerical optimisation of target pressure distributions for subsonic and transonic airfoil design*, AGARD CP 463, 1989

[4] J.P. van Wageningen and A.J. van der Wees, *Selection of an optimisation for highly non-linear problems. Evaluation of MINOS for some standard test problems and an aerodynamic design problem*, NLR TR 91336 L, 1991

[5] C.H.M. van Kemenade, J.M. van den Akker, and J.N. Kok. “Evolutionary Air Traffic Flow Management for Large 3D-problems. To appear in “*Proceedings of the fourth international conference on Parallel Problem Solving from Nature (PPSN* IV*), 22-27 September, 1996, Berlin, Lecture Notes in Computer Science*”, Springer Verlag, 1996