

GENERATION OF REALISTIC AIR TRAFFIC SCENARIOS USING A GENETIC ALGORITHM

Robert D. Oaks, Signal Corporation

Mike Paglione, Federal Aviation Administration

William J. Hughes Technical Center, Atlantic City International Airport, NJ 08405

Abstract

Traffic flow management decision support tools such as the User Request Evaluation Tool (URET), developed by the MITRE Center for Advanced Aviation Systems Development, and the Center-TRACON Automation System (CTAS), developed by the National Aeronautics and Space Administration/Ames Research Center, use simulation as a tool for development, technical assessment, and field evaluation. Air traffic scenarios based on recorded live data are used to test these decision support tools. Frequently the scenarios need to be modified in order to create aircraft-to-aircraft encounters and conflicts that are not present in the live data. This paper presents an implementation of a genetic algorithm that is being used to time shift the flights within an air traffic scenario to create encounters with specific constrained characteristics. These constraints are the distributions of the horizontal and vertical closest points of approach, the encounter angle at the closest point of horizontal approach, and the vertical type of encounter. This paper describes how the genetic algorithm was implemented, including a description of the solution chromosome and of the fitness function used to measure the potential solutions. After describing the implementation a specific example of its use is presented.

Introduction

Both the User Request Evaluation Tool (URET), developed by the MITRE Center for Advanced Aviation Systems Development, and the Center-TRACON Automation System (CTAS), developed by the National Aeronautics and Space Administration/Ames Research Center, are decision

support tools (DSTs) that support en route air traffic controllers. Each has a conflict probe function that predicts aircraft-to-aircraft and aircraft-to-airspace conflicts.

In 1996 the Federal Aviation Administration's Traffic Flow Management Branch (ACT-250) established the Conflict Probe Assessment Team (CPAT) to evaluate the accuracy of the conflict probes in these DSTs. In 2002, CPAT became a part of the Simulation and Modeling Group (ACB-330). Over the past six years CPAT has measured the conflict prediction accuracy of URET [1], measured the trajectory modeling accuracy of both URET and CTAS [2], and assisted in the accuracy testing of URET Current Capability Limited Deployment (CCLD) [3, 4], which is the operational implementation of URET.

Air Traffic Scenarios

For each of these tasks CPAT used air traffic scenarios, which are data files describing the flow of aircraft traffic over a period of time. The files contain time-stamped planning and advisory information and track data. The planning and advisory information describe the aircraft's planned flight; which includes its flight plan and flight plan amendments, interim altitude clearances, and hold information. The track data represents the aircraft's actual flight path. It consists of several fields including the flight's time-stamped horizontal coordinates and altitude.

Encounters and Conflicts

An aircraft-to-aircraft encounter is an instance when the relative spatial distance between two aircraft is less than some parametric value. This

distance is usually specified in two dimensions: its projection onto a horizontal plane and its projection onto a vertical axis. The values for defining an encounter in this paper are 25 nautical miles (nm) in the horizontal plane and 5000 feet vertically.

An aircraft-to-aircraft conflict is an aircraft-to-aircraft encounter for which these horizontal and vertical distances also violate published air traffic control standards. In en route airspace the horizontal separation standard is 5 nm and the vertical separation standard is either 2000 feet if both aircraft are above FL290 or 1000 feet if one or both aircraft are below FL290. Since encounters and conflicts, as defined in this paper, differ only with regards to distance parameters, the terms are used interchangeably.

Time Shifting

Two specific requirements for the URET CCLD accuracy testing were that the air traffic scenarios had to be based on recorded field data and that these scenarios had to contain a specified minimum number of encounters and conflicts [5].

Recorded field data will contain aircraft-to-aircraft encounters, but under normal operating conditions this data will not contain aircraft-to-aircraft conflicts. In order to meet the URET CCLD accuracy test requirement, CPAT time shifted the flights in the recorded field data.

This time shifting consisted of determining a flight specific time increment that was added to all the events associated with the flight. This caused each flight to follow its recorded flight profile, but at a different time. This caused aircraft-to-aircraft encounters and conflicts to occur in the scenarios that did not exist in the field.

For the URET CCLD accuracy scenarios CPAT developed software that calculated these time increments using time compression and random time adjustment. For time compression the time increment is derived by multiplying a constant times the difference between a flight's start time and a base time that precedes all the start times in the scenario. For random time adjustment the time increment is randomly selected. A more detailed description of these techniques and an overview of CPAT's scenario generation process are presented in Reference [6].

This approach was satisfactory for the URET CCLD accuracy testing, but CPAT realized that the distribution of key encounter parameters (e.g., encounter angle) was not controlled by these techniques. In order to control these parameters CPAT investigated the feasibility of using a genetic algorithm to determine a set of delta times (i.e., flight specific time increments) that can be applied to the flights in a scenario so that the distribution of aircraft-to-aircraft encounters and conflicts meets user defined distribution constraints. The results of this investigation are documented in Reference [7].

The Genetic Algorithm

The genetic algorithm (GA) was invented by John Holland at the University of Michigan in the 1960s and 1970s. GAs are a specific case of a broad class of algorithms called *Random Heuristic Search* [8] algorithms and are considered the most prominent example of evolutionary programming. Comprehensive information regarding the history, study, application, and theory of GAs can be found in the literature. Most of CPAT's implementation of a GA is based on material gleaned from References [9], [10], and [11].

GAs derive their behavior from a metaphor of the biological processes associated with evolution. There is no specific GA; instead a GA is an approach to solving a problem. But all GA approaches have the following traits in common: a population of chromosomes, selection according to fitness, crossover to create new offspring, and random mutation.

CPAT implemented a GA in a program named *Cat*,¹ which was developed using:

- *gcc* Version 2.7.2.3, the GNU C/C++ compiler
- *libg+* Version 2.7.2, the GNU C/C++ libraries
- *Pro*C/C++* Version 8.1.6, the Oracle preprocessor that provides a software interface to tables within an Oracle Version 8.1.6 relational database

¹ *Cat* was named for the character *Cat* on the British television series *Red Dwarf*. *Cat* is a humanized feline; the result of 3,000,000 years of evolution on the space ship *Red Dwarf* after all but one of its crew were killed by a radiation leak.

The goal of *Cat* is to find a set of delta times that can be applied to the flights in a scenario so that the distribution of parameters characterizing aircraft-to-aircraft conflicts meets user defined distribution constraints. The parameters chosen for *Cat* are: the number of conflicts, the horizontal separation distance at closest approach, the vertical separation distance at closest approach, the encounter angle at closest approach, and the vertical type of encounter, which refers to whether the aircraft are in level flight or transitioning vertically. CPAT did not choose these bins arbitrarily. They were chosen based on the bins used for URET CCLD Accuracy Testing [5] and on conflict properties discussed in Reference [12].

The following subsections describe how each of the traits common to all GAs were implemented in *Cat*.

Population of Chromosomes

The first common trait found in all GAs is a population of chromosomes. In a GA, a chromosome is defined as an array of bits or characters that represent a potential solution to a problem. These bits or characters are defined as the chromosome's genes. The values these genes can assume are defined as alleles. A population of these chromosomes is a subset of all solutions to the problem. Usually the initial population is selected randomly.

In *Cat* a chromosome is defined to be a sequence of delta times. A chromosome may be represented by the tuple

$$\langle \Delta t_1, \Delta t_2, \dots, \Delta t_n \rangle$$

where a delta time is associated with each flight and the number of genes is equal to the number of flights in the scenario. These delta times represent the flight specific time shift increment. The granularity of the delta times is 10's of seconds because the track data used by *Cat* has been preprocessed and interpolated to 10-second intervals. For example, the chromosome

$$\langle 0, -75, 9, \dots \rangle$$

means to start the first flight at its original time, to start the second flight 750 seconds earlier than its original start time, to start the third flight 90 seconds later, etc. Since a scenario may contain

thousands of flights, each chromosome may contain thousands of delta times.

The number of chromosomes contained in the population maintained by *Cat* is an input parameter. The initial population consists of chromosomes in which the delta times are selected randomly either from a uniform distribution with a user specified upper and lower range or from a normal distribution with a mean of zero and a user specified standard deviation.

Selection According to Fitness

The second common trait found in all GAs is selection according to fitness. This requires that the implementer must define a fitness function. The fitness function in a GA produces a score for each chromosome, which is a measure of how well the chromosome solves the problem. The fitness of a population may be defined either as the average of all of the fitness scores of the population's chromosomes or as the fitness score of the best individual chromosome in the population. The goal of the GA is to evolve its population until its fitness reaches some desired value.

The fitness of a chromosome in *Cat* is based on how well the distribution of encounters found in a scenario generated with the time shifted flights specified in the chromosome meets the user defined distributions. For *Cat* these distributions are specified in terms of a number of constraint bins.

Constraint Bins

Table 1 identifies the 20 constraint bins used by *Cat*. These bins specify the total number of encounters the user desires and the distribution of those encounters. Note that *Cat* can be run in either of two modes. In the conflict mode the bin sizes represent conflict separation criteria. In the encounter mode the bin sizes are larger.

- Number of conflicts. One constraint bin specifies the number of desired conflicts or encounters (denoted *NbrEncounters*). This constraint bin is specified by a minimum and a maximum bound. For example in a large scenario the user may specify the following range:

$$\circ \quad 247 \leq NbrEncounters \leq 302$$

Table 1. Constraint Bins

Constraint Bins	Conflict Mode	Encounter Mode
1 bin for number of conflicts	1 bin specified by a minimum and maximum bound	
5 bins for horizontal separation	1 nm increments from 0 to 5 nm	5 nm increments from 0 to 25 nm
5 bins for vertical separation	400 ft increments from 0 to 2000 ft	1000 ft increments from 0 to 5000 ft
6 bins for encounter angle	30 degree increments from 0 to 180°	
3 bins for vertical type of encounter	Level-level Level-transitioning Transitioning-transitioning	

- Horizontal constraint bins. There are five horizontal constraint bins, which specify the distribution of the horizontal separation parameter. This parameter refers to the distance measured in the horizontal plane at the closest approach in the horizontal plane (denoted *MinSepHorz*). The bin size differs for encounters and for conflicts. For encounters these bins are in 5 nm increments for 0 nm to 25 nm. For conflicts these are in 1 nm increments from 1 nm to 5 nm. For example in a large scenario the user might specify the following distribution:

 - 60 to 74 conflicts where the *MinSepHorz* is less than 1 nm.
 - 51 to 63 conflicts where the *MinSepHorz* is greater than or equal 1 nm and less than 2 nm.
 - 50 to 61 conflicts where the *MinSepHorz* is greater than or equal 2 nm and less than 3 nm.
 - 55 to 67 conflicts where the *MinSepHorz* is greater than or equal 3 nm and less than 4 nm.
 - 31 to 37 conflicts where the *MinSepHorz* is greater than or equal 4 nm and less than or equal 5 nm
- Vertical constraint bins. There are five vertical constraint bins, which specify the distribution of the vertical separation parameter. This parameter refers to the altitude difference between two aircraft as measured at the closest approach in the horizontal plane (denoted *MinSepVert*). Again, the bin size differs for encounters and for conflicts. For encounters these are 1000 foot increments from 0 feet to 5000 feet. For conflicts these are 400 foot increments from 0 feet to 2000 feet. For example in a large scenario the user might specify the following distribution:

 - 204 to 250 conflicts where the *MinSepVert* is less than 400 feet.
 - 16 to 20 conflicts where the *MinSepVert* is greater than or equal 400 feet and less than 800 feet.
 - 15 to 19 conflicts where the *MinSepVert* is greater than or equal 800 feet and less than 1200 feet.
 - 9 to 11 conflicts where the *MinSepVert* is greater than or equal 1200 feet and less than 1600 feet.
 - 2 to 2 conflicts where the *MinSepVert* is greater than or equal 1600 feet and less than or equal 2000 feet.

Note in this example that this distribution can be satisfied with as few as 246 conflicts yet the number of conflicts bin specifies a minimum requirement of 247 conflicts. This shows that these bins are defined independently, but must both be satisfied to meet the constraints.

Note also that the lower and upper bounds for a constraint bin can be the same. This example shows that the user wants exactly two conflicts in the fifth constraint bin.
- Encounter angle constraint bins. There are six encounter angle constraint bins, which specify the distribution of the encounter angle parameter. This parameter refers to the angle in

the horizontal plane measured at the closest approach in the horizontal plane (denoted *angleClosestApproach*). This angle is 0° for head-on encounters and 180° for in-trail encounters. For either encounters or conflicts these are 30° increments from 0° to 180°. For example in a large scenario the user might specify the following distribution:

- 73 to 90 conflicts where the *angleClosestApproach* is less than 30°.
- 31 to 39 conflicts where the *angleClosestApproach* is greater than or equal 30° and less than 60°.
- 31 to 39 conflicts where the *angleClosestApproach* is greater than or equal 60° and less than 90°.
- 26 to 33 conflicts where the *angleClosestApproach* is greater than or equal 90° and less than 120°.
- 42 to 53 conflicts where the *angleClosestApproach* is greater than or equal 120° and less than 150°.
- 44 to 48 conflicts where the *angleClosestApproach* is greater than or equal 150° and less than or equal 180°.

• Vertical type of encounter constraint bins.

There are three vertical type of encounter constraint bins, which specify the distribution of the vertical type of encounter parameter. This parameter refers to whether the flights are in level flight or if they are either climbing or descending. For either encounters or conflicts these are level-level, level-transitioning, and transitioning-transitioning, where level refers to a flight in level flight and transitioning refers to a flight that is either climbing or descending. For example in a large scenario the user might specify the following distribution:

- 90 to 110 conflicts where both aircraft are level.
- 97 to 119 conflicts where one aircraft is level and the other aircraft is transitioning.
- 59 to 73 conflicts where both aircraft are transitioning.

The ranges for these constraint bins are input to *Cat* as 20 pairs of constraint bounds – a low bound and a high bound for each of the constraint bins. These bounds are denoted *lobound_i* and *hibound_i* for $i = 1, 20$, and are used in Equations (4), (5), and (6).

Fitness Function

Two fitness functions are available in *Cat*: a linear fitness function and an exponential fitness function. Each of these functions reward both individual and multiple instances where the tallied counts, denoted *count_i* for $i = 1, 20$, fall within these constraint bounds. At the same time the functions penalize instances in which the measured value is below the low bound or above the high bound.

The linear fitness function is defined as:

$$F = \frac{X}{20} \quad (1)$$

The exponential fitness function is defined as:

$$F = \frac{2^X}{2^{20}} \quad (2)$$

Where for both Equation (1) and Equation (2) X is defined as:

$$X = \sum_{i=1}^{20} x_i \quad (3)$$

The variable x_i represents the individual contribution that bin i provides to the fitness score. The calculation for x_i depends on three conditions.

1. If the tallied count for a bin lies below the lower bound (i.e., if $count_i < lobound_i$) then

$$x_i = \left(\frac{count_i}{lobound_i} \right)^2 \quad (4)$$

2. If the tallied count for a bin falls within the lower and upper bounds (i.e., if $lobound_i \leq count_i < hibound_i$) then

$$x_i = 1 \quad (5)$$

3. If the tallied count falls above the upper bound (i.e., if $hibound_i < count_i$) then

$$x_i = \left(\frac{hibound_i}{count_i} \right)^2 \quad (6)$$

For each of these equations, $lobound_i$ refers to constraint bin i 's low bound, $hibound_i$ refers to its high bound, and $count_i$ refers to the number of encounters tallied to be in the constraint bin.

The value of x_i provides a value of 1.0 if the tallied count in its bin is between the low and high bounds and decreases to 0.0 as the count gets further away from either of the bounds. This is seen in Figure 1, which is a plot showing the individual contribution for a constraint bin as a function of tallied count ($count_i$). In this figure the two vertical dashed lines represent the bin's low bound and upper bound.

Once the individual values of x_i are computed for each of the 20 constraint bins, X is computed as their floating-point sum as defined in Equation (3). This results in X being a weighted sum representing the number of bins in which the constraints have been satisfied.

The fitness (F) is then calculated as defined in either Equation (1) or Equation (2), depending on user input. Figures 2 and 3 are plots of how fitness (F) varies as a function of the sum of the individual contributions (X) when using Equation (1) and Equation (2), respectively. Note in both of these plots the sum of the individual contributions ranges from 0 to 20, where 20 is the maximum value for X which is achieved when all constraints have been met.

Selection Technique

The first step a GA takes in evolving a new generation of chromosomes is to select parent chromosomes from the current population. In all GAs the chance of a chromosome being selected to be a parent is based on the chromosome's fitness.

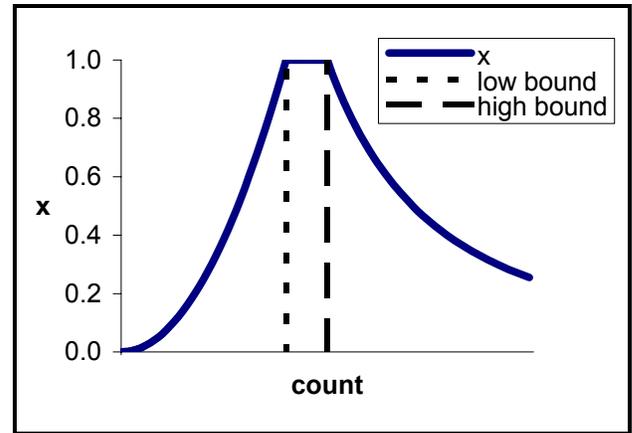


Figure 1. Constraint Bin Contribution to Fitness

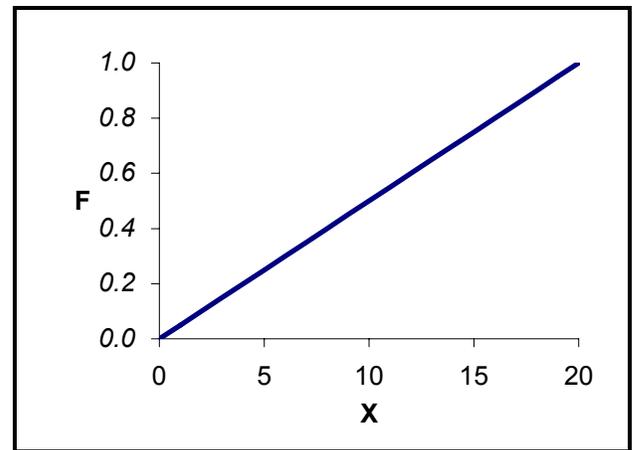


Figure 2. Linear Fitness Function

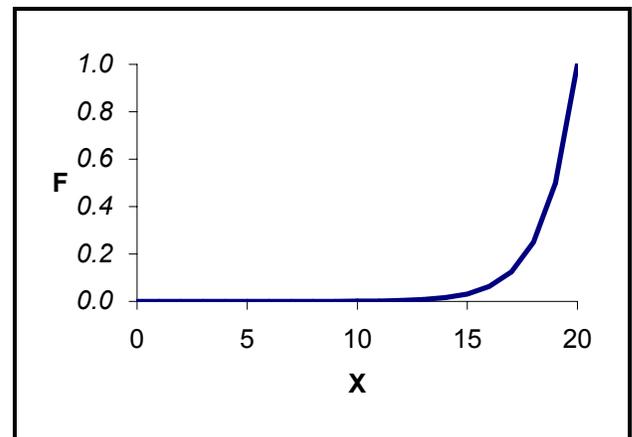


Figure 3. Exponential Fitness Function

In *Cat* the number of parent chromosomes selected is the same as the number of chromosomes in the population and the specific selection technique used by the GA is a user option. The following selection techniques are available in *Cat*:

- **Fitness Proportionate Selection**. This is the original selection technique developed by Holland. In this technique the chance of a chromosome being selected is directly proportional to the chromosome's contribution to the total fitness of the population.
- **Sigma Scaled Selection**. This technique favors chromosomes with a fitness value close to the average fitness of the population.
- **Boltzmann Selection**. This technique uses the Boltzmann distribution so that the selection criteria vary with time. When using this selection technique a GA is similar to simulated annealing.
- **Rank Selection**. In this technique the chromosomes are ranked according to their fitness. Then the chance of a chromosome being selected depends on the rank order of the chromosome. This technique keeps a GA from converging too quickly.
- **Tournament Selection**. In this technique pairs of chromosomes are randomly chosen and the chromosome with the higher fitness value is selected.

During any of these selection processes it is statistically possible that none of the most fit chromosomes will get selected. Because of this possibility *Cat* uses an algorithm called *stochastic universal sampling* with the Fitness Proportionate Selection, Sigma Scaled Selection, and Boltzmann Selection techniques, since each of these techniques select chromosomes based on each chromosome's expected representation in the selected population. The *stochastic universal sampling* algorithm ensures that fit chromosomes are not statistically lost in the selection process.

Crossover to Produce New Offspring

The third common trait found in all GAs is crossover to produce new offspring. In this step the chromosomes in the selected parents are paired and some of their genes may be swapped.

In *Cat* the selected parents are paired serially and the occurrence of crossover depends on an input parameter P_c , which is the probability of crossover. When crossover occurs, the genes between two parent chromosomes are swapped using either single-point or two-point crossover depending on user input. If crossover does not occur the selected parent's chromosomes become the offspring chromosomes.

Single-Point Crossover

In single-point crossover a single locus point is randomly selected and the genes beyond that point are swapped between the two parent chromosomes. For example given the two parent chromosomes

<1, 2, 3, 4, 5, 6, 7, ...>

<90, 80, 70, 60, 50, 40, 30, ...>

if the randomly selected locus point is between the third and fourth genes, the resultant offspring chromosomes would be

<1, 2, 3, **60, 50, 40, 30**, ...>

<90, 80, 70, **4, 5, 6, 7**, ...>

where the swapped genes are identified in the bold font.

This is the simplest form of crossover, but has limitations including an end point effect that is the result of always including a chromosome's end genes in the swap.

Two-Point Crossover

In two-point crossover two loci points are randomly selected and the genes between those two points are swapped between the two parent chromosomes. This technique reduces the end point effect, which may occur when using single-point crossover. For example given the two parent chromosomes

<1, 2, 3, 4, 5, 6, 7, ...>

<90, 80, 70, 60, 50, 40, 30, ...>

if the first randomly selected locus point is between the second and third genes and the second randomly selected locus point is between the sixth and seventh genes, the resultant offspring chromosomes would be

<1, 2, **70, 60, 50, 40**, 7, ...>

<90, 80, **3, 4, 5, 6**, 30, ...>

where the swapped genes are identified in the bold font.

Random Mutation

The fourth common trait found in all GAs is random mutation. This is the final step in evolving a new generation of chromosomes during which each of the genes in the chromosomes of the offspring population is considered for mutation.

In *Cat* when mutation occurs the gene is randomly changed to another valid value (allele), which is a delta time selected randomly from either a uniform distribution in a user specified interval or a normal distribution with a zero mean and a user defined standard deviation. In *Cat* the occurrence of mutation depends on an input parameter P_m , which is the probability of mutation. For example given the following chromosome

<1, 2, 3, 4, 5, 6, 7, ...>

if the fourth gene were randomly selected to mutate, it may be changed to

<1, 2, 3, **90**, 5, 6, 7, ...>

where the mutated gene is identified in the bold font.

Additional Features

Two additional features that CPAT implemented in *Cat* are elitism and a resume capability.

Elitism

Elitism is a technique that cheats the GA's metaphor. As implemented in *Cat* a user specified number of the highest ranking chromosomes are retained prior to the selection step. Then, after the selection-crossover-mutation steps, the worst chromosomes are replaced by these elite chromosomes. As a result *Cat* can be ensured that the best chromosomes in a given generation are present in the succeeding generation. Elitism has been proven to improve a GA's performance.

Resume Capability

When running *Cat* sufficient information is written to a file during each generation so that another run can be initiated starting with that generation. This capability has proved to be useful because it provides the ability to start up a run that

may have terminated prematurely. It also provides the ability to restart a run with different constraint bin values.

Pseudocode

Figure 4 contains high-level pseudocode that describes *Cat*.

```

1 Initialization
2 do {
3   evaluate the fitness of each chromosome
4   save the elite
5   select a parent population
6   create offspring using crossover
7   randomly mutate offspring chromosomes
8   save the new population.
9 } while (termination criteria not met) ;

```

Figure 4. Genetic Algorithm Pseudocode

- Line 1 represents *Cat*'s initialization. During initialization user input is processed and flight information is selected from tables within the Oracle database. This flight information is stored in memory so the program no longer needs to access the tables.
- Lines 2 through 9 represent the processing loop. Each iteration represents the evolution of a single generation.
- Line 3 represents the evaluation of the fitness score for each chromosome in the population. The fitness calculation requires that each track point for every flight must be compared with each track point for every other flight. Even though gross filter techniques are used this function is $O(n^2)$. To achieve a reasonable run time *Cat* calculates the fitness of each chromosome in a child process that returns its result via interprocess communication.
- Line 4 represents the elitism implementation. The chromosomes are first rank ordered, and then a user specified number of the best chromosomes are saved. In each successive generation these saved chromosomes replace the worst chromosomes in the population.
- Lines 5, 6, and 7 represents the selection, crossover, and mutation processes.

- Line 8 represents when data associated with the current generation is output to a file. This provides the resume capability. Additional files are also created that can be viewed by the user to monitor *Cat*'s progress.
- Line 9 represents the end of the processing loop. The loop terminates if a user specified maximum number of generations is achieved or if the value of one of the chromosomes reaches a user specified value.

Example

The following example is based on field data recorded at the Atlanta Air Route Traffic Control Center. The non-time shifted scenario covers a time period of approximately four hours and contains 1545 distinct flights. As expected this field data contains no aircraft-to-aircraft conflicts.

Cat was given the data in the columns labeled Low and High in Table 2 as the low and high constraint bounds. This represents an increase from no conflicts to a significant number of conflicts (viz. between 247 and 302) with user specified distributions for the horizontal distance at closest approach, vertical distance at closest approach, encounter angle, and transition mix. (The column labeled Count represents the solution count and will be discussed later.)

Additional user input to *Cat* specified the following:

- The population size was 20.
- Fitness was calculated using the exponential fitness function defined in Equation (2).
- The Sigma Scaled selection technique was used.
- The Probability of Crossover was set to 0.75.
- Two-point crossover was used.
- The Probability of Mutation was set to 0.01
- A normal distribution with a mean of 0.0 and a standard deviation of 300 seconds was used for establishing the initial population and for random mutation.
- Four elite chromosomes were retained for each successive generation.

Cat was launched with this input on a Sun Ultra 60 workstation with dual 450 MHz processors under the Solaris 8 operating system interfacing with an Oracle 8.1.6 relational database.

After 126 generations the population had found a solution (i.e., one of the chromosomes had a fitness score of 1.0) and the average fitness score of population was 0.94003. The column labeled Count in Table 2 presents the actual conflict count for each of the constraint bins in the solution. In all cases the conflict distribution lies within the user requested bounds. This took approximately 7 hours and 45 minutes.

Table 2. Example Constraint Bins

Constraint	Low	Count	High
Number of Conflicts	247	269	302
Horiz: 0 to 1 nm	60	62	74
Horiz: 1 to 2 nm	51	54	63
Horiz: 2 to 3 nm	50	59	61
Horiz: 3 to 4 nm	55	60	67
Horiz: 4 to 5 nm	31	34	37
Vert: 0 to 400'	204	220	250
Vert: 400 to 800'	16	19	20
Vert: 800 to 1200'	15	17	19
Vert: 1200 to 1600'	9	11	11
Vert: 1600 to 2000'	2	2	2
Angle: 0 to 30°	73	90	90
Angle: 30 to 60°	31	32	39
Angle: 60 to 90°	31	35	39
Angle: 90 to 120°	26	26	33
Angle: 120 to 150°	42	42	53
Angle: 150 to 180°	44	44	48
Level-level	90	90	110
Level-transitioning	97	113	119
Transitioning-transitioning	59	66	73

Since each of the chromosomes for this run contains 1545 genes, space prohibits presenting the individual chromosomes in this example. However, Table 3 presents the individual sorted fitness scores for the first three generations. As expected, the first generation (Gen #0) has low fitness scores since these 20 chromosomes contain random genes. The chromosome with the best fitness in generation #0 had a fitness score of 0.0886440 and the average fitness of the population was 0.0235534. The successive generations show an overall improvement in the fitness scores with best fitness values of 0.0911046 and 0.1023070 and average

fitness scores of 0.0438816 and 0.0651789, respectively. Table 3 also shows how the elite chromosomes from a previous generation are retained; these elite chromosomes are represented by the bold font in the columns labeled Gen #1 and Gen #2

Table 3. Fitness Scores for Initial Generations

Gen #0	Gen #1	Gen #2
0.0886440	0.0911046	0.1023070
0.0488699	0.0886440	0.0953004
0.0321908	0.0863275	0.0912814
0.0313339	0.0712683	0.0911046
0.0274708	0.0706393	0.0886440
0.0264061	0.0547135	0.0884806
0.0241474	0.0499142	0.0863275
0.0241063	0.0488699	0.0846547
0.0196521	0.0412677	0.0826284
0.0194637	0.0321908	0.0803263
0.0189504	0.0313339	0.0733514
0.0182859	0.0312214	0.0712683
0.0156847	0.0292689	0.0465129
0.0144459	0.0288748	0.0415218
0.0132846	0.0266635	0.0331802
0.0110749	0.0264580	0.0323595
0.0105811	0.0181483	0.0321876
0.0100104	0.0177470	0.0305996
0.0084683	0.0166889	0.0291958
0.0079964	0.0162869	0.0223448

All but 38 of the 1545 flights were time shifted by *Cat*. However, the amount of time shift was not excessive. The average time shift was only -7.5 seconds (earlier in time) with a standard deviation of 301.9 seconds. The most a flight was shifted earlier in time was 950 seconds. The most a flight was shifted later in time was 880 seconds. These results are consistent with the user input. Figure 5 shows a histogram showing the frequency distribution of the time shifts for all of the flights. This means that *Cat* was able to derive data to generate a scenario with the desired number of conflicts and the desired distribution of conflict parameters by time shifting the flights no more than 16 minutes from their original times.

Figure 6 shows a plot of fitness versus generation. The thicker line represents the fitness value of the chromosome with the highest fitness for in a generation. The thinner line represents the average fitness value for the generation.

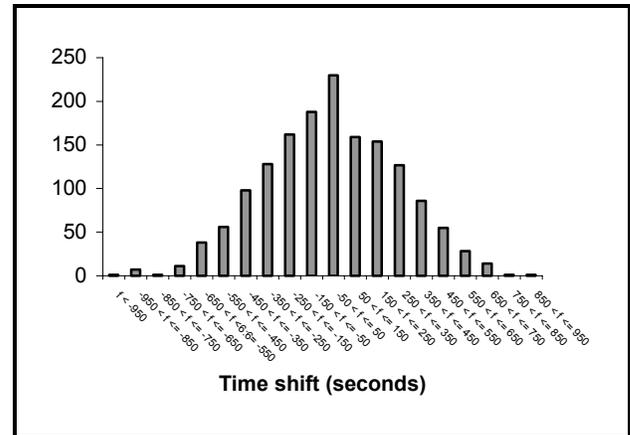


Figure 5. Frequency Distribution of Time Shift

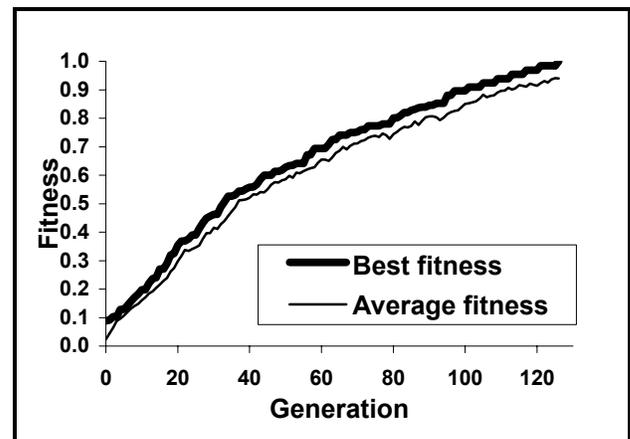


Figure 6. Fitness vs. Generation

Acronyms

- CCLD Core Capability Limited Deployment
- CPAT Conflict Probe Assessment Team
- CTAS Center TRACON Automation System
- DST Decision Support Tool
- FL Flight Level
- GA Genetic algorithm
- GNU GNU's Not Unix
- URET User Request Evaluation Tool

References

- [1] Cale, Mary Lee, Michael Paglione, Dr. Hollis Ryan, Dominic Timoteo, Robert Oaks, April 1998, "URET Conflict Prediction Accuracy Report," DOT/FAA/CT-TN98/8, WJHTC/ACT-250.
- [2] Paglione, Mike, Dr. Hollis F. Ryan, Robert D. Oaks, J. Scott Summerill, Mary Lee Cale, May 1999, "Trajectory Prediction Accuracy Report User Request Evaluation Tool (URET)/Center-TRACON Automation System (CTAS)," DOT/FAA/CT_TN99/10, WJHTC/ACT-250.
- [3] Conflict Probe Assessment Team, November 2000, CD-ROM, "URET CCLD Final Accuracy Scenario Delivery Refresh Data, Revision 1," Federal Aviation Administration, Engineering and Integration Branch, ACT-250, William J. Hughes Technical Center, New Jersey, 08405.
- [4] Conflict Probe Assessment Team, October 2001, CD-ROM, "URET CCLD Final Accuracy Scenario Delivery Scenario Data, Revision G," Federal Aviation Administration, Engineering and Integration Branch, ACT-250, William J. Hughes Technical Center, New Jersey, 08405.
- [5] Lockheed Martin Air Traffic Management, August, 1998, "User Request Evaluation Tool (URET) Core Capability Limited Deployment (CCLD) System Specification (SSS), Volume I, Part 2: Conflict Probe," Lockheed Martin Air Traffic Management, Rockville, MD.
- [6] Oaks, Robert, Mike Paglione, Fall 2001, "Generation of Realistic Air Traffic Scenarios Based on Recorded Field Data," *46th Annual Air Traffic Control Association Conference Proceedings*, Arlington, VA, pp.142-146.
- [7] Oaks, Robert D., August 2002, "A Study on the Feasibility of Using a Genetic Algorithm to Generate Realistic Air Traffic Scenarios Based on Recorded Field Data," paper presented at the AIAA Guidance, Navigation, and Control Conference, Monterey, CA
- [8] Vose, Michael D., 1999, *The Simple Genetic Algorithm: Foundations and Theory*, Cambridge, MA, The MIT Press.
- [9] Goldberg, David E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA, Addison-Wesley.
- [10] Michalewicz, Zbigniew, 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, Third Edition, New York, NY, Springer-Verlag.
- [11] Mitchell, Melanie, 1998, *An Introduction to Genetic Algorithms*, Cambridge, MA, The MIT Press.
- [12] Bilimoria, K. D. and H. Q. Lee, August, 2001, "Properties of Air Traffic Conflicts for Free and Structured Routing," paper presented at the AIAA Guidance, Navigation, and Control Conference, Montreal, Canada.