JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 2, March–April 2010

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

Jennifer Davis,[*] Mario G. Perhinschi,[†] and Hever Moncayo[*]
West Virginia University, Morgantown, West Virginia 26508

DOI: 10.2514/1.46126

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. A detector is defined as a subregion of the hyperspace formed by relevant system parameters at abnormal conditions. The utility is a part of an integrated set of methodologies for the detection, identification, and evaluation of a wide variety of aircraft subsystem abnormal conditions. The process of generating and optimizing detectors has several phases. 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 nonself. A second phase consists of a classic genetic algorithm that attempts to optimize the number of detectors, overlapping between detectors, and coverage of the nonself 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 hyperrectangles, hyperspheres, and hyperellipsoids. 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. The desirable performance of the proposed methodology is demonstrated by comparing the detection results for aircraft actuator failures of two unoptimized detector sets with the detection results of an optimized detector set. These results show that the algorithm can determine equal or better detection performance while using fewer detectors to cover the nonself.

Nomenclature

a	=	semi-axis length vector
c	=	center
D	=	detector, genotypes
d	=	semi-side length vector, applicable to rectangles
L	=	lower, or worse, limit
N	=	population size
PI	=	performance-index value
p	=	probability of selection
q	=	cumulative probability of selection
r	=	radius
$\underline{S}$	=	self
$\bar{\underline{S}}$	=	nonself
TF	=	total fitness
U	=	upper, or better, limit
W	=	performance-index weight
Σ	=	universe, all possible solutions

Subscripts

coverage	=	coverage performance-index criterion
E	=	hyperellipsoids
e	=	hyperellipsoids
i	=	referring to a value for a particular individual
number	=	number of detectors performance-index criterion
overlap	=	overlap performance-index criterion
S	=	hyperspheres
s	=	hyperspheres
R	=	hyperrectangles
RE	=	hyperrotational ellipsoids
r	=	hyperrectangles
re	=	hyperrotational ellipsoids

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 subsystems over the entire flight envelope has been widely acknowledged [1–4] and has become a major objective of NASA's Aviation Safety Program [5]. Previous methods, including state estimation or observer-based schemes [6–8] relying on Kalman or other filters and artificial neural-network-based schemes [9,10], have focused on specific failures and limited areas of the flight envelope. The complexity and extremely high dimensionality of the problem of detecting various aircraft subsystem failures over the entire flight envelope require adequate tools. Recently, a new concept inspired from the biological immune system was proposed for aerospace systems failure detection [11,12]. The artificial immune system (AIS)-based fault detection operates in a similar manner, as does its biological counterpart (according to the principle of self/nonself discrimination) when it distinguishes between entities that belong to the organism and entities that do not. This paradigm can potentially directly address the complexity and multidimensionality of aircraft dynamic response in the context of abnormal conditions and can 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 [13]: that is, the definition of regions in the hyperspace 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 online detection and to ensure maximum coverage of the self/nonself without overlapping for good detection performance makes

Received 24 June 2009; revision received 22 September 2009; accepted for publication 16 October 2009. Copyright © 2009 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0731-5090/10 and $10.00 in correspondence with the CCC.

[*]Graduate Student, Department of Mechanical and Aerospace Engineering.

[†]Assistant Professor, Department of Mechanical and Aerospace Engineering. Senior Member AIAA.

evolutionary or genetic algorithms [14,15] 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 [16] 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. 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 paper is organized in the following manner. An introduction to the basics of AIS and evolutionary algorithms (EAs) is presented in Secs. II and III, respectively. Section IV describes the general architecture of the computational tools developed for the generation and optimization of AIS detectors. Each of the algorithm modules is described in detail in Sec. V. The interactive utility is presented in Sec. VI, including the outline of the design options available. Results illustrating the functionality and performance of the EA are discussed in Sec. VII. Finally, some conclusions regarding the EA's performance are summarized in Sec. VIII.

II. Artificial-Immune-System Paradigm

The biological immune system has the capability to detect microbial and nonmicrobial exogenous entities while not reacting to the self cells. T cells [17] are the component of the system with the most important role in this process. T cells are first generated through a pseudorandom 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 for which the 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 [13,18,19]. The basic idea of this new computational paradigm is that an abnormal situation (i.e., failure of one of the aircraft subsystems) can be declared when a current configuration of identifiers or features does not match with any configuration from a predetermined set known to correspond to normal situations. These identifiers, which can be likened to the identifiers present on biological T cells, 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. Extensive experimental data are necessary to determine the self or the hyperspace of normal conditions, which is comparable to the body's own cells in the biological immune system. Adequate numerical representations of the self/nonself 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 [16] is presented in Fig. 1.

The AIS concept has shown a promising potential for a variety of applications [20], such as anomaly detection in computer operation [21], pattern recognition [22], data mining [23], computer security [24], and adaptive control [25]. KrishnaKumar [11] and Dasgupta [12] 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 [26] 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/nonself 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 [27,28] are a class of artificial intelligence techniques primarily developed for parameter optimization [29]. 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 toward 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 plays 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

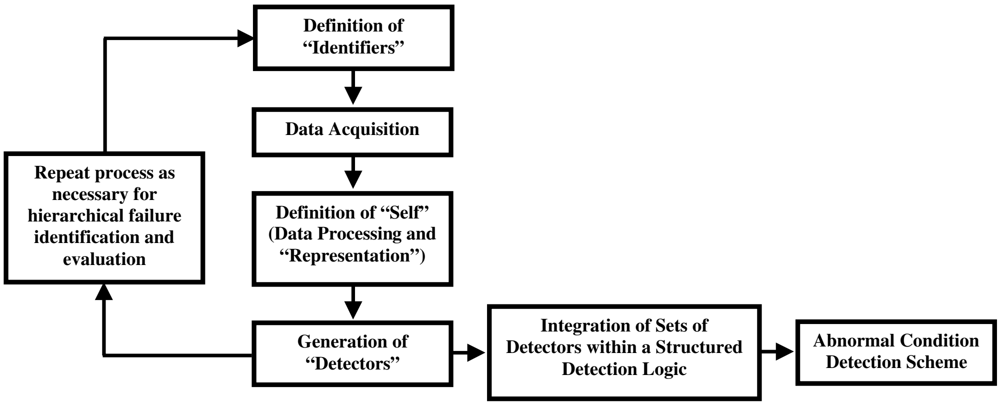

Definition of "Identifiers"
Data Acquisition
Definition of "Self" (Data Processing and "Representation")
Generation of "Detectors"
Repeat process as necessary for hierarchical failure identification and evaluation
Integration of Sets of Detectors within a Structured Detection Logic
Abnormal Condition Detection Scheme

Fig. 1 Artificial-immune-system-based abnormal-condition detection.

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 preset maximum number of iterations is reached. The block diagram of a typical EA is presented in Fig. 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 the following:

1) EAs are characterized by 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 suboptimum 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.

2) EAs can handle high-complexity 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.

3) The search for a solution within an EA is global. 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.

4) EAs are robust to high 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 subsystem FDIE.

5) EAs are also characterized by 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

A. Definitions

Several key concepts will be used throughout this paper with the following meanings. The *solution space* Σ is the entire *universe*, or every possible solution within the solution set, 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 identifiers can be any pertinent system measurement (such as system states) or computed variables (such as automatic control-law compensation or state estimates). For this application, the system is an aircraft, but this algorithm is not system-specific. The identifier values are normalized between 0 and 1, covering the entire possible range under normal conditions. The *self* S is the subset of Σ corresponding to normal flight conditions, and the *nonself* $\bar{S}$ corresponds to abnormal conditions. Ideally, the self and the nonself are disjunct sets and completely cover the solution space:

$$\bar{S} \cap S = 0 \quad \text{and} \quad \bar{S} \cup S = \Sigma \tag{1}$$

For computational convenience, the self and nonself are typically represented as sets of geometrical hyperbodies referred to as *clusters* and *detectors*, respectively. In other words, the clusters are subregions of the self, and the detectors are subregions of the nonself.

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 nonself, as illustrated in Fig. 3. 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-based utility for FDGO includes three main modules, as shown in Figure 4:

1) Data preprocessing is normalization, duplicate data removal, and clustering.

2) Phase I is the generation of the initial population of solutions through an iterative algorithm.

3) Phase II is the optimization of the solution through a classic genetic algorithm.

The main purpose of the preprocessing is to reduce the memory and computation time needed for the FDGO. The EA has two phases. The first phase consists of an iterative algorithm that creates an initial set of detectors that do not overlap with the self and achieve a desired

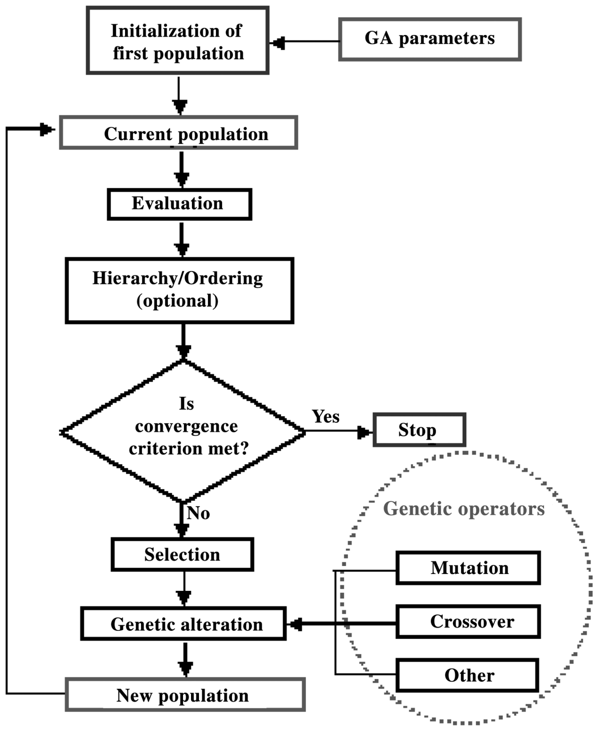


Fig. 2 Block diagram of an evolutionary algorithm.

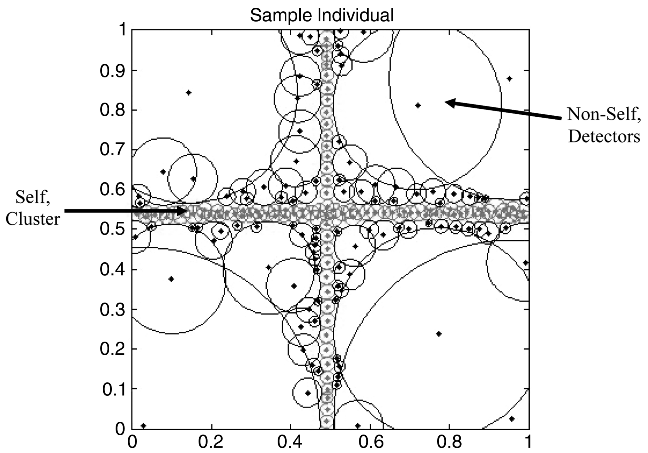


Fig. 3 Example of an individual in two dimensions.

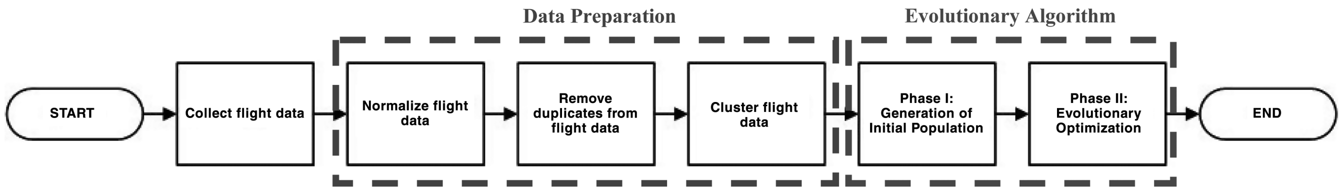

Data Preparation
Evolutionary Algorithm
START
Collect flight data
Normalize flight data
Remove duplicates from flight data
Cluster flight data
Phase I: Generation of Initial Population
Phase II: Evolutionary Optimization
END

Fig. 4 Flowchart of optimization processes.

level of nonself 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 uncovered areas in the nonself, minimum overlapping among detectors, and a minimum number of detectors, while maintaining no overlapping between nonself detectors and self. As compared to existing algorithms (some of them included within phase I), the proposed approach performs detector-set optimization based on relevant criteria to improve performance of the detection scheme and reduce the computational effort. The FDGO utility provides a comprehensive framework in terms of design parameters and options for AIS detector-set development that is application-independent.

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 sets, as 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, nonself 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-mean* [30] clustering method is implemented within the WVU immunity-based failure-detector optimization and testing (IFDOT) tool. The clusters are eventually represented as either hyperspheres or hyperrectangles. This allows flexibility in the generation of detectors as either hyperspheres, hyperellipsoids, or hyperrectangles. The reduction of empty space is achieved through an iterative clustering algorithm [31] 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 I of the EA, an initial population of potential solutions (sets of detectors) is generated. Currently, two methods for hyperspherical detectors and one method for hyperrectangular detectors are implemented within the IFDOT tool. Phase I uses several representative existing approaches for detector generation. It is the objective of phase II to start from and improve upon these approaches.

The first method implemented for hyperspherical detector generation is a negative-selection algorithm with real representation and variable (NSA-RV) detector size [32]. The flowchart of this algorithm is presented in Fig. 5. 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:

1) The maximum allowed number of detectors is reached.

2) The maximum number of consecutively generated candidate detectors overlapping other detectors or self clusters is reached (shows likelihood of adequate coverage of the nonself).

3) The 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 hyperspherical detector generation is an enhanced NSA-RV, which integrates NSA-RV with detector moving and cloning [12]. 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 Fig. 6.

The method implemented for hyperrectangular detector generation is similar to the NSA-RV with the following differences. Each hyperrectangle 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 hypersphere 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 (minD) depends on the number of iterations the algorithm has performed and on a decay parameter τ, which is set by the user. The equation for the decay parameter is

$$\text{minD} = Rss_0 \times e^{-\text{iteration}/\tau} \tag{2}$$

where Rss_0 is the base minimum radius provided by the user. A flowchart for this function is provided in Fig. 7.

C. Phase 2: Optimization of Detectors

1. *Algorithm Layout*

Phase II of the EA is a classic genetic algorithm that uses sets of detectors generated in phase I as individuals in the population. Each detector may be considered as a gene within the chromosome. Several options are available for the detector representation as geometric hyperbodies. A three-criterion 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 [15,28]. This evolutionary search for the optimum solution continues for the specified number of generations. The flowchart of the EA is presented in Fig. 8.

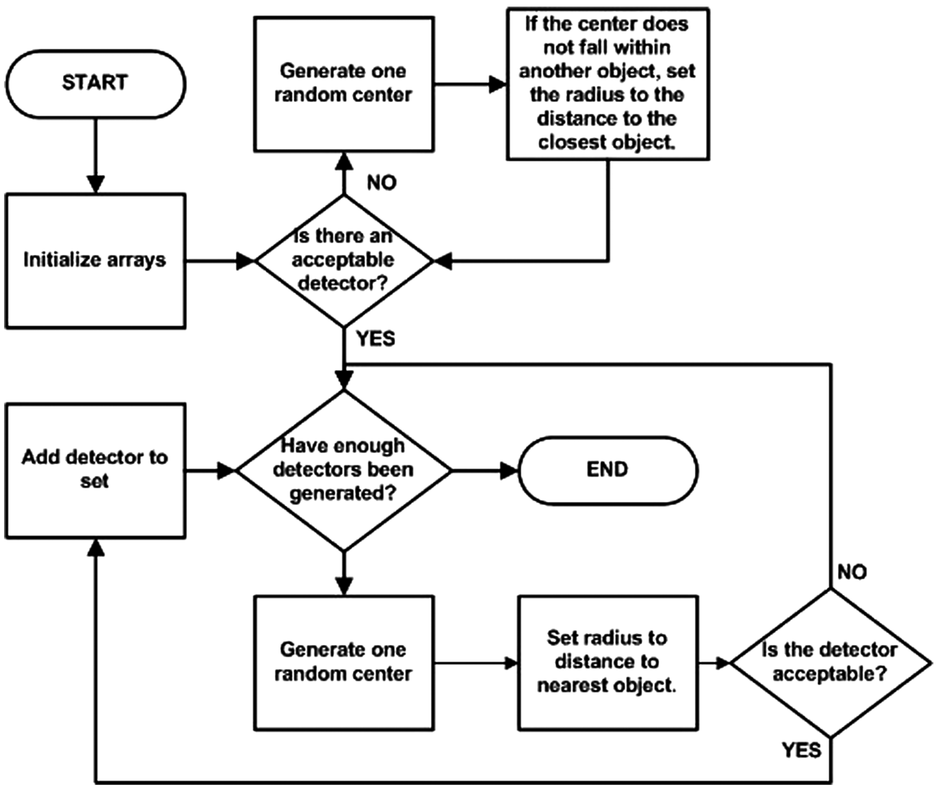

START
Initialize arrays
Generate one random center
If the center does not fall within another object, set the radius to the distance to the closest object.
Is there an acceptable detector?
NO
YES
Add detector to set
Have enough detectors been generated?
END
Generate one random center
Set radius to distance to nearest object.
Is the detector acceptable?
NO
YES

Fig. 5 Flowchart of detector generation using the NSA-RV.

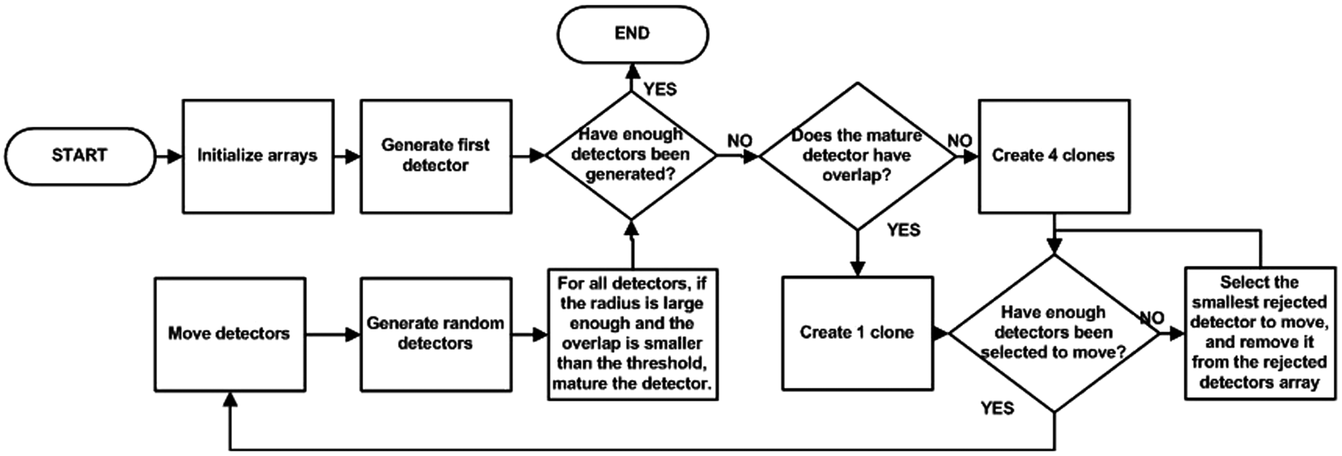

END
YES
START
Initialize arrays
Generate first detector
Have enough detectors been generated?
NO
Does the mature detector have overlap?
NO
Create 4 clones
YES
Move detectors
Generate random detectors
For all detectors, if the radius is large enough and the overlap is smaller than the threshold, mature the detector.
Create 1 clone
Have enough detectors been selected to move?
NO
Select the smallest rejected detector to move, and remove it from the rejected detectors array
YES

Fig. 6 Flowchart of detector generation using the enhanced NSA-RV.

2. *Representation of the Individual*

Each individual is a set of detectors. Because of 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. Hyperspherical detectors D_S are defined as

$$D_S = (c_s, r_s) \tag{3}$$

where $c_s \in \Re^n$ is the location of the center of the detector, $r_s \in \Re$ is the radius of the detector, and n is the dimension of the solution space. Hyperrectangular detectors D_R are defined as

$$D_R = (c_r, d_r) \tag{4}$$

where $c_r \in \Re^n$ is the location of the center of the detector, $d_s \in \Re^n$ is the side length of the detector in each dimension, and n is the dimension of the solution space. Hyperellipsoidal detectors are defined as

$$D_E = (c_e, a_e) \tag{5}$$

where $c_e \in \Re^n$ is the location of the center of the ellipse, $a_e \in \Re^n$ is the axes vector for all dimensions, and n is the dimension of the solution space. Finally, rotational hyperellipsoidal detectors are defined as

$$D_{\mathrm{RE}} = (c_{\mathrm{re}}, a_{\mathrm{re}}) \tag{6}$$

where $c_{\mathrm{re}} \in \Re^n$, is the location of the center of the detector, $a_{\mathrm{re}} \in \Re^2$, is the axes vector, and n is the dimension of the solution space. Note that, unlike hyperellipsoids, which may have different axes for each dimension, only one preferential axis may differ from the others.

3. *Fitness Function*

The fitness of the individuals is evaluated based on the following criteria: 1) number of detectors in the set, 2) percentage of nonself

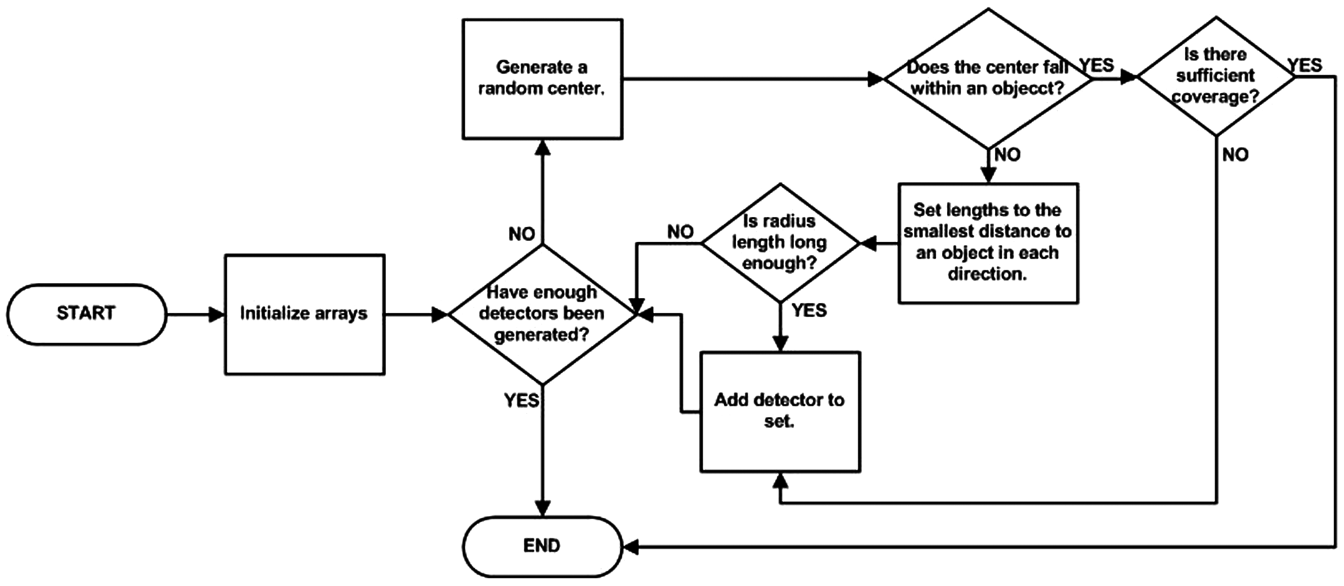

START
Initialize arrays
Have enough detectors been generated?
NO
YES
Generate a random center.
Does the center fall within an objecct?
YES
NO
Is there sufficient coverage?
YES
NO
Set lengths to the smallest distance to an object in each direction.
Is radius length long enough?
NO
YES
Add detector to set.
END

Fig. 7 Flowchart of detector generation for hyperrectangles.

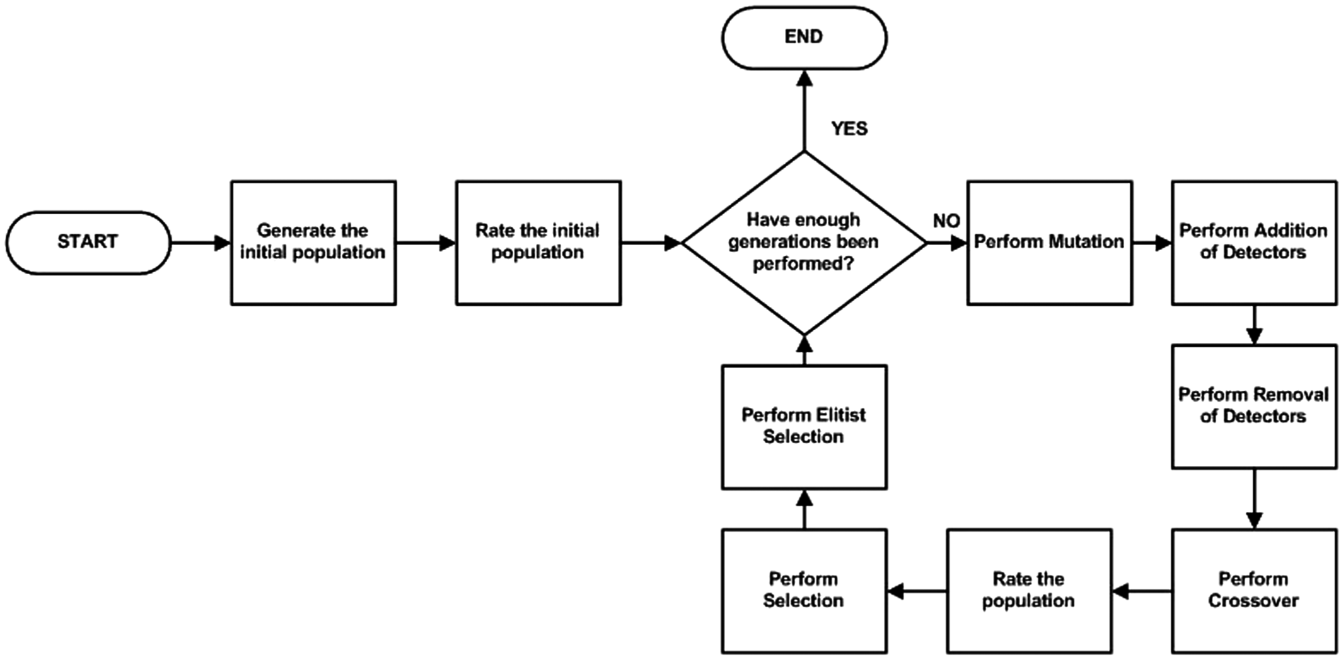

END
YES
START
Generate the initial population
Rate the initial population
Have enough generations been performed?
NO
Perform Mutation
Perform Addition of Detectors
Perform Removal of Detectors
Perform Crossover
Rate the population
Perform Selection
Perform Elitist Selection

Fig. 8 Flowchart of evolutionary algorithm.

that is covered by detectors, and 3) percentage of overlapping that occurs within the detector set.

Better individuals ideally have a small number of detectors, no overlap, and cover the entire nonself. 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 in Eqs. (7–9):

$$\mathrm{PI}_{\mathrm{coverage}i} = \frac{1}{U_{\mathrm{coverage}} - L_{\mathrm{coverage}}}\mathrm{coverage}_i - \frac{L_{\mathrm{coverage}}}{U_{\mathrm{coverage}} - L_{\mathrm{coverage}}} \tag{7}$$

where $\mathrm{PI}_{\mathrm{coverage}i}$ is the performance index of the individual i with respect to the coverage criterion, $\mathrm{coverage}_i$ is the coverage of the individual, L_{coverage} is the lowest acceptable coverage, and U_{coverage} is the highest expected coverage:

$$\mathrm{PI}_{\mathrm{overlapping}i} = \frac{1}{U_{\mathrm{overlapping}} - L_{\mathrm{overlapping}}}\mathrm{overlapping}_i + \frac{U_{\mathrm{overlapping}}}{U_{\mathrm{overlapping}} - L_{\mathrm{overlapping}}} + 1 \tag{8}$$

where $\mathrm{PI}_{\mathrm{overlapping}i}$ is the performance index of the individual i with respect to the overlap criterion, $\mathrm{overlapping}_i$ is the overlapping of the individual i, $L_{\mathrm{overlapping}}$ is the highest acceptable overlapping, and $U_{\mathrm{overlapping}}$ is the lowest expected overlapping:

$$\mathrm{PI}_{\mathrm{number}i} = \frac{1}{U_{\mathrm{number}} - L_{\mathrm{number}}}\mathrm{number}_i - \frac{U_{\mathrm{number}}}{U_{\mathrm{number}} - L_{\mathrm{number}}} + 1 \tag{9}$$

where $\mathrm{PI}_{\mathrm{number}i}$ is the performance index of the individual i with respect to the number of detectors criterion, number_i is the number of detectors in the individual i, L_{number} is the highest acceptable number of detectors, and U_{number} is the lowest expected number of detectors.

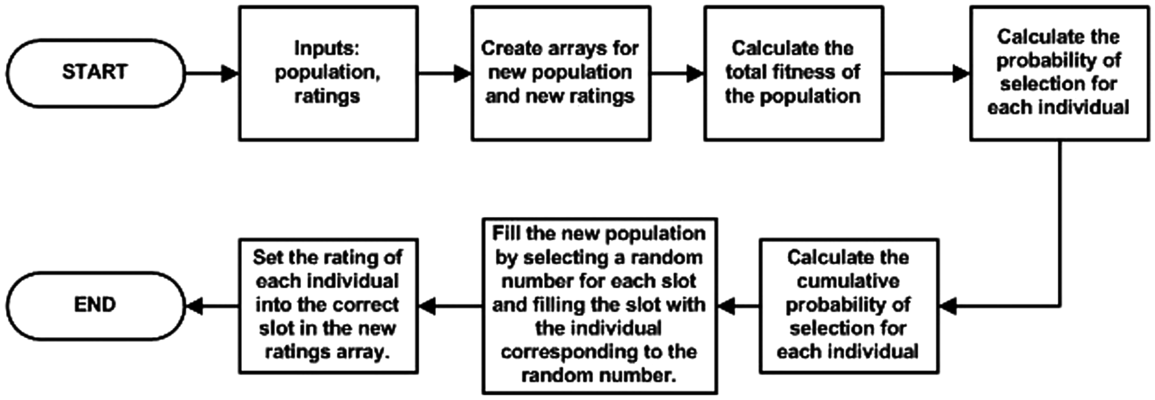

START
Inputs: population, ratings
Create arrays for new population and new ratings
Calculate the total fitness of the population
Calculate the probability of selection for each individual
Calculate the cumulative probability of selection for each individual
Fill the new population by selecting a random number for each slot and filling the slot with the individual corresponding to the random number.
Set the rating of each individual into the correct slot in the new ratings array.
END

Fig. 9 Flowchart of the roulette-wheel selection algorithm.

Each individual in the population has a performance index $\mathrm{PI}_{\mathrm{individual}}$ computed as

$$\mathrm{PI}_{\mathrm{individual}i} = W_{\mathrm{overlap}} \times \mathrm{PI}_{\mathrm{overlap}} + W_{\mathrm{number}} \times \mathrm{PI}_{\mathrm{number}} + W_{\mathrm{coverage}} \times \mathrm{PI}_{\mathrm{coverage}} \tag{10}$$

where W_{overlap}, W_{number}, and W_{coverage} are relative weights for the importance of each of the criteria.

A small number of detectors implies reduced computational requirements. It will also require larger size of the detectors, which is acceptable for general detection when only good coverage of the nonself 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 nonself 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. Roulette-wheel selection [28] is the method used in this application. Each individual is assigned a performance index between 0 and 1, using Eqs. (7–10). The total-performance-index TF is the sum of all of the performance indices for all individuals in the population:

$$\mathrm{TF} = \Sigma_{i=1}^{N} \mathrm{PI}_i \tag{11}$$

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:

$$p_i = \frac{\mathrm{PI}_{\mathrm{individual}}}{\mathrm{TF}} \tag{12}$$

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

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

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 Fig. 9.

5. *Variation Operators*

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

The *mutation operator* 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 this operator is displayed in Fig. 10. An illustration of the mutation operator is shown in Fig. 11. After mutation, one detector has been moved and another has been resized.

The initial population as generated in phase I can only include hyperspherical detectors. The mutation operator is used to alter the shape of the detectors to create ellipsoids or rotational ellipsoids.

If the mutation causes the detector to overlap the self clusters, the mutation is not performed ensuring that, at all times, zero overlapping with the self is maintained.

The *gene addition* operator, a function that performs a special type of mutation, 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

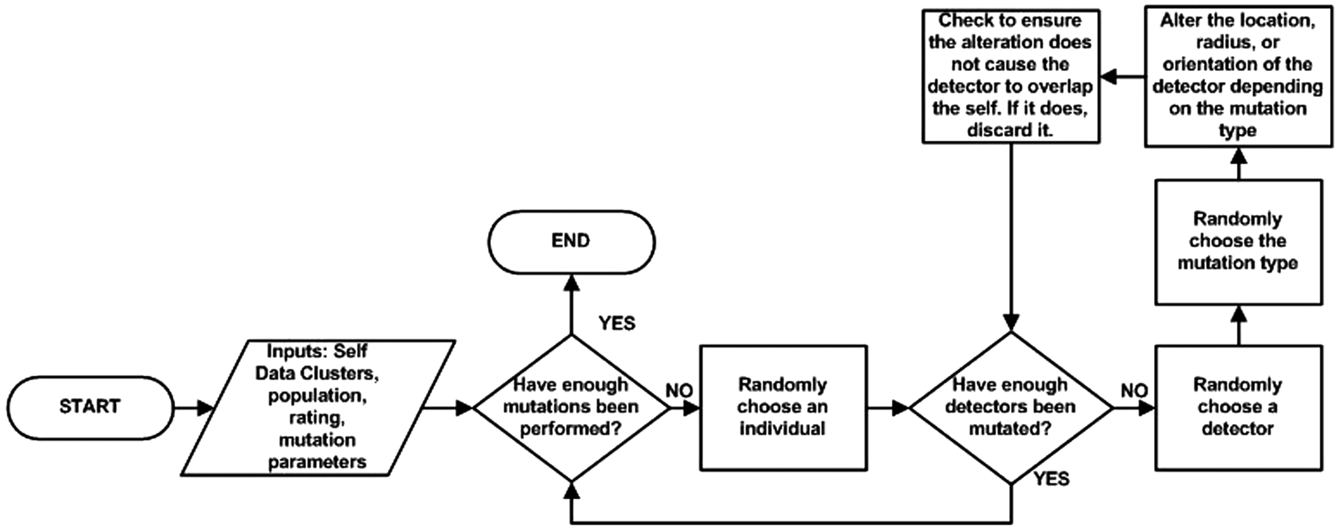

START
Inputs: Self Data Clusters, population, rating, mutation parameters
Have enough mutations been performed?
YES
END
NO
Randomly choose an individual
Have enough detectors been mutated?
NO
YES
Randomly choose a detector
Randomly choose the mutation type
Alter the location, radius, or orientation of the detector depending on the mutation type
Check to ensure the alteration does not cause the detector to overlap the self. If it does, discard it.

Fig. 10 Flowchart of the mutation genetic operator.

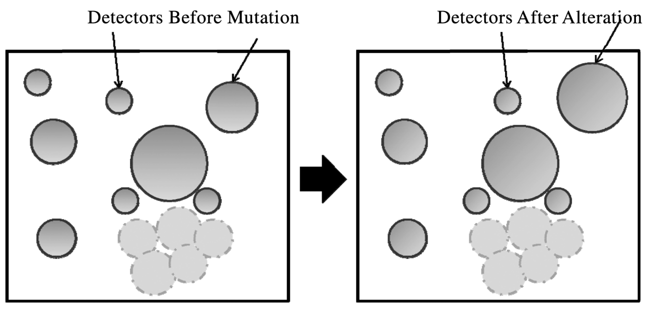


Fig. 11 Illustration of the mutation operator.

performance index of the individual. This process is repeated until the addition rate is satisfied for each pass of the operator. A flowchart of the addition operator is shown in Fig. 12. An illustration of this operator is presented in Fig. 13. Notice that after the operator has been applied, new detectors were included that do not overlap any of the previously existing detectors or the self.

Two variations exist for this operator. 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 operator 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 the scheme to distinguish between failures in the same category.

The *gene removal* operator, as a different type of mutation, 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 describing the logic of the remove operator is presented in Fig. 14. An illustration of this simple genetic operator is shown in Fig. 15, in which a detector, which exhibited a high amount of overlapping, was removed. This operator must be used with caution. Removing detectors may significantly decrease the coverage, as seen in the illustration.

To apply the *crossover operator*, 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 nonself. 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 operator is applied. Therefore, nonoverlapping with the self, following this operator, is guaranteed. The flowchart of the crossover operator is presented in Fig. 16. Figure 17 shows the two individuals before and after crossover has been performed. Notice that the detectors in the top right of the individuals have been exchanged between the two individuals.

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

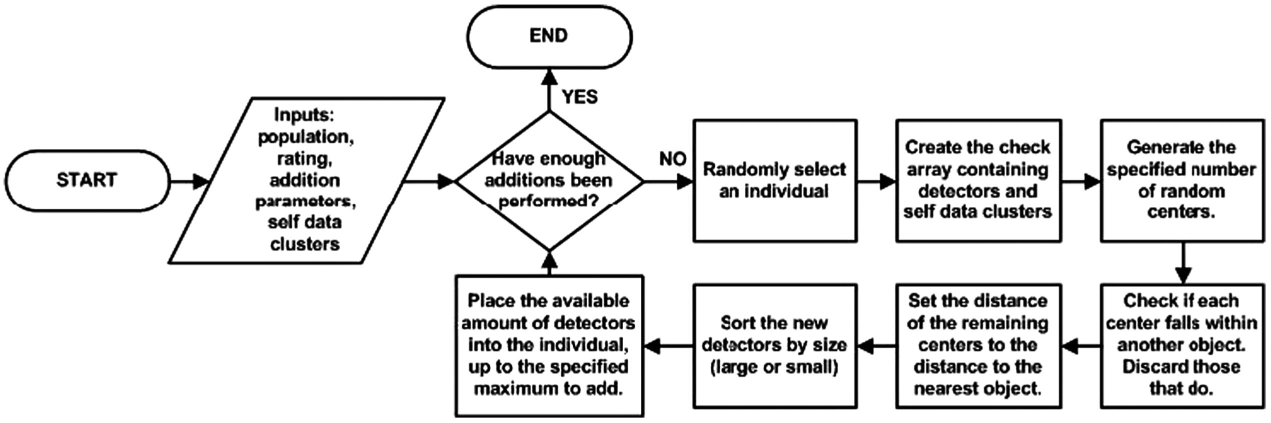

START
Inputs: population, rating, addition parameters, self data clusters
Have enough additions been performed?
YES
END
NO
Randomly select an individual
Create the check array containing detectors and self data clusters
Generate the specified number of random centers.
Check if each center falls within another object. Discard those that do.
Set the distance of the remaining centers to the distance to the nearest object.
Sort the new detectors by size (large or small)
Place the available amount of detectors into the individual, up to the specified maximum to add.

Fig. 12 Flowchart of the addition genetic operator.

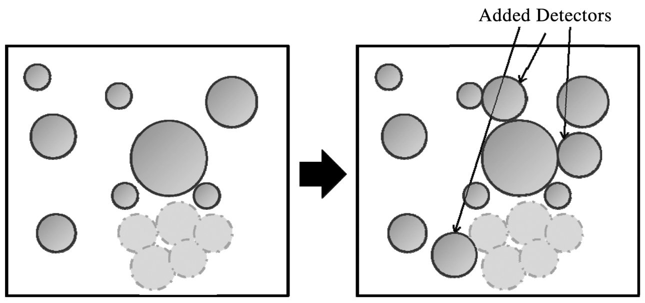


Fig. 13 Illustration of the gene addition operator.

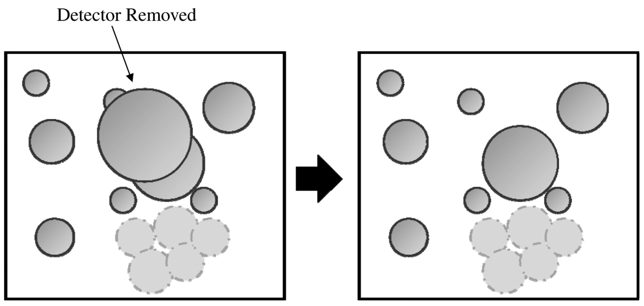


Fig. 15 Illustration of detector gene removal operator.

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 is developed in MATLAB/Simulink and 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 algorithm can be optionally run multi-threaded, in which case the Parallel Computing Toolbox is needed.

The main menu presented in Fig. 18 is the portal to the design environment. At any point, the user can return here to repeat certain operations or redirect the design sequence. The File drop-down menu allows the selection of one of the following design tasks: 1) normalization of flight data, 2) elimination of duplicates within data files, 3) clustering of flight data to produce a self with two shape options, 4) generation of positive and negative selection-detection schemes with two shape options, 5) optimization of negative selection-detection schemes with four shape options, 6) results reporting, 7) file merging, and 8) 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 setup 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 before the use of the IFDOT tool. This raw data file should contain only the time histories of the identifiers to be used for FDGO.

B. Preparing Data

Several parameters are available for processing data. These are as follows: 1) normalization limit, 2) normalization grace margin (only for applicable method), and 3) 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 are 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: hyperspheres or hyperrectangles. 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 spherical *number-imposed clustering*.

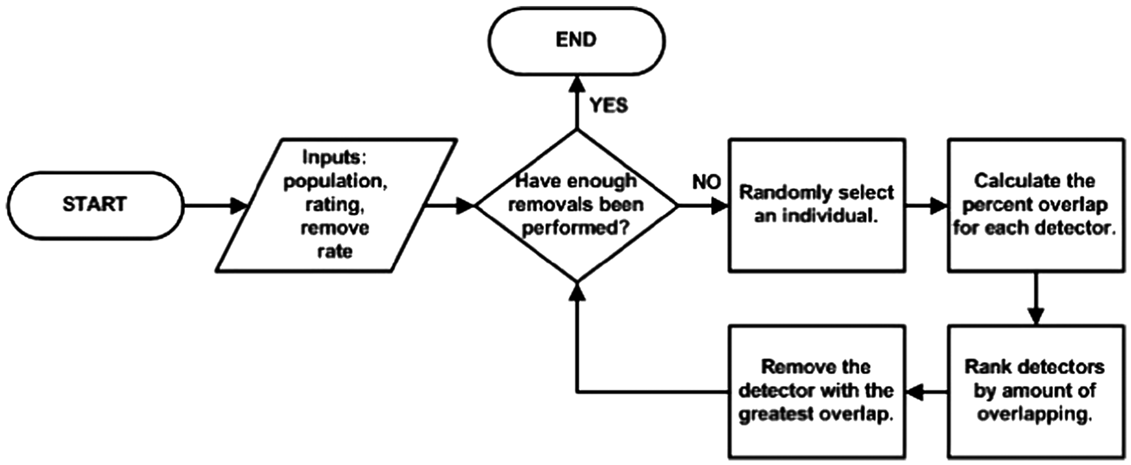

END
YES
START
Inputs: population, rating, remove rate
Have enough removals been performed?
NO
Randomly select an individual.
Calculate the percent overlap for each detector.
Rank detectors by amount of overlapping.
Remove the detector with the greatest overlap.

Fig. 14 Flowchart of the remove genetic operator.

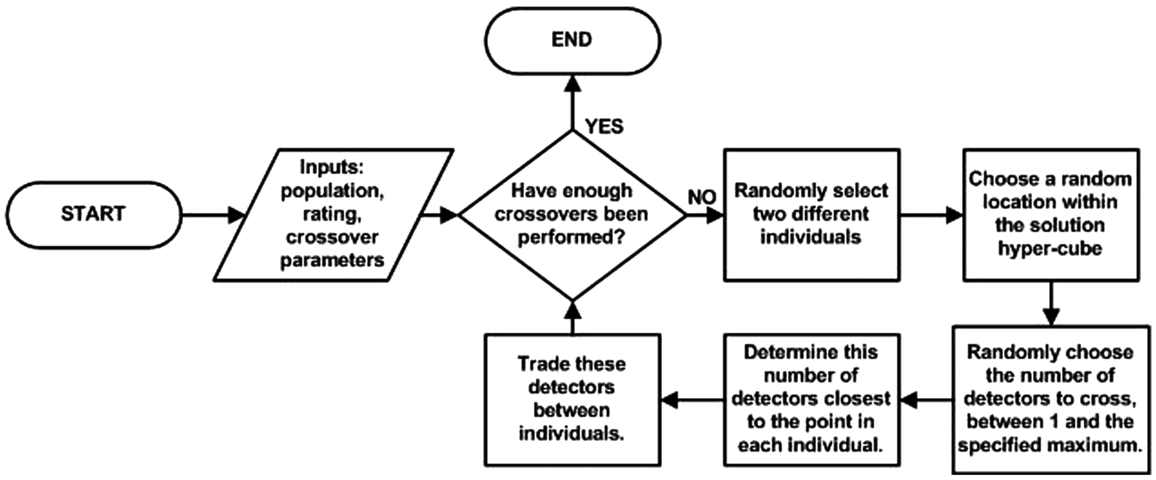

END
YES
START
Inputs: population, rating, crossover parameters
Have enough crossovers been performed?
NO
Randomly select two different individuals
Choose a random location within the solution hyper-cube
Randomly choose the number of detectors to cross, between 1 and the specified maximum.
Determine this number of detectors closest to the point in each individual.
Trade these detectors between individuals.

Fig. 16 Flowchart for the crossover genetic operator.

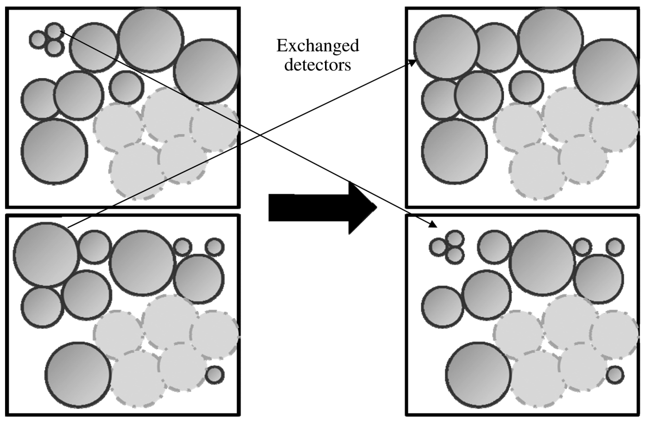


Fig. 17 Illustration of crossover genetic operator.

For the spherical number-imposed clustering method, the following parameters need to be specified:

1) The desired number of clusters specifies the maximum number of clusters that may be used.

2) The minimum cluster radius ensures that no cluster is too small.

3) The confidence percentage is the stopping criterion for the Monte Carlo method [33] for overlapping and coverage calculation.

4) The permitted error is the 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:

1) The initial number of clusters is the starting number of clusters.

2) The cluster increase step is the additional number of clusters to be investigated at each iteration.

3) The maximum number of clusters is the allowable number of clusters in the set.

4) The point radius is the confidence radius around each data point that is assumed to belong to the self.

5) The acceptable empty percentage is the allowable amount of empty space a cluster may contain.

6) The confidence percentage is the stopping criterion for the Monte Carlo method.

7) The permitted error is stopping criterion for the Monte Carlo method.

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

1) The desired number of clusters is the maximum number of clusters that can be used in the set.

2) The minimum cluster dimension specifies the smallest allowable side length for the cluster.

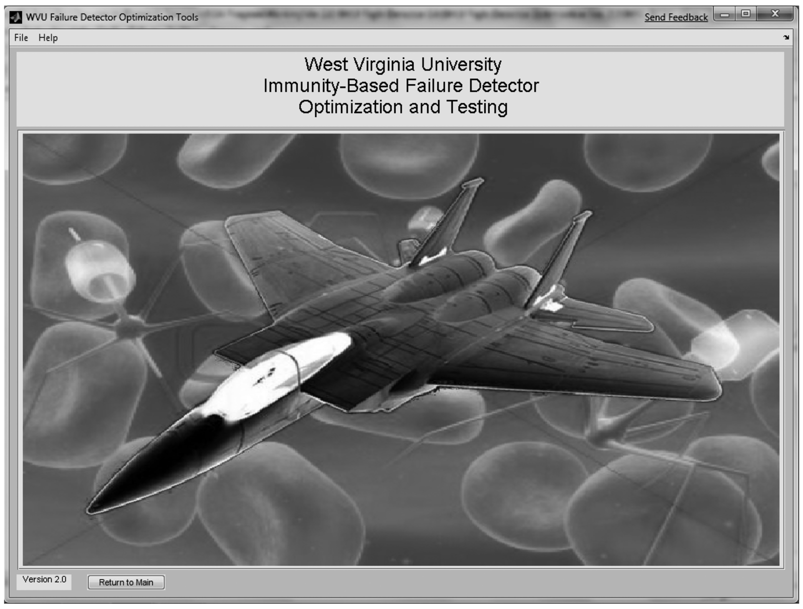


Fig. 18 Main menu of the WVU IFDOT design environment.

3) The confidence percentage is as above.
4) The permitted error is as above.

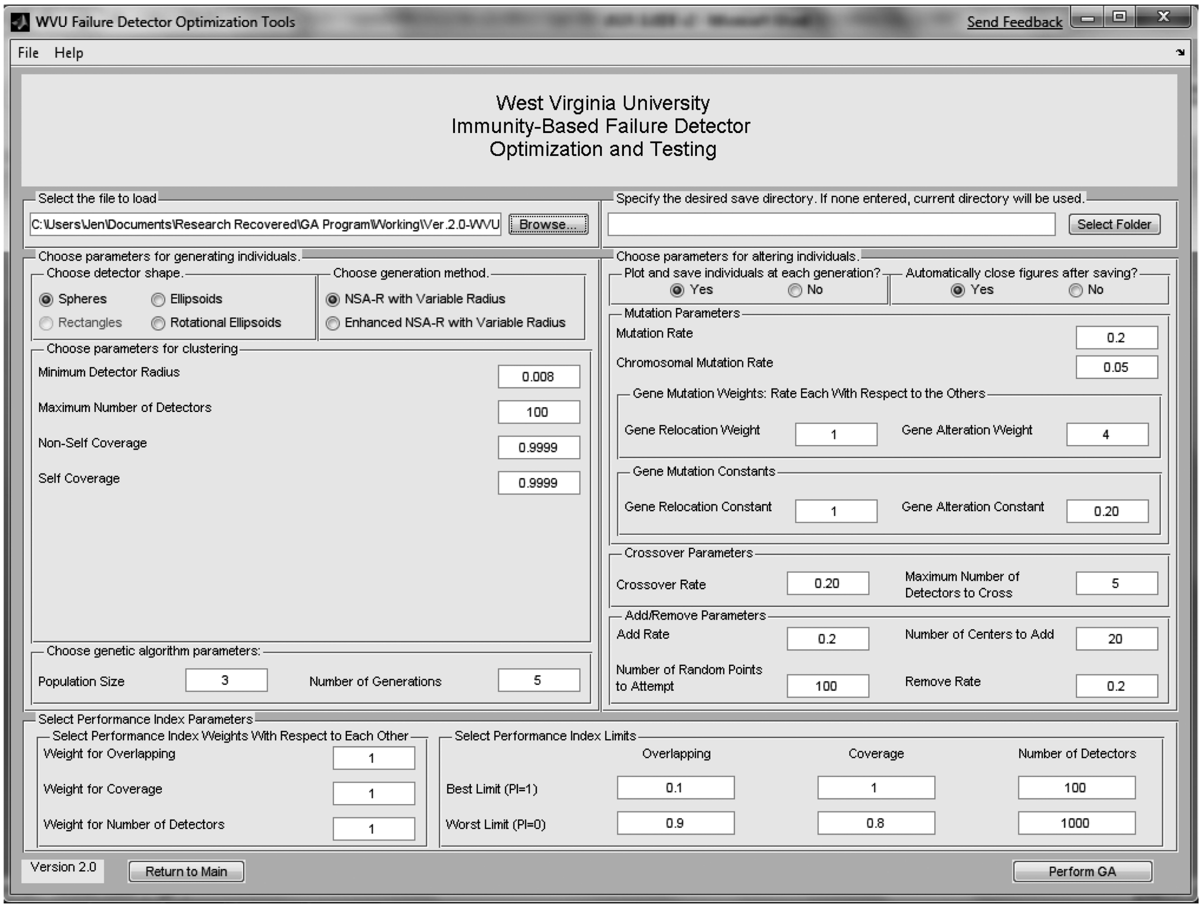
All clustering methods will return the final number of clusters, their overlap, and their coverage of the total hypercube (solution space). These values are useful to the user when determining the parameters for the genetic algorithm.

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 in Fig. 19. 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, then 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 in Fig. 20. In addition, if enhanced NSA-RV is used, the detector-generation parameter-selection menu will be the one presented in Fig. 21.

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.



West Virginia University
Immunity-Based Failure Detector
Optimization and Testing

Select the file to load: C:\Users\Jen\Documents\Research Recovered\GA Program\Working\Ver.2.0-WVU

Specify the desired save directory. If none entered, current directory will be used.

Choose parameters for generating individuals.

Choose detector shape: [✓ Spheres] Ellipsoids Rectangles Rotational Ellipsoids

Choose generation method: [✓ NSA-R with Variable Radius] Enhanced NSA-R with Variable Radius

Choose parameters for clustering

Minimum Detector Radius	0.008
Maximum Number of Detectors	100
Non-Self Coverage	0.9999
Self Coverage	0.9999

Choose genetic algorithm parameters:

Population Size	3	Number of Generations	5

Choose parameters for altering individuals.

Plot and save individuals at each generation? [✓ Yes] No

Automatically close figures after saving? [✓ Yes] No

Mutation Parameters

Mutation Rate	0.2
Chromosomal Mutation Rate	0.05

Gene Mutation Weights: Rate Each With Respect to the Others

Gene Relocation Weight	1	Gene Alteration Weight	4

Gene Mutation Constants

Gene Relocation Constant	1	Gene Alteration Constant	0.20

Crossover Parameters

Crossover Rate	0.20	Maximum Number of Detectors to Cross	5

Add/Remove Parameters

Add Rate	0.2	Number of Centers to Add	20
Number of Random Points to Attempt	100	Remove Rate	0.2

Select Performance Index Parameters

Select Performance Index Weights With Respect to Each Other

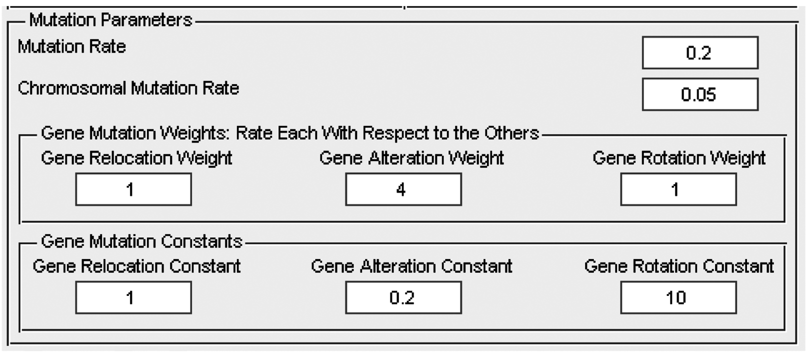
Weight for Overlapping	1
Weight for Coverage	1
Weight for Number of Detectors	1

Select Performance Index Limits

	Overlapping	Coverage	Number of Detectors
Best Limit (PI=1)	0.1	1	100
Worst Limit (PI=0)	0.9	0.8	1000

Version 2.0

Fig. 19 Detector-optimization main menu.



Mutation Parameters

Mutation Rate	0.2
Chromosomal Mutation Rate	0.05

Gene Mutation Weights: Rate Each With Respect to the Others

Gene Relocation Weight	Gene Alteration Weight	Gene Rotation Weight
1	4	1

Gene Mutation Constants

Gene Relocation Constant	Gene Alteration Constant	Gene Rotation Constant
1	0.2	10

Fig. 20 Ellipsoidal mutation parameters.

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

1) The maximum number of detectors is the maximum allowed in any individual.

2) The minimum detector radius is the smallest allowable detector size.

3) The maximum self coverage is the stopping criterion that occurs when enough detectors that are not of an acceptable size are generated.

4) The maximum nonself coverage is the stopping criterion that occurs after too many random centers are rejected for falling in the existing self clusters or nonself detectors.

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

1) The minimum detector radius is the smallest allowable detector size.

2) The initial number of detectors is the starting point of the algorithm.

3) The maximum number of detectors limits the algorithm and the size of the individual.

4) The maximum number of iterations is the stopping criterion for the detector-generation algorithm.

5) The number of random detectors in each iteration defines the number of random detectors that will be added each iteration.

6) The number of detectors to move each iteration is the number of rejected detectors that will be moved.

7) The initial adaptation rate determines the distance detectors will be moved.

8) The decay parameter decreases the distance detectors are moved each iteration.

9) The permitted overlapping threshold is the amount of overlapping a detector can have and become matured.

10) The number of nearest points considered in cloning determines how many clones will be produced at each iteration.

11) The number of nearest points considered in moving determines how many detectors will be moved in each iteration.

l2) The initial adaptation distance used to locate new clones determines the distance away that clones will be generated.

13) The cloning decay parameter reduces at each iteration the distance away clones will be generated.

The parameters for generating rectangular detectors are as follows:

1) The maximum number of detectors is the size limit of each individual.

2) The initial minimum detector dimension is the smallest acceptable detector size.

3) The decay parameter decreases the smallest acceptable detector size at each iteration.

4) The expected coverage is the 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:

1) The coverage weight determines the relative importance of coverage.

2) The overlapping weight determines the relative importance of overlapping.

3) The number of detectors weight determines the relative importance of the number of detectors.

4) Coverage upper bound is the best expected coverage.

5) Coverage lower bound is the worst expected coverage.

6) The overlapping upper bound is the lowest expected overlapping.

7) The overlapping lower bound is the largest expected overlapping.

8) The number upper bound is the lowest expected number of detectors.

9) The number lower bound is the 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 as follows:

1) The crossover rate is the percentage of individuals selected for this GO.

2) The maximum detectors to cross is the maximum number of detectors to undergo this GO.

The parameters defining the add/remove GOs consist of the following:

1) The add rate is the percentage of individuals selected for this GO.

2) The remove rate is the percentage of individuals selected for this GO.

3) The number of centers to attempt is the number of centers generated by the algorithm.

4) The maximum number of detectors to add is the largest number of detectors to be added to an individual at one time.

The genetic algorithm properties are as follows:

1) Population size is the number of individuals.

2) The number of generations is the stopping criterion for the FDGO algorithm.

Mutation parameters depend on the shape of the detectors. For spheres, the mutation parameters will be as follows:

1) The mutation rate is the percentage of individuals selected for this GO.

2) The chromosomal mutation rate is the number of detectors in a selected individual that will be mutated.

3) The gene-alteration constant defines the maximum amount by which the radius can be altered.

4) The gene-relocation constant determines the maximum distance a center can be moved.

5) The gene-alteration weight determines the probability of gene alteration.

6) The gene-relocation weight determines the probability of gene relocation.

For ellipsoids, the following parameters are requested:

1) The mutation rate is the percentage of individuals selected for this GO.

2) The chromosomal mutation rate is the number of detectors in a selected individual that will be mutated.

3) The gene-alteration constant defines the maximum amount by which the radius can be altered.

4) The gene-relocation constant determines the maximum distance a center can be moved.

5) The gene-rotation constant is the maximum number of degrees a detector can be rotated.

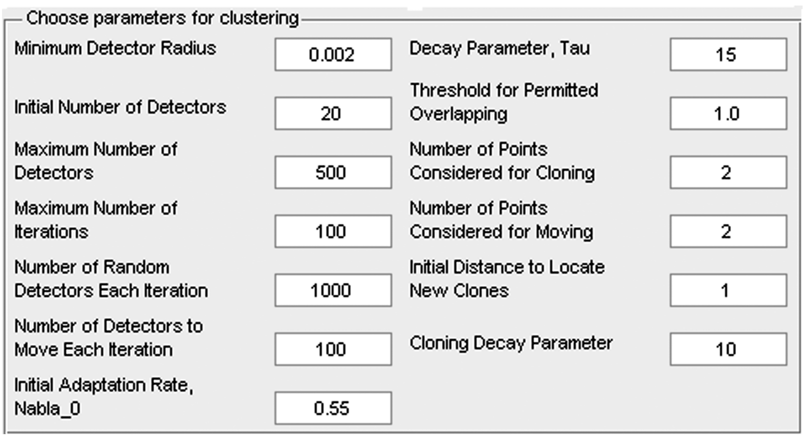
6) The gene-alteration weight determines the probability of gene alteration.

7) The gene-relocation weight determines the probability of gene relocation.

8) The 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 two dimensions and six 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 three-dimensional solution space (using the parameters of 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. This second set of results is based upon data collected from the WVU


Choose parameters for clustering

Minimum Detector Radius	0.002	Decay Parameter, Tau	15
Initial Number of Detectors	20	Threshold for Permitted Overlapping	1.0
Maximum Number of Detectors	500	Number of Points Considered for Cloning	2
Maximum Number of Iterations	100	Number of Points Considered for Moving	2
Number of Random Detectors Each Iteration	1000	Initial Distance to Locate New Clones	1
Number of Detectors to Move Each Iteration	100	Cloning Decay Parameter	10
Initial Adaptation Rate, Nabla_0	0.55		

Fig. 21 Enhanced detector-generation parameters.

F-15 flight simulation [34], performed using the WVU Motus six-degree-of-freedom flight simulator. The air vehicle model implemented emulates the NASA Intelligent Flight Control System F-15 research aircraft. Lookup tables covering large regions of the flight envelope are used to compute aerodynamic and propulsion forces and moments. The lookup tables are divided to represent the aerodynamic contributions from individual control surfaces (including canards and dual rudder) to accommodate the modeling of actuator failures. Direct adaptive fault-tolerant control laws [34] are included, generating commands for the differential canard, collective and differential stabilator, differential aileron, and rudder.

A. Two-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. The number of individuals in the population, the number of detectors for each individual, and the number of generations have been limited to keep the computational time reasonable, considering the fact that the purpose of this example is to demonstrate the functionality of the algorithm and not to achieve high detection performance. A 30% mutation rate, crossover rate, and add rate were 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 in Fig. 22. This individual contains 54 detectors, due to multiplication. It has a coverage of 63.5% and an overlap of 100%.

Figure 23 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 that 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.

At the end of 50 generations, the best individual improved to the solution is shown in Fig. 24. 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 Fig. 25.

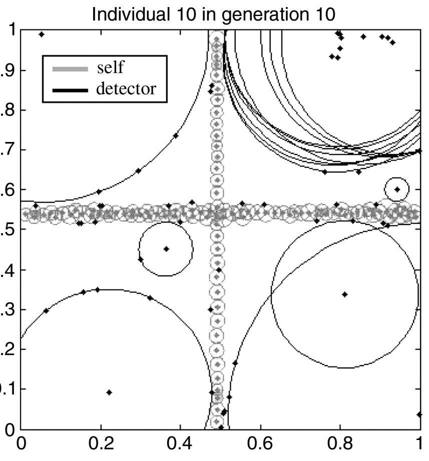


Fig. 23 Intermediate individual.

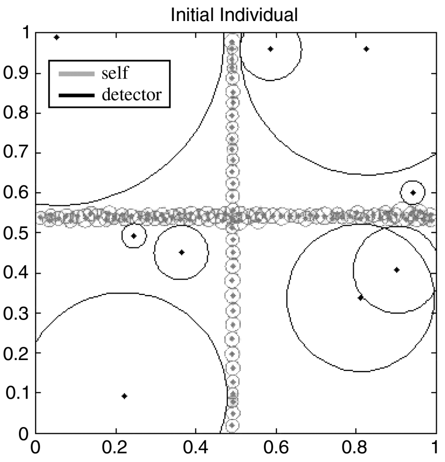


Fig. 22 Initial individual.

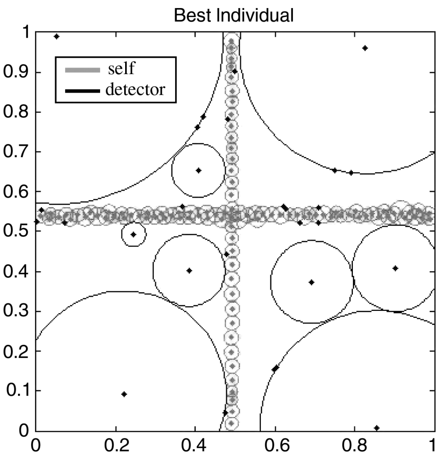


Fig. 24 Final best individual.

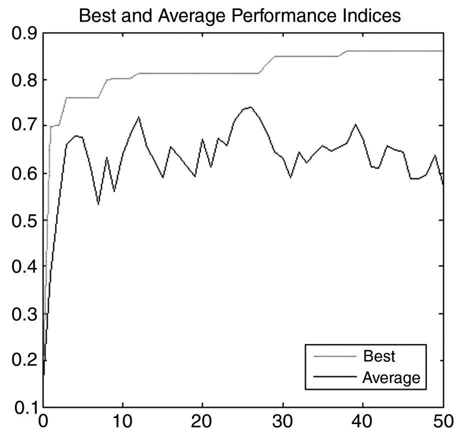


Fig. 25 Performance indices for the 2-D example.

B. Six-Dimensional Example

The purpose of this next example is to illustrate the operation of the IFDOT tool in higher dimensions. A six-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 in Fig. 26. This plot suggests that an increasing larger number of generations will be required to reach values of the performance similar to those 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. Therefore, the analysis is focused on the relative performance with versus without optimization and not on the absolute performance of the detection scheme. In this example, a three-dimensional self is considered, consisting of neural-network outputs for roll, pitch, and yaw rates (NNp, NN$_q$, and NN$_r$). The neural networks are part of a fault-tolerant controller [34], and their outputs in terms of compensatory angular accelerations have been shown to possess good detection capabilities [35]. 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 deg. 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.

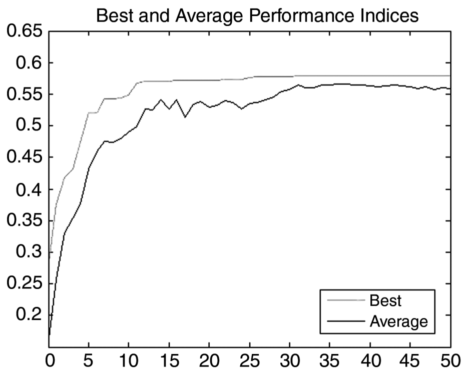


Fig. 26 Performance indices for the 6-D example.

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 27.41%. 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 nonself, as 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. This can be seen in the instance that the detection rate remains only around 50% for all rudder failure cases, whereas the stabilator and aileron failures are detected nearly 100% of the time by the better-covered sets (second and third in Table 1). This set of identifiers seems to be inadequate for capturing the signature of the rudder failure. However, it is possible that increasing the coverage even more could improve the detection of the rudder failure. 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 [35]. In any case, the number of detectors may become critical to the performance of the FDIE scheme. In the case of the small phase I set, only 4.74% detection rate is achieved for the right-aileron failure achieves, whereas for the left-aileron failure, 90.65% detection rate is achieved. This shows that although the

Table 1 Rates of detection before and after optimization of detector set

Resulting detector-set characteristics			Actuator failure 8 deg						Sensor failure						Structural failure 35%	
			Stabilator		Aileron		Rudder		LSB			LFDB				
Antibodies	Coverage, %	Overlap, %	L	R	L	R	L	R	p	q	r	p	q	r	L	R
Phase I detector-set generation method																
248	78.71	27.41	99.6	98.05	90.65	4.74	54.45	47.94	10.29	6.61	51.51	3.66	15.07	75.31	53.36	92.51
Phase I detector-set generation method																
495	89.35	49.81	99.97	98.02	99.96	99.51	55.62	49.72	17.88	7.13	75.39	6.29	17.04	83.49	99.63	99. 98
Phase II detector-set generation method																
298	89.19	58.80	99.97	99.80	99.95	95.74	57.76	50.16	12.18	9.74	62.91	6.81	20.05	82.96	99.06	99.98

identifiers have the ability to capture this failure, there exists a lack of coverage in the region of the nonself space where the right-aileron failure occurs.

VIII. Conclusions

An interactive design tool for artificial immune system detector generation and optimization has been developed in MATLAB and tested. The user-friendly graphical interface and the substantial library of alternative algorithms allow maximum flexibility and effectiveness in the design.

The proposed two-phase evolutionary algorithm optimizes the set of detectors for coverage of the nonself, number of detectors, and overlapping, while maintaining no overlapping with the self. All relevant optimization criteria are thus addressed.

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. The design tool, when properly applied, is capable of producing sets of detectors that are more effective than those produced using existing detector-generation methods, for an artificial immune system.

Although the examples presented in this paper address aircraft failure detection, detector sets could be produced using this utility for any system, provided there are adequate data available to properly define the self.

Acknowledgments

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

[1] KrishnaKumar, K., Viken, S., and Nguyen, N., "Stability, Maneuverability, and Safe Landing in the Presence of Adverse Conditions," NASA Aeronautics Research Mission Directorate, 2009; available online at http://www.aeronautics.nasa.gov/nra_pdf/irac_-tech_plan_c1.pdf [retrieved Nov. 2009].

[2] Srivastava, A. N., Mah, R. W., and Meyer, C., "Automated Detection, Diagnosis, Prognosis to Enable Mitigation of Adverse Events During Flight," NASA Aeronautics Research Mission Directorate, 2008; available online at http://www.aeronautics.nasa.gov/nra_pdf/ivhm_-tech_plan_c1.pdf [retrieved Nov. 2009].

[3] Young, S. D., and Quon, L., "Integrated Intelligent Flight Deck," NASA Aeronautics Research Mission Directorate, 2007; available online at http://www.aeronautics.nasa.gov/nra_pdf/iifd_tech_plan_c1.pdf [retrieved Nov. 2009].

[4] Young, R., and Rohn, D., "Aircraft Aging and Durability Project," NASA Aeronautics Research Mission Directorate, 2007; available online at http://www.aeronautics.nasa.gov/nra_pdf/aad_tech_plan_c1.pdf [retrieved Nov. 2009].

[5] Jones S. M., and Reveley M., "An Overview of the NASA Aviation Safety Program Assessment Process," 3rd AIAA Aviation Technology, Integration, and Operations (ATIO), AIAA Paper 2003-6706, Denver, CO, 17–19 Nov. 2003.

[6] Marcos, A., Ganguli, S., and Balas, G., "Application of Fault Detection and Isolation to a Boeing 747-100/200 Aircraft," AIAA Guidance, Navigation, and Control Conference, AIAA Paper 02-4944, Monterey, CA, Aug. 2002.

[7] Shin, J. Y., Wu, N. E., and Belcastro, C., "Linear Parameter Varying Control Synthesis for Actuator Failure, Based on Estimated Parameter," AIAA Guidance, Navigation, and Control Conference, AIAA Paper 02-4546, Monterey, CA, Aug. 2002.

[8] Narendra, K. S., and Balakrishnan, J., "Adaptive Control Using Multiple Models," *IEEE Transactions on Automatic Control*, Vol. 42, No. 2, 1997, pp. 171–187. doi:10.1109/9.554398

[9] Lou, S. J., Budman, H., and Duever, T. A., "Comparison of Fault Detection Techniques: Problem and Solution," *Proceedings of the American Control Conference*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2002, pp. 4513–4518.

[10] Napolitano, M. R., Younghawn, A., and Seanor, B., "A Fault-Tolerant Flight Control System for Sensor and Actuator Failures Using Neural Networks," *Aircraft Design*, Vol. 3, No. 2, 2000, pp. 103–128. doi:10.1016/S1369-8869(00)00009-4

[11] KrishnaKumar, K., "Artificial Immune System Approaches for Aerospace Applications," 41st Aerospace Sciences Meeting and Exhibit, AIAA Paper 2003-0457, Reno, NV, 2003.

[12] Dasgupta, D., KrishnaKumar, K., Wong, D., and Berry, M., "Negative Selection Algorithm for Aircraft Fault Detection," *Proceedings from the 3rd International Conference on Artificial Immune Systems 2004*, Springer, Berlin, Sept. 2004, pp. 1–13.

[13] Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R., "Self Nonself Discrimination in a Computer," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 202–212.

[14] Holland, J. H., "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence," Univ. of Michigan Press, Ann Arbor, MI, 1975; rev. 1992.

[15] Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991, pp. 1–22.

[16] Perhinschi, M. G., Moncayo, H., and Davis, J., "Integrated Framework for Aircraft Sub-System Failure Detection, Identification, and Evaluation Based on the Artificial Immune System Paradigm," AIAA Guidance, Navigation, and Control Conference, AIAA Paper 2009-6261, Chicago, IL, Aug. 2009.

[17] Coico, R., Sunshine, G., and Benjamini, E., *Immunology, A Short Course*, 5th ed., Wiley-Liss, New York, 2003, pp. 1–10.

[18] Farmer, J., Norman, H., Packard, S., and Perelson, A. S., "The Immune System, Adaptation, and Machine Learning," *Physica D*, Vol. 22, No. 1–3, North-Holland, Amsterdam, 1986, pp. 187–204. doi:10.1016/0167-2789(86)90240-X

[19] Dasgupta, D., and Attoh-Okine, N., "Immunity-Based Systems: A Survey," *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, Oct. 1997, pp. 396–374.

[20] Dasgupta, D., and Forrest, S., "Artificial Immune Systems in Industrial Applications," *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, July 1999, pp. 257–267.

[21] Zhi-Tang, L., Yao, L., and Li, W., "A Novel Fuzzy Anomaly Detection Algorithm Based on Artificial Immune System," *Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05), 2005*, IEEE Computer Society, Washington, D.C., July 2005, pp. 481–486.

[22] De Castro, L., and Timmis, J., "Artificial Immune Systems: A Novel Paradigm to Pattern Recognition", *Artificial Neural Networks in Pattern Recognition*, edited by J. M. Corchado, L. Alonso, and C. Fyfe, Univ. of Paisley, Paisley, Scotland, U.K., 2002, pp. 67–84.

[23] Dasgupta, D., and Majumdar, N., "Anomaly Detection in Multidimensional Data Using Negative Selection Algorithm," *Proceedings of the Congress on Evolutionary Computation CEC '02*, Vol. 02, IEEE Computer Society, Washington, D.C., 2002, pp. 1039–1044.

[24] Kim, J., Greensmith, J., Twycross, J., and Aickelin, U., "Malicious Code Execution Detection and Response Immune System Inspired by the Danger Theory," *Adaptive and Resilient Computing Security Workshop (ARCS-05)*, 2005.

[25] Ko, A., Lau, H., and Lau, T., "An Immuno Control Framework for Decentralized Mechatronic Control," *Proceedings of the Third International Conference on Artificial Immune Systems*, Springer, Berlin, Sept. 2004, pp. 91–105.

[26] Verleysen, M., and François, D., "The Curse of Dimensionality in Data Mining and Time Series Prediction," *Computational Intelligence and Bioinspired Systems (IWNAN 2005)*, Lecture Notes in Computer Science, Vol. 3512, Springer–Verlag Berlin, 2005, pp. 758–770.

[27] Goldberg, D. E., "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison Wesley, Reading, MA, 1989, pp. 1–24.

[28] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer–Verlag, Berlin, 1992, pp. 13–42.

[29] KrishnaKumar, K., and Goldberg, D. E., "Control System Optimization Using Genetic Algorithms," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 3, 1992, pp. 735–740. doi:10.2514/3.20898

[30] Elkan, C., "Using the Triangle Inequality to Accelerate *k*-Means," *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, AAAI Press, Menlo Park, CA, 2003, pp. 147–153.

[31] Perhinschi, M. G., and Moncayo, H., "Integrated System for Immunity-Based Failure Detection, Identification, and Evaluation," NASA Langley Research Center, Hampton, VA, Nov. 2008.

[32] Ji, Z., and Dasgupta, D., "Augmented Negative Selection Algorithm with Variable-Coverage Detectors," *Proceedings from the Conference of Genetic and Evolutionary Computation, 2004 (CEC2004)*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2004, pp. 1081–1088.

[33] *Introduction to Monte Carlo Methods* [online book], Oak Ridge National Lab., Oak Ridge, TN, 1995, http://www.ipp.mpg.de/~rfs/comas/Helsinki/helsinki04/CompScience/csep/csep1.phy.ornl.gov/mc/mc.html [retrieved Nov. 2009].

[34] Perhinschi, M. G., Napolitano, M. R., and Campa, G., "A Simulation Environment for Design and Testing of Aircraft Adaptive Fault-Tolerant Control Systems," *Aircraft Engineering and Aerospace Technology*, Vol. 80, No. 6, Dec. 2008, pp 620–632.

[35] Moncayo, H., Perhinschi, M. G., and Davis, J., "Immunity-Based Aircraft Failure Detection and Identification Using an Integrated Hierarchical Multi-Self Strategy," AIAA Guidance, Navigation, and Control Conference, AIAA Paper 2009-5878, Chicago, IL, Aug. 2009.