

Air Traffic Control Using Genetic Search Techniques

V.H.L. Cheng, L.S. Crawford, and P.K. Menon

Optimal Synthesis Inc.
Palo Alto, California

Abstract

Genetic search techniques constitute an optimization methodology effective for solving discontinuous, non-convex, nonlinear, or non-analytic problems. This paper explores the application of such techniques to a non-analytic event-related air traffic control problem, that of runway assignment, sequencing, and scheduling of arrival flights at an airport with multiple runways. Several genetic search formulations are developed and evaluated with a representative arrival traffic scenario. The results exemplify the importance of the selection of the chromosomal representation for a genetic-search problem.

I. Introduction

Trial-and-error or cut-and-try methods for finding the answer to a problem are well accepted in every field of science. The success rate of cut-and-try approaches depends greatly on the experience the user has with the problem under consideration. Genetic search methods described in this paper represent a logical approach to cut-and-try search methods. Continuing improvements in the computational power of desktop machines make genetic search methods a practical methodology for finding the answers to difficult problems.

The main difficulty in using cut-and-try methods is that it is unclear as to how to proceed from one guess answer to the next. In the case of a numerical search involving smooth functions, it may be possible to use the first and second derivatives of the functions with respect to the search variables to generate new guesses. However, if the functions are non-smooth, alternate strategies must be found.

Under these conditions, one is often tempted to employ random search methods. In the random search methodology, the problem is first parameterized so that a random number generator can be used to come up with random guesses. Random answers are selected and evaluated to determine the suitability with respect to the conditions that a correct answer should satisfy. In the limit as the number of trials goes to infinity, there is a good chance that the right answer will be found. Several variations of this basic random search strategy have been advanced in the literature, including those that change the probability distribution as the search advances. The

objective of such refinements is to attempt to increase the probability of finding the right answer as the number of trials increases.

Genetic search techniques represent an improvement over random search techniques. The main premise behind these techniques is that the successful biological processes observed in nature such as mutation and crossover can be simulated on the computer to generate a *population* of guess answers in cut-and-try problems. In this paradigm, each candidate answer or chromosome is considered to belong to a population. Associated with each chromosome is a *fitness* that describes the suitability of the chromosome with respect to the conditions describing the correct answer to the problem. Genetic operations such as mutation and crossover are used to synthesize new chromosomes, which are then evaluated with respect to the fitness criterion. A *selection* process is employed to choose the chromosomes from the population that will be used in carrying out these simulated biological operations. Periodically, the chromosomes that show poor fitness are deleted to keep the population from becoming too large, leading to improvement in the average fitness. This process of simulated biological selection-creation-destruction is repeated for a certain number of *generations*. The best member of this *population* then represents the answer. Figure 1 shows a simplified but typical flow chart for genetic search.

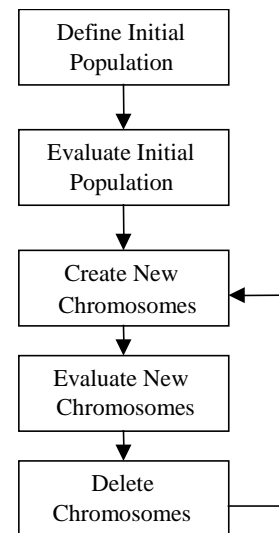


Figure 1. Typical Genetic-Search Program Flowchart

Due to its lack of restrictions, genetic search techniques are suitable for implementing nontraditional search processes. This approach has been found to be exceedingly successful in a wide variety of problems. This paper explores the use of such techniques in a time-based air traffic control (ATC) problem motivated by the automation system of [1]. The genetic search schemes described in this paper are based on the functionality provided by the Genetic Search Toolbox™ [2]. The functionality of this software has its basis in textbooks by Holland [3], Fogel [4], Goldberg [5], and Koza [6], and it encompasses the varied approaches used in the disciplines of genetic algorithms, genetic programming, and evolutionary programming.

II. Problem Description

The ATC problem discussed in this paper concerns the runway assignment, sequencing, and scheduling of n arrival flights to m runways. The problem is formulated with the following conditions:

- For each of the n arrival flights, an estimated time of arrival (ETA) to each of the m runways is available as the minimum time to get to the runway. Any restriction in aircraft performance and flight pattern is assumed to have been included in the determination of these ETA values. The schedule of a flight to a runway cannot be earlier than the corresponding ETA.
- All the flights scheduled to land on a runway have to observe specific separation rules between the leading aircraft and the trailing aircraft based on the aircraft types. Three aircraft types are assumed: Heavy (H), Large (L), and Small (S). The separations are assumed to be given in terms of time separation determined at the runway threshold.
- When a flight is scheduled to a runway, the delay as a result of the schedule is defined as the difference between the scheduled time of arrival (STA) and the earliest ETA among all the runways for that flight.

Four genetic search schemes are studied in Section III to explore the different characteristics associated with the different ways to formulate the genetic search. A simple scenario has been arbitrarily generated for evaluating the four schemes, with 12 flights and 3 runways (Figure 2). For identification purposes, the call signs of the flights are given by the list

call_sign = (UA138, UA532, UA599, NW358,
UA2987, AA128, UA1482, NW357, AA129,
UA2408, UA805, AA399);

and the corresponding aircraft types are given by the list

ac_type = (H, L, H, H, S, H, L, H, H, S, H, L)

The list of ETAs for the 12 flight and 3 runways are given by the following 12×3 matrix

$$\text{eta} = \begin{bmatrix} 11 & 10 & 9 \\ 15 & 17 & 19 \\ 6 & 7 & 8 \\ 6 & 7 & 8 \\ 9 & 12 & 15 \\ 7 & 6 & 5 \\ 15 & 17 & 19 \\ 6 & 7 & 8 \\ 6 & 7 & 8 \\ 9 & 12 & 15 \\ 7 & 6 & 5 \\ 9 & 7 & 5 \end{bmatrix}$$

Any landing sequence must satisfy the requirements for aircraft separation, which are often specified in terms of a separation matrix. The matrix used in this problem is:

$$\text{sep} = \begin{bmatrix} 1 & 1.5 & 2 \\ 1 & 1.5 & 1.5 \\ 1 & 1 & 1 \end{bmatrix}$$

where the rows represent the leading aircraft, and the columns represent the trailing aircraft, with both dimensions arranged in the order of Heavy, Large, and Small aircraft types. For instance, a Small aircraft following a Heavy aircraft will require 2 time units of separation, while a Large aircraft trailing a Small aircraft will require only 1 time unit of separation.

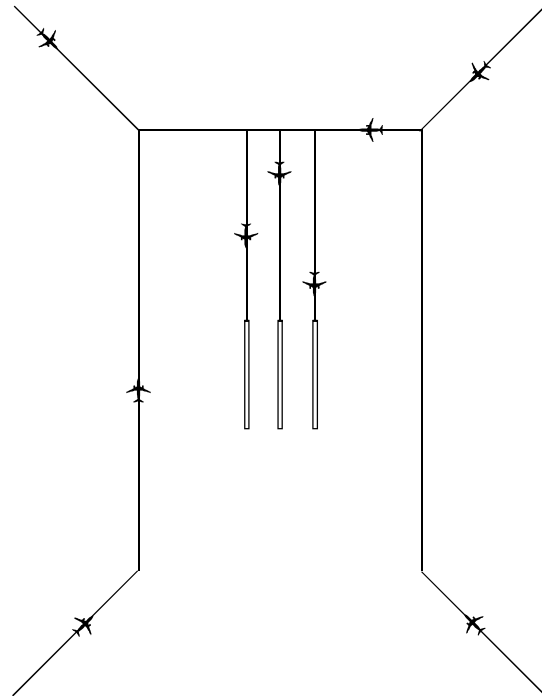


Figure 2. Simple Arrival Traffic Pattern

III. Genetic Search Techniques

This section illustrates four different ways in which the ATC runway assignment, sequencing, and scheduling problem can be formulated as a genetic search.

III.1 Scheme 1: Two-Chromosome Representation for Sequence and Runway Assignment

The first scheme uses two separate chromosomes to represent the flight landing sequence and the runway assignment, respectively. The two chromosomes in the pair together determine the runway assignment, sequencing and scheduling of the flights. For an n -flight problem, each of the chromosomes would have n genes. The *sequence chromosome* defines the relative order of the flights. This relative order between two flights is important only if the two flights are assigned to the same runway. Each flight is assigned the runway indicated by the corresponding gene entry in the *runway chromosome*. The schedule of the flight is determined by ensuring that its STA is minimized as determined by the schedule of the assigned runway. The separation requirements between successive landing flights are enforced in determining the STA. The fitness of the chromosome pair is defined as the sum of all the delays squared. The square of the delay is used instead of just the delay itself in order to encourage dividing up the delay among multiple flights rather than loading up one flight with a long delay. Since the fitness appears as a cost function rather than a benefit function, this problem seeks the chromosome pair with the lowest fitness.

The implementation of this scheme is tested with merely four initial chromosome pairs: The first sequence chromosome is defined by

(1 2 3 4 5 6 7 8 9 10 11 12)

and the first runway chromosome is defined by

(1 2 3 1 2 3 1 2 3 1 2 3)

The remaining three chromosome pairs are

(i) (1 3 5 7 9 11 2 4 6 8 10 12)
(1 1 1 1 1 1 1 1 1 1 1 1)

(ii) (8 9 10 11 12 7 6 5 4 3 2 1)
(2 2 2 2 2 2 2 2 2 2 2 2)

(iii) (8 9 10 11 12 7 6 5 4 3 2 1)
(3 3 3 3 3 3 3 3 3 3 3 3)

Since the sequence has to include all flights without duplication, the sequence chromosome can be any permutation of the flights. On the other hand, the runway chromosome is made up of genes representing the runways, and the same runway can appear a number of times in the chromosome. Of the four initial chromosome pairs, the first pair has the lowest fitness value, which is 691.25. This chromosome means that flight #4, i.e. NW358, would be scheduled as the second flight to land on Runway 1, for example. The overall schedule corresponding to this chromosome pair is given by the following table.

Runway # 1		Runway # 2		Runway # 3	
UA138	11	UA532	17	UA599	8
NW358	12	UA2987	18.5	AA128	9
UA1482	15	NW357	19.5	AA129	10
UA2408	16.5	UA805	20.5	AA399	11.5

To arrive at the runway assignment, sequence, and schedule, a relatively simple schedule decoder is required to interpret the chromosomes. The decoder basically uses the sequence and the runway assignment of each flight to determine the earliest schedule for the flight by taking into account the ETA of the flight to the assigned runway as well as the separation requirement given the preceding flight for the same runway.

Since any permutation of the sequence chromosome would produce a valid sequence list, the mutation operation is used to generate new sequence chromosomes (Figure 3). On the other hand, a crossover operation is used to generate new runway chromosomes. Figure 4(a) describes a general crossover operation. Since the number of flights to be controlled is fixed by the problem, the two-segment crossover operation in Figure 4(b) is used instead in order to preserve the lengths of the chromosomes. The selection of chromosome pairs for operations is randomized by emphasizing low fitness values. Each time through the loop, two new offspring chromosome pairs are generated. In order to keep the population size from growing too large, the two chromosome pairs with the highest fitness values within the population are deleted for every iteration when the population size is above 1000.

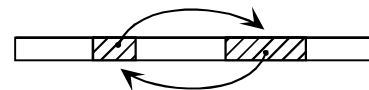


Figure 3. Illustration of Mutation Operation

The following table lists the lowest fitness value of the population after every 1000 generations.

Generations	Lowest Fitness Value
1000	30.25
2000	19
3000	13.5
4000	13.5
5000	13
6000	11.25

After 6000 generations, the sequence chromosome turns out to be

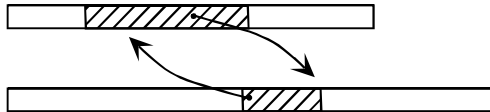
(3 4 12 6 9 11 8 10 1 5 2 7)

with a corresponding runway chromosome

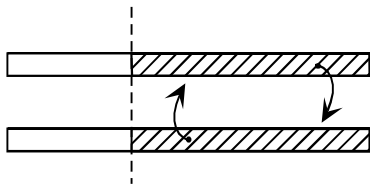
(1 1 3 2 2 3 3 1 3 1 1 1)

This chromosome pair corresponds to the following runway sequence and schedule.

Runway # 1		Runway # 2		Runway # 3	
UA599	6	AA128	6	AA399	5
NW358	7	AA129	7	UA805	6
UA2408	9			NW357	8
UA2987	10			UA138	9
UA532	15				
UA1482	16.5				



(a) General Crossover Operation



(b) Two-Segment Crossover for Preserving Lengths

Figure 4. Illustration of Crossover Operations

III.2 Scheme 2: One-Chromosome Representation for Sequencing and Runway Selection Priority

Unlike the first scheme, this scheme uses only one chromosome definition for the complete runway assignment, sequencing, and scheduling problem. Each chromosome represents a priority list for the flights: Going down the list, each flight is assigned the runway and schedule so that its STA, hence the delay, is minimized. For an n -flight problem, the chromosome would have n genes. The separation requirements between successive landing flights are enforced in determining the STA. As before, the fitness of the chromosome is defined as the sum of all the delays squared, and the problem seeks the chromosome with the lowest fitness.

The implementation of this scheme can be started with a single chromosome, e.g.:

(8 9 10 11 12 7 6 5 4 3 2 1)

This chromosome means that flight #8, i.e. NW357, would be scheduled first. The overall schedule corresponding to this chromosome is given by the following table, and corresponds to a fitness of 29.5.

Runway # 1		Runway # 2		Runway # 3	
NW357	6	AA128	6	UA805	5
AA129	7	UA2987	12	AA399	6.5
UA2408	9			NW358	8
UA1482	15			UA599	9
UA532	16.5			UA138	10

To arrive at the runway assignment, sequence and schedule, a different schedule decoder is required to interpret the chromosome by successively selecting the runway that would provide each flight on the priority list the earliest STA.

Since any permutation of the chromosome would produce a priority list of the flights, the mutation operation is used to generate new schedules. At every iteration two chromosomes are randomly selected for carrying out the mutation operation, and the fitness of the offspring chromosomes is determined. The following table lists the lowest fitness value of the population after every 100 generations.

Generations	Lowest Fitness Value
100	15.5
200	14.25
300	12.5
400	12.5
500	12.5
600	11.25

After 600 generations, the chromosome with the lowest fitness turns out to be

(8 12 11 4 3 10 5 6 7 9 1 2)

This chromosome has a fitness value of 11.25, and it corresponds to the following runway sequence and schedule.

Runway # 1		Runway # 2		Runway # 3	
NW357	6	UA805	6	AA399	5
NW358	7	UA599	7	AA128	6
UA2408	9	AA129	8	UA138	9
UA2987	10				
UA1482	15				
UA532	16.5				

It is interesting to point out that this scheme achieves the best fitness value found by Scheme 1 in fewer generations, even though this scheme selects the chromosomes randomly for the creation operation, while the earlier scheme selects them in a “randomized” fashion emphasizing desirable (i.e. low) fitness values. This improved search performance is made possible by a much smaller search space than the one for Scheme 1. Specifically, the search space for the *population* of the Scheme 2 example involves $12!$ permutations, while the search space for the *sequence population* and the *runway population* used by Scheme 1 together involves $12! \times 3^{12}$ possibilities.

III.3 Scheme 3: Variant of Scheme 2 with Chromosome Selection Based on Fitness

This scheme is identical to Scheme 2 except that selection of the chromosomes for operation is now based on a randomized approach that emphasizes desirable fitness values. The lowest fitness value of the population after every 5 generations is listed in the following table.

Generations	Lowest Fitness Value
5	20.5
10	14.25
15	14.25
20	14.25
25	11.25

It is obvious that in this particular case that the randomized selection approach has the effect of focusing the search on desirable chromosomes and thus a desirable solution can likely be obtained much more quickly. This example shows yet another runway assignment, sequence, and schedule that are different from those obtained by Schemes 1 and 2 but have the same fitness value.

Runway # 1		Runway # 2		Runway # 3	
NW358	6	UA805	6	AA399	5
AA129	7	UA599	7	AA128	6
UA2408	9	NW357	8	UA138	9
UA2987	10				
UA532	15				
UA1482	16.5				

The results from this scheme demonstrate the benefits of using a randomized criterion related to fitness for selecting chromosomes for creation of offspring chromosomes. Other selection methods such as tournament and rank selections can also be used for similar purposes [2].

III.4 Scheme 4: Chromosome Representation of an Algebraic Metric

The previous three schemes all define the chromosomes to represent the flight sequences or runway assignments given a list of arrival flights. This last scheme is different from the previous ones in that chromosomes are defined as mathematical expressions that represent different options for a metric, which would then be used in a recursive algorithm to achieve the runway assignment, sequencing, and scheduling. In other words, the previous schemes need the use of genetic search on-line to perform the runway assignment, sequencing, and scheduling functions, while this scheme uses genetic search off-line to determine a formula defining a metric, which would then be used by an on-line algorithm to carry out the functions. Hence genetic search is not needed for on-line processing.

In contrast to the first three schemes, which are based on *genetic algorithm* formulations, this scheme

manipulates chromosomes representing operations and functions of variables in the spirit of *genetic programming*. This off-line genetic search program generates chromosomes to formulate metrics based on the following four basic terms:

flt_eta - Estimated Times of Arrival for the flights at the individual runways.

min_eta - Minimum Estimated Times of Arrivals among all runways.

last_sta - Scheduled Times of Arrival for the last flights currently scheduled for the runways.

sep_req - Required Separations between the last scheduled flight for the runways and the new flights to be scheduled.

For the metric to be used in selecting the next flight for scheduling and runway assignment, these basic terms are computed for all flights and runways being considered. Hence these terms are stored in $n_f \times n_r$ matrices, where

n_f is the number of flights being considered and n_r is the number of runways. Since the min_eta is the minimum ETA among all runways, all the columns of the matrix are defined to be identical. Similarly, all the rows of the matrix last_sta should be identical since the quantity is independent of the new flights to be scheduled.

An initial set of chromosomes has been defined to promote the generation of different metrics through cross-over operations (Figure 4(a)). The mathematical operations that are most likely to appear in a good solution are incorporated into the chromosome definitions. Thus, addition (+), subtraction (-), element-wise multiplication (*), maximum, minimum, absolute value, and the signum function are included. An initial population of functions is selected as follows:

```

flt_eta
last_sta
sep_req
min_eta
sep_req .* sep_req
last_sta + sep_req
max(flt_eta, sep_req)
min(min_eta, flt_eta)
- sep_req
abs(flt_eta + min_eta)
sign(flt_eta - last_sta)
min_eta .* flt_eta
sign(flt_eta - sep_req)
flt_eta + sep_req
min_eta - last_sta .* max(flt_eta,
last_sta)
sep_req + min(min_eta, last_sta)

```

Given a chromosome and a list of flights to be scheduled, a function subroutine is called to use the chromosome as the basis for defining the metric for

selecting the next flight for scheduling and runway assignment. After the flight with the smallest metric value has been selected and scheduled, it is removed from the list and the function subroutine will call itself in a recursive manner to schedule the remaining flights.

Since this particular implementation selects the flights one at a time without going deeper into a tree search for successive flight scheduling, the resulting schedule is not expected to exhibit the same level of optimality as the ones from the previous examples. On the other hand, after an optimal metric has been defined with this method, the recursive function subroutine can be used to schedule any other list of flights without running the off-line genetic search routine again.

Among the initial chromosomes tested in this scheme, the one with the best fitness is given by `flt_eta`, which corresponds to a first-come first-served schedule. Furthermore, since the minimum ETA for each flight corresponds to landing at either Runway #1 or Runway #3, all of the landings are scheduled for these two runways. The resulting fitness for the given list of flights is 36.5.

Runway # 1		Runway # 2		Runway # 3	
UA599	6			AA128	5
NW358	7			UA805	6
NW357	8			AA399	7.5
AA129	9			UA138	9
UA2987	11				
UA2408	12				
UA532	15				
UA1482	16.5				

After 1000 generations, the Genetic Search routine arrived at a chromosome given by:

$$\max((\text{flt_eta} + -(\text{sep_req})), \text{last_sta})$$

with a fitness value of 12.5. As discussed above, this fitness value is slightly larger than the value of 11.25 provided by the previous schemes. Upon initial inspection, this metric does not represent any physically meaningful quantity. However, the effect of this metric on the recursive selection algorithm is equivalent to that of the following metric:

$$\max(\text{flt_eta}, \text{last_sta} + \text{sep_req})$$

This metric can be obtained from the former one by adding `sep_req` to both terms in the max operation, and hence it does not alter the selection process. This modified metric, however, corresponds to selecting the next flight and runway allocation with the earliest possible scheduled time of arrival. The resulting runway assignment and schedule are given by:

Runway # 1		Runway # 2		Runway # 3	
UA599	6	UA805	6	AA128	5
NW358	7	NW357	7	AA399	6.5
UA2987	9	AA129	8	UA138	9
UA2408	10				
UA532	15				
UA1482	16.5				

IV. Concluding Remarks

Four different genetic-search formulations have been developed and applied to the runway assignment, sequencing, and scheduling problem for multiple flights arriving at an airport with multiple runways. The first formulation includes two separate chromosomal representations, separately for flight sequencing and runway assignment. Scheduling is done automatically to minimize delay given the flight sequence and runway assignment.

The second formulation uses only one chromosomal representation to define an overall sequence or priority list. Runway assignment and scheduling are accomplished automatically, again to minimize delay for each flight successively according to the priority list. Although both of the first two schemes produced results with the same optimum fitness, the second scheme produced the result in substantially fewer generations, as its formulation deals with a much smaller search space.

The third scheme uses the same chromosomal representation as the second scheme, but differs from the second scheme by using a fitness-based probabilistic selection process for identifying “good” chromosomes for genetic operations. The result is an even faster convergence to an optimal solution.

Evaluation results from the first three schemes exemplify the importance of an appropriate chromosomal representation and an appropriate selection process.

The fourth scheme differs in principle from the first three. Its chromosomal representation is formulated with mathematical operations and functions to define a metric, which is used as part of a runway assignment, sequencing, and scheduling computer program. The schedule resulting from applying this program to the example scenario does not produce a solution as good as the previous three schemes. This is an anticipated inherent limitation of the one-flight-look-ahead design of the program. It is conceivable that other genetic-search formulations can produce other programs that would look deeper into the scheduling process, with the potential to automatically create programs capable of providing superior ATC solutions.

References

- [1] H. Erzberger, T.J. Davis, and S. Green, “Design of Center-TRACON Automation System,” *Proceedings of the 56th AGARD Symposium on Machine Intelligence*

in Air Traffic Management, Berlin, Germany, 1993, pp. 11-1–11-12.

- [2] *Genetic Search Toolbox User's Manual*, Optimal Synthesis Inc., 1998-99.
- [3] J.H. Holland, *Adaptation in Natural and Artificial Systems*, Cambridge, MA: The MIT Press, 1993.
- [4] D.B. Fogel, *Evolutionary Computation*, Piscataway, NJ: IEEE Press, 1995.
- [5] D.E. Goldberg, *Genetic Algorithms*, Reading, MA: Addison-Wesley, 1989.
- [6] J.R. Koza, *Genetic Programming*, Cambridge, MA: The MIT Press, 1992.