Evolutionary Algorithm for Artificial Immune System-Based Failure Detector Generation and Optimization

Jennifer Davis*, Mario G. Perhinschi†, Hever Moncayo‡

West Virginia University, Morgantown, WV, 26508

The development of an evolutionary algorithm and accompanying software for the generation and optimization of artificial immune system-based failure detectors using the negative selection strategy is presented in this paper. The utility is a part of an integrated set of methodologies for the detection, identification, and evaluation of a wide variety of aircraft sub-system abnormal conditions. A preliminary phase consists of processing data from flight tests for “self” definition including normalization, duplicate removal, and clustering. A first phase of the evolutionary algorithm produces, through an iterative process, a set of detectors that do not overlap with the “self” and achieve a prescribed level of coverage of the “non-self”. A second phase consists of a classic genetic algorithm that attempts to optimize the number of detectors, overlapping between detectors, and coverage of the “non-self” while maintaining no overlap with the “self”. For this second phase, an initial individual is a set of detectors obtained in the first phase. Specific genetic operators have been defined to accommodate different detector shapes, such as hyper-rectangles, hyper-spheres, and hyper-ellipsoids. An interactive design environment has been developed in Matlab that relies on an advanced user-friendly graphical interface and on a substantial library of alternative algorithms to allow maximum flexibility and effectiveness in the design of detector sets for artificial immune system-based abnormal condition detection.

I. Introduction

In recent years, the need for a solution to the failure detection, identification, and evaluation (FDIE) problem for aerospace vehicles that includes all sub-systems over the entire flight envelope has been widely acknowledged and has become a major objective of NASA’s Aviation Safety Program. The complexity and extremely high dimensionality of the problem require adequate tools. Recently, a new concept inspired from the biological immune system was proposed for aerospace systems failure detection. The Artificial Immune System (AIS)-based fault detection operates in a similar manner as does its biological counterpart - according to the principle of self/non-self discrimination - when it distinguishes between entities that belong to the organism and entities that do not. This paradigm can potentially address directly the complexity and multi-dimensionality of aircraft dynamic response in the context of abnormal conditions and provide the tools necessary for a comprehensive/integrated solution to the FDIE problem.

A critical issue for the AIS-based detection is the generation of adequate detectors, that is the definition of regions in the hyper-space of relevant parameters (identifiers) that are only reached when abnormal conditions are present. To date, there is no deterministic method to perform this task and available algorithms rely on random location of detectors and search for uncovered regions. In addition, the need to computationally optimize the detector set for on-line detection and to ensure maximum coverage of the self/non-self without overlapping for good detection performance, makes evolutionary or genetic algorithms a promising solution for the generation of AIS detectors.

An integrated set of methodologies for AIS-based detection, identification, and evaluation of a wide variety of aircraft sensor, actuator, propulsion, and structural failures/damages is currently under development at West Virginia University (WVU) within NASA’s Aviation Safety Program. As part of this research effort, computational tools were developed using evolutionary algorithms for the generation and optimization of AIS-based detectors.

* Graduate Student, Dept. of Mechanical and Aerospace Engineering
† Assistant Professor, Dept. of Mechanical and Aerospace Engineering, AIAA Senior Member.
‡ Graduate Student, Dept. of Mechanical and Aerospace Engineering
These tools were implemented within an interactive integrated design environment in Matlab/Simulink. The main purpose of this paper is to present the development and operation of this design environment.

The main aspects of the AIS paradigm and its application are presented in Section II. Section III consists of a brief review of the evolutionary or genetic algorithms. The general architecture of an integrated system of tools developed in Matlab/Simulink for the generation and optimization of AIS-based detectors is discussed in Section IV. Details on the main components and phases of the algorithm are presented in Section V. The graphical user interface and the options available for the AIS detector set design are presented in Section VI. In section VII, the functionality of the design tool is illustrated through some results. Finally, the conclusions are summarized in Section VIII followed by acknowledgements and a reference list.

II. The Artificial Immune System Paradigm

The biological immune system has the capability to detect microbial and non-microbial exogenous entities while not reacting to the self cells. T-cells are the component of the system with the most important role in this process. T-cells are first generated through a pseudo-random genetic rearrangement mechanism, which ensures high variability of the new cells in terms of biological identifiers. Typically, these identifiers are specific molecular strings of organic compounds such as proteins or polysaccharides. A selection process takes place in the thymus resulting in the destruction of the T-cells whose identifiers match the self. Eventually, only those T-cells that are “different” are allowed to leave the thymus and proliferate. This process is referred to as negative selection. The surviving T-cells can now circulate throughout the body to detect intruders and mark them for destruction.

The mechanisms and processes of the biological immune system are the inspiration for the AIS, as a new artificial intelligence technique for fault detection. The basic idea of this new computational paradigm is that an abnormal situation (i.e. failure of one of the aircraft sub-systems) can be declared when a current configuration of “identifiers” or “features” does not match with any configuration from a pre-determined set known to correspond to normal situations. These “identifiers” can include various sensor outputs, states estimates, statistical parameters, or any other information expected to be relevant to the behavior of the system and able to capture the signature of abnormal situations. Adequate numerical representations of the self/non-self must be used and the data processed such that they are manageable given the computational and storage limitations of the available hardware. The artificial counterpart of the T-cells - the detectors - must then be generated and optimized. This process may be repeated to generate several sets of detectors as part of a hierarchical scheme that allows failure isolation and evaluation. Finally, a detection logic must be designed for real time operation with a high detection rate and low number of false alarms. The block diagram of the AIS design process for fault detection is presented in Figure 1.

![Figure 1. Artificial Immune System-Based Abnormal Condition Detection](image-url)

The AIS concept has shown a promising potential for a variety of applications such as anomaly detection in computer operation, pattern recognition, data mining, computer security, and adaptive control. Dasgupta and KrishnaKumar have pioneered the use of AIS for fault detection for aerospace systems.
The success of the AIS-based FDIE scheme will depend on the capability of the parameters selected as identifiers (e.g. aircraft states and pilot input) to capture the dynamic signature of each and every type of failure. When the number of failure classes that are targeted is high, a large number of identifiers is necessary, thus increasing the dimensionality of the detector space and exposing the entire process to specific issues that can potentially have a negative impact on the performance of the FDIE scheme. For adequate detection performance and reduced computational effort, disjunct complete coverage of “self”/“non-self” and minimal overlapping between detectors must be accomplished with a reduced number of detectors. Deterministic methods are not available to solve this generation and optimization problem and current approaches rely on random initialization of candidate detectors and subsequent censoring to achieve sets of optimization criteria. In this context, evolutionary algorithms can potentially provide the necessary tools.

III. Evolutionary Algorithms

Evolutionary or genetic algorithms (EA) are a class of artificial intelligence techniques primarily developed for parameter optimization. Many important problems in sciences, engineering, and economics can be formulated as parameter optimization problems. EAs iteratively search the solution space relying on analogies to natural biological processes. They simulate the evolution of species and individual selection based on Darwin’s theories to direct the search towards a global optimum. EAs work simultaneously on a set (population) of potential solutions (individuals) to the problem at hand. Within the EA paradigm, individuals are also referred to as “chromosomes”. A set of design requirements and constraint (DRC) must be defined. They may be expressed mathematically, logically (binary or fuzzy), or in a descriptive manner and can be totally independent from one another. This capability is one of the strong points of EAs. This set of DRC play the role of the environment. Based on the “survival of the fittest” principle, the degree to which solutions meet the performance requirements and constraints is evaluated and used to select “surviving” individuals that will “reproduce” and generate a new population. Individuals will now undergo alterations similar to the natural genetic mutation, crossover, and possibly other genetic operators. The next iteration starts with this new population (new set of possible solutions). The process continues until there is no more significant increase in the performance of the best solution or a pre-set maximum number of iterations is reached. The block diagram of a typical EA is presented in Figure 2.

Optimization has become a major field of EAs applicability. As compared to the widely used gradient methods and enumerative schemes, EAs are global and robust over a broad spectrum of problems. EAs have shown excellent potential for solving highly complex nonlinear parameter optimization problems. In general, the advantages of using EAs include:

- **Parallelism.** The gradient-based methods are capable of searching for the solution in one direction at a time. If the algorithm does not converge or a local sub-optimum is reached the search must be reinitiated. The EAs perform the search in multiple directions due to the fitness-based selection and the randomness of the changes from one generation to the next.
• **Complexity of performance criteria.** The performance criteria (or fitness) do not have to be expressed as an analytical function. There are no requirements for continuity, derivability, or bijectivity. The formulation of the constraints can be arbitrary.

• **Globality.** If properly designed, EAs can escape the trap of local extrema and given enough iterations they will reach the global extremum. Related to this property, it should be noted that EAs can achieve a good balance between *exploration* and *exploitation*. “Exploration” is the process of browsing the solution space in search of “better” solution and “exploitation” is the process of tuning a “good” solution through searching within its vicinity.

• **Dimensionality.** The EAs are capable of efficiently handling problems with huge numbers of parameters and objectives (and/or constraints). In particular, this characteristic makes them attractive for AIS-based detector generation for aircraft sub-system FDIE.

• **Generality.** EAs are not necessarily dependent on the problem they are supposed to solve. They do not have to use previously known domain-specific information. This lack of preconceptions allows the EAs to investigate all possible search pathways and possibly search through regions excluded by more conventional algorithms. However, domain specific information can be used to constrain the search and avoid wasting computational time in regions that are not expected to lead to the solution.

IV. General Architecture of the EA for Failure Detector Generation and Optimization

A. Definitions

Several key concepts will be used throughout this paper with the following meanings. The *solution space* $\Sigma$ is the entire *universe* as defined by the identifiers considered within various phases of the FDIE process including both normal and abnormal flight conditions. The dimension of the solution space is equal to the number of identifiers. The *self* $S$ is the sub-set of $\Sigma$ corresponding to normal flight conditions, while the *non-self* $\overline{S}$, corresponds to abnormal conditions. Ideally, the self and the non-self are disjunct sets and completely cover the solution space:

$$\overline{S} \cap S = \emptyset \text{ and } \overline{S} \cup S = \Sigma$$

(1)

For computational convenience, the self and non-self are typically represented as sets of geometrical hyper-bodies referred to as *clusters* and *detectors*, respectively.

Within the EA, an *individual* is a potential solution to the failure detector generation and optimization (FDGO) problem, which is a single set of detectors covering the non-self. Several such individuals form the *population*.

B. Algorithm Architecture

A large repository of self data is necessary to the creation of a complete and comprehensive self. For increased computational and algorithmic effectiveness, the general structure of the EA for FDGO includes three main modules as shown in Figure 3:

• Data Preprocessing (normalization, duplicate data removal, and clustering);
• Phase I (generation of initial population of solutions through an iterative algorithm);
• Phase II (optimization of the solution through a classic EA).

**Figure 3.** Flowchart of Optimization Processes

The main purpose of the preprocessing is to reduce the memory and computation time needed for the FDGO. The first phase of the EA consists of an iterative algorithm that creates an initial set of detectors that do not overlap with the self and achieve a desired level of non-self coverage. Phase I is repeated as many times as necessary to produce an initial population for the classic genetic algorithm that represents Phase II. The solution is optimized to achieve minimum un-covered areas in the non-self, minimum overlapping among detectors, and a minimum number of detectors, while maintaining no overlapping between non-self detectors and self.

American Institute of Aeronautics and Astronautics
V. Description of Evolutionary Algorithm Modules

A. Preprocessing

The preprocessing of flight data includes: normalization, duplicate removal, and clustering. As a result of the normalization, each dimension (identifier measured values) is scaled to values between 0 and 1. Therefore, the solution space becomes a unit hypercube. The normalization factor for each dimension is determined as the span of the flight data plus a percentage margin. Alternatively, desired maximum and minimum values can be specified in the computation of the normalization factor. This approach is particularly useful when additional sets of self data are to be combined with previously acquired/processed ones and the same normalization factors must be used.

Removing duplicate points reduces redundancy within the data and can substantially increase the speed of the algorithm. A threshold must be selected that defines the vicinity of any data point within which all points are assumed to belong to the self. Any other data point that falls in this vicinity is therefore considered a duplicate and is removed. It should be noted that if the threshold is too large, non-self points may be included as self, which leads to detection errors. If the threshold is too small, then too much data redundancy may be allowed, which can increase the computational requirements. Pertinent values of this threshold can be obtained through analysis of the average distance between consecutive measurement points at adequate sampling rates.

Once the duplicate points have been removed, additional reduction of the memory and computational requirements can be achieved through clustering of the normalized flight data. An optimized version of the k-means\(^25\) clustering method is implemented within the WVU immunity-based failure detector optimization and testing (IFDOT) tool. The clusters are eventually represented as either hyper-spheres or hyper-rectangles. This allows flexibility in the generation of detectors as either hyper-spheres, hyper-ellipsoids, or hyper-rectangles. The reduction of empty space is achieved through an iterative clustering algorithm\(^26\) in which the number of clusters is progressively increased until the desired level of empty space is reached.

B. Phase 1—Generation of Detectors

Within Phase 1 of the EA, an initial population of potential solutions – sets of detectors – is generated. Currently, two methods for hyper-spherical detectors and one method for hyper-rectangular detectors are implemented within the IFDOT tool.

The first method implemented for hyper-spherical detector generation is a negative selection algorithm with real representation and variable detector size\(^27\) (NSA-RV). The flowchart of this algorithm is presented in Figure 4. Candidate detectors are first initialized by random generation of their centers. If the center does not fall within the self, the algorithm assigns a radius to it based on the nearest distance to the self. If this distance is greater than the minimum desirable detector radius, the candidate detector is accepted. The following stopping criteria exist for this algorithm:

- maximum allowed number of detectors is reached
- maximum number of consecutively generated candidate detectors overlapping other detectors or self-clusters is reached (shows likelihood of adequate coverage of the non-self)
- maximum number of detectors with radii smaller than a threshold are attempted (indicates that adequate coverage of small areas, such as between clusters, has been achieved).

The second method for hyper-spherical detector generation is an enhanced NSA-RV, which integrates NSA-RV with detector moving and cloning\(^7\). Detectors are generated iteratively. The algorithm begins by creating an initial number of detectors, in the same manner as the first method. Overlapping is calculated for each detector. Detectors are either matured or rejected based on an overlapping threshold. Rejected detectors are moved so that they can improve their overlapping. New candidate detectors are created in the vicinity of mature detectors. This process is referred to as “cloning”. New detectors are also inserted randomly, as in the initial process. The algorithm stops when there are enough mature detectors, or the maximum number of iterations has been performed. The flowchart of the enhanced NSA-RV is presented in Figure 5.

The method implemented for hyper-rectangular detector generation is similar to the NSA-RV with the following differences. Each hyper-rectangle detector contains a side-length corresponding to each identifier, which is measured from the center to the edge of the detector in each dimension. This differs from hyper-sphere detectors, since these detectors measure the same radius for all dimensions. For this reason, the distance for a dimension is set based on the shortest distance to the self from the center in each dimension. Because of the varying dimensions, the minimum distance in each dimension depends on the number of iterations the algorithm has performed and on a decay parameter, \(\tau\), which is set by the user. The equation for the decay parameter is:

\[
\text{minimum distance in each dimension} = \text{Resp} \times e^{-\frac{t}{\tau}}
\]  

(2)
where $R_{ss0}$ is the base minimum radius provided by the user. A flowchart for this function is provided below in Figure 6.

**Figure 4.** Flowchart of Detector Generation Using the NSA-RV

**Figure 5.** Flowchart of Detector Generation Using the Enhanced NSA-RV

**Figure 6.** Flowchart of Detector Generation for Hyper-Rectangles
C. Phase 2—Optimization of Detectors

1. Algorithm Layout

Phase II of the EA is a classic genetic algorithm that uses as individuals in the population, sets of detectors generated in Phase I. Each detector may be considered as a gene within the chromosome. Several options are available for the detector representation as geometric hyper-bodies. A three criteria performance index is used to assess the “fitness” of each individual. Four customized genetic operators have been defined. A new population is selected at each iteration based on the comparative fitness of each individual using the roulette wheel selection method enhanced with elitist strategy\textsuperscript{10,23}. This evolutionary search for the optimum solution continues for the specified number of generations. The flowchart of the EA is presented in Figure 7.

![Flowchart of Evolutionary Algorithm](image)

2. Representation of the Individual

Each individual is a set of detectors. Due to the specific nature of this application and the identifiers considered, real value representation for all dimensions of the detectors was used. Depending on the shape, the detectors are defined as follows. Hyper-spherical detectors \( D_S \) are defined as:

\[
D_S = (c_s, r_s)
\]

where \( c_s \in \mathbb{R}^n \) is the location of the center of the detector, \( r_s \in \mathbb{R} \) is the radius of the detector, and \( n \) is the dimension of the solution space. Hyper-rectangular detectors \( D_R \) are defined as:

\[
D_R = (c_r, d_r)
\]

where \( c_r \in \mathbb{R}^n \) is the location of the center of the detector, \( d_r \in \mathbb{R}^n \) is the side length of the detector in each dimension, and \( n \) is the dimension of the solution space. Hyper-ellipsoidal detectors are defined as:

\[
D_E = (c_e, a_e)
\]

where \( c_e \in \mathbb{R}^n \) is the location of the center of the ellipse, \( a_e \in \mathbb{R}^n \) is the axes vector for all dimensions, and \( n \) is the dimension of the solution space. Finally, rotational hyper-ellipsoidal detectors are defined as:

\[
D_{RE} = (c_{re}, a_{re})
\]

where \( c_{re} \in \mathbb{R}^n \) is the location of the center of the detector, \( a_{re} \in \mathbb{R}^2 \) is the axes vector, and \( n \) is the dimension of the solution space. Note that, unlike hyper-ellipsoids, which may have different axes for each dimension, only one preferential axis may differ from the others.

3. Rating the Population

The “fitness” of the individuals is evaluated based on the following criteria:
• number of detectors in the set
• percentage of non-self that is covered by detectors
• percentage of overlapping that occurs within the detector set

Better individuals ideally have a small number of detectors, no overlap, and cover the entire non-self. Each of these factors must be balanced according to their importance in order to produce the optimized set of detectors, thus a weight factor, $W$, must be specified by the designer for each of the three criteria. The evaluation function for each of these performance criteria is linear, scaled from a user-specified lower limit to a user-specified upper limit. These relationships are given below in equations 7, 8, and 9.

$$P_{\text{coverage}} = \frac{1}{L_{\text{coverage}} - U_{\text{coverage}}} \sum_{k=1}^{n} \left( \frac{k}{L_{\text{coverage}}} - \frac{k}{U_{\text{coverage}}} \right)$$ (7)

where $P_{\text{coverage}}$ is the performance index of the individual $i$ with respect to the coverage criterion, coverage is the coverage of the individual, $L_{\text{coverage}}$ is the lowest acceptable coverage, and $U_{\text{coverage}}$ is the highest expected coverage.

$$P_{\text{overlapping}} = \frac{1}{L_{\text{overlapping}} - U_{\text{overlapping}}} \sum_{k=1}^{n} \left( \frac{k}{L_{\text{overlapping}}} - \frac{k}{U_{\text{overlapping}}} \right) + 1$$ (8)

where $P_{\text{overlapping}}$ is the performance index of the individual $i$ with respect to the overlap criterion, overlapping is the overlapping of the individual $i$, $L_{\text{overlapping}}$ is the highest acceptable overlapping, and $U_{\text{overlapping}}$ is the lowest expected overlapping.

$$P_{\text{number}} = \frac{1}{L_{\text{number}} - U_{\text{number}}} \sum_{k=1}^{n} \left( \frac{k}{L_{\text{number}}} - \frac{k}{U_{\text{number}}} \right) + 1$$ (9)

where $P_{\text{number}}$ is the performance index of the individual $i$ with respect to the number of detectors criterion, $L_{\text{number}}$ is the highest acceptable number of detectors, and $U_{\text{number}}$ is the lowest expected number of detectors.

A small number of detectors implies reduced computational requirements. It will also require larger size of the detectors, which is acceptable for general detection where only good coverage of the non-self is necessary. However, for failure identification, smaller detectors may be preferable as they provide better resolution and may be able to distinguish between failures within the same category.

High coverage is absolutely necessary to achieve high detection rates. Any areas of the non-self that are not covered by detectors will be considered self and not trigger detection. Typically, in order to obtain acceptable coverage, a large number of detectors are needed.

Overlapping is not desirable. Although it can be argued that it is better to have overlapping in an area than no coverage, overlapping produces redundancy and increases calculation time.

4. Selecting the Next Generation

Selection of the new population for the next generation is performed based on the performance index of each individual, relative to the total performance of the population. The roulette-wheel selection\(^\text{23}\) is the method used in this application. Each individual in the population has a performance index $P_{\text{individual}}$ computed as:

$$P_{\text{individual}} = \sum_{k=1}^{n} \left( \frac{k}{L_{\text{coverage}}} - \frac{k}{U_{\text{coverage}}} \right) + \sum_{k=1}^{n} \left( \frac{k}{L_{\text{overlapping}}} - \frac{k}{U_{\text{overlapping}}} \right) + \sum_{k=1}^{n} \left( \frac{k}{L_{\text{number}}} - \frac{k}{U_{\text{number}}} \right)$$ (10)

The total performance index $TF$ is the sum of all of the performance indices for all individuals in the population:

$$TF = \sum_{i=1}^{n} P_{i}$$ (11)

The performance index of each individual is divided by the total performance index of the current population to obtain the probability of selection for each individual $p_i$:

$$p_i = \frac{P_{\text{individual}}}{TF}$$ (12)

The cumulative probability is calculated next for each of the individuals, as the sum of the probabilities of all precedent individuals:

$$q_i = \sum_{j=1}^{i} p_j$$ (13)

Since the population size is maintained constant throughout the algorithm, the population in each generation can only contain the same number of individuals. Each available spot in the new population is filled by generating a random number and selecting the individual for which the random number is less than its cumulative probability but greater than the cumulative probability of the preceding individual. Therefore, individuals with higher performance indices will get larger cumulative probability intervals, and more chances for multiple copies in the new generation. The algorithm continues until the next generation is fully populated. A flowchart of this process is presented in Figure 8.
5. Genetic Operators

Four distinct genetic operators (GO) - mutation, addition, removal, and crossover - are performed on the population according to GO rates established by the designer. The individuals that are subject to genetic alteration are selected randomly.

Mutation

The mutation GO was designed with the intention to produce small alterations of the individuals, in an effort to focus the search in the vicinity of existing solutions. In general, this operator may change the overlap and coverage values of an individual/set of detectors by altering the location, radius, or orientation of a single detector/gene by a small increment. The individual and the gene subject to mutation are selected randomly.

For spheres and rectangles, there are two types of mutation: gene alteration and gene relocation. Gene alteration consists of randomly increasing or decreasing the radius of the detector by a random amount within a range specified by the designer. In the event that the detector is rectangular, the dimension to be altered is also selected at random. Gene relocation involves randomly selecting an axis of the detector and moving the center of the detector a random amount up to a multiple of the radius, as specified by the user.

For ellipsoids and rotational ellipsoids, an additional type of mutation is available, which produces alterations of the orientation of the detector. An axis is selected at random and the detector is rotated an amount at random about that axis. A flowchart of the mutation GO is shown in Figure 9.
Gene Addition

The gene addition GO is aimed at increasing coverage without increasing overlapping. As a result of this operator, new detectors are added to a particular individual, chosen at random from the population. New points are generated randomly in the solution space until one is found that falls neither within the self region nor within another detector. This becomes the center of the added detector. To this center, a radius or side length equal to the distance to the nearest object is assigned. In this way no overlapping is produced and any new detectors cover area that was not previously covered by any other detector. Up to a user-specified amount of detectors can be added to an individual at one time. Note that adding new detectors may conflict with the requirement of reducing the number of detectors and reduce the performance index of the individual. This process is repeated until the addition rate is satisfied for each pass of the genetic operator. A flowchart of this GO is presented in Figure 10.

Two variations exist for this GO. One version favors and inserts larger detectors. Large detectors are better for detection in that they cover more of the solution space and tend to reduce the number of detectors needed to achieve a certain level of coverage. The other version of this GO favors and inserts smaller detectors. Small detectors tend to be better for identification. Since they cover a smaller region of the solution space, they provide better resolution and allow to distinguish between failures in the same category.

Gene Removal

The gene removal GO is intended to decrease overlapping within an individual. This algorithm randomly chooses an individual, and then calculates the overlapping for each detector within the individual. The detectors are ranked according to their percentage of overlap with other detectors and the detector with the greatest overlap is removed. This continues until the detector removal rate is satisfied. A flowchart of this simple genetic operator is shown in Figure 11. This GO must be used with caution. Removing detectors may significantly decrease the coverage.
Crossover

To apply the crossover GO, two individuals are chosen at random. A random number of detectors \( N_{CO} \) - up to a maximum that is initially set by the user – is first established. The crossover point \( P_{CO} \) is randomly selected as an \( n \)-dimensional point in the non-self. Then, \( N_{CO} \) detectors closest to \( P_{CO} \) from the two individuals are interchanged. The \( N_{CO} \) detectors from both individuals maintain the same location within the solution space after the crossover GO is applied. Therefore, non-overlapping with the self, following this GO, is guaranteed. Figure 12 contains a flowchart of this operator.

![Flowchart of Crossover Genetic Operator](image)

Figure 12. Flowchart of Crossover Genetic Operator

6. Elitist Selection

Elitist selection is a strategy that allows the best individual to survive unaltered from one generation to the next. The best individual of the current generation is simply introduced into the next. This ensures that good solutions obtained at different moments during the EA are not lost without finding a better one, thus emphasizing exploitation and accelerating convergence.

VI. Description of the Interactive Utility for AIS Detector Design

The interactive design environment for FDGO developed in Matlab/Simulink relies on an advanced user-friendly graphical interface and on a substantial library of alternative algorithms to allow maximum flexibility and effectiveness in the design of detector sets for artificial immune system-based abnormal condition detection.

The main menu presented in Figure 13 is the portal to the design environment. At any point, the user can return here to repeat certain operations or re-direct the design sequence. The ‘File’ drop-down menu allows the selection of one of the following design tasks:

- Normalization of flight data;
- Elimination of duplicates within data files;
- Clustering of flight data to produce a “self” with two shape options;
- Generation of positive and negative selection detection schemes with two shape options;
- Optimization of negative selection detection schemes with four shape options;
- Results reporting;
- File merging;
- Detection testing.

The ‘Help’ menu provides access to a detailed User’s Guide including descriptions of available algorithms, required parameters, and instructions on performing various tasks.

A. Selecting Files

Typically, for any task that is performed using the interactive utility, an appropriate data file containing data and/or set-up parameters must be available. These files can be created using the Interactive Utility. However, at least one data file containing flight test data collected under normal conditions is necessary to perform the FDGO
and must be generated prior to the use of the IFDOT tool. This raw data file should contain only the time histories of the identifiers to be used for FDGO.

Figure 13. Main Menu of the WVU IFDOT Design Environment

B. Preparing Data
Several parameters are available for processing data. These are as follows:

- Normalization limit;
- Normalization grace margin (only for applicable method);
- Duplicate removal tolerance.

The data normalization menu allows the user to complete one of the methods for normalizing the data. Data can be normalized to a percentage margin of the file data or normalized to specified limits. If the data is to be normalized to specific limits, these can be specified manually, or by retrieving the limits from a previously processed data file.

Duplicates are removed using only one method. Duplicates should be removed to eliminate redundant data and speed up future algorithms. This is done by specifying a similarity tolerance, called the duplicate removal tolerance.

Two raw data files can also be combined in this section. This may be performed before the files are processed. Other options exist throughout the program to allow combination of other types of data files as well.

C. Clustering Data
Test data can be clustered using two shapes, hyper-spheres or hyper-rectangles. Spherical clusters are used to generate spherical, ellipsoidal, and rotational ellipsoidal detectors. Rectangular clusters can only be used in conjunction with generation of rectangular detectors. Spheres can be clustered using two different methods. Number-Imposed Clustering is generally the quicker option; however, it does not optimize the empty space within the clusters. Empty Space-Optimized Clustering generally results in clusters with more accurate self coverage, but takes considerably longer to compute. For clustering using rectangles, only one method is available, which is equivalent to the spheres Number-Imposed Clustering.

For the spherical Number-Imposed Clustering Method, the following parameters need to be specified:

- Desired number of clusters (specifies the maximum number of clusters that may be used);
- Minimum cluster radius (ensures that no cluster is too small);
- Confidence percentage (stopping criterion for the Monte-Carlo Method for overlapping and coverage calculation);
For the spherical Space-Optimized Clustering Method, the following parameters need to be specified:

- Initial number of clusters (starting number of clusters);
- Cluster increase step (additional number of clusters to be investigated at each iteration);
- Maximum number of clusters (the allowable number of clusters in the set);
- Point radius (confidence radius around each data point that is assumed to belong to the self);
- Acceptable empty percentage (the allowable amount of empty space a cluster may contain);
- Confidence percentage (stopping criterion for the Monte-Carlo Method);
- Permitted error (stopping criterion for the Monte-Carlo Method).

The rectangle clustering method requires that the following parameters are specified:

- Desired number of clusters (the maximum number of clusters that can be used in the set);
- Minimum cluster dimension (specifies the smallest allowable side length for the cluster);
- Confidence percentage (as above);
- Permitted error (as above).

All clustering methods will return the final number of clusters, their overlap, and their coverage of the total hyper-cube (solution space). These values are useful to the user when determining the parameters for the genetic algorithm.

**D. Performing the Genetic Algorithm:**

This section is intended to cover performing the genetic algorithm optimization. It assumes that clustering has already been performed, and that the necessary data files are available to the user. Performing the genetic algorithm has two phases. First the detectors are generated to create the initial population, and then they are altered using the genetic algorithm, to arrive at an optimized set. The genetic algorithm main menu is shown below in Figure 14. Several parameters must be selected here in order to perform the genetic algorithm. The genetic algorithm may be performed using several different detector shapes, such as spheres, ellipsoids, rotational ellipsoids, and rectangles. Note that the clustered data file will determine which of these options is adequate. If the data was clustered using spherical clusters, spheres, ellipsoids, and rotational ellipsoids will be available. If the data was clustered using rectangle clusters, only rectangles will be available.

If ellipsoid or rotational ellipsoid detectors are chosen, the mutation parameters will be different from above and will appear as shown below in Figure 15. In addition, if enhanced NSA-RV is used, the detector generation parameters selection menu will be the one presented in Figure 16.

The parameters for each method will be listed next, starting with Phase I, detector generation. Three methods are implemented for detector generation. Two different methods are capable of producing spherical detectors. One additional method is available to produce rectangular detectors.

The simple method of generating spheres, - NSA-RV - requires the following inputs:

- Maximum number of detector (maximum allowed in any individual);
- Minimum detector radius (smallest allowable detector size);
- Maximum self-coverage (stopping criterion that occurs when enough detectors that are not of an acceptable size are generated);
- Maximum non-self coverage (stopping criterion that occurs after too many random centers are rejected for falling in the existing self clusters or non-self detectors).

The second detector generation method – the enhanced NSA-RV - requires the following parameters to be specified:

- Minimum detector radius (smallest allowable detector size);
- Initial number of detectors 9 starting point of the algorithm);
- Maximum number of detectors (limits the algorithm and the size of the individual);
- Maximum number of iterations (stopping criteria for the detector generation algorithm);
- Number of random detectors each iteration (defines the number of random detectors that will be added each iteration);
- Number of detectors to move each iteration (number of rejected detectors that will be moved);
- Initial adaptation rate (determines the distance detectors will be moved);
- Decay parameter (decreases the distance detectors are moved each iteration);
- Permitted overlapping threshold (the amount of overlapping a detector can have and become matured);
- Number of nearest points considered in cloning (determines how many clones will be produced at each iteration);
- Number of nearest points considered in moving (determines how many detectors will be moved in each iteration);
- Initial adaptation distance used to locate new clones (determines the distance away clones will be generated);
- Cloning decay parameter (reduces at each iteration the distance away clones will be generated).

The parameters for generating rectangular detectors are as follows:
- Maximum number of detectors (the size limit of each individual);
- Initial minimum detector dimension (the smallest acceptable detector size);
- Decay parameter (decreases the smallest acceptable detector size at each iteration);
- Expected coverage (coverage desired from the detector set).
Phase II requires a considerable number of input parameters. For all shapes, the performance index, crossover parameters, add/remove parameters, and GA parameters are the same. Only for the mutation parameters does the detector shape change the necessary inputs.

The performance index parameters consist of the following:
- Coverage weight (determines the relative importance of coverage);
- Overlapping weight (determines the relative importance of overlapping);
- Number of detectors weight (determines the relative importance of number of detectors);
- Coverage upper bound (best expected coverage);
- Coverage lower bound (worst expected coverage);
- Overlapping upper bound (lowest expected overlapping);
- Overlapping lower bound (largest expected overlapping);
- Number upper bound (lowest expected number of detectors);
- Number lower bound (maximum number of detectors).

The performance index weights are normalized within the program. Note that for all bound limits, the lower bound corresponds to a rating of 0, and the upper bound corresponds to a rating of 1.

The parameters defining the crossover GO are:
- Crossover rate (percentage of individuals selected for this GO);
- Maximum detectors to cross (maximum number of detectors to undergo this GO).

The parameters defining the add/remove GOS consist of:
- Add rate (percentage of individuals selected for this GO);
- Remove rate (percentage of individuals selected for this GO);
- Number of centers to attempt (number of centers generated by the algorithm);
- Maximum number of detectors to add (largest number of detectors to be added to an individual at one time).

The genetic algorithm properties are:
- Population size (number of individuals);
- Number of generations (stopping criterion for the FDGO algorithm).

Mutation parameters depend on the shape of the detectors. For spheres, the mutation parameters will be as follows:
- Mutation rate (percentage of individuals selected for this GO);
- Chromosomal mutation rate (number of detectors in a selected individual that will be mutated);
- Gene alteration constant (defines the maximum amount the radius can be altered by);
- Gene relocation constant (determines the maximum distance a center can be moved);
- Gene alteration weight (determines the probability of gene alteration);
- Gene relocation weight (determines the probability of gene relocation).

For ellipsoids, the following parameters are requested:
- Mutation rate (percentage of individuals selected for this GO);
- Chromosomal mutation rate (number of detectors in a selected individual that will be mutated);
- Gene alteration constant (defines the maximum amount the radius can be altered by);
- Gene relocation constant (determines the maximum distance a center can be moved);
- Gene rotation constant (maximum number of degrees a detector can be rotated);
- Gene alteration weight (determines the probability of gene alteration);
- Gene relocation weight (determines the probability of gene relocation);
- Gene rotation weight (determines the probability of gene rotation).

VII. Example Results Using the Design Environment for AIS Detector Generation and Optimization

In this section, two sets of results obtained through the use of the IFDOT tool will be presented. The first set of results consists of examples in 2-dimensions and 6-dimensions to illustrate the functionality and effectiveness of the EA. The initial population for each of these examples was prepared by generating initial individuals with poor coverage. For each individual, the detectors present in the initially-generated set were multiplied to arrive at initial individuals with full overlap, higher numbers of detectors, and poor coverage. A second set of results involves a 3-dimensional solution space, using the parameters roll, pitch, and yaw rate neural network outputs, to show that the EA can potentially improve the performance of the detection scheme. Comparisons between detection rates for optimized and unoptimized schemes are presented.
A. 2-Dimensional Example

In this example, a general set of self data containing roll-rate and pitch-rate is used. This example was performed for 50 generations on a population of 10 individuals. The maximum number of detectors was limited to 54. A 30% mutation rate, crossover rate, and add rate was used, as well as a 200% remove rate. These are rather aggressive alteration rates, intended to increase the exploration within the solution space. The initial individual is shown below in Figure 17. This individual contains 54 detectors, due to multiplication. It has a coverage of 63.5% and an overlap of 100%.

![Initial Individual](image1)

Figure 17. Initial Individual

![Intermediate Individual](image2)

Figure 18. Intermediate Individual

![Final Best Individual](image3)

Figure 19. Final Best Individual

Figure 18 shows an individual from an intermediate population. This individual is used to illustrate the movement and removal of detectors during optimization. Several detectors can be seen which are nearly overlapping. These remain from the initial population in which these detectors were identical. The high removal rate used for this example is intended to eliminate all occurrences of such extreme detector overlap.

![Performance Indices](image4)

Figure 20. Performance Indices for the 2-D Example
At the end of 50 generations, the best individual improved to the solution shown in Figure 19. It contains 28 detectors, exhibiting a coverage of 73.34% and no overlap. All parameters in this example improved from the initial individual: number of detectors decreased, coverage increased, and overlap was eliminated entirely. The best individual and the average performance index are shown in Figure 20.

B. 6-Dimensional Example
The purpose of this next example is to illustrate the operation of the IFDOT tool in higher dimensions. A 6-dimensional initial set of individuals is created as in the previous example. The genetic alteration rates are the same, except for the remove rate, which is 30%. The best initial individual in this population contains 66 detectors and has a coverage of 34.17% and overlapping of 100%. After 50 generations, the best solution consists of 244 detectors, with a coverage increased to 58.33% and overlapping reduced to 70.00%. The plot of best performance index and average performance index versus generation is shown below in Figure 21. This plot suggests that an increasing larger number of generations will be required to reach similar values of the performance as in the previous example. It may also suggest that the genetic alteration rates are low, thus favoring exploitation at the expense of exploration leading to increased risks of convergence to local extrema. These observations are consistent with the higher dimensionality of the solution space.

C. Effects of Detector Optimization on the Detection Performance
The purpose of this analysis is to illustrate the improvement in detection performance that can potentially be achieved through Phase II optimization. In this example, a 3-dimensional self is considered, consisting of neural network outputs for roll, pitch, and yaw rates (NNp, NNq, and NNr). The neural networks are part of a fault tolerant controller29 and their outputs in terms of compensatory angular accelerations have been shown to possess good detection capabilities29. The actuator failures considered are left (L) or right (R) control surface, which can be stabilator, aileron, or rudder, locked at trim position plus 8 degrees. The sensors affected by failures are the angular rate sensors on the three channels, roll (p), pitch (q), and yaw (r). Two types of failures have been investigated: large step bias (LSB) of 10 deg/s and large fast drifting bias (LFDB) reaching 10 deg/s in 0.3 s. Finally, a structural failure is considered as well consisting of a 35% reduction in aerodynamic coefficients and mass of the left or right wing. Detection has been performed using two different sets of detectors obtained as the result of Phase I. One set contains a relatively high number of detectors, 495, with a coverage of 89.35%, and an overlap of 49.81%. The second Phase I set contains a relatively low number of detectors, 248, with a coverage of 78.71%, and an overlap of...

![Figure 21. Performance Indices for the 6-D Example](image-url)
A third detection scheme resulting from Phase II of the EA after 100 generation has a coverage of 89.19%, overlapping of 58.80%, and contains 298 detectors. This individual contains nearly the same number of detectors as the smaller first set generated by Phase I, but exhibits equivalent or better detection than either set of detectors generated by Phase I as presented in Table 1. In other words, the Phase II optimization can potentially achieve similar detection performance with fewer detectors and/or better performance with the same number of detectors as the sets obtained using Phase I algorithms. It has been observed that Phase I is more exposed to missing coverage of important areas of the non-self as it can be inferred from the poor detection of the right aileron failure in Table 1.

When analyzing these results, it is important to note the failure specificity of each set of identifiers. This means that different failures may require different parameters to capture their signatures. Therefore, a given set of identifiers may have good potential in detecting certain classes of failure and may not be appropriate for others simply due to their nature. The optimization of detector set does not change the nature of the identifiers; therefore, it may help reach their potential but does not create such capabilities when they do not exist. Another consequence of the failure specificity is that successful detection schemes can be developed using several sets of detectors (based on different identifiers) each of lower dimension instead of one large, high-dimensional set. In any case, the number of detectors may become critical to the performance of the FDI scheme.

Table 1—Rates of Detection Before and After Optimization of Detector Set

<table>
<thead>
<tr>
<th>Phase</th>
<th>Actuator Failure 8 deg</th>
<th>Sensor Failure</th>
<th>Structural Failure 35%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stabilator</td>
<td>Aileron</td>
<td>Rudder</td>
</tr>
<tr>
<td>Phase I</td>
<td>248 Antibodies, coverage 78.71%, overlap 27.41%</td>
<td>L: 99.96</td>
<td>L: 90.65</td>
</tr>
<tr>
<td></td>
<td>R: 98.05</td>
<td>R: 4.74</td>
<td>R: 47.94</td>
</tr>
<tr>
<td></td>
<td>R: 98.02</td>
<td>R: 99.51</td>
<td>R: 49.72</td>
</tr>
<tr>
<td>Phase II</td>
<td>298 Antibodies, coverage: 89.19%, overlap 58.80%</td>
<td>L: 99.97</td>
<td>L: 99.95</td>
</tr>
<tr>
<td></td>
<td>R: 99.80</td>
<td>R: 95.74</td>
<td>R: 50.16</td>
</tr>
</tbody>
</table>

VIII. Conclusions

An interactive design tool for AIS detectors generation and optimization has been developed in Matlab and tested. It includes a user-friendly graphical interface and a substantial library of alternative algorithms to allow maximum flexibility and effectiveness in the design.

A two phase evolutionary algorithm optimizes the set of detectors for coverage of the non-self, number of detectors, and overlapping, while maintaining no overlapping with the self.

A limited number of simplified tests were performed demonstrating the correct functionality of the algorithms and the expected improvement in overall detection performance after detector set optimization.

Acknowledgment

This research effort was sponsored by NASA Aviation Safety Program through a grant within the Integrated Resilient Aircraft Control project and by the NASA West Virginia Space Grant Consortium through a grant within the Research Seed Grant program.

References


