Reverse Engineering of Foreign Missiles via Genetic Algorithm

Jon D. Wollam

AFIT/GSE/ENY/99D-01

REVERSE ENGINEERING OF FOREIGN MISSILES VIA GENETIC ALGORITHM

THESIS

Jon D. Wollam
AFIT/GSE/ENY/99D-01

AFIT/GSE/ENY/99D-01

Disclaimer Statement

AFIT/GSE/ENY/99D-01

REVERSE ENGINEERING OF FOREIGN MISSILES VIA GENETIC ALGORITHM

THESIS

Presented to the Faculty of the Graduate School of Engineering of

the Air Force Institute of Technology

Air University

in Partial Fulfillment of the Requirements for the

Degree of Master of Science (Systems Engineering)

Jon D. Wollam, B.S.

December, 1999

AFIT/GSE/ENY/99D-01

REVERSE ENGINEERING OF FOREIGN MISSILES VIA GENETIC ALGORITHM

Jon D. Wollam, B.S.

Approved:

[illegible] 15 NOV 99

Lt Col Stuart Kramer, PhD — Date

[illegible] 22 NOV 99

Maj David R. Jacques, PhD — Date

E. Price Smith 19 NOV 99

Lt Col E. Price Smith, PhD — Date

Table of Contents

page

List of Figures

List of Tables

List of Symbols

a	–	altitude
A_f	–	fin area
A_w	–	wing area
D	–	diameter
$F(x)$	–	fitness function
i	–	individual missile subscript
j	–	particular waypoint subscript
L	–	length overall
L_n	–	length of nose section
L_e	–	length of equipment section
L_w	–	length of warhead section
L_p	–	length of propellant section
L_{bt}	–	length of boat tail
M	–	mass
N	–	number of waypoints
$nose$	–	nose cone profile
o	–	observed missile subscript
R	–	range
t	–	time
T/W	–	thrust-to-weight ratio
v	–	velocity
X_w	–	location of wing
ρ	–	density

AFIT/GSE/ENY/99D-01

Abstract

One mission of the National Air Intelligence Center (NAIC) is the reverse engineering of foreign missile weapon systems from incomplete observational data. In the past, intuition and repeated runs of a missile performance model were required to converge to a solution compatible with observed flight characteristics. This approach can be cumbersome and time-consuming, as well as being subject to undesirable influences from the analyst's preconceptions and biases.

An alternative approach has been created to apply genetic algorithm (GA) techniques to allow automation of the process, wider exploration of the design space, and more optimal solutions matching the observational data. The GA, when interfaced with a missile performance model, was able to identify a set of missiles that very closely matched the observed performance of a given sample missile. The approach was able to provide the analyst with multiple candidate missiles for further analysis that would have been missed by the previous trial and error approach.

REVERSE ENGINEERING OF FOREIGN MISSILES VIA GENETIC ALGORITHM

I. Introduction

1.1 Background

The National Air Intelligence Center (NAIC) is an organization of the United States Air Force located at Wright-Patterson Air Force Base in Dayton, Ohio that works closely with the Defense Intelligence Agency of the U.S. government. These agencies are responsible for understanding the military capabilities of foreign countries.

New aerodynamic weapon systems are continually being designed and tested by foreign governments. In order for the U.S. government to be prepared for any future conflict against these potential adversaries, it is important that we understand other military weapon systems and can defend against them. The problem is that foreign militaries do not volunteer all the information that is needed to sufficiently describe the abilities of their weapon systems. If they did, it would be possible to counter them with defensive systems of our own thus giving the U.S. military an advantage. Therefore, intelligence production organizations such as NAIC are necessary for the determination of foreign military potential.

The Air Force collects information with a variety of methods, most of which are heavily classified. Engineering performance measurements on foreign missiles may be obtained by public release of information or by human intelligence. Additionally, missile telemetry data may be determined using signal intelligence. Each method provides certain features of the missile or its performance to the NAIC analyst. For example, waypoint information can be generated from an observed test flight such as altitude, range, and velocity versus time. It is the job of the analyst to take these knowns about the missile and reverse engineer its design and estimate its complete military capabilities.

In the past, human intuition and repeated runs of a missile performance program were required to converge to a solution compatible with observed flight characteristics. This approach required multiple user-defined iterations in order to arrive at a reasonable fit to the observed data.

The analyst would first choose particular values of missile design variables such as diameter, length, and mass based upon prior experience and knowledge of missile aerodynamics. The analyst would then attempt to fly this unrefined missile along an observed flight profile. A predicted flight trajectory was calculated using the missile design software, Missile Integrated Design Analysis System (MIDAS), and deviations between the predicted and the observed trajectories were noted by the analyst. Changes to the missile design were then made with the goal of improving the trajectory match. The analyst would then repeat this process until a design was arrived at which more closely performed as the observed missile.

This iterative approach is cumbersome and time-consuming, and is subject to undesirable influences based on the analyst's preconceptions and biases. Because of this methodology, a single solution may be produced when many different possible solutions exist. The entire design space was not being explored very well in this trial and error approach. An alternative approach would be to automate the process and take the human out of the loop.

1.2 Problem Statement

The basic problem of matching missile flight trajectories can be viewed as an optimization problem. The goal of this process is to identify a missile design that can exactly match the waypoints of the observation. It could be viewed as performing a statistical least squares fit of the observed waypoints with a predicted missile trajectory [Mendenhall 1992].

The hypothesis of this thesis is that a genetic algorithm (GA) can solve this type of problem. Various design parameters of the missile can be chosen randomly, then run through the same trial and error process as the analyst had done. The genetic algorithm learns from successive design iterations, retains the characteristics of the missiles that better match the observation, and attempts to improve the match with automatic successive approximations. The advantage to using this approach is not only a quicker solution of the design problem, but also an unbiased and more in-depth search of the potential missile space.

1.3 Scope and Limitations

This study was designed to show proof of concept. The analysts at NAIC are interested in whether or not the genetic algorithm can assist in the preliminary stages of reverse engineering design. Therefore, a hypothetical missile observation has been fabricated that does not represent any actual foreign missile.

1.4 Thesis Overview

This thesis is organized into four main sections. The first section provides background in optimization and the general theory of genetic algorithms, and briefly introduces the missile design software MIDAS. The second section deals with the methodology of the study. This section explains how the missile design parameters were chosen and how the genetic algorithm and MIDAS were tied together into a single process. The remaining two sections discuss two sample reverse engineering design problems. The first problem is based on very little information about the foreign missile observation, while the second assumes that more information is available. After a brief summary, suggestions for further follow-on study are proposed.

II. Fundamentals

2.1 General Optimization

Optimization, simply stated, is the process of finding the minimum or maximum of some mathematical function. An optimization problem involving a single variable can usually be solved easily. A first course in calculus teaches that taking the derivative of an analytical function, setting it equal to zero, and solving results in the critical points of that function including the optimal minimum and maximum. Functions of two or more variables can be solved in a similar method. However, functions of multiple variables can sometimes be very difficult to solve.

Functions may also have multiple extrema, or local maximum or minimum points. Standard optimization techniques such as hill-climbing and branch-and-bound, have difficulty locating multiple peaks or dips. An example of this is the Himmelblau function of two variables shown in Figure 1 [Reklaitis 1983]. This function has four maximums of equal value. In trying to locate these extrema, standard hill-climbing techniques would start at some location and climb to the top of one of these peaks. This single peak would be declared a local maximum. Searching with this approach provides no information about the other peaks of equal value, nor does it prove anything about the global maximum. The global extrema could be located somewhere else entirely in the design space.

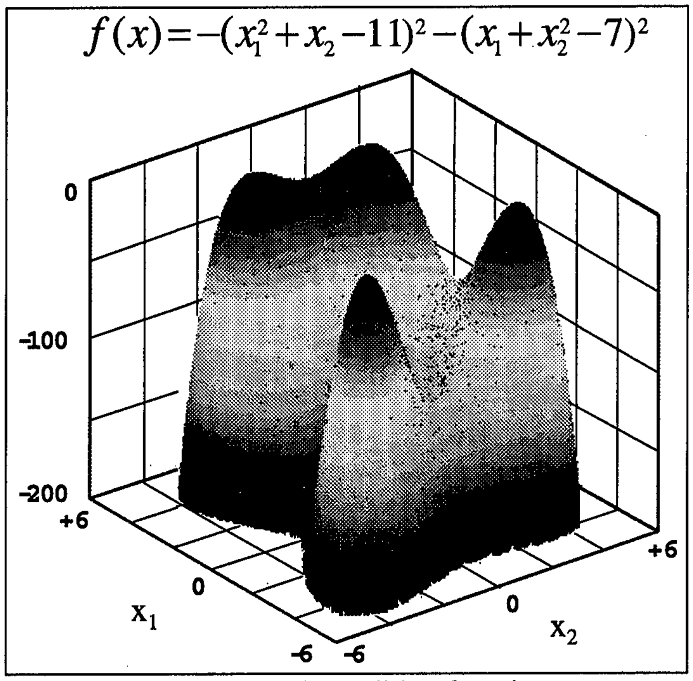

Figure 1 – Himmelblau function

Genetic algorithms are designed to search a much wider area of the design space, and potentially could provide a set of optimal solutions to a given problem. The GA approach was selected for this study because multiple extrema were expected in the reverse engineering problem. It was believed that several different missile designs could match a given set of observations. It is important for the intelligence analyst to explore as much of the potential design space as possible, before selecting a single missile design.

2.2 Genetic Algorithm Theory

The GA is based upon Charles Darwin's survival of the fittest theory of evolution. It copies the natural selection and reproduction processes of biological populations in order to strengthen the fitness of the overall population.

In biological evolution, the individuals of the population are each different. They may be differentiated by their height, weight, eye color, hair color, skin color, strength, intelligence, and so on. These characteristics or variables are encoded as chromosomes within a string of DNA that completely defines the individual. The measure of performance, in this biological example, is the fitness of the individual as defined by the environment in which the population lives. The individuals with a better or higher fitness tend to thrive and reproduce more readily, thereby allowing their desired traits to be passed on to their children, while the weaker individuals perish. Over several generations, the population becomes optimal in its environment as defined by their fitness [Darwin 1859].

The GA, as an optimization methodology, is set up in the same manner. That is, individuals are defined by some binary encoding of variables and compete with the rest of their population for survival. The most fit individuals reproduce and pass the desired traits on to future generations, while less fit individuals perish. After several generations, the population tends to cluster around the optimum.

The characteristics or design variables which describe the individuals in a population are binary encoded into what essentially represents a string of DNA. Consider for a moment a population of automobiles. Discrete variables such as the automobile color can be individually modeled using binary encoding, while continuous variables such as automobile fuel economy can be discretized within a bounded region.

As an example of discrete variable encoding, assume that there are only 4 different automobile colors to choose from: blue, red, yellow, and green. The binary encoding of these colors could be defined as: 00=blue, 01=red, 10=yellow, and 11=green. Only 2 binary bits are needed to completely describe the color of an individual automobile. If more than 4 colors are desired, more bits in the binary code would be required. For example, 3 bits represent 8 different colors, 4 bit represent 16 different colors, and so on, in powers of 2.

As an example of continuous variables, consider fuel economy. Assume the bounds for a particular design to be within 10 and 30 miles-per-gallon (mpg). If an accuracy of 1.0 mpg is required, a binary encoding of 5 bits is needed for 32 increments between the minimum and maximum. The bits of 00000 would represent the minimum of 10 mpg, while the bits of 11111 would represent the maximum of 30 mpg. Each binary increment between these bounds would be 0.625 mpg apart, thus making the 32 increments between the bounds. If more precision were needed, a discretization of 6 binary digits would give 64 increments each 0.3125 mpg apart, 7 digits would give 0.15625, and so on.

Once the traits of an individual are defined, a method is needed to determine the relative goodness of individuals. This is accomplished by creating a cost function or fitness function, *F(x)*, that depends upon the values of each of the design variables. In essence, this is the standard objective function that any optimization technique requires. More weight may be placed on some design variables than others, but each contributes some positive or negative contribution to the overall fitness of the individual.

In order to create subsequent generations, the individuals must compete against each other for the rights of procreation. The more fit individuals tend to reproduce more often, while the less fit tend to die out. This exists within the genetic algorithm as well as in biological populations. The reproduction strategy of tournament selection ranks the population from most to least fit, and begins a random process of selecting parents with the goal of creating children. Two parents are selected to produce two children. Parent individuals, with a high fitness, will be chosen to reproduce more often than individuals with a lower fitness. Over time, the stronger traits will be retained while the weaker traits vanish.

The children in most GA techniques replace the parents in the next generation. Some selection techniques, however, allow parents to compete with their children for entry into the next generation. Also, an elitist strategy may ensure that the best individual of the current generation is cloned into the next generation. This prevents the best traits from accidentally dying off.

Crossover techniques define the chromosomal make-up of the children by mimicking the natural processes of DNA reproduction. The binary strings of both parents are combined in some way to produce their children. There are two basic types of reproduction: single-point and uniform crossover. Single-point crossover forces a single break in the binary code of the parents so that each child obtains some chromosomes from each parent. A break is made randomly somewhere within the parents' binary string. One child gets the binary code of one parent to the left of the break, while the binary code to the right of the break comes from the other parent. The other child gets the opposite. Each child carries on certain traits from both parents in this manner. In a similar manner, uniform crossover allows multiple breaks in the binary code instead of just one.

After selection and crossover, mutations are also permitted to explore regions of the design space that may have already become extinct or never been explored. A jump mutation swaps two random bits within the child's binary string, and a creep mutation randomly selects a bit to be changed. The standard GA flow that has just been described is shown in Figure 2. The exit criteria may be set at some given number of generations, or after some measure of convergence has been reached.

Eventually, the population will tend to converge to a common point. This would occur if the GA were allowed to run for an extended amount of time. An illustration of this can be seen in human biological evolution. The Europeans and Africans each progressed down separate paths. The Europeans developed pale skin, while the Africans developed dark skin. These isolated populations are said to have converged because each individual, within the separated populations, holds a common trait. The GA, as an optimization tool, can also arrive at this kind of convergence in design within the design parameters.

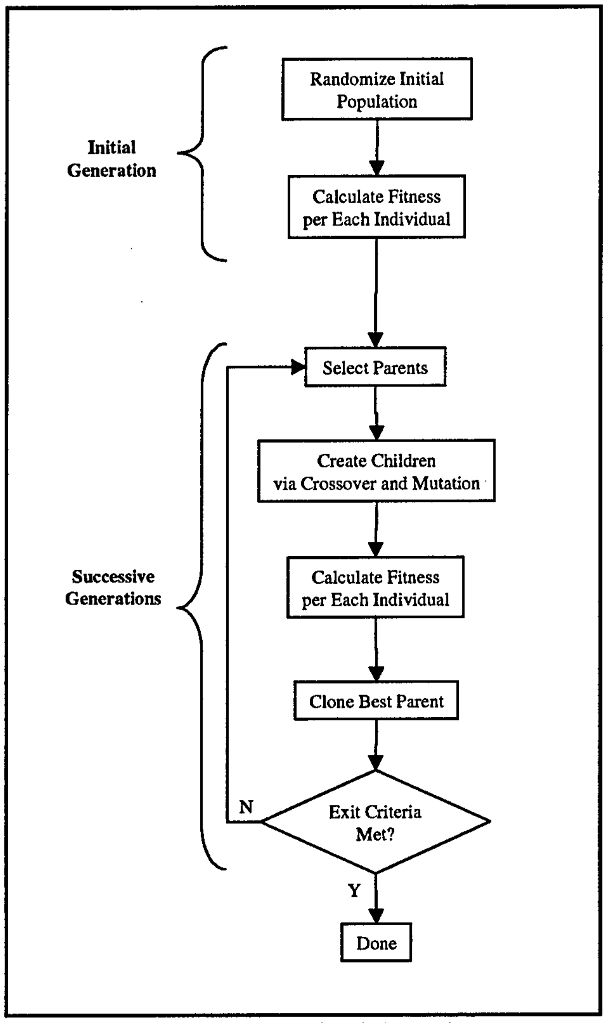
Initial Generation
Randomize Initial Population
Calculate Fitness per Each Individual
Successive Generations
Select Parents
Create Children via Crossover and Mutation
Calculate Fitness per Each Individual
Clone Best Parent
Exit Criteria Met?
N
Y
Done

Figure 2 – Standard GA Flow

The GA should also be run several times to account for this convergence and the inherent random processes. The initial creation of individuals, selection of parents, crossover reproduction, and child mutations are all based on random number draws. Different results can be expected between one initial randomization seed and another. On the other hand, these differences are not guaranteed and different seeds could end with the same results. It is important that several different initialization seeds be performed for any given problem that has some degree of randomization [Banks 1996].

The genetic algorithm used for this study [Carroll 1998] incorporates tournament selection, binary coding, both jump and creep mutation, and either single-point or uniform crossover. Inputs to the GA allowed for variation in the number of variables, bounds on the variables, and numerous other options (see Appendices A and C).

Based on the work of others, the population was set at 100 individuals [Nadon 1996, Millhouse 1998]. The uniform crossover rate was 50%, the jump mutation rate 1%, and the creep mutation rate 6%. These settings were held constant throughout the study because they produced reasonable results.

2.3 Introduction to MIDAS

MIDAS is a multi-disciplinary system of computer routines that design, size, and analyze missiles. Different engineering disciplines that are considered include mass properties, aerodynamics, propulsion, performance, thermal, sensors, and radar cross section.

MIDAS is particularly applicable to studies performed during the mission analysis and concept exploration phases of the missile design process. Good buzzwords to describe its fidelity include "first-cut" or "first-order", and not "finite element" or "simulation". The emphasis is on characterizing details rather than analyzing them [Lockheed Martin 1995].

MIDAS is useful in a parametric approach to missile design. The design process consists of choosing a candidate design based on mission requirements, determining the capability, and comparing it to the mission requirements in order to see how well it performs. After numerous trial and error iterations of this process, a good solution within the design space can be identified.

MIDAS can be run from a graphical user interface or by sequential data file inputs. Many aspects of the design must be defined in these input files with respect to the multiple engineering disciplines as mentioned above. Some of the general top-level inputs include planform geometry, type of propulsion, aerodynamic coefficients, and material properties.

A missile design is defined by a number of specific design parameters within these disciplines such as number of stages, diameter, propellant type, chamber pressure, nose fineness, fin configuration and size, propellant weight, burn rate, control scheme, and so on. Many of the parameters are determined to meet various subsystem requirements such as maximum range or average speed, or to produce a better design in terms of production cost or other measures of effectiveness.

Initially, some of the geometry and system details may not be known, but are defined implicitly by the laws of physics and assumed engineering judgements. For example, the motor case thickness could be determined by a chamber pressure requirement, or a required design point thrust is used to determine the nozzle design. The functional responsibilities of MIDAS are divided into six primary modules to balance these different engineering disciplines.

- the Problem Control Module is responsible for sequencing the analyses

- the Configuration Module performs geometric analysis
- the Mass Properties Module sizes structural elements to meet load requirements
- the Aerodynamics Module sizes control surfaces to meet maneuver requirements
- the Propulsion Module designs a motor to deliver a given thrust
- the Performance Module flies a specified trajectory

The Problem Control Module takes the inputs as provided by the analyst, and searches for syntax errors and performs other sanity and balancing checks using the other modules. MIDAS loops on design then provide iterations for configuration balancing between the different disciplines. The loops cycle through the design modules, and match particular components that have been specified in terms of a requirement for another module in order to provide a complete and consistent configuration definition. Through this process, MIDAS can intelligently define a missile that meets the feasibility constraints within each of the disciplines based upon a select few input design parameters.

III. Methodology

3.1 Overview

The current trial and error approach of reverse engineering the foreign missile had to be converted into an optimization problem that could be solved by a genetic algorithm. The fundamental outcome from this study was the development of a procedure that could direct this optimization process. The issues were in how to simplify the entire missile design into a short list of design parameters, calculate the candidate missile's trajectory by using these parameters, and then devise a fitness function that captured the quality of the match of the observation. The entire missile design, with some preliminary assumptions, had to be described in mathematical terms that completely described an individual missile.

Certain preliminary assumptions of the missile had to be made in order to keep the parameter list manageable during the optimization problem. Examples of these early assumptions include the type of propellant (liquid or solid), the number of stages, the number of wings or fins (if any), and a large amount of other aerodynamic and material properties. The analyst would be required to make some early assumptions, such as these, in order to keep the problem within a realistic design space. Even with multiple assumptions, there would still be a plethora of candidate missiles that existed. MIDAS input files include the preliminary assumptions that were used during this study and are listed in Appendix B.

Another accomplishment of this study involves the connection between the GA and MIDAS. The entire interface had to be created to link the design parameters between the two models. The design variables were passed from the GA into MIDAS along with the early assumptions as described above. MIDAS would then calculate a trajectory using the given information. The GA would then calculate a fitness function that compared the candidate's predicted trajectory back to the observation. Using this information, successive generations of the GA would try to improve the previous generation by changing the values of the design parameters.

3.2 Selection of Design Variables

MIDAS was used in a simplified manner. Although the total MIDAS database consists of about 2000 variables, and a typical data set involves values for about 50-80 variables, a single planform was chosen which limited the number of design variables to 12. This kept the problem manageable, while still allowing a large potential design space. These design variables formed the parameter vector that defined each individual missile.

The diameter (D), mass (M), and length (L) are intuitively critical parameters of a missile design. The lengths of individual sections of the missile were also determined to be important. Therefore, the length of the nose (L_n), equipment (L_e), warhead (L_w), propellant (L_p), and boat tail (L_{bt}) sections were also selected as design variables. In addition, the engine was assumed to have solid propellant with a design variable of the thrust-to-weight (T/W) ratio.

The assumed planform consisted of a single stage and included four wings and four fins spaced evenly around a circular cross section fuselage. Each wing and fin was identical, respectively, with certain assumed parameters based upon aerodynamic rules of thumb. The leading edge sweep was 68 degrees; the aspect ratio was 1.2632; and the chord tip-to-root ratio was 0.185. The trailing edge was assumed to attach perpendicular to the fuselage, and the leading edge of the root chord was located a distance (X_w) from the nose. With these parameters assumed, the respective areas of the wings (A_w) and fins (A_f) would then completely describe the planform configuration. These assumptions of the missile planform can be considered early assumptions or can represent known observational data.

The last design parameter was the shape of the nose cone. This was included in the study in order to demonstrate the implementation of a discrete design variable. The four options for the curve of the nose cone were circular tangent, Von Karman, cone, or ¾ power Newtonian.

The initial set of design variables was therefore:

$$x = [D, M, L_n, L_e, L_w, L_p, L_{bt}, X_w, A_w, A_f, T/W, nose].$$

The bounds of each of these variables were selected to be within realistic ranges for the particular observation. These bounds represent the early observational data or can be based on prior experience of missile design. To give an exaggerated example, a missile with a diameter of a centimeter would never make it 100km. Correspondingly, a missile with a 3m diameter may be able to reach intercontinental ranges and was therefore too large for the actual missile being observed. For this reason, the diameter was bounded to be within 20cm and 150cm. The bounds for the remaining variables will be given later in Table 1.

It was discovered early during the analysis that different combinations of the design variables could create infeasible missile designs. This would occur even though the values for all of the parameters are within realistic bounds. For example, the smallest diameter, when paired with the longest lengths, produces a missile that resembles a pencil the length of a telephone pole. This is of course an exaggeration, but long, thin missiles and short, fat missiles are not realistic. This is true even when the individual design variables are within realistic bounds. The conclusion here is that the diameter and length of the missile are somewhat dependent on each other. The result of this dependency is a large proportion of unrealistic or infeasible missiles.

A method was needed to reduce the dependency of the parameters. To accomplish this, each of the length parameters were normalized by dividing by the diameter. This method resembles dimensional analysis that is used throughout the engineering sciences. The design parameters that represent length then became L_n/D, L_e/D, L_w/D, L_p/D, and L_{bt}/D.

The other design variables were also investigated for dependencies. Intuitively, large missiles would tend to be heavier than small missiles. Therefore, a dependency must exist between the diameter, lengths, and mass of the missile. The mass seems somewhat proportional to volume, therefore the density, ρ, of the missile was chosen to replace the mass. Also, aerodynamic control surfaces are theoretically related. Knowing this, wing area and location might be related to the fin area. However, even with several results from MIDAS, a simple relationship among the design variables for the aerodynamic controls could not be found.

These changes to the basic set of design variables were not expected initially and were proposed only after initial optimization runs failed. A useful lesson is the importance of reducing the dependencies between design variables. The final set of quasi-independent design variables was:

$$x = [D, \rho, L_n/D, L_e/D, L_w/D, L_p/D, L_{bt}/D, X_w, A_w, A_f, T/W, nose].$$

The variables (with the exception of ρ and T/W) are graphically shown in Figure 3. (Note the nose cone is not accurately represented.)

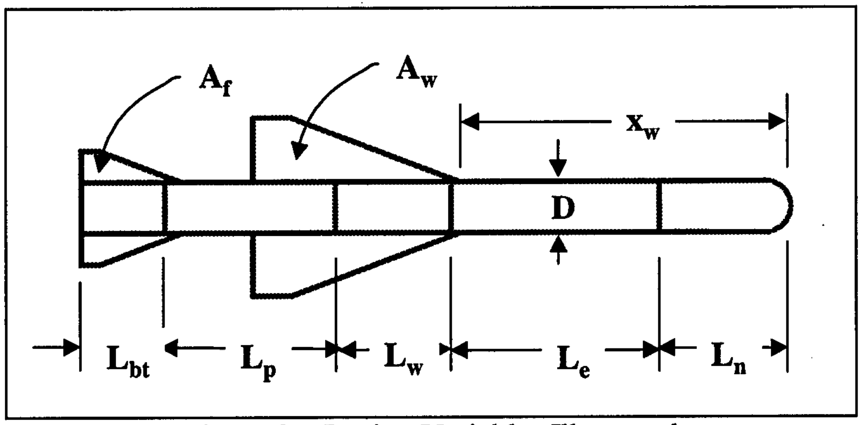

Figure 3 – Design Variables Illustrated

The initial minimum and maximum bounds for these design variables are listed in Table 1. Also included in the table, is the number of bits per variable used for the binary encoding used throughout this study. The number of bits is determined by the desired precision for the parameters. For example, the precision of the diameter was set at about 1cm, thereby requiring 7 bits ((150-20) / 2^7 = 1.015625cm). A more precise measurement was deemed unnecessary for this proof of concept.

Table 1 – Initial Design Variable Bounds

Variable	min	max	units	#bits
D	20	150	cm	7
ρ	1000	3000	kg/m^3	8
L_n/D	1	3	-	8
L_e/D	2	4	-	8
L_w/D	1	3	-	8
L_p/D	0	4	-	8
L_{bt}/D	0	2	-	9
X_w	L_n	$L_n + L_e$	cm	9
A_w	0.05	1.0	m^2	7
A_f	0.05	1.0	m^2	7
T/W	5	25	-	7
nose	1	4	-	2

The total number of bits for the entire missile was 88, which made the design space equivalent to $2^{88} = 3.1\text{x}10^{26}$ different possible missiles. In order to put this number into perspective, an analyst would have to evaluate a million designs per second for 10 trillion years, in order to enumerate each and every possible design. Hence, there is a need for a much quicker, optimized exploration of the potential design space.

3.3 Trajectory Calculation via MIDAS

The predicted trajectory of the missile must be compared to the observation in order to calculate the fitness of each individual. This means that the design variables, as determined by the GA, must be run through MIDAS. This process required three distinct steps: 1) the creation of a MIDAS input file based upon the design variables, 2) the external execution of MIDAS, and 3) the data collection of the predicted trajectory. These steps defined how the data was gathered that would later be used for the fitness calculation.

A generic MIDAS input file was constructed which included missile geometry definition, aerodynamics, propulsion, mass properties, as well as the desired trajectory profile that would be flown. Several of these MIDAS files included further assumptions of early observations or other realistic assumptions about the type of missile that was trying to be matched. The idea was to create a template for a given missile that could be filled in with an individual missile's assumed geometry. See Appendix B for these input files.

Once the input files were created, MIDAS was executed from within the GA. This was done with an external execution command. After MIDAS had finished, the GA was then able to access the MIDAS output files.

It was soon discovered that there would still be unrealistic missiles that defied some laws of physics or hardwired MIDAS assumptions, even with the realistic variable bounds and the diminished variable dependencies. MIDAS would fail and not attempt to fly the missile along the desired profile in some circumstances.

The MIDAS *err.log* file was therefore first inspected by the GA fitness algorithm. If the error file showed the run had not been successfully completed, the particular missile was considered infeasible. An infeasible missile, however, still needed an assigned fitness value. In order to discourage any infeasible missile from reproducing and procreating its undesired traits, the fitness was assigned a very poor value. In this way, the infeasible trait of the missile would have a very small chance of continuing into future generations.

For missiles that were feasible, the MIDAS output files had to be input into the GA fitness array. Because of difficulty in reading directly from standard MIDAS output files, a new output file was created within MIDAS. This new output file included the relevant information about the predicted trajectory including range, altitude, speed, and time. If other data was of interest, the MIDAS code would need edited and recompiled, while adding the relevant parameters to the output print statement. The data retrieved from the MIDAS output files was then used for the specific fitness functions that are described for the sample problems in the following sections.

IV. Analysis - Low Information

4.1 Introduction

The objective of the sample problems was to investigate whether a genetic algorithm could successfully reverse engineer a missile. The first problem was intended to represent a case with little observed data. The data for each sample problem was generated from a hypothetical missile created by analysts at NAIC at the unclassified level. This allowed the analysis to remain unclassified because it merely represented a nominal foreign missile and was not a real one. The design was then flown through MIDAS along a realistic flight profile to represent an intelligence gathered telemetry.

The profile that the actual missile was flown against in MIDAS provided the observation for trajectory comparisons. The commanded profile began with a launch from an aircraft at an altitude of 5km, at a distance of 103km from its target. After a 5 second delay, it climbed to an altitude of 15km. This altitude was maintained until the missile was 40km from its target. From this point forward, the missile guidance performed standard proportional navigation until target impact. This was also the profile that MIDAS used to command the candidate missiles during trajectory prediction.

To ensure that the author was unbiased throughout the study, the actual design dimensions of this observed missile were withheld until the very end of the analysis. This was comparable to the situation being mimicked – the attempted reverse engineering of a foreign missile. The only facts that were provided were the altitude, speed, and range of the observed trajectory as a function of time. For the low information analysis, only the impact conditions were collected from the observation.

4.2 Impact Fitness Function

The goal of this sample case was to find the missile or missiles that could match the observed impact conditions. This low information approach represented a missile that was advertised by a foreign government that was not actually observed in flight, or was merely observed at target impact after the launch. The missile was advertised or observed to travel 103km, while following the commanded profile, and impacting after 341 seconds at a speed of 254.5m/s.

This observed impact was then compared against the candidate missile's impact in order to calculate an appropriate fitness. The fitness function was therefore a combination of the range, speed, and time of the candidate missile at impact as compared to that of the observation. An equal weighting of the percentage difference squared of each of these factors was chosen to represent the penalty of not reaching the goal conditions. This is basically a least squares fit of the sample data against that of the known. Hence, the low information fitness function was defined as:

$$F(x) = -\left(\frac{R_i - R_o}{R_o}\right)^2 - \left(\frac{v_i - v_o}{v_o}\right)^2 - \left(\frac{t_i - t_o}{t_o}\right)^2 .$$

The maximum value of this fitness function is zero. This occurred when each of an individual's impact conditions exactly matched those of the observation. If there was any deviation, then a value was subtracted from its fitness, proportional to the percentage squared of that deviation. The larger the negative value, the worse the fitness. A very low value of -999.9 was given to individuals that could not be evaluated by MIDAS and were considered infeasible.

4.3 Preliminary Results

The preliminary results revealed a serious problem - a majority of missiles were still infeasible. The fraction of feasible missiles within the initial generation was 20-25%. This number varied due to the randomization effects of the initialization seed. The number of feasible missiles, per each generation, for a single random seed is shown in Figure 4.

Over successive generations, the missiles that were initially feasible reproduced with corresponding missiles of high fitness. This increased the overall quantities of individuals that were feasible over time. This trend is clearly shown within the figure by the increasing data. However, random effects of crossover and mutations still allowed a varied population. After approximately twenty generations the percentage of feasible missiles had finally reached an acceptable level of around 80% feasibility.

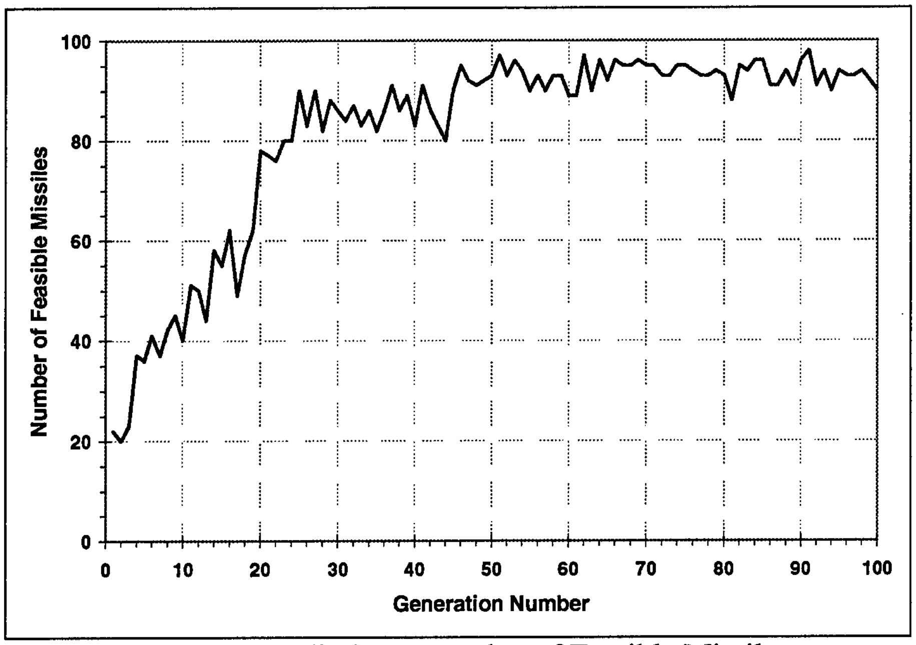

Figure 4 – Preliminary Number of Feasible Missiles

A consequence of the high number of initial infeasible missiles was a much less efficient exploration of the design space. In effect, the first generation was five times smaller than if all of the missiles had been feasible. The useful population was only twenty feasible individuals. The missiles that were initially feasible dominated the results of the GA and the design space was not being explored well.

4.4 Increasing Feasible Solutions

The first step that helped increase the number of feasible solutions was tightening the design variable bounds. The lower bounds of the propulsion and boat tail lengths were raised, based upon further scrutiny and engineering judgements. Next, the bounds on the diameter and the wing and fin areas were adjusted in order to exclude areas where none of the designs were feasible. These updated variable bounds are listed in Table 2. The number of bits was held constant to help refine the reduced design space and increase parameter precision.

Table 2 – Final Design Variable Bounds

Variable	min	max	units	#bits
D	**25**	**60**	cm	7
ρ	1000	3000	kg/m^3	8
L_n/D	1	3	-	8
L_e/D	2	4	-	8
L_w/D	1	3	-	8
L_p/D	**2**	4	-	8
L_{bt}/D	**1**	2	-	9
X_w	L_n	$L_n + L_e$	cm	9
A_w	**0.1**	**0.7**	m^2	7
A_f	0.05	**0.5**	m^2	7
T/W	5	25	-	7
nose	1	4	-	2

In addition, checks on the feasibility of the remaining missiles were added to the genetic algorithm. If MIDAS could not evaluate a missile against the commanded flight profile, it was not allowed into the missile population. In effect this was "a brave new world" method of screening out savages before their birth [Huxley 1932]. A small percentage of infeasible missiles, however, were allowed to survive with the overall goal set at 80-90% feasibility. This allowed some small chance of a mutation to survive into future generations. This new feasibility check was added during the creation of the initial generation as well as during reproduction between the two parents. The revised genetic algorithm flow is given in Figure 5.

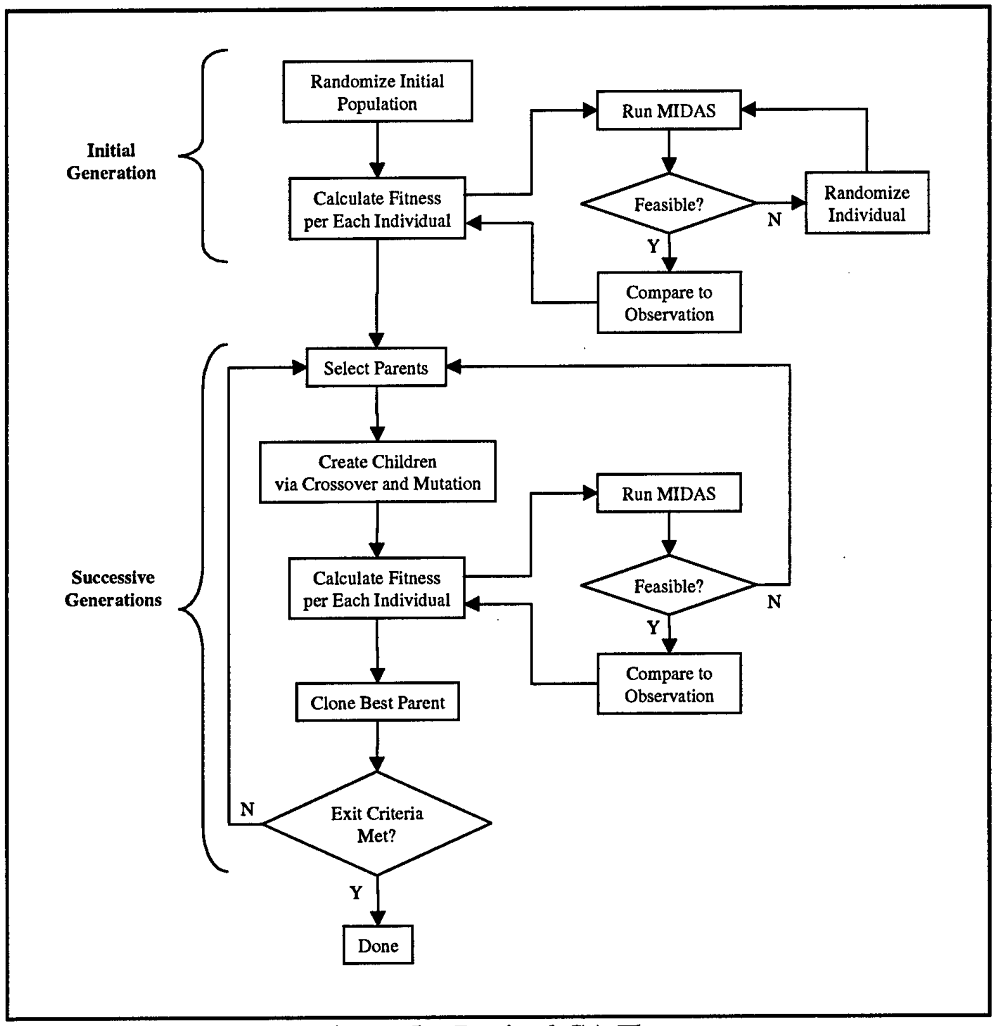

Figure 5 – Revised GA Flow

During each check of feasibility, a random draw was made which could overrule the feasibility check and accept a certain percentage of the infeasible missiles. During initialization, 15% of the infeasible missiles were accepted. In reproduction during successive generations, 60% were accepted. These values kept the number of feasible missiles above the desired 80-90% goal as shown in Figure 6.

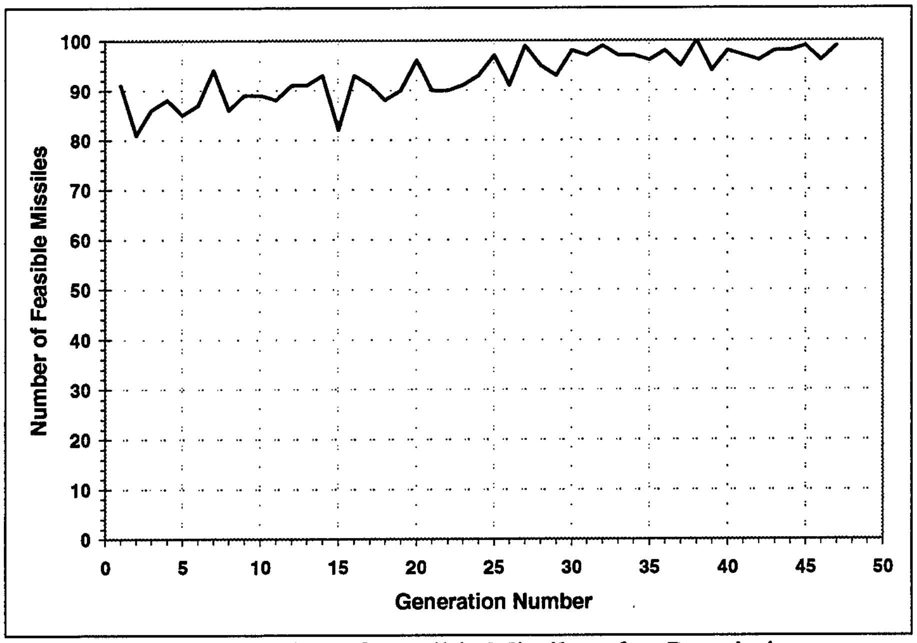

Figure 6– Number of Feasible Missiles after Restrictions

4.5 Low Information Results

With the number of feasible missiles at an acceptable level, the process was performed again. Next, the relative fitness of each of the missiles was studied. After approximately 25 generations, the GA had found many different missile designs that had nearly matched the observed conditions. The maximum fitness value per generation steadily increased up to this point. The population history of the best and 4th best missile fitness values is shown in Figure 7. The average, overall population fitness is shown on a much larger scale in Figure 8.

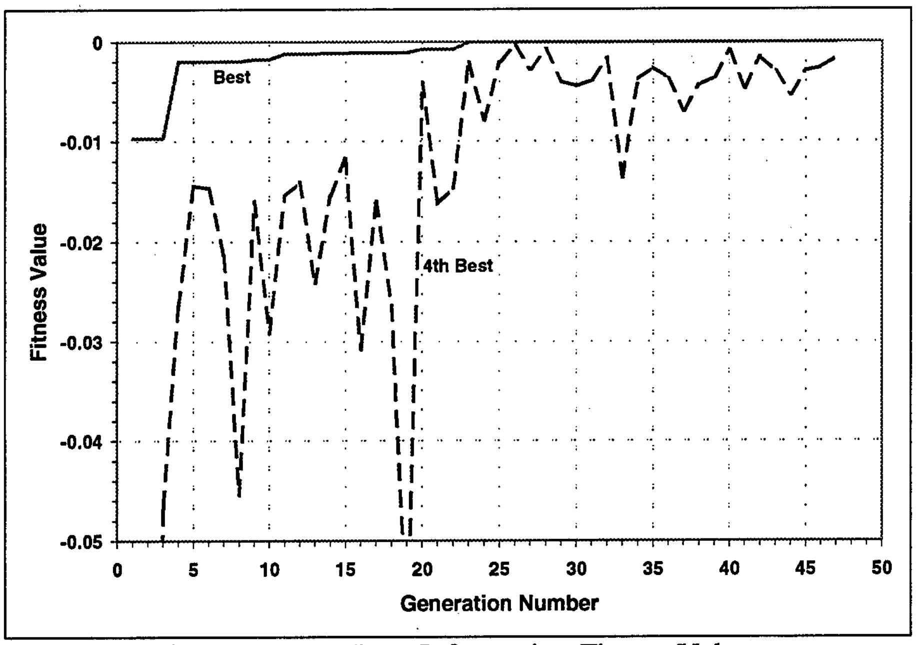

Figure 7 – Best Low Information Fitness Values

The 4th best missile was chosen because the top four or more missiles seemed to produce good results across many different randomization seeds. The 4th best missile could also be considered the 96th percentile point of the population. This provided a better view of the population by throwing out possible outliers that had the very best fitness values. It also proves that the top individual was not completely driving the overall results.

Elitist strategy forces the best fitness value to increase, or at least remain the same, from one generation to the next as seen in Figure 7. Random effects of crossover and mutation cause the 4th best fitness values to vary significantly between generations, but generally increasing over successive generations. In Figure 8, the entire population can be seen converging in fitness as well, but not necessarily in design. The average population fitness continues to increase through, at least, the 25th generation.

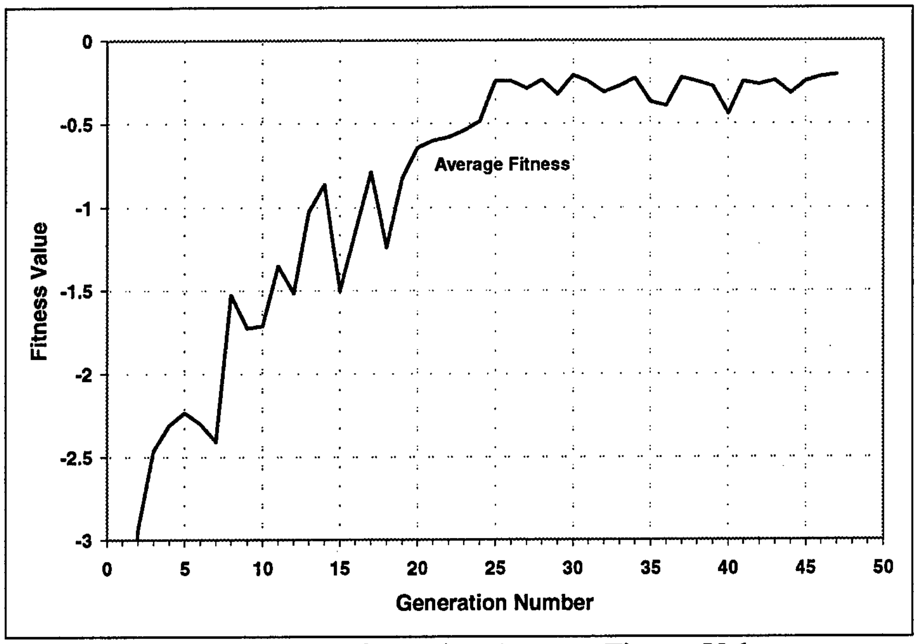

Figure 8 – Low Information Average Fitness Values

The success story of Figure 7 is that multiple missiles, at least four, are coming very close to matching the impact conditions of the observation. With fitness values greater than –0.005, the conditions are nearly a perfect match.

After about 40 generations, it was found that the population would begin to converge in design as well, and several missiles would begin to look similar. For this reason, detailed study of the best individual missiles was made at the 25th generation when there were still a variety of design parameter values that were different among the top missiles. The impact conditions for the best four missiles at the 25th generation are listed in Table 3, with their respective design characteristics.

Table 3 – Low Information Best Missiles

	best	2nd best	3rd best	4th best
D (cm)	31.6	32.7	30.8	30.0
L (cm)	392.6	315.4	312.3	372.2
L_n (cm)	32.9	34.0	51.1	41.7
L_e (cm)	118.0	96.0	69.5	118.0
L_w (cm)	69.3	39.6	49.6	68.7
L_p (cm)	110.1	99.6	110.4	110.9
L_{bt} (cm)	62.4	46.2	31.8	32.9
X_w (cm)	56.9	68.0	66.0	91.3
M (kg)	639.9	549.4	485.5	538.7
A_w (m^2)	0.402	0.591	0.435	0.601
A_t (m^2)	0.128	0.061	0.078	0.075
T/W	20.6	23.1	21.2	21.5
nose	1	2	3	1
time (sec)	339.81	332.14	333.15	332.36
speed (m/s)	255.2	256.4	260.6	264
range (km)	103.01	103.009	103.009	103.009
fitness	0.0000	-0.0007	-0.0010	-0.0020

A fitness value of –0.002 means that the missile's impact conditions are less than 0.2%, different from the observed missile as a sum of squares. Therefore, each of these missiles is close enough to match the observed conditions that they should all be considered strong candidates as a match for the actual missile. They should warrant further detailed analysis by the intelligence analysts. Also, these are most likely better matches than an analyst could accomplish through the trial and error process.

Casual analysis of Table 3 reveals that some of the design variables between each missile are nearly the same, while others are quite different. The missile diameter, for example, is about 30.8-32.7 cm for each of the top four missiles. This would lead one to conclude that the diameter of the actual missile is also within this range. Many of the other variables are different, and can be visualized in Figure 9. The overall length is shown to vary greatly, as well as the location and size of the forward wing. The nose contour was found to be a very minor contributor to the overall fitness, and is not shown in the diagram.

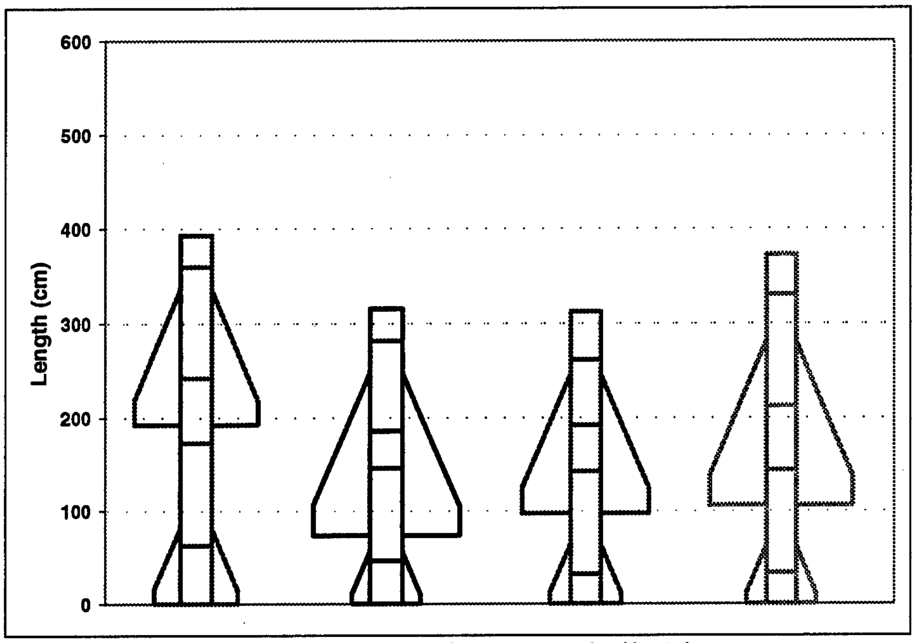

Figure 9 – Low Information Best Missile Diagrams

This set of optimal missiles leads to the conclusion that the genetic algorithm is successfully locating missiles, within the design space, that can match the observation, and that there are various different designs that can do so.

The question then arose whether other matching designs existed within the design space in areas that had not yet been explored. Also remembering that this is a random process, several different randomization runs should be performed anyway. The initial randomization seed was changed in the GA, and another missile evaluation GA run was conducted. The results of the 25th generation of the second randomization seed are given in Table 4 and Figure 10.

Table 4 – Other Low Information Missiles

	best	2nd best	3rd best	4th best
D (cm)	38.2	39.3	39.3	38.5
L (cm)	419.6	455.8	476.6	411.6
L_n (cm)	51.1	39.6	40.3	38.8
L_e (cm)	109.4	95.9	137.0	90.0
L_w (cm)	110.2	114.9	111.8	114.9
L_p (cm)	101.0	155.5	116.6	92.4
L_{bt} (cm)	47.8	49.8	70.9	75.5
X_w (cm)	59.0	46.4	128.7	81.8
M (kg)	577.8	577.6	617.6	590.0
A_w (m^2)	0.639	0.464	0.629	0.506
A_f (m^2)	0.057	0.054	0.050	0.050
T/W	22.5	13.8	11.1	20.6
nose	2	2	1	2
time (sec)	345.07	348.44	346.86	325.37
speed (m/s)	257.2	262.2	265.1	252.6
range (km)	103.01	103.01	103.009	103.009
fitness	-0.0003	-0.0014	-0.0020	-0.0022

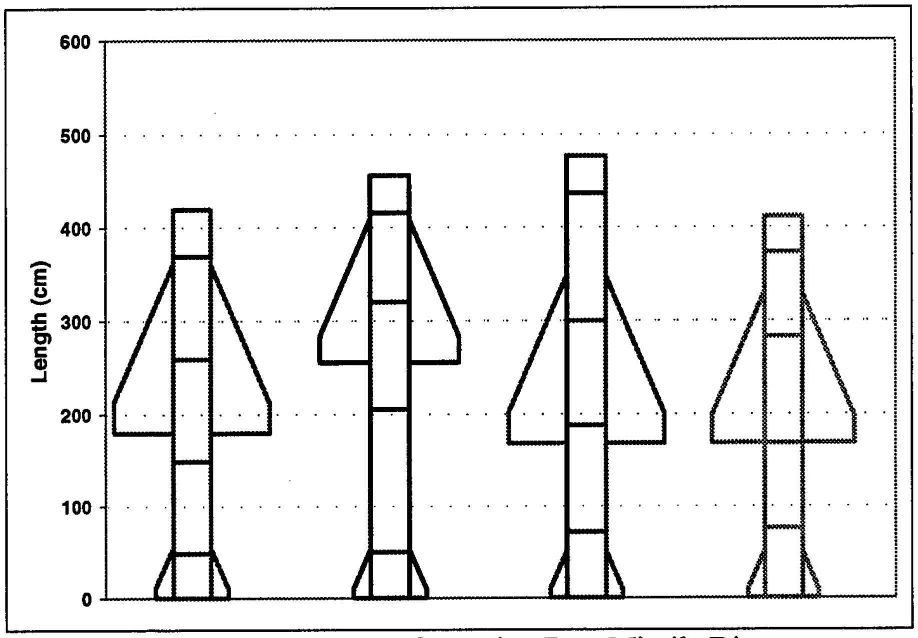

Figure 10 – Other Low Information Best Missile Diagrams

These missiles are clearly different from those of the first seed, but also match the observed missile impact conditions. A significant difference however, is that the diameter of these missiles is about 38.2-39.3 cm. This is drastically different from the common diameter of the first seed.

The nearly identical diameters, per each seed, shows that a convergence in population design has already begun, even before the 25th generation. This is because there are many different missile designs that are capable of matching the observed missile characteristics based solely on the impact conditions. Once a single missile was discovered by the GA that nearly made a match, that missile became the most sought after design for reproduction. Many descendents continued to carry on its traits.

4.6 Summary

The genetic algorithm process of reverse engineering missile designs can work when a few steps are added to the standard GA flow. Tightening of design parameter bounds, investigation of missile infeasibility regions, and random feasibility checks were necessary to maintain a high percentage of feasible missiles within the population. With these changes, it was proven that several missiles could be designed to match the sample observation. For this low information problem, many of these matching solutions may not have been found without the use of the genetic algorithm.

V. Analysis – High Information

5.1 Introduction

The second sample problem included more information from the observation than just the impact conditions. Knowledge of multiple waypoints along the entire trajectory including time, range, speed, and altitude for each point was collected during the observed missile's flight. The question remained, for this problem, whether the GA could reverse engineer a missile that matched the points along the given trajectory.

As with the first sample problem, the hypothetical missile was fabricated by NAIC and was run through MIDAS along the same commanded profile, which simulated the observation. The collected data for the observed missile is shown in 10 sec increments in Figure 11, and is listed numerically in Appendix C within the *GOALS1.inp* file.

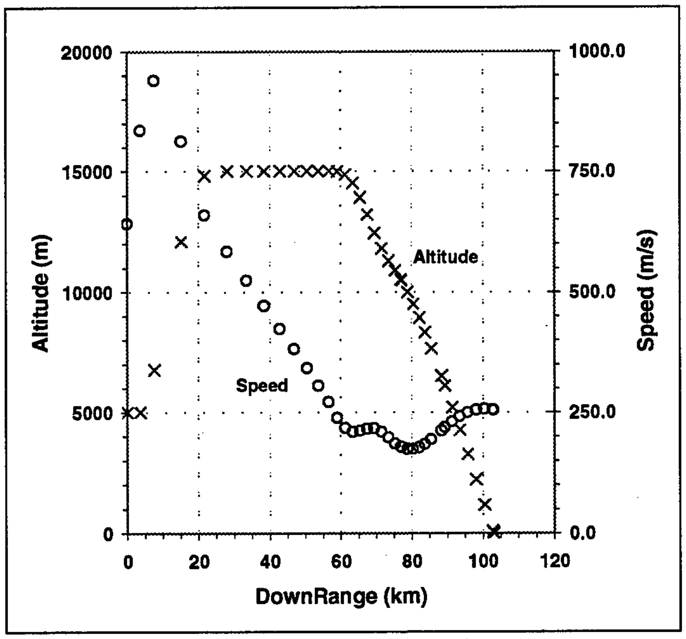

Figure 11 – Observed Trajectory

5.2 Multiple Waypoints Fitness Function

Several terms had to be added to the low information fitness function in order to compare the match of an individual missile's trajectory against the entire observation. As before, the squared differences of the range and speed were included. The squared difference of the altitude relative to the observation was now added. These terms were included at each of the *N* observed waypoints along the trajectory, and were averaged by dividing by *N* thus producing an averaged fitness as:

$$F(x)=\frac{1}{N}\sum_{j=1}^{N}\left[-\left(\frac{R_{ij}-R_{oj}}{R_{oj}}\right)^2-\left(\frac{v_{ij}-v_{oj}}{v_{oj}}\right)^2-\left(\frac{a_{ij}-a_{oj}}{a_{oj}}\right)^2\right].$$

There was a minor problem with this simple least squares fit. There was no guarantee that the individual missile would have the same time of flight as the observed missile. It was possible that the individual missile might take a much longer or much shorter length of time to accomplish the commanded profile. This depended on average speed, which in turn depended on mass, T/W, and so on. In fact, there was a small chance of the trajectories taking exactly the same amount of time. The problem then, was in how to match up each of the *N* waypoints in order to compare the two trajectories.

As an example, if the observed missile flew for 351 seconds and the individual missile candidate flew for only 300, the remaining 51 seconds of flight would be treated as severe penalties to the fitness function. This was because there was no data from the individual missile to compare against for nearly the last minute of flight. In order to solve this problem, a lengthening, or shortening, of the individual's trajectory was performed to make each of the trajectories have the exact same duration.

Once the individual trajectory had been sized to exactly fit the observation, then the range, speed, and altitude of the individual missile were interpolated at the time of the collected waypoints. This permitted a one-for-one comparison between the trajectories, without the waypoints of one trajectory extending beyond the impact point of the other. In order to penalize the individual missile for this trajectory fitting, the final term of the fitness function accounted for the time adjustment. The resulting high information fitness function was thus defined as:

$$F(x)=\frac{1}{N}\sum_{j=1}^{N}\left[-\left(\frac{R_{ij}-R_{oj}}{R_{oj}}\right)^2-\left(\frac{v_{ij}-v_{oj}}{v_{oj}}\right)^2-\left(\frac{a_{ij}-a_{oj}}{a_{oj}}\right)^2\right]-\left(\frac{t_{iN}-t_{oN}}{t_{oN}}\right)^2$$

Once again, the maximum fitness value was zero and an infeasible missile was assigned a very low fitness of –999.9 to discourage reproduction. Direct comparison of the values of this high information fitness function with the low information fitness function is not appropriate because there are many more terms in the high information fitness function. The importance of the relative differences between the low and high information fitness values is also much different.

5.3 High Information Results

The high information results tended to show the same properties as those of the low information results. The best through 4th best missiles increased in fitness through the 20-25th generation, and then leveled off. The best missiles also began to converge in design by the 40th generation. Figure 12 shows the high information fitness values as they increased over time for a given initial randomization seed.

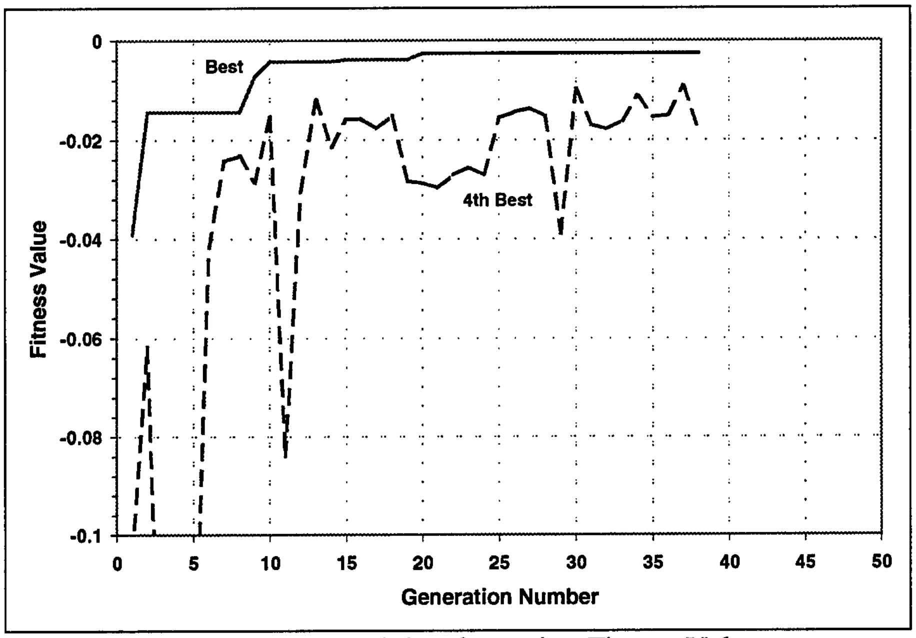

Figure 12 - Best High Information Fitness Values

Table 5 lists the missile design parameters for the best four missiles at the 25th generation, and Figure 13 shows the missile diagrams. The missiles are different in most parameters including section lengths, wing location and position, and overall mass. The diameters, however, are nearly the same at 36.6-37.7 cm.

Table 5 - High Information Best Missiles

	best	2nd best	3rd best	4th best
D (cm)	37.7	36.6	37.7	36.6
L (cm)	398.6	458.4	466.9	411.9
L_n (cm)	59.5	56.9	42.1	59.2
L_e (cm)	138.9	143.4	137.1	133.4
L_w (cm)	51.0	85.9	107.7	73.6
L_p (cm)	91.6	109.6	122.3	88.6
L_{bt} (cm)	57.6	62.5	57.6	57.1
X_w (cm)	107.7	120.7	90.1	118.8
M (kg)	501.9	543.8	620.4	492.1
A_w (m^2)	0.369	0.530	0.492	0.218
A_f (m^2)	0.061	0.089	0.103	0.068
T/W	17.4	15.4	15.2	23.3
nose	3	4	4	4
fitness	-0.0026	-0.0081	-0.0150	-0.0156

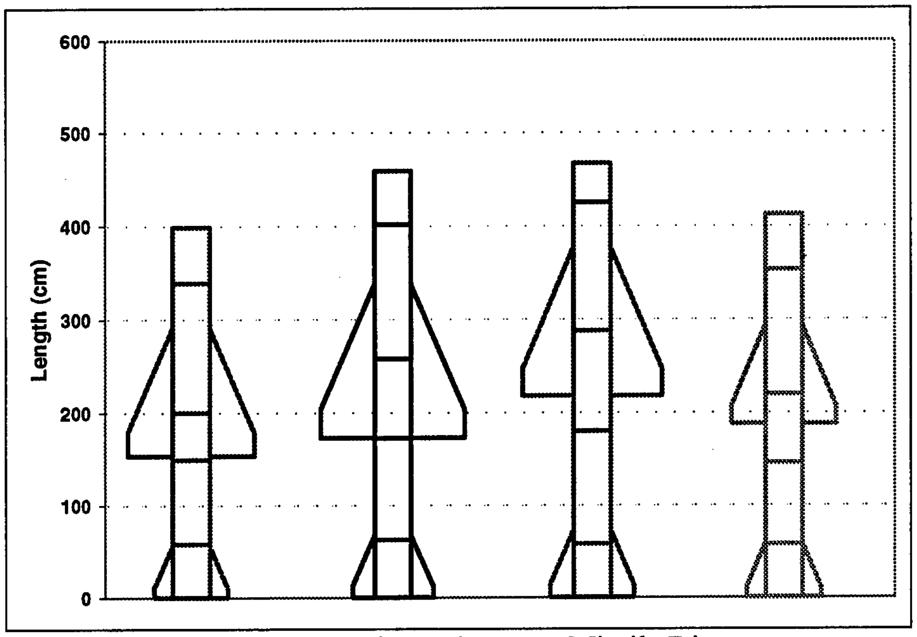

Figure 13 - High Information Best Missile Diagrams

In order to get a feel for how good these fitness values are, the predicted trajectory for the best missile is plotted overtop of the observed missile waypoints in Figure 14. The X's and O's are the points collected from the observed missile as in Figure 11, while the continuous lines are the predicted trajectory results. As can be seen, the reverse engineered missile has performance almost identical to the actual missile during the observation. The range, altitude, and speed are nearly a perfect match. Not shown in the figure is also a very good match in time of flight as well.

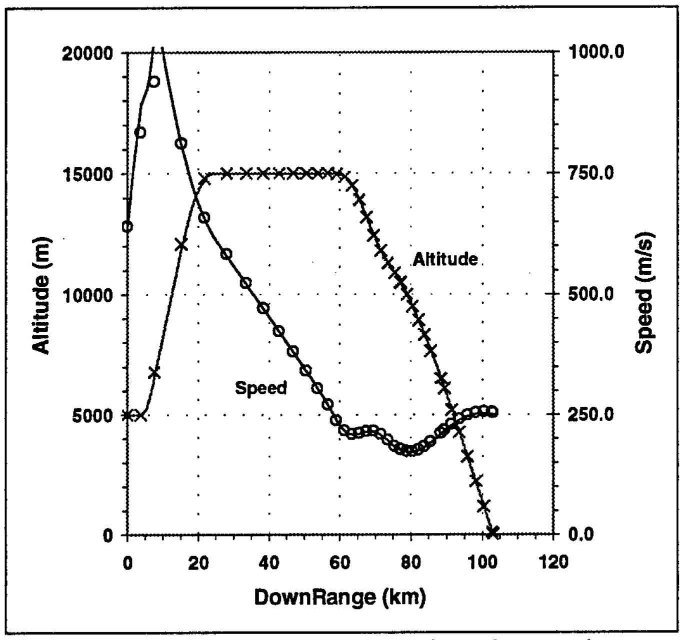

Figure 14 – Trajectory Waypoints Comparison

In addition to this best missile, several others at the 25th generation could also be considered strong candidates for being the observed missile.

As with the low information problem, several different initialization seeds were run due to the inherent randomization processes of the GA. The resulting best missiles per each seed varied somewhat between the populations, but both were extremely good matches of the observation. The best missiles for seven different initialization seeds are listed in Table 6.

Table 6 – Multiple Seed Best Missiles

	seed1	seed2	seed3	seed4	seed5	seed6	seed7
D (cm)	37.7	34.9	34.1	38.2	31.6	34.1	34.9
L (cm)	398.6	380.2	417.2	354.3	379.0	403.4	342.9
L_n (cm)	59.5	66.1	78.5	52.9	75.3	64.8	41.8
L_e (cm)	138.9	78.9	107.2	79.8	94.2	105.6	100.2
L_w (cm)	51.0	35.5	84.6	47.8	77.2	81.4	74.6
L_p (cm)	91.6	132.0	100.0	131.0	86.3	88.0	86.3
L_{bt} (cm)	57.6	67.7	46.8	42.7	46.0	63.5	40.0
X_w (cm)	107.7	93.8	131.2	84.8	111.2	94.6	50.0
M (kg)	501.9	472.4	418.1	661.2	455.9	469.2	559.8
A_w (m^2)	0.369	0.327	0.346	0.487	0.313	0.398	0.393
A_t (m^2)	0.061	0.068	0.054	0.142	0.075	0.075	0.093
T/W	17.4	22.0	14.9	24.2	22.6	17.8	22.0
nose	3	4	3	4	1	4	4
fitness	-0.0026	-0.0011	-0.0038	-0.0010	-0.0034	-0.0026	-0.0013

Among the independent cases, it still appeared that certain traits such as diameter had begun to converge by the 25th generation, but were very different between the seeds. This once again was because many different designs were capable of matching the given observation, even with the multiple waypoint objectives. Certain good design traits were randomly located in early generations, then were passed on to a larger and larger percentage of children throughout future generations. This would begin the converging process of design.

5.4 Summary

Even with the increased restrictions of multiple waypoints, the genetic algorithm was able to locate several missiles within the design space that matched the observation. The high information fitness function that included trajectory sizing and output value interpolation seemed to work well in ranking the individuals and directing the optimization. Many of the best missiles that were designed using this process could be considered for future detailed engineering analysis, and might not have been found without the use of the genetic algorithm.

VI. Conclusions and Recommendations

This approach of applying genetic algorithms to the reverse engineering missile design problem was successful. The interface between the missile design program MIDAS and the GA permitted an automated and timely search of the design space. The GA was also able to identify a set of missiles that very closely matched the observations and the intelligence analyst would be able to consider for further detailed analysis. Many of these designs would have been missed by the previous trial and error approach; or would have taken months to find.

This study demonstrated a useful procedure for reverse engineering any type of design. A fundamental step is the selection of design parameters to fully describe a candidate design, yet are limited enough to keep the problem manageable. The model interface between the GA and MIDAS demonstrated how input and output collection algorithms could assist with the fitness calculation process. Finally, the fitness functions themselves were able to successfully direct the optimization search toward more optimum designs that matched an observation including special handling of infeasibility. This GA approach might also be useful in mission design problems where specific mission goals are known and a system must be developed to accomplish that mission.

In order to reduce the number of candidate missiles further, additional information would probably be needed to help constrain the design problem. Additional observational data such as another observed trajectory, or tighter restrictions on the design variable bounds should diminish the number of matching missiles. The exact missile length or diameter, for example, might be obtained from a photograph, significantly reducing the feasible design regions. The goal of the genetic algorithm should be to help the intelligence analysts consider a set of candidate missiles for further detailed analysis, but should not overwhelm them with too many options.

It was also discussed during the defense of this thesis, that replacing the design variables that were used here with certain aerodynamic coefficients might be a good idea. This would create a two-step approach where the aerodynamic characteristics of the missile are first determined, then mapped into a realistic geometric planform. This could limit the number of matching missile solutions.

Finally, the design parameters that were used in this study for the actual missile during the observation are given in Table 7. This missile is also diagramed as the example missile in Figure 3 in Section 3.2. The actual missile is much longer and weighs much less than any of the best missiles that the GA was able to locate in the low/high information analyses. This was probably due to the plethora of missile designs that were able to match the observation. Eventually, an initialization seed probably could have found the actual missile, but it did very well finding many others. The process was proven to work, and is recommended for future analysis in this as well as other design areas.

Table 7 – Actual Missile

	ACTUAL
D (cm)	35.0
L (cm)	445.0
L_n (cm)	70.0
L_e (cm)	135.0
L_w (cm)	75.0
L_p (cm)	110.0
L_{bt} (cm)	55.0
X_w (cm)	200.0
M (kg)	505.7
A_w (m^2)	0.348
A_f (m^2)	0.084
T/W	19.6
nose	1
fitness	0.0000

Bibliography

Arora, Jasbir S. *Introduction to Optimum Design.* New York: McGraw-Hill, 1989.

Banks, Jerry and John S. Carson, II and Barry L. Nelson. *Discrete-Event System Simulation.* Upper Saddle River NJ: Prentice-Hall, 1996.

Carroll, David L. "FORTRAN Genetic Algorithm (GA) Driver." Online, 1998. Available: http://www.staff.uiuc.edu/~carroll/ga.html

Crossley, William A. and David H. Laananen. "Conceptual Design of Helicopters via Genetic Algorithms," *Journal of Aircraft, 33*(6):1062-1070 (November-December 1996).

Darwin, Charles. *On the Origin of Species.* London: John Murray, Albemarle Street, 1859.

Dasgupta, D. and Z. Michalewicz. *Evolutionary Algorithms in Engineering Applications* New York: Springer-Verlag Berlin Heidelberg, 1997.

Fonseca, Carlos M. and Peter J. Fleming. "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization." *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo CA: Morgen Kaufmann Publishers, Inc., July 1993.

Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading MA: Addison Wesley, 1989.

Huxley, Aldous. *Brave New World.* Garden City NY: Doubleday, Doran & Co. Inc., 1932.

Lockheed Martin. *MIDAS – Missile Integrated Design Analysis System.* Dallas TX: April 1995.

Mendenhall, William and Terry Sincich._*Statistics for Engineering and the Sciences.* San Francisco CA: Dellen Publishing Co., 1992.

Millhouse, Paul T. *Improving Algorithmic Efficiency of Aircraft Engine Design for Optimal Mission Performance.* MS thesis, AFIT/GOR/ENY/98M-02. School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, 1998.

Mitsuo, Gen and Runwei Cheng. *Genetic Algorithms and Engineering Design.* New York: Wiley, 1997.

Nadon, Luc J.J.P. *Multidisciplinary and Multiobjective Optimization in Conceptual Design for Mixed-Stream Turbofan Engines.* MS thesis, AFIT/GAE/ENY/96D-6. School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, 1996.

Nadon, Luc J.J.P, Stuart C. Kramer and Paul I. *King Multi-Objective Optimization of Mized-Stream Turbofan Engines.* Technical Report AIAA-98-0910, AIAA, 1998.

Powell, David and Michael M. Stolnick. "Using genetic algorithms in engineering design optimization with non-linear constraints." *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo CA: Morgen Kaufmann Publishers, Inc., July 1993.

Reklaitis, G.K., A. Ravindran, KM Ragsdell *Engineering Optimization Methods.* New York: John Wiley, 1983.

Van Veldhuizen, David A. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD dissertation AFIT/DS/ENG/99-01, Air Force Institute of Technology, Technology, Wright-Patterson Air Force Base, 1999.

About the Author

Jon Wollam is a military operations analyst at Veridian Engineering located in Dayton, Ohio. He has over 6 years of constructive modeling and simulation experience in the operational effectiveness of military weapon systems, GPS navigation, combat identification, and low observable mission planning. He received a BS in Aerospace Engineering in 1994 from the University of Cincinnati, and began working toward his Systems Engineering degree in 1996 at the Air Force Institute of Technology.

permanent address: 6273 East Law Rd.
Valley City, OH 44280

current e-mail: jwollam@dytn.veridian.com

Appendix A – How to use GA with MIDAS

The genetic algorithm and the missile design software MIDAS are located on an NAIC mainframe, UNIX based computer at Wright-Patterson AFB. There are two required input files for the genetic algorithm, and several required for MIDAS. Example MIDAS input files that were used for this study can be found in Appendix B, and the GA input files are in Appendix C.

The MIDAS input files include all that is required for the execution of MIDAS. This includes all of the early assumptions about the missile including basic planform geometry, type of propulsion, aerodynamic coefficients, material properties, and commanded profile for matching the trajectories. The only inputs that are missing from these files are the design variables that must be defined via the GA. The GA adds the design variables to the MIDAS input files when the fitness function is calculated. If the set of design variables is to be changed, the GA code would have to be edited, recompiled, and linked.

The first GA input file *GOALS1.inp* contains the data from the observation. This is the objective of the optimization process. The file is simply a table of the flight path angle gamma, speed, altitude, and range as a function of time. Gamma was initially part of the analysis, but was removed when it was realized that the flight path angle was an input to MIDAS and was not really being calculated within. If other types of data are desired to be part of the fitness function, MIDAS itself would have to be edited to add the required information.

The second GA input file defines the design parameters, and the attributes of the genetic algorithm. The design parameters are described to the GA as numerical minimum and maximum bounds, and number of bits required for each. Other significant GA attributes include the size of the population, the types and rates of crossover and mutation, niching and elitism options, and the initial randomization seed. Each of these inputs is described in detail on the last page of Appendix C.

In addition to the standard MIDAS output files, the NAIC version of MIDAS now includes the output file *gadata.out*. This file simply lists the predicted trajectory characteristics of gamma, speed, altitude, range, weight, thrust, and alpha of the missile as a function of time. The GA inputs this file in order to calculate the fitness of the individual design.

In order to obtain the GA code used for this study, contact the author where mentioned earlier, or:

Department of Aeronautics and Astronautics
School of Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH
45433-6583

Appendix B –MIDAS input files

These are example input files which are required to use the GA with MIDAS:
* *PFMAIN1.inp* is the commanded missile profile in MIDAS format.

The other files are required by MIDAS and represent other assumptions about the missile. The only variables that need to be added to these files, to run MIDAS, are the respective GA design variables. The GA driver automatically adds these to the MIDAS input files when the fitness function is being calculated.

PFMAIN1.inp
```
ALFMAX=4*15., 16*20.,
ALTI=5000.,
ALTMAX=30000.,
ATNSVR=2*-1., 3., 17*-1.,
ATNVAL=2*0., 15000., 17*0.,
CRALT=5000., 0., 2*15000., 16*0.,
FPCDLV=20*0.,
FPCON=2*1., 6., 7., 8., 15*0.,
FPCVAL=0., 10., 8*0., 35., 4., 28*0., 1., 4., 1., 157*0.,
FPPARM=4*0., 4., 15*0.,
FPPVAL=0., 9.999, 10., 197*0.,
GRAIN=1., 2., 18*1.,
NZPRIO=20*0,
PRINTB=1.,
SIDTAB=20*0.,
SKIDTR=0.,
STPVAL=50., 12000., 0., 30., 0.6096, 15*0.,
STPVAR=9., 3., 1., 10., 3., 15*0.,
TARGET=0., 103., 5., 6*0., 1.E+30, 2*0.,
THROTL=20*1.,
TPHASE=5., 19*1000.,
TTITLE=' 30km  0.6M',
TTOTAL=20*1000.,
VELI=0.09,
XMACHI=2.,
```

AEINPT.inp

```
ALTT=0., 12200., 20000., 30000., 0.,
CDMULT=1.,
CLMULT=1.,
NALTT=4,
NMACHT=12,
XMACHT=0.6, 0.8, 0.95, 1.05, 1.2, 1.6, 2., 2.5, 3.1, 3.8, 4.6, 6.,
8*0.,
```

AETABL.inp

```
ALTD=0., 12200., 20000., 30000., 21*1.E+30,
CDOTAB=0.314099, 0.296853, 0.411895, 0.627558, 0.665841, 0.596392,
    0.537607, 0.475743, 0.408831, 0.355904, 0.304482, 0.243878,
    8*1.E+30, 0.36947, 0.347861, 0.459611, 0.672329, 0.708492,
    0.634379, 0.571708, 0.505731, 0.434719, 0.377928, 0.323052,
    0.258214, 8*1.E+30, 0.439203, 0.411841, 0.519325, 0.728277,
    0.76173, 0.681731, 0.614215, 0.543151, 0.467094, 0.40556,
    0.346454, 0.276454, 8*1.E+30, 0.570442, 0.531561, 0.630707,
    0.832415, 0.860656, 0.769524, 0.693004, 0.612596, 0.527345,
    0.457215, 0.390478, 0.311273, 428*1.E+30,
CLATAB=0.338774, 0.343989, 0.351738, 0.377208, 0.401914, 0.381042,
    0.327528, 0.292395, 0.272205, 0.244477, 0.2156, 0.178665,
    88*1.E+30,
NALTD=4, 4*0,
NMACHD=4*12, 21*0,
NMACHL=12, 4*0,
XCPTOT=237.503, 237.965, 234.003, 237.449, 241.675, 243.434, 247.2,
    253.075, 259.217, 263.088, 266.212, 271.255, 88*0.,
XMACHD=0.6, 0.8, 0.95, 1.05, 1.2, 1.6, 2.,
    2.5, 3.1, 3.8, 4.6, 6., 8*1.E+30, 0.6, 0.8, 0.95, 1.05, 1.2, 1.6,
    2., 2.5, 3.1, 3.8, 4.6, 6., 8*1.E+30, 0.6, 0.8, 0.95, 1.05, 1.2,
    1.6, 2., 2.5, 3.1, 3.8, 4.6, 6., 8*1.E+30, 0.6, 0.8, 0.95, 1.05,
    1.2, 1.6, 2., 2.5, 3.1, 3.8, 4.6, 6., 428*1.E+30,
XMACHL=0.6, 0.8, 0.95, 1.05, 1.2, 1.6, 2., 2.5, 3.1, 3.8, 4.6, 6.,
    88*1.E+30,
```

AEUNTS.inp

```
AMETIN=2.,
AMETOT=2.,
```

CFAFIN.inp

```
AELE=0.,
ALCD=45., 135., 225., 315., 4*-1.,
AR=1.111,
CPAR=0.,
DTEFWD=-0.01,
FTE=-1.,
INMS=1,
INSS=0,
ISTG=1,
ITRIM=1,
NBR=4,
PCIN=0.,
RADR=0.0619999,
RADT=0.0619999,
SAIN=68.,
SLE=-1.,
SPAN=0.,
TCR=0.047,
TCT=0.088,
TR=0.,
```

CFBASC.inp
```
IMTYPE=1,
```

CFBOD1.inp
```
BTADIA=30.,
BTDMR=1.833,
HIBSTB=5.08,
HPEQIP=124.968,
ITBOAT=1,
MASTAK=1, 3, 2, 8, 9,  5*0,
MATBST=2,
MATLB=7,
MATNOS=4,
STHEO=0.,
```

CFWING.inp
```
ALCD=45., 135., 225., 315., 4*-1.,
AR=1.111,
DWLR=0.092,
IPT=5,
ISTG=1,
NBR=4,
PCIN=0.,
RADR=0.0254,
RADT=0.0619999,
SAIN=68.,
SPAN=-43.5,
STE=-1.,
TCR=0.033,
TCT=0.048,
TR=0.185,
TWLR=0.3,
```

MPAFIN.inp
```
ISOL=2,
IWCT=4,
IWOR=2,
MSK=7,
MWCC=6,
```

MPBOD1.inp
```
FBMISC=0.05,
FSFFL=1.25,
FSFFS=1.25,
IDMAP=2,
ISABOD=1,
MNCAP=4,
RNX=14.,
RNZ=25.,
TRIM=0.,
```

MPCG.inp
```
ICG=1,
```

MPCTRL.inp
```
DCELIC=0.5, 2*1.,
HMOM=3*0.,
ICSTYP=3, 2*1,
ISACTL=1,
ISURFC=3, 2*0,
OTMAXI=150., 2*10.,
SRIN=100., 2*300.,
```

MPGUID.inp
```
IGTYPE=2,
IRTCAD=1,
RAQRNG=277.8,
RNGIMU=40.0001,
RTARGA=9.99999,
RTDUTY=0.2,
```

MPSTAT.inp
```
EPR=5.,
ICMIB=1,
IDESY=3*0,
IFRDGU=2,
ISFACR=0,
RNZU=3*25.,
WWARH=115.,
```

MPWING.inp
```
ISOL=2,
IWCT=1,
IWOR=2,
MSK=6,
MWCC=7,
```

PCMAIN.inp
```
BANNER=0.,
CADCMF=2.,
CARPET=0.,
CDODES=-1., 3*0.,
CLADES=-1., 3*0.,
COMPAE=1.,
COMPCF=1.,
COMPMP=1.,
COMPPF=1.,
COMPPR=1.,
NPRDES=1,
PASSES=10,
PRALFA=0., 2.7, 2*-99.,
PRALT=7000., 3*0.,
PRAXG=9.8, 3*0.,
PRFRAC=-0.03, 3*1.,
PRGW=150., 3*-1.,
PRMACH=1.2, 3*2.,
PRTHR=4*-1.,
SIZER=1.,
WTPROP=3*0.,
```

PCUNTS.inp
```
METIN=1,
METOUT=1,
```

PCVARY.inp
```
INDEX=0, 20000, 28*0,
NMLIST=' ','$PFMAIN', '$PCMAIN', 27*' ',
NPARMS=0,
UNITS=' ','m', 28*' ',
VAR=' ', 'ALTI','P3', 27*' ',
```

PFUNTS.inp
```
PFMETN=2.,
PFMETO=2.,
```

PRMAIN.inp
```
IPRLEV=2, 2*1,
IPRSYS=3*1,
```

PRSR.inp
```
AEAT=8., 3*1.,
AEXIT=4*0.,
ATHR=4*0.,
CSTR1=4*1490.,
FBFS=4*1.,
ISRDS=3, 3*1,
ISRND=4, 3*1,
ISROPT=2, 3*1,
NGRN=2, 3*1,
PC=5000., 3*6894.76,
RHOP=4*0.001645,
WBWT=4*1.,
```

PRSRBP.inp
```
AEXIT=4*0.,
APAT=4*1.5,
DINSUL=0.0014, 3*0.,
DLINER=0.0014, 3*0.,
PCMAX=8700.01, 3*6894.76,
TINSUL=0.1, 3*0.,
TLINER=4*0.,
XLBT=20., 3*0.,
```

PRTABL.inp
```
ITABLE=0,
```

Appendix C – GA input files

These are example input files which are required to use the GA:
* *GOALS1.inp* is the missile observation.
* *ga.inp* has the GA driver required inputs.

After the *ga.inp* file are David L. Carroll's own description's on the use of the GA variables [Carroll 98].

GOALS1.inp

```
Time(sec),Gamma(deg),speed(m/s),Alt(km),Range(km)
0,0,640.9,5000,0
5,0,835,5000,3.693
10,34.53,939.2,6779,7.543
20,34.53,811.9,12085,15.124
30,9.11,659.3,14808,21.781
40,0,583.4,15000,27.937
50,0,523.4,15000,33.451
60,0,470.8,15000,38.405
70,0,423.8,15000,42.864
80,0,381.2,15000,46.876
90,0,342,15000,50.482
100,0,305,15000,53.707
110,0,271.2,15000,56.58
120,-0.84,238.6,14993,59.119
130,-6.08,217.3,14871,61.378
140,-13.27,209.5,14515,63.465
150,-18.84,211.5,13926,65.475
160,-20.76,215.8,13190,67.478
170,-19.19,215.9,12443,69.505
180,-15.85,209.1,11799,71.532
190,-12.97,197.2,11297,73.499
200,-12.42,184.8,10889,75.357
209,-14.3,178.6,10511,76.944
210,-14.49,178.1,10466,77.117
220,-16.27,174.4,9999,78.81
230,-17.8,173.9,9489,80.47
240,-19.14,177,8934,82.128
250,-20.31,183.6,8326,83.82
260,-21.35,193.4,7656,85.578
275,-22.62,211.7,6518,88.388
280,-22.97,218.1,6101,89.378
290,-23.56,230.5,5214,91.437
300,-23.98,241.2,4262,93.595
310,-24.26,249.3,3259,95.834
320,-24.41,254.3,2220,98.13
330,-24.46,256.1,1163,100.455
340,-24.46,254.8,104,102.782
340.56,-24.46,254.6,45,102.911
340.98,-24.45,254.5,1,103.01
```

ga.inp

```
$ga
 icreep=1,
 idum=-1001,
 ielite=1,
 iend=  0,
 iniche=0,
 irestrt=0,
 iskip= 0,
 itourny=1,
 iunifrm=1,
 kountmx=1,
 maxgen= 100,
 microga=0,
 nchild=2,
 nichflg=1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 nowrite=1,
 nparam= 12,
 npopsiz= 100,
 nposibl=256, 128, 256, 256, 256, 256, 512, 128, 128, 512, 128, 4,
 parmax=4.0, 60.0, 3.0, 3.0, 4.0, 2.0, 3000.0, 1.0, 0.50, 1.0, 25.0, 4,
 parmin=2.0, 25.0, 1.0, 1.0, 2.0, 1.0, 1000.0, 0.1, 0.05, 0.0,  5.0, 1.0,
 pcreep=0.06,
 pcross=0.5,
 pmutate=0.01,
$end
```

GA input variable descriptions

```
c  icreep   = 0 for no creep mutations
c           = 1 for creep mutations; creep mutations are recommended.
c  idum     = The initial random number seed for the GA run.  Must equal
c               a negative integer, e.g. idum=-1000.
c  ielite   = 0 for no elitism (best individual not necessarily
c               replicated from one generation to the next).
c           = 1 for elitism to be invoked (best individual replicated
c               into next generation); elitism is recommended.
c  iend     = 0 for normal GA run (this is standard).
c           = number of last population member to be looked at in a set
c               of individuals.  Setting iend-0 is only used for debugging
c               purposes and is commonly used in conjunction with iskip.
c  iniche   = 0 for no niching
c           = 1 for niching; niching is recommended.
c  irestrt  = 0 for a new GA run, or for a single function evaluation
c           = 1 for a restart continuation of a GA run.
c  iskip    = 0 for normal GA run (this is standard).
c           = number in population to look at a specific individual or
c               set of individuals.  Setting iskip-0 is only used for
c               debugging purposes.
c  itourny  = No longer used.  The GA is presently set up for only
c               tournament selection.
c  iunifrm  = 0 for single-point crossover
c           = 1 for uniform crossover; uniform crossover is recommended.
c  kountmx  = the maximum value of kount before a new restart file is
c               written; presently set to write every fifth generation.
c               Increasing this value will reduce I/O time requirements
c               and reduce wear and tear on your storage device
c  maxgen   = The maximum number of generations to run by the GA.
c               For a single function evaluation, set equal to 1.
c  microga  = 0 for normal conventional GA operation
c           = 1 for micro-GA operation (this will automatically reset
c               some of the other input flags).  I recommend using
c               npopsiz=5 when microga=1.
c  nchild   = 1 for one child per pair of parents (this is what I
c               typically use).
c           = 2 for two children per pair of parents (2 is most common)
c  nichflg  = array of 1/0 flags for whether or not niching occurs on
c               a particular parameter.  Set to 0 for no niching on
c               a parameter, set to 1 for niching to operate on parameter.
c               The default value is 1, but the implementation of niching
c               is still controlled by the flag iniche.
c  nowrite  = 0 to write detailed mutation and parameter adjustments
c           = 1 to not write detailed mutation and parameter adjustments
c  nparam   = Number of parameters (groups of bits) of each individual.
c             Make sure that nparam matches the number of values in the
c             parmin, parmax and nposibl input arrays.
c  npopsiz  = The population size of a GA run (typically 100 works well).
c             For a single calculation, set equal to 1.
c  nposibl  = array of integer number of possibilities per parameter.
c             For optimal code efficiency set nposibl=2**n, i.e. 2, 4,
c             8, 16, 32, 64, etc.
c  parmax   = array of the maximum allowed values of the parameters
c  parmin   = array of the minimum allowed values of the parameters
c  pcreep   = The creep mutation probability.  Typically set this
c             as (nchrome/nparam)/npopsiz.
c  pcross   = The crossover probability.  For single-point crossover, a
c             value of 0.6 or 0.7 is recommended.  For uniform crossover,
c             a value of 0.5 is suggested.
c  pmutate  = The jump mutation probability.  Typically set = 1/npopsiz.
```

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE Dec 99	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE REVERSE ENGINEERING OF FOREIGN MISSILES VIA GENETIC ALGORITHM			5. FUNDING NUMBERS
6. AUTHOR(S) DAGSI Jon D. Wollam			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT/ENY 2950 P St Wright-Patterson AFB, OH 45433-7765			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GSE/ENY/99D-01
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Mr. Skip Campbell NAIC/TANW 4180 Watson Way Wright-Patterson AFB OH 45433-5648 DSN 757-8800 and 937-257-8800			10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES Lt Col Stuart C Kramer DSN 785-3636 x 4318			
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release Distribution Unlimited			12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

One mission of the National Air Intelligence Center (NAIC) is the reverse engineering of foreign missile weapon systems from incomplete observational data. In the past, intuition and repeated runs of a missile performance model were required to converge to a solution compatible with observed flight characteristics. This approach can be cumbersome and time-consuming, as well as being subject to undesirable influences from the analyst's preconceptions and biases.

An alternative approach has been created to apply genetic algorithm (GA) techniques to allow automation of the process, wider exploration of the design space, and more optimal solutions matching the observational data. The GA, when interfaced with a missile performance model, was able to identify a set of missiles that very closely matched the observed performance of a given sample missile. The approach was able to provide the analyst with multiple candidate missiles for further analysis that would have been missed by the previous trial-and-error approach.

14. SUBJECT TERMS Genetic Algorithm, optimization, air-to-ground, missile			15. NUMBER OF PAGES 71
			16. PRICE CCDE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL