2010

The design of an evolutionary algorithm for artificial immune system based failure detector generation and optimization

Jennifer N. Davis
West Virginia University

The Design of an Evolutionary Algorithm for Artificial Immune System Based Failure Detector Generation and Optimization

by

Jennifer N. Davis

Thesis submitted to the College of Engineering and Mineral Resources at West Virginia University in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Approved by:

Dr. Larry E. Banta

Dr. Powsiri Klinkhachorn

Dr. Mario G. Perhinschi, Committee Chairperson

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia

2010

Keywords: Evolutionary Algorithm, Artificial Immune System, Failure Detection and Identification

Abstract

The Design of an Evolutionary Algorithm for Artificial Immune System Based Failure Detector Generation and Optimization

Jennifer N. Davis

The development of an evolutionary algorithm and accompanying software for the generation and optimization of artificial immune system-based failure detectors is presented in this thesis. These detectors use the Artificial Immune System-based negative selection strategy. 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. The evolutionary algorithm and accompanying software discussed in this document is concerned with the creation, optimization, and testing of failure detectors based on the negative selection strategy. 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 evolutionary algorithm that attempts to optimize the number of detectors, overlapping between detectors, and coverage of the "non-self" while maintaining no overlapping with the "self." For this second phase, the initial population is composed of sets of detectors, called individuals, obtained in the first phase. Specific genetic operators have been defined to accommodate different detector shapes, such as hyper-rectangles, hyper-spheres, hyper-ellipsoids and hyper-rotational-ellipsoids. The output of this evolutionary algorithm consists of an optimized set of detectors which is intended for later use as a part of a detection, identification, and evaluation scheme for aircraft sub-system failure.

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. This user interface is designed for use with Windows and MATLAB 7.6.0, although measures have been taken to maintain compatibility with MATLAB version 7.0.4 and higher, with limited interface compatibility. This interface may also be used with UNIX versions of MATLAB, version 7.0.4 or higher.

The results obtained show the feasibility of optimizing the various shapes in 2, 3, and 6 dimensions. Hyper-spheres are generally faster than the other three shapes, though they do not necessarily exhibit the best detection results. Hyper-ellipsoids and hyper-rotational-ellipsoids generally show somewhat better detection performance than hyper-spheres, but at a higher calculation cost. Calculation time for optimization of hyper-rectangles seems to be highly susceptible to dimensionality, taking increasingly long in higher dimensions. In addition, hyper-rectangles tend to need a higher number of detectors to achieve adequate coverage of the solution space, though they exhibit very little overlapping among detectors. However, hyper-rectangles are consistently and considerably quicker to calculate detection for than the other shapes, which may make them a promising candidate for online detection schemes.

Dedication

To my advisor, Mario Perhinschi, the best advisor a student could ask for, for having more faith in me than I had in myself and more patience than any person has a right to.

To my committee, for all their input and expertise.

To my family, for their love and support, always and continuously.

And to my love, Brenton Wilburn, for being there through it all.

Acknowledgements

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.

Table of Contents

List of Figures

List of Tables

List of Symbols

Symbol		Definition
a	=	Semi-axis length vector
c	=	Center
D	=	Detector, genotypes
d	=	Semi-side length vector, applicable to hyper-rectangles
GB	=	Gigabytes (millions of bytes)
GHz	=	Gigahertz (millions of samples per second)
Hz	=	Hertz (samples per second)
L	=	Lower, or worse, limit
N	=	Population size
p	=	Probability of selection
PI	=	Performance Index value
q	=	Cumulative probability of selection
r	=	Radius
RAM	=	Random Access Memory
S	=	Self
$\overline{S}$	=	Non-self
TF	=	Total Fitness
U	=	Upper, or better, limit
W	=	Performance Index weight
$\sum$	=	Universe; all possible solutions
	Subscripts	
coverage	=	Coverage performance index criterion
E	=	Hyper-ellipsoids
e	=	Hyper-ellipsoids
i	=	Referring to a value for a particular individual
number	=	Number of detectors performance index criterion
overlap	=	Overlap performance index criterion
S	=	Hyper-spheres
s	=	Hyper-spheres
R	=	Hyper-rectangles
r	=	Hyper-rectangles
RE	=	Hyper-rotational-ellipsoids
re	=	Hyper-rotational-ellipsoids
	Acronyms	
AIS	=	Artificial Immune System
BIS	=	Biological Immune System
DRC	=	Design Requirements and Constraints
DR	=	Detection Rate
EA	=	Evolutionary, or genetic, Algorithm
FA	=	False Alarm Rate
FDGO	=	Failure Detector Generation and Optimization

FDIE	=	Failure Detection, Identification, and Evaluation
GA	=	Genetic, or evolutionary, Algorithm
GBM	=	Gradient Based Method
HNIS	=	Hybrid Neural-Immune System
IFCS	=	Intelligent Flight Control System
IFDOT	=	Immunity-based Failure Detector Optimization and Testing
LFDB	=	Large Fast Drifting Bias
LSB	=	Large Step Bias
NASA	=	National Aeronautics and Space Administration
NSA-RV	=	Negative Selection Algorithm with Real representation and Variable radius
sGA	=	Structured Genetic Algorithm
WVU	=	West Virginia University

Note: All variables are normalized.

1 Introduction

In recent years, failure detection, identification, and evaluation (FDIE) of aerospace vehicles and their sub-systems over the full flight envelope has become recognized as an imminent necessity (1) (2) (3) (4) and has become a major objective of NASA's Aviation Safety Program (5). Previous attempts (6) (7) (8) (9) have been limited to detection in only certain areas of the flight envelope, or of only a few types of failure. The research effort which encompasses the subject matter of this thesis is intended for use with aircraft flight systems, and is capable of detecting the occurrence of a failure at any point in the flight envelope and affecting any subsystem, identifying and compensating for known failures, and reevaluating the safe operation flight envelope of the craft. Although the research effort is focused upon aircraft applications, the material discussed in this thesis, namely the optimization of immunity-based failure detectors, is intended to remain general. Thus the optimization methodologies described in this thesis could be applied to non-aerospace systems as well with little to no customization, though the focus will remain on applicability to aerospace systems.

The FDIE problem requires adequate tools capable of handling the complexity and potentially high dimensionality associated with it. A new artificial intelligence technique inspired by the biological immune system (BIS), called Artificial Immune System (AIS), has been proposed for use in detecting failure in aerospace systems (10) (11). The AIS-based fault detection paradigm operates similarly to the BIS in that it uses the principle of self/non-self discrimination to distinguish whether an entity belongs to a system or not. The AIS can potentially directly address the issues associated with the design of a comprehensive and integrated set of methodologies for FDIE.

A set of methodologies utilizing AIS-based failure detection, identification, and evaluation for a wide variety of aircraft sensor, actuator, propulsion, and structural failures and damages (10) is currently under development at West Virginia University (WVU) within NASA's Aviation Safety Program (12) (13). A critical issue for this detection scheme is generating adequate detectors (14), or obtaining sufficient description of regions of the hyper-space as defined by the identifiers only reached in the presence of adverse conditions.

To date, there is no deterministic method to generate detectors over the non-self region of the hyper-space, 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 and non-self regions without these overlapping for good detection performance makes evolutionary or genetic algorithms (15) (16) a promising solution for the generation of AIS detectors.

The genetic algorithm presented in this thesis is intended to produce and optimize detectors for an AIS-based fault detection and identification scheme. These tools were implemented within an interactive integrated design environment in MATLAB. The main objective of this thesis is to present the development and operation of the Immunity-Based Failure Detector Optimization and Testing (IFDOT) design environment, intended to create and optimize detectors for the purposes mentioned above. In addition, this thesis is intended to present results comparing some of the various design possibilities afforded by this design environment and analyze some important aspects such as improved detection performance and detector computation time. Note that the range and number of options prohibits full exploration of all of the design parameters of this program within this thesis.

Section 2 presents previous research efforts related to FDIE and this algorithm, and defines how this algorithm improves upon earlier approaches. Section 2.1 presents some support for the

development of FDIE. The main aspects of the AIS paradigm and its applications in the field of fault detection are presented in Section 2.2. Section 2.3 consists of a brief review of evolutionary or genetic algorithms, including applications of EAs to optimization of fault detection schemes. The general architecture of an integrated system of tools developed in MATLAB for the generation and optimization of AIS-based detectors is discussed in Section 3. Details on the main components and phases of the algorithm are presented in Section 4. The graphical user interface and the options available for the AIS detector set design are presented in Section 5. In section 6, results demonstrating the functionality and benefits of this approach are discussed. Finally, the conclusions are summarized in Section 7 followed by a reference list.

2 Literature Review

2.1 Reasons for Failure Detection, Identification, and Evaluation

In order to increase the safety and survivability of aircraft, a method for detecting and evaluating failures is required. It is not feasible to train pilots to recognize all potential failures, nor is it always possible for a pilot to recognize that a failure has occurred without performing an action which has been hampered by the failure. This could cause disastrous results if the pilot, unaware of the situation, attempts a maneuver the aircraft is no longer capable of performing. For this reason, a detection scheme implemented in the control scheme of the aircraft is needed.

Several methods have previously been developed to detect and evaluate aircraft failures. These include parameter identification algorithms (17), impedance estimation (6), state estimation (18), Kalman Filtering (19), and neural networks (7) (20) (21) (22) (23). However, none of these previous methods is capable of detecting failures, both known and unknown, over the entire flight envelope and correctly evaluating and reacting to the various failures. Each of these research attempts is limited to detecting specific failures or detecting failures over only a limited range of the flight envelope.

A new method has recently emerged which shows promise in detecting various types of failures, both known and unknown, over the entire flight envelope. This method is an artificial intelligence technique referred to as artificial immune system. This paradigm draws inspiration from the robustness and adaptability of the biological immune system found in mammals. It has already been applied to aircraft fault detection (11) (24), with promising results.

2.2 Artificial Immune System for Fault Detection

In recent years, the biological immune system has been the focus of great research interests to mathematicians and engineers, due to its robustness and vast information processing abilities (25). From this research, a branch of artificial intelligence dubbed the artificial immune system has surfaced. This technology possesses great potential for solving complex problems related to detection of abnormal conditions. This section will describe the functions of the BIS, address the relationship between the BIS and the AIS paradigm, and give examples of applications involving AIS for fault detection.

"The biological immune system is a complex adaptive system that has evolved in vertebrates to protect them from invading pathogens" (26). The BIS has the ability to detect microbial and non-microbial exogenous entities while not reacting to the body's own cells. It is the body's first line of defense against viruses, infections, and other intruders. The most important function of the biological immune system is self/non-self discrimination (27) in which the immune system must be able to identify and destroy potentially-harmful exogenous entities without harming the body. Specialized antibodies called T-cells are the component of the system with the most important role in this process. T-cells (11), which circulate throughout the body, are equipped with biological identifiers, or specific molecular strings of organic compounds such as proteins or polysaccharides. These cells are used to differentiate between self cells and non-self entities. When a T-cell comes in contact with an entity which matches its identifiers, it bonds to that entity, marking it for destruction.

T-cells are generated through a pseudo-random genetic rearrangement mechanism (25) in the bone marrow. This ensures high variability of the new cells in terms of the biological identifiers. T-cells then migrate to the thymus. The purpose of the thymus is to produce mature T-cells (28). In the thymus, the immature T-cells undergo a process sometimes called "thymic education" (28), in which T-cells whose identifiers match the self are destroyed or altered. Eventually, only those T-

cells that are "different" from the body's own signatures are allowed to mature 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 strength of the biological immune system lies in its complexity and adaptability. The biological immune system is largely a distributed system (29), meaning that unlike other systems, it does not focus around a single organ. This system, with no single point of failure, is highly robust, dynamic, error-tolerant, self-monitoring, and adaptable (30). T-cells have the ability to work independently, including destroying entities and reproducing new antibodies to fight infection. Because the contribution of each T-cell is small, some mistakes may be made without catastrophic effect to the body (30).

The mechanisms and processes of the biological immune system are the inspiration for the AIS as a new artificial intelligence technique for fault detection (14) (15). Artificial Immune System is the name for all efforts to develop computational models inspired by biological immune systems (27). Three basic principles (31) from the biological immune system have been adopted and adapted into the artificial intelligence technique known as the artificial immune system: immune network theory, clonal selection principles, and negative-selection strategy.

Immune network theory encompasses a network of antibodies capable of reproducing. Jerne (32) (33), who has greatly contributed in this area, defines a network in which initial antibodies can create new antibodies, the new antibodies can create an additional generation, and so on. Such a network is adaptive, dynamic, and self-aware, much in the same ways as the BIS.

Clonal selection principles are based on the biological immune system's ability to learn from previous experiences (34), such that it can better detect antigens it has been exposed to, yet it can also detect unknown antigens. This allows the algorithm to learn using known training data, but also be effective against conditions that are yet unknown.

Negative selection, the key AIS principle of importance in this thesis, is a computational process used to simulate biological self/non-self discrimination (27). Negative selection algorithms work by generating a self from "normal" data, then using this self to randomly generate detectors for the non-self (14). Any detectors that detect, or cover, the self region are discarded, leaving only detectors that cover the non-self. Negative selection looks for activation of detectors in the non-self to determine when abnormal conditions have occurred.

The artificial immune system paradigm has, in recent years, been applied to numerous applications, such as data mining (35), pattern recognition (36) (37), computer security (38) (39) (40) (41), fraud detection (42) and adaptive control (43). One significant area of interest is AIS for fault detection. The negative selection algorithm (14) is one application of the AIS paradigm, which is used to distinguish self, or normal operating conditions, from non-self, or abnormal operating conditions. Such an algorithm can be flexibly applied to fault detection, or detection of anomalous conditions. Some examples of such applications include fault diagnosis in brushless machines (44), robot control and fault tolerance (45) (46) (47) (48), hardware fault tolerance (49), and aircraft fault detection using real-valued negative selection (11) (24).

The basic concept of the AIS paradigm for fault detection 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. Extensive experimental data are necessary to determine the "self" or the hyper-space of normal conditions. 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 (50) (51). Finally, a detection logic must be designed for real time operation with a high detection rate and low number of false alarms (52). The block diagram of the AIS design process for fault detection is presented in Figure 2.1.

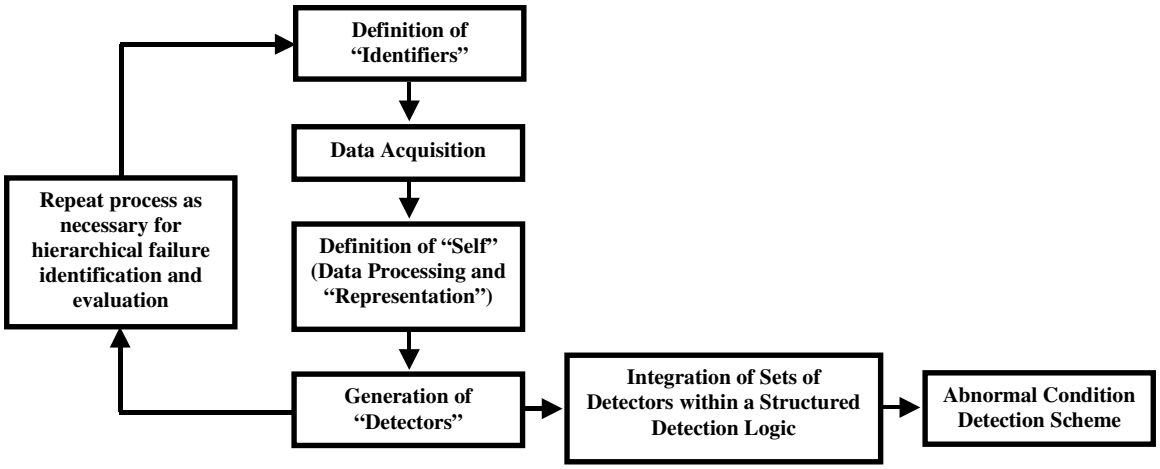

Figure 2.1—Artificial Immune System-Based Abnormal Condition Detection (53) (54)

Dasgupta and KrishnaKumar have pioneered the use of AIS for fault detection for aerospace systems (10) (11). Certain issues about dimensionality and limitations raised by Stibor et. al. (55) (56) for the use of negative selection algorithms were refuted by Ji and Dasgupta (57). A discussion of previous applications of AIS for aircraft fault detection is needed to define the benefits and differences of the methods contained in this thesis.

One application of AIS for aircraft fault detection by Gonzalez and Dasgupta (24) describes the use of a real-valued negative selection algorithm. It provides reasoning that detector sets are smaller, identifying anomalies after detection is easier, and applying additional immune techniques is simpler than in the case of using binary representation. Using this method, both normal and abnormal conditions can be mapped onto the solution space. The solution space is reduced to a unit hypercube by scaling all identifiers to values between zero and one. This method is an inspiration for many of the methods utilized in this research effort, including data representation, basic clustering, and basic detector generation. However, this application only utilizes one possible detector shape and, as is the focus of this thesis, does not use an EA to optimize the non-self detectors.

An additional technique used by the Gonzalez and Dasgupta (24) is anomaly detection using self-organizing maps. This type of neural network is used to organize the normal condition data, similar to clustering, which is discussed in Section 4.2. This tends to use a pseudo-fuzzy approach to anomaly detection. If a data point is near enough to the self, it is considered normal. Otherwise, it is abnormal.

An application of AIS for aircraft fault detection (11) involves a real-valued negative selection algorithm, utilizing hyper-sphere detector definition with variable radius. A self is defined using test data obtained from a C-17 man-in-the-loop flight simulator, such that it encompasses the

entire space occupied by normal operating conditions. The data is scaled to between values of 0 and 1, then clustered using the k-means algorithm.

Detectors are then generated to fill the areas not covered by the self, with two goals: maximize coverage of the non-self while minimizing coverage of the self. This detector generation algorithm is more complex than those seen previously. It not only generates detectors at random, but incrementally moves invalid detectors away from the self until the detector becomes valid. In order to assess the effectiveness of the detection scheme on the C-17, identifiers of roll-, pitch-, and yaw-rate commands and measurements were used to identify 5 simulated faults. Once it is determined that the scheme is capable of detecting a failure, the activated detectors corresponding to a specific failure are noted, so that later the type of failure can be determined, not just that a failure has occurred. When tail and wing damages are simulated, detection rates of 89% and 92% are achieved, with few false alarms. Tests also showed that the number of false positives is inversely proportional to the number of detectors used in the scheme.

The previous method is later expanded into a method called Multilevel Immune Learning Detection (58), in which multiple levels of immunity-based detectors are implemented. The first detection level is broad, to encompass a wide variety of potentially unknown faults. Another level of detection is implemented with smaller detectors to detect known failures. Once the fault has been declared and identified, an adaptive control scheme uses the fault data to compensate for the failure. This scheme was tested on engine, tail, and wing failures using roll-, pitch-, and yaw-rate as identifiers. Average results over 10 trials of each failure type showed detection between 91.8% and 97.8%, with false alarms less than 1.04%.

Common afflictions of AIS algorithms are high-dimensionality and lack of coverage of the non-self areas. The success of the AIS-based FDIE scheme will depend on the ability of the parameters selected as identifiers (e.g. aircraft states and pilot input) to capture the dynamic signature of every targeted 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 solution space to hyper dimensions and exposing the entire process to specific issues (59) that can potentially have a negative impact on the performance of the FDIE scheme. Calculation in these higher dimensions is often complex, and involves non-conventional means. High dimensionality also reduces the likelihood of generating detectors with high coverage of the solution space. Low coverage leads to poor performance of the AIS in detecting abnormal conditions.

In order to combat high-dimensionality at its root, new methods for the reduction of dimensionality are being developed (60) (61) (62). The purpose of these methods is to reduce the dimensionality of a data set, while retaining the integrity of the identifiers. Many of these methods are emerging, though most involve highly-complex non-linear conversions and are not fully developed. If these reduction of dimensionality methods are to be applied to the problem discussed in this thesis, they will be handled separately, prior to preprocessing with the utility discussed in this document, and will therefore not be discussed further.

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 (11). In this context, evolutionary algorithms can potentially provide the tools necessary for optimizing detectors in a high-dimensionality solution space.

2.3 Evolutionary Algorithm for Detection Rule Optimization

Evolutionary, or genetic, algorithms (63) (64) (EAs) are a class of artificial intelligence techniques which are based on the biological principles in Darwin's theories regarding evolution of species. These techniques are focused upon parameter optimization (65), and are applicable to a wide variety of science, engineering, and economics problems. EAs use an iterative approach directed to search the solution space for a global optimum, or the fittest individual in the population.

Several key components of Darwin's theories help to shape the EA paradigm. Individuals are capable of surviving to reproduce new offspring and continue their genetic line based on their aptitude for adapting to the surrounding environment. Thus, better-fit individuals live longer and produce more offspring. Individuals in an EA must behave in the same way.

Within an EA, an individual is a potential solution to a parameter optimization problem. The population is composed of a number of individuals, all competing for offspring in the next generation. This competition can be approached in varying ways. Individuals can have a lifespan of one or many generations. This lifespan can be the same for each individual, or vary depending on the fitness of the individual. An individual also produces offspring, or copies of itself, based on the individual's fitness. An individual may receive, one, many or no copies in the next population. This mimics the extinction or proliferation of a genetic lineage. The EA's population size may remain fixed or vary from generation to generation.

Within the EA paradigm, individuals are also referred to as "chromosomes". In order to define the parameter optimization problem to be solved, a set of design requirements and constraints (DRC) is formulated. The set of DRC acts as the environment within the EA paradigm. Each DRC is assigned a performance function, and an individual is evaluated based on its performance with respect to the DRC to determine its fitness, or performance index. The performance index is used to determine the next generation. Several algorithms for deciding the new population are available, including the two most common roulette-wheel selection and tournament selection.

One of the strengths of the EA paradigm is the open-ended nature of the DRC. The DRC may be expressed mathematically, logically (binary or fuzzy), or descriptively. In addition, the DRC may have no relationship with each other. This allows EAs to easily solve many problems that would be difficult or impossible to solve analytically.

An individual, or chromosome, does not necessary stay the same throughout the generations of the EA. If this were the case, multiple generations would not be needed. Instead, chromosomes undergo probabilistic alteration through the implementation of genetic operators, such as mutation and crossover. These mimic the natural mutations that can occur occasionally within genes during the reproduction of cells. Within the EA, these operations are intended to explore new areas of the solution space.

In general, the EA begins with a randomly assigned initial population of solutions. These guesses undergo alteration by the genetic operators. They are then assigned a fitness based on the DRC, and a new population is generated based on this fitness. 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 2.2.

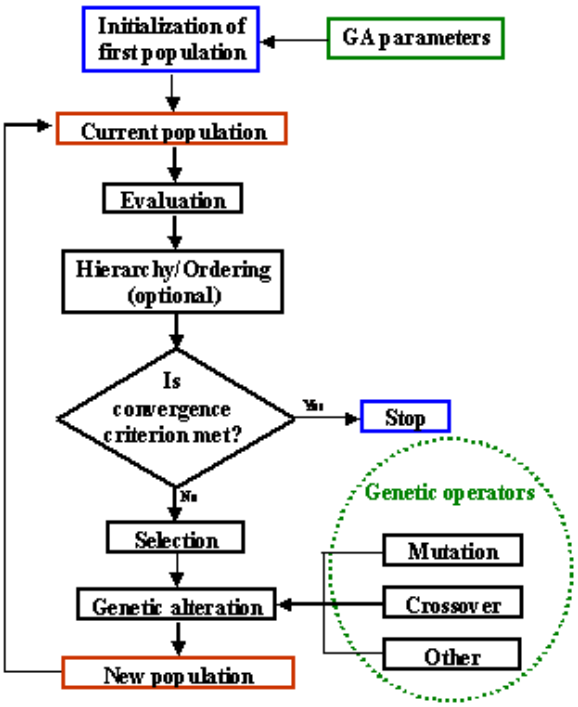

Figure 2.2—Block Diagram of an Evolutionary Algorithm (53) (54)

Evolutionary algorithms are widely used for parameter optimization, due to their superiority in many respects over the traditional gradient-based methods (GBMs). Evolutionary algorithms prove to be global and robust over a wide variety of problems, with excellent potential for solving, highly complex, nonlinear problems, including those with high-dimensionality. Several characteristics of EAs set them apart from traditional GBMs, and often make them a better or the only choice. Unlike GBMs, EAs can search the solution space in many directions at once, including in high-dimensions. This is because the solutions are randomly altered using the genetic operators, and the search is directed using the fitness rating of the individuals. The performance criteria used to determine the fitness rating of the individuals are highly customizable. There are no constraints on the formulation of these criteria. They may be analytical, logical, or descriptive. There is no need for continuity, derivability, or bijectivity of the fitness functions. Properly designed EAs also avoid local extrema by using an adequate balance between exploration of the solution space and exploitation of good existing solutions. Finally, EAs are able to find better global solutions since they are not dependent upon the problem they are intended to solve. Although domain-specific information can be used to guide or steer the EA, it is not necessary, and a lack of this information can potentially lead to the discovery of a good solution that would have otherwise been excluded by more traditional methods.

In this thesis, new research is discussed in which an evolutionary algorithm is used to optimize the detector sets for detecting various aircraft failures over a wide range of the flight envelope for the design criteria specified above. Other research efforts similar to this approach have been attempted. Some examples of these approaches will be discussed, noting their differences from the material of this thesis.

One application of EA for optimization of AIS-based fault detectors comes from Amaral, et. al. (66). In this research effort, negative selection strategy is applied to the detection of faults in analog circuits. An evolutionary algorithm is used to optimize a small number of detectors, rather than using a larger number of detectors to cover the solution space. No overlapping is allowed to

occur with the self, though overlapping between detectors is permissible. Testing of this detection scheme revealed almost perfect detection performance.

Gonzalez et. al. (67) used a genetic algorithm to optimize fuzzy detectors for computer intrusion detection based on the artificial immune system paradigm. Unlike the research presented in this thesis which compares the use of four different hyper-shapes as detectors, this attempt only used rectangular detectors. In addition, the EA is maximizing the coverage of the non-self space and minimizing the coverage of the self space. This is contrary to the methods presented in this thesis in that this research tolerates no overlapping between detectors and the self space, since this could introduce false alarms and decrease the reliability and effectiveness of the finalized detection scheme. The research presented by Gonzalez et. al. also does not optimized the number of detectors or overlapping between detectors.

Research presented by Shapiro et. al. (68) suggests that hyper-ellipsoid detectors produce better detection performance than hyper-sphere detectors. This approach generates hyper-ellipsoids for AIS-based detection and optimizes them using a genetic algorithm. Unlike the EA presented in this thesis, an individual is a single ellipsoid rather than a full set of detectors. The hyper-ellipsoids are optimized for coverage of the solution space using only mutation to alter the hyper-ellipsoids. This method achieves good detection results using approximately half the number of detectors as needed for spherical detectors.

Research presented by Gao et. al. (69) utilized the genetic algorithm to optimized detectors for an AIS-based scheme. This research attempts to optimize coverage of the non-self while allowing neither overlapping with the self nor overlapping among detectors. However, rather than comparing multiple sets of detectors, the genetic algorithm compares individual detectors against each other. Since no overlapping is allowed, the location of the center of the detector determines the radius of the hyper-sphere detectors. Thus the performance index is assigned based solely on the size of the radius of the detectors.

Another similar attempt to that presented in this thesis is presented by Balachandran et. al. (70) (71). In this research effort, an EA is used to maximize the coverage of the non-self and minimize the overlap with the self of an AIS scheme for pattern recognition. In the effort presented by Balachandran et. al., multiple shapes are organized into a data representation called a Structured Genetic Algorithm (sGA) (72). A structured genetic algorithm is a type of EA which allows an individual detector to take different shapes throughout the evolution so that a single detector set contains multiple shapes. Again, this EA does not optimize for the number of detectors in the set or for overlapping between detectors. Like Gonzalez et. al., overlapping is also allowed to occur with the self. In addition, Balachandran's research uses three shapes: hyper-spheres, hyper-rectangles, and hyper-ellipsoids. The research presented in this thesis introduces another shape: the hyper-rotational-ellipsoid.

While each of these methods has some similarity to the research presented in this thesis, the key difference between the methods discussed in this thesis and the applications discussed in the previous paragraphs is that these methods are not optimized for coverage of the non-self, number of detectors, or overlapping between detectors. In order to produce good results using the AIS paradigm for fault detection, detectors must provide adequate coverage of the non-self without overlap with the self However, the detection rules must also be capable of running in real-time, when applied to the aircraft it is protecting. This requires the detector set to have a low number of detectors. In addition, overlap between detectors is not beneficial and should also be minimized for efficiency. These aspects require additional optimization of the detector sets, beyond generation.

3 General Architecture of Evolutionary Algorithm for Failure Detector Generation and Optimization

This section will cover the theory and techniques involved in the evolutionary algorithm for failure detector generation and optimization (FDGO) algorithm. This includes preparation steps such as normalization, clustering, and generation of the initial population (Phase I) for the evolutionary algorithm, which are not technically part of the actual evolutionary algorithm, but are necessary steps in order to perform the evolutionary algorithm, and which are included in the interactive utility called West Virginia University Immunity-Based Failure Detector Generation, Optimization, and Testing, referred to as IFDOT.

3.1 Problem Definition

Research is currently under investigation at WVU to produce a comprehensive and integrated set of methodologies for aircraft FDIE. The AIS paradigm has shown promising capabilities for producing such a solution. The AIS paradigm is to be used to detect failure over the partial or full flight envelope for a given aircraft. These methodologies are general, although they are applied to a particular system, the IFCS WVU research F15 aircraft model. In order for this scheme to obtain good detection results, adequate coverage of the solution space is needed, which requires flexible and extensive design tools for detector generation and optimization. Several design options including multiple cluster generation and detector generation algorithms spanning four shape options have been implemented. The self is defined using flight data collected from the flight simulator under normal operating conditions using the IFCS WVU research F15 model. This data is processed to produce the self clusters, which define the self region for the AIS. Detection rules, or detectors, will be produced using one of several algorithms. These will then be optimized by the FDGO, which itself allows a great deal of user design flexibility.

3.2 Definitions

Several key concepts will be used throughout this thesis with the following meanings. The s*olution space* Σ 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 Σ corresponding to normal flight conditions, while the n*on-self* $\bar{S}$, corresponds to abnormal conditions. Ideally, the self and the non-self are disjunct sets and completely cover the solution space:

$$\bar{S} \cap S = \phi \; and \; \bar{S} \cup S = \Sigma \qquad 1$$

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 problem, which is a single set of detectors covering the non-self. Several such individuals form the *population*. An example of a typical individual, as represented in 2 dimensions, is presented in Figure 3.1 below, in order to illustrate the self definition, or clusters, and non-self definition, or detectors.

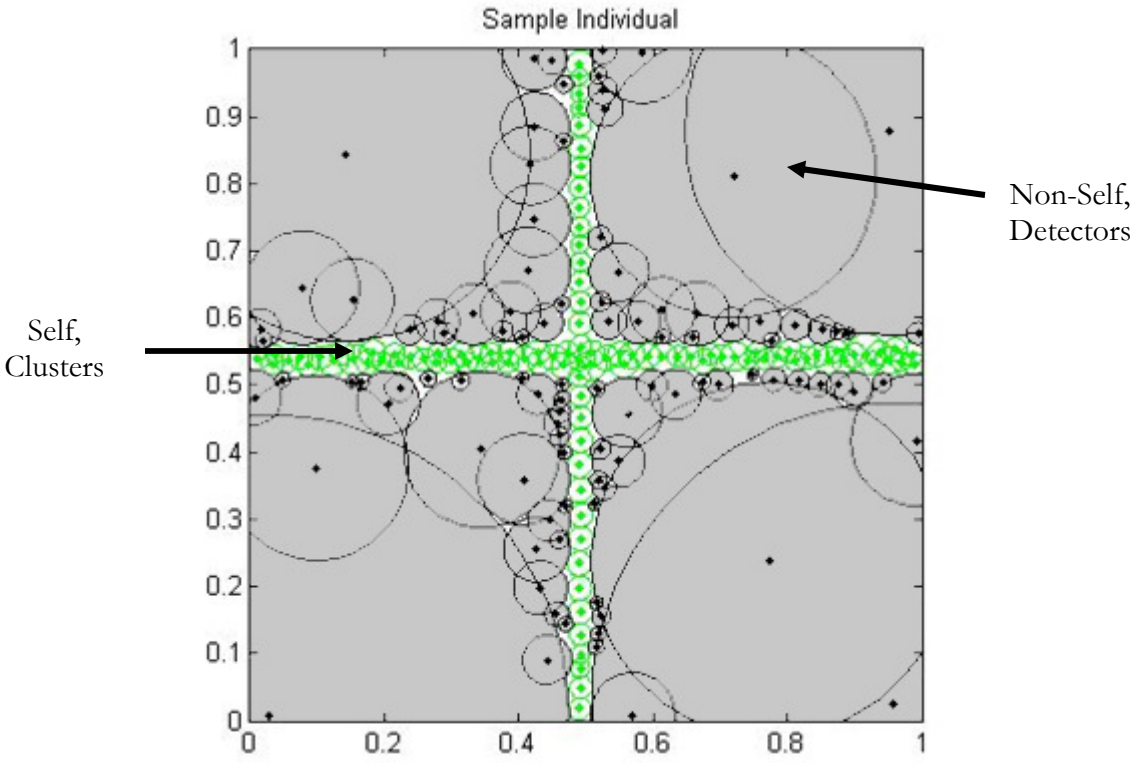

Figure 3.1—Illustration of a Typical Individual (53)

3.3 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 3.2:

- 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).

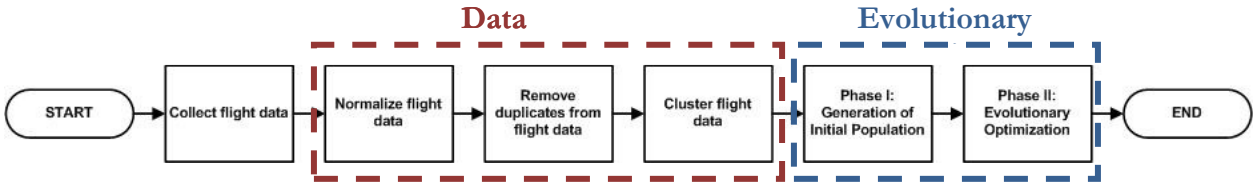

Figure 3.2—Flowchart of Optimization Processes (53) (54)

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.

3.4 Pertinent Mathematical Techniques

Some important mathematical techniques are necessary to the calculations performed throughout the FDGO algorithm. These will be discussed below in detail pertinent to the understanding of the algorithms encompassed in this thesis.

3.4.1 Distance calculations

Two important distance calculations are needed throughout these algorithms. These are Euclidean (73) distance and Mahalanobis (74) distance. Euclidean distance is used to calculate the straight-line distance between two points, for example, the distance between the center of a detector (non-self) and the center of a cluster (self) when determining whether a the detector may overlap the self. Euclidean distance, in 2 dimensions, is defined using the equation below.

$$D = (x^2 + y^2)^{\frac{1}{2}} \qquad 2$$

In the above equation D represents distance, and x and y are distances in each dimension. When Euclidean distance is calculated in higher dimensions, the format is defined using the equation below.

$$D = (x_1^2 + x_2^2 + \cdots + x_n^2)^{\frac{1}{2}} \qquad 3$$

In this equation, D represents distance, x is a distance in a dimension, and subscripts 1 to n indicate which dimension, with n equal to the number of dimensions in the hyper-space. The Euclidean distance is used in conjunction with hyper-sphere and hyper-rectangle detectors.

Mahalanobis distance is a distance calculation applicable to hyper-ellipsoids and hyper-rotational-ellipsoids. For the purposes of this research, Mahalanobis distance determines whether a point in the solution space falls inside or outside of a hyper-ellipsoid or hyper-rotational-ellipsoid detector. In calculating the Mahalanobis distance in this algorithm, the MATLAB function, *mahaldist,* is used. This function intakes the location of a point, the center of the ellipsoid, and the A matrix as the weighting function, defined below. It outputs a numerical value, which if less than 1, indicates the point falls within the hyper-ellipsoid or hyper-rotational-ellipsoid detector.

$$A = \Lambda V \Lambda^T \qquad 4$$

$$V = \begin{bmatrix} 1/a_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/a_n^2 \end{bmatrix} \qquad 5$$

where A is the weighting matrix for the Mahalanobis distance calculation, V is the length matrix of the hyper-ellipsoid or hyper-rotation ellipsoid detector, and Λ is the orientation matrix of the detector. The orientation matrix is discussed further in defining the representation of hyper-ellipsoids and hyper-rotational-ellipsoids.

3.4.2 Volume Estimation

Due to the nature of using hyper-shapes, and that the hyper-shapes may overlap one another, it is not possible to calculate overlapping among detectors and coverage of the non-self region analytically. For this reason, the overlapping and coverage are estimated numerically using Monte Carlo-type volume estimation techniques (75) (76) (77). The Monte Carlo Volume Estimation algorithm generates points at random throughout the defined hyperspace, then tests the number of points that fall within certain objects to approximate the volume relative to the hyper-

space of an object. The number of points needed for a certain level of accuracy is given by the relationships below.

$$\gamma = 1 - \frac{C}{100} \quad (6)$$

$$N = \frac{1}{4 * \gamma * err^2} \quad (7)$$

In these equations, C represents the confidence interval, which is given as a value between 0 and 100, and err is the error permitted in the final solution. Increasing C to nearly 100 gives better results, but the calculation takes significantly longer. This is also true of decreasing err close to 0. Recommended values for these parameters as used in the algorithm are C=98 and err=0.01. These may be adjusted as needed, however, these values were primarily chosen due to computational loading and time constraints. As a comparison, the Table 3.1—Monte Carlo Calculation Time Comparison below shows the time in seconds needed to calculate coverage and overlapping using the Monte Carlo method implemented for this research for a single set of detectors. These are intended for comparison only, as computational power greatly influences the time needed for this calculation. The dashed combination was not completed. Due to its extremely extended calculation time, this combination of parameters was deemed inadequate for use in this research. Table and Table show the coverage and overlapping values calculated for each of these time trials, to show the similarity accuracy of each.

Table 3.1—Monte Carlo Calculation Time Comparison, in Seconds

err\conf	98	99	99.5	99.9
0.01	54.89	111.36	216.85	1072.53
0.005	220.42	434.78	862.35	4283.18
0.001	5515.88	10979.65	21476.37	---

Table 3.2—Monte Carlo Calculation Coverage Comparison, in Percent of Solution Space

err\conf	98	99	99.5	99.9
0.01	94.81	94.77	94.82	94.76
0.005	94.81	94.75	94.77	94.78
0.001	94.78	94.77	94.78	---

Table 3.3—Monte Carlo Calculation Overlapping Comparison, in Percent of Solution Space

err\conf	98	99	99.5	99.9
0.01	72.99	73.17	73.09	73.13
0.005	73.16	73.15	73.10	73.14
0.001	73.12	73.12	73.14	---

3.4.3 Bisection Method

Bisection method is a numerical method for determining the root of an equation, f(x) , given a bound [a,b], which contains only one root of the equation. Thus f(a)*f(b)<0. An approximation of the root, c, is specified by:

$$c = \frac{a+b}{2} \qquad 8$$

For this point, the value of f(c) is calculated and the new interval containing the root is determined. This process continues until the root is determined to a specified accuracy measured as the size of the interval containing the root.

4 Description of Evolutionary Algorithm Modules

This section will cover in detail each of the modules, or functions, present in the Interactive Utility. This includes preprocessing and clustering, generation of detectors, all segments of the evolutionary algorithm itself, and detection evaluation functions.

4.1 Preprocessing

Preprocessing of data takes place prior to the performance of the evolutionary algorithm. These algorithms are carried out in an effort to reduce the computational load upon the system performing the calculations. Self data sets can and should be quite large, in order to well-define the self portion of the solution space. However, it is neither practical nor feasible to work with such a large data set in its raw form. Normalization is carried out so that the solution space may be easily defined as a unit hyper-cube. Removal of duplicates and clustering are performed to reduce the size of the array that defines the self, helping the evolutionary algorithm to run more quickly.

4.1.1 Normalizing Data

Normalization is performed in order to make working with the potentially large number of dimensions easier and more intuitive. Otherwise the dimensions would range to a different scale for each of the identifiers. Not only would the ranges for each of the identifiers need to be kept track of throughout the course of the genetic algorithm, but this would cause distortion of the detector shapes, for instance a sphere with a constant radius, which is not preferable.

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 or failure sets are to be used for detection testing, as the same normalization factors must be used.

4.1.2 Eliminating Duplicates

Removing duplicate points reduces redundancy within the data and can substantially increase the speed of the clustering 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 or necessary self points may be removed, which could lead 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.

4.2 Clustering

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* (78) 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 hyper-spheres, hyper-ellipsoids, hyper-rotational-ellipsoids, or hyper-rectangles. The

reduction of empty space is achieved through an iterative clustering algorithm (12) (13) (79) in which the number of clusters is progressively increased until the desired level of empty space is reached.

4.2.1 Clustering Algorithms

Several clustering methods have been attempted prior to this AIS research. Initially clustering was performed using a fixed, uniform radius for all clusters and detectors. However, it was found that using clusters and detectors with variable radii allowed for more efficient coverage of the self and non-self solution space (80). Therefore this research uses variable size clusters and detectors.

Additional improvements to the clustering algorithm have been implemented (81) (82). When the self is defined using clusters, some "empty space" will necessarily be included. Empty space is defined as the portion of the solution space covered by a cluster that is not covered by a data point and its point radius. Empty space within clusters approaches zero as the number of clusters approaches the number of data points in the raw data set.

4.2.2 Clustering with Hyper-Spheres

Two clustering methods are integrated into the IFDOT utility for the purpose of producing hyper-sphere clusters. Hyper-sphere clusters are compatible with hyper-sphere, hyper-ellipsoid, and hyper-rotational ellipsoid detectors. The first method produces variable size clusters, without attention to empty space. The stopping criterion for this algorithm relies only on the number of clusters that have been generated.

Cluster centers are generated using the k-means method to find the correct number of logical centers within the self data. This also associates the data points with the center nearest each data point. Thus, the radius of the clusters is set using the distance from the self to the point for the farthest point associated with a particular center. Note that for this clustering method, any two sets of clusters generated using the same data set and the same number of clusters will be identical.

The other clustering method optimizes for empty space within the clusters. The improved clustering algorithm produces clusters iteratively, using the same method as above, but increases the number of clusters each iteration until the level of allowable empty space meets a specified requirement.

4.2.3 Clustering with Hyper-Rectangles

Clustering with hyper-rectangles is similar to the simple method for clustering hyper-spheres. Empty space is not taken into account when generating rectangle clusters. Centers are generated in the same manner as the previous methods, using the k-means algorithm. Self data points are associated in the same way. However, rather than assigning a constant radius, as is the case for hyper-spheres, this algorithm assigns a semi-axis length for each dimension, based on the distance to the farthest point in each dimension associated with each center. This produces hyper-rectangles. These can be likened to hyper-ellipsoids, whereas hyper-cubes, with the same length in each dimension, could be likened to hyper-spheres. Giving the hyper-rectangles independent lengths in each dimension allows them more flexibility, thus allowing them potentially better approximation of the self, or as detectors, more flexible coverage of the non-self.

4.3 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 hyper-spherical detectors and one method for hyper-rectangular detectors are implemented within the IFDOT tool. These are used to generate the initial population for Phase II of the EA.

4.3.1 Detector Generation with Hyper-Spheres

Two methods were implemented for generating hyper-spherical detectors. Hyper-sphere detectors make up the initial population for the EA when using hyper-spheres, -ellipsoids, or –rotational-ellipsoids, which will be discussed later. The first method implemented for hyper-spherical detector generation is a negative selection algorithm with real representation and variable detector size (29) (83) (84) (NSA-RV). The flowchart of this algorithm is presented in Figure 4.1. Candidate detectors are first initialized by random generation of their centers. If the center does not fall within the self or any previously generated and matured detectors, 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).

These stopping criteria are specified by desired coverage of the self and of the non-self. These are only approximate coverage estimates, and are not found using the Monte Carlo method. These criteria can be calculated using the following equations.

$$t \geq \frac{1}{1-Co} \qquad 9$$

In the above equation, t is the number of random centers that have fallen within another object and Co is the desired coverage of the non-self. When too many points have fallen within another detector or the self, this signifies that a desired coverage threshold has been reached.

$$T \geq \frac{1}{1-SCmax} \qquad 10$$

In equation 10, T is the number of detectors that have been attempted with a radius smaller than the desired threshold and SC_{max} is the desired coverage of the self, or more specifically, the small areas between clusters.

The second method for hyper-spherical detector generation is an enhanced NSA-RV, which integrates NSA-RV with detector moving and cloning (81) (82). This method attempts to limit overlapping among the detectors. 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, using the equations below.

$$W(d) = \sum_{d \neq d'} w(d, d') \qquad 11$$

$$w(d, d') = \left(e^{\delta} - 1\right)^{m} \qquad 12$$

$$\delta = \frac{r_d + r'_d - D}{2r_d} \qquad 13$$

In the above equations, d is the detector for which overlapping is being measured, d' is the detector it is being compared with, m is the number of dimensions in the solution space (the number of identifiers), r_d is the radius of the detector, r_d' is the radius of the compared detector, and D is the center-to-center distance between the two detectors. Detectors are either matured or rejected based on an overlapping threshold. Rejected detectors are moved so that they can improve their overlapping. Detectors are moved using the following equation.

$$\eta = \eta_o e^{iter/\tau} \qquad 14$$

The factor calculated using the equation above, η, is added to the new detector center to move the center. Based on the iteration number, iter, and the decay factor, τ, this factor will decrease over time until the detector will finally be rejected if it is moved in a number of generations and is still not acceptable.

New candidate detectors are created in the vicinity of mature detectors. This process is referred to as "cloning". The number of new clones is inserted based on the overlapping of the detector being cloned. If the detector has some overlap, only one clone is generated at a random angle from the original. If the original detector exhibits no overlap, four clones are created. The first is placed at a random angle from the original, and the other three are generated at 90° intervals from the random clone. 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 4.2.

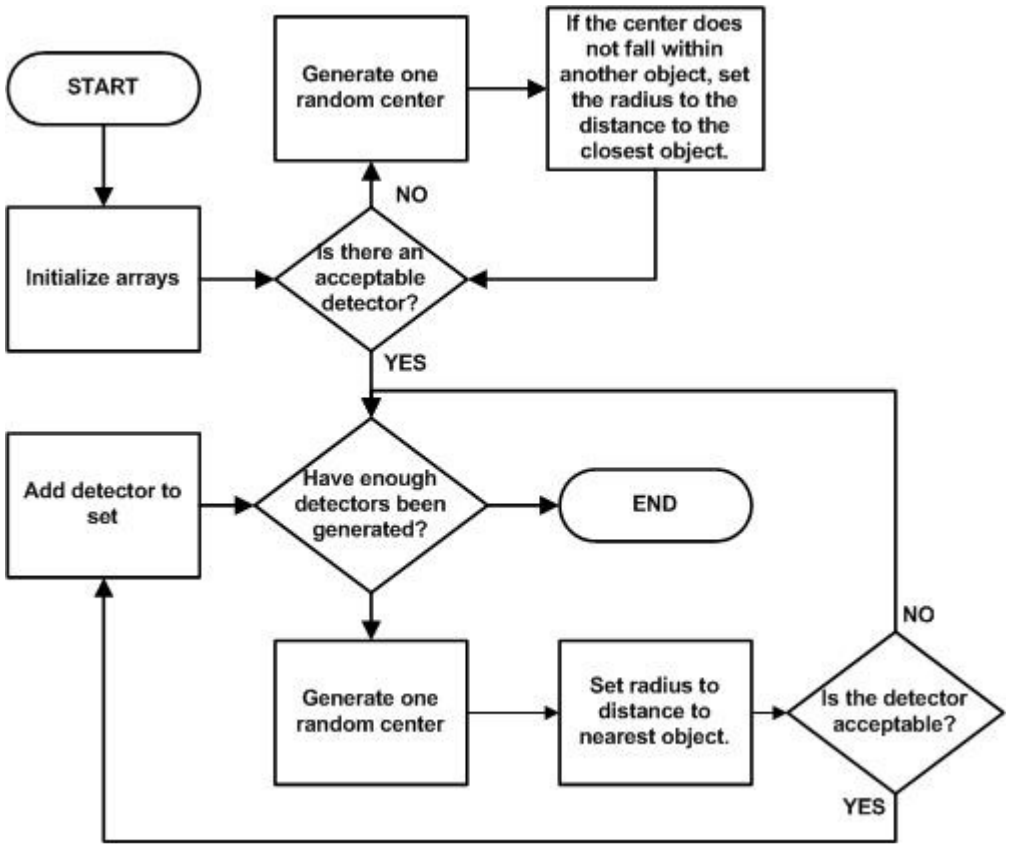

Figure 4.1—Flowchart of Detector Generation Using NSA-RV (53) (54)

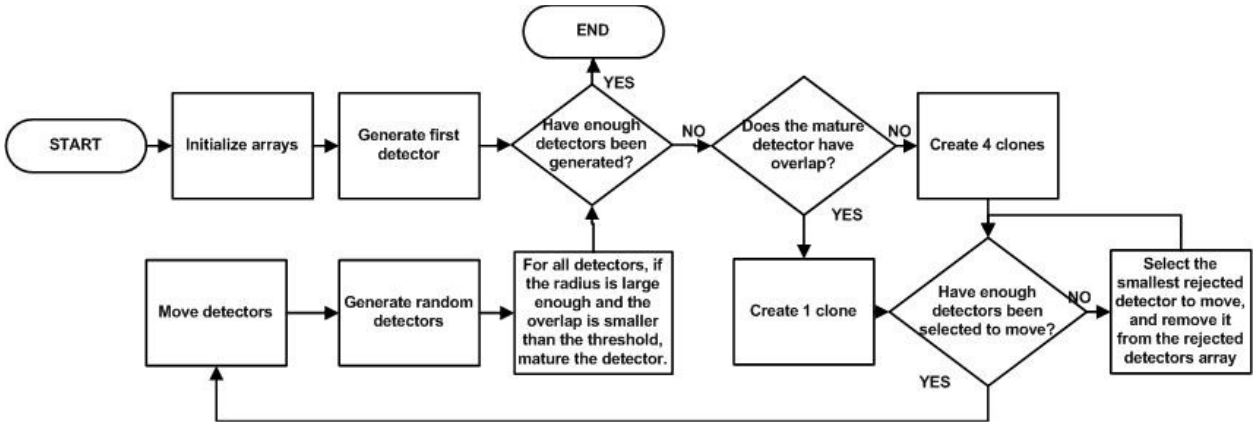

Figure 4.2—Flowchart of Detector Generation Using the Enhanced NSA-RV (53) (54)

4.3.2 Detector Generation with Hyper-Ellipsoids and Hyper-Rotational-Ellipsoids

Three shapes depend on hyper-sphere detector generation for their creation: hyper-spheres, hyper-ellipsoids, and hyper-rotational-ellipsoids. For hyper-spheres, the process is simple. However, hyper-spheres are a special case of hyper-ellipsoid in which all semi-axis lengths are equal. This principle is used to convert hyper-spheres into hyper-ellipsoids or hyper-rotational-ellipsoids.

Hyper-ellipsoids are defined as having a location in each dimension, which defines the center. In addition, the hyper-ellipsoid also has a semi-axis length for each dimension. This is different from the hyper-sphere which has only one radius, which is used for all dimensions. In order to convert the hyper-spheres into hyper-ellipsoids, the radius is duplicated for each dimension needed. Since hyper-ellipsoids have the capability of being rotated, an orientation matrix for each detector is also generated. The orientation matrix is a square matrix of equal dimension to that of the solution space. The elements of this matrix contain the rotation of the ellipsoid in each plane. When the detector is not rotated about any plane, the orientation matrix is the identity matrix of appropriate dimension.

Hyper-rotational-ellipsoids, like hyper-ellipsoids, have a semi-axis length for each dimension. However, similar to the hyper-sphere, the rotational ellipsoid only has one independent semi-axis length; the rest are dependent, or the same as each other. This means that rather than having the same radius for all dimensions, the rotational ellipsoid has the same radius for all dimensions but one.

The hyper-rotational-ellipsoids are defined as having a location in each dimension, the center. They have 3 additional parameters, besides the center. These are the independent semi-axis length, the dependent semi-axis length, and the location of the independent dimension. All dimensions aside from the one defined in the last parameter have a semi-axis length equal to the dependent length, while the independent dimension has the length of the independent semi-axis length. Hyper-rotational-ellipsoids have the capability of being rotated about the axes of the solution space, in the same manner as the hyper-ellipsoids. Therefore, each hyper-rotational-ellipsoid detector is also accompanied by an orientation matrix, defined in the same way as the orientation matrix for hyper-ellipsoids.

4.3.3 Detector Generation with Hyper-Rectangles

The detector generation 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, τ, which is set by the user. The equation for the decay parameter is:

$$\text{minimum distance in each dimension} = Rss_0 * e^{-\text{iteration}/\tau} \quad (15)$$

where Rss_0 is the base minimum radius provided by the user. The stopping criterion for this detector generation algorithm is reaching the expected coverage of the non-self portion of the solution space. This is calculated using equation 10 above. A flowchart for this function is provided below in Figure 4.3.

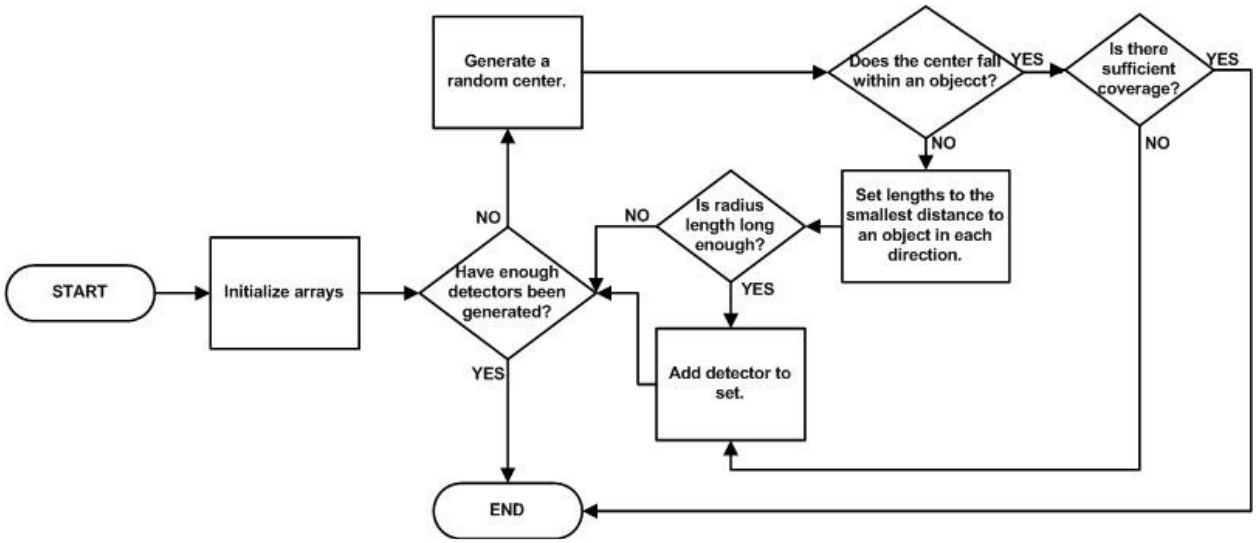

Figure 4.3—Flowchart of Detector Generation for Hyper-Rectangles (53) (54)

4.4 Phase 2—Optimization of Detectors

4.4.1 Evolutionary 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 hyper-bodies. A three-criterion performance index which plays the roles of the environment is used to assess the "fitness" of each individual. Four customized genetic operators, or variation 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 (16) (64). This evolutionary search for the optimum solution continues for the specified number of generations. The flowchart of the EA is presented in Figure 4.4.

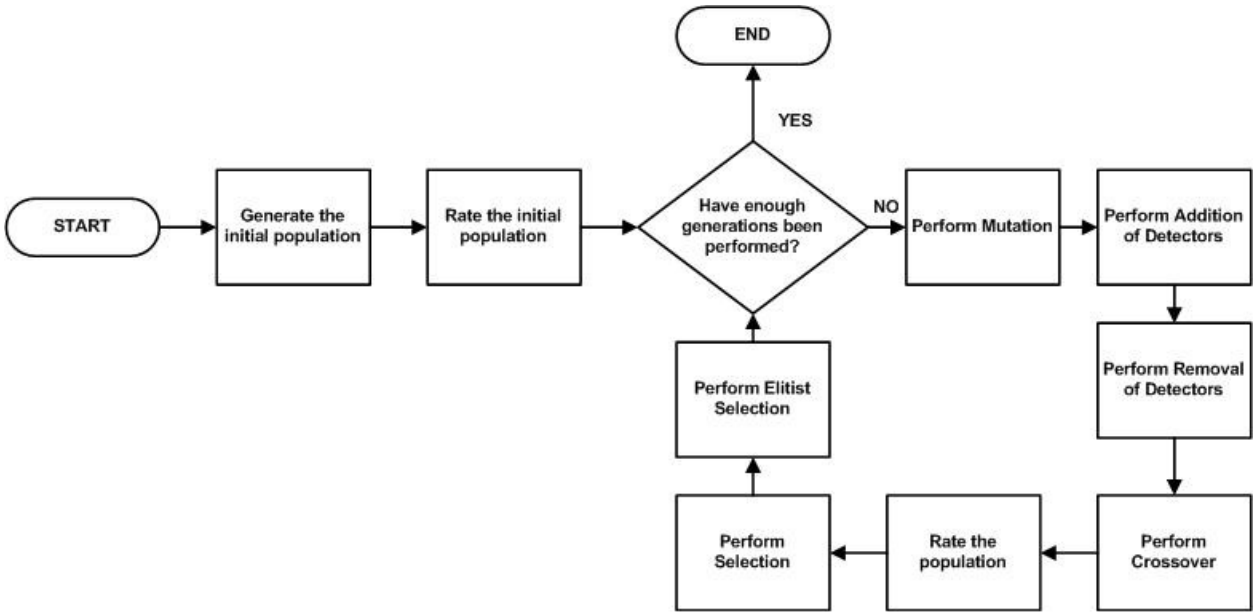

Figure 4.4—Flowchart of Evolutionary Algorithm (53) (54)

4.4.2 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:

$$D_S = (c_s, r_s) \quad 16$$

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. Hyper-rectangular detectors D_R are defined as:

$$D_R = (c_r, d_r) \quad 17$$

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

$$D_E = (c_e, a_e) \ , \Lambda_E \quad 18$$

where $c_e \in \Re^n$ is the location of the center of the ellipse, $a_e \in \Re^n$ is the semi-axes vector for all dimensions, n is the dimension of the solution space, and Λ_E is a square matrix of dimension n defining the orientation of the detector. Finally, rotational hyper-ellipsoidal detectors are defined as:

$$D_{RE} = (c_{re}, a_{re}), \Lambda_{RE} \quad 19$$

where $c_{re} \in \Re^n$, is the location of the center of the detector, $a_{re} \in \Re^2$, is the semi-axes vector, n is the dimension of the solution space, and Λ_{RE} is a square matrix of dimension n defining the orientation of the detector. Note that, unlike hyper-ellipsoids, which may have different axes for each dimension, only one preferential axis may differ from the others for hyper-rotational-ellipsoids.

4.4.3 Genetic Operators

Four distinct genetic operators - mutation, addition, removal, and crossover - are performed on the population according to the genetic operation rates established by the designer. The individuals that are subject to genetic alteration are selected randomly. In this section each of the four genetic operators will be discussed in detail. Since although each of the four genetic operators perform the same tasks, different hyper-shapes require that the genetic operators be carried out in different ways. For this reason, for each of the operators, the purpose and overview will first be discussed, and then the methods for each of the hyper-shapes will be described.

4.4.3.1 Mutation

The mutation genetic operator was designed with the purpose of producing small alterations to 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 hyper-spheres and hyper-rectangles, there are two types of mutation: gene alteration and gene relocation. Hyper-ellipsoids and hyper-rotational-ellipsoids also have a third type of mutation: gene rotation. 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 has multiple defining lengths, the dimension to be altered is also selected at random. This is the case for hyper-rectangles, hyper-ellipsoids, and hyper-rotational-ellipsoids. Hyper-rotational-ellipsoids also undergo random reassignment of the independent dimension. 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 hyper-ellipsoids and hyper-rotational-ellipsoids, rotation of the detector, called gene rotation, is capable by alteration of the orientation matrix. An axis is selected at random and the detector is rotated an amount at random about that axis. A flowchart of the mutation genetic operator is shown in Figure 4.5.

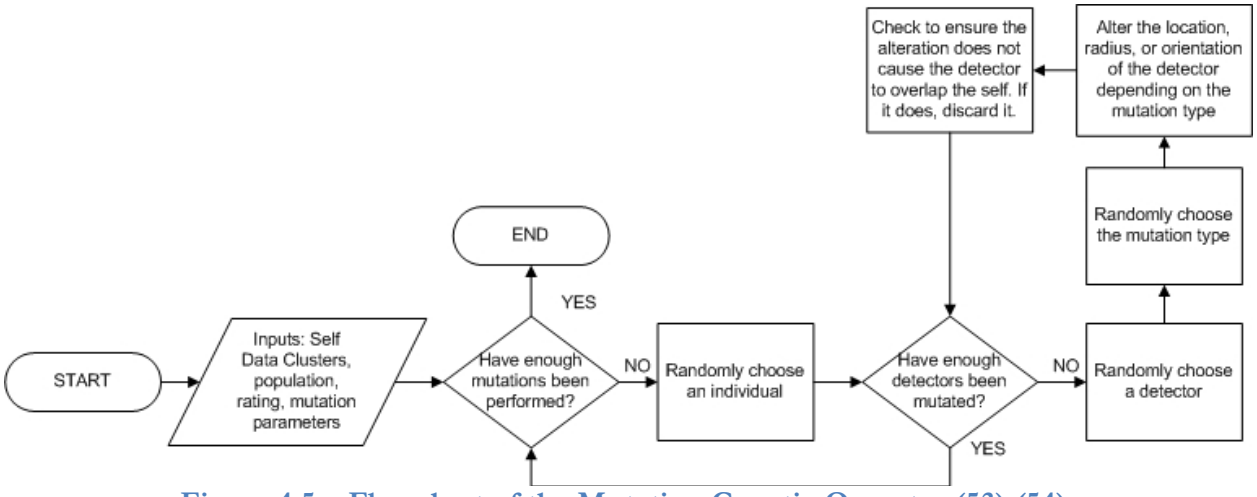

Figure 4.5—Flowchart of the Mutation Genetic Operator (53) (54)

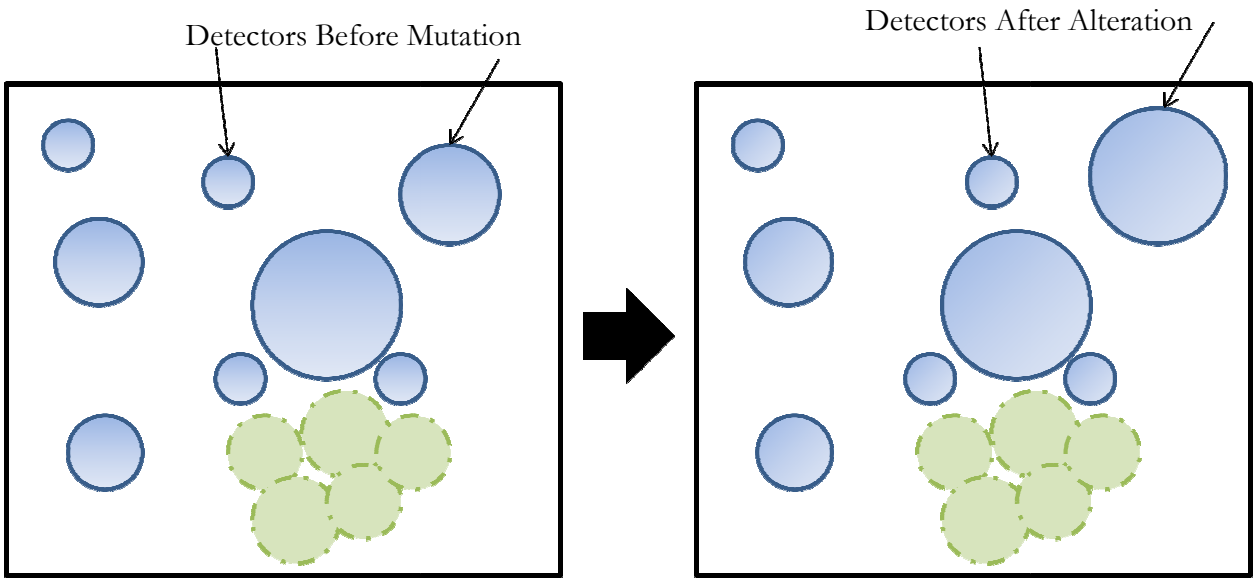

Figure 4.6—Diagram of the Mutation Genetic Operator (53)

4.4.3.1.1 Mutation with Hyper-Spheres

The mutation of the hyper-sphere detectors is the simplest case. Note that all values and selections are made at random. An individual is selected and within it a detector is selected. The mutation type is chosen at random, based on the weights provided by the user. For gene alteration, the radius is multiplied by a value whose maximum is given by the user, and this result is either added to or subtracted from the current radius, limited in such a way that the radius cannot become zero. For gene relocation, the center location of the hyper-sphere is moved in a random distance in a single direction. The distance is determined by multiplying the radius by a value whose maximum is specified by the user, and the result is added to or subtracted from the center location of the appropriate dimension. This is limited in such a way that the center cannot leave the unit-hyper cube of the solution space. Any alteration to a detector is checked for overlap with the self before it is finalized and added to the detector set. 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.

4.4.3.1.2 Mutation with Hyper-Rectangles

Mutation with of hyper-rectangles is similar to mutation of hyper-spheres, except that now there are multiple side lengths for which to account. For both gene alteration and gene relocation with hyper-rectangles, when a side length is needed, either to be altered, or as a factor is moving the center, the side length is chosen randomly.

4.4.3.1.3 Mutation with Hyper-Ellipsoids and Hyper-Rotational-Ellipsoids

Since the initial population as generated in Phase I only includes hyper-spherical detectors with the capability of becoming hyper-ellipsoids, the mutation genetic operator is responsible for creating the alterations to the detectors that result in hyper-ellipsoids and hyper-rotational-ellipsoids. In this way it is ensured that the initial solution is good, and that modifications are used to improve on the initial solution, which has the weakness of constant radius in all dimensions of a detector.

Gene alteration and gene relocation are carried out in the same way as for the hyper-rectangles, except that for hyper-rotational-ellipsoids the independent semi-axis length is reassigned with the gene alteration operator. Gene rotation is also applicable to hyper-ellipsoids and hyper-rotational-ellipsoids. Gene rotation is performed by selecting at random the orientation matrix of a single detector within an individual, and multiplying two of its dimensions by the rotation matrix. This rotates the detector in a single plane at a time for any mutation. The rotation matrix is defined as the identity matrix of the same dimension as the number of identifiers defining the solution space. The elements corresponding to the two axes selected at random are replaced by the rotation matrix elements, which are given in equation 20 below.

$$\begin{array}{c} \\ axis\ 1 \\ axis\ 2 \end{array} \begin{array}{c} \begin{array}{cc} axis\ 1 & axis\ 2 \end{array} \\ \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \end{array} \qquad 20$$

The orientation matrix of the detector is multiplied by the rotation matrix defined above, where $\alpha \in [0,\alpha_{MAX}]$ is the desired angle of rotation and α_{MAX} is the maximum desired rotation provided by the user. As before, each alteration is checked to ensure that the modifications have not caused the detector to overlap with the self before the changes are finalized and added to the detector set.

4.4.3.2 Crossover

To apply the crossover genetic 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 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 genetic operator is applied. Therefore, non-overlapping with the self, following this genetic operator, is guaranteed. This genetic operator is carried out identically for all hyper-shapes, with the exception that the hyper-ellipsoids and hyper-rotational-ellipsoids must trade orientation matrices attached to the detectors being swapped as well. Figure 4.7 contains a flowchart of this operator, and Figure 12 shows a diagram of the operator.

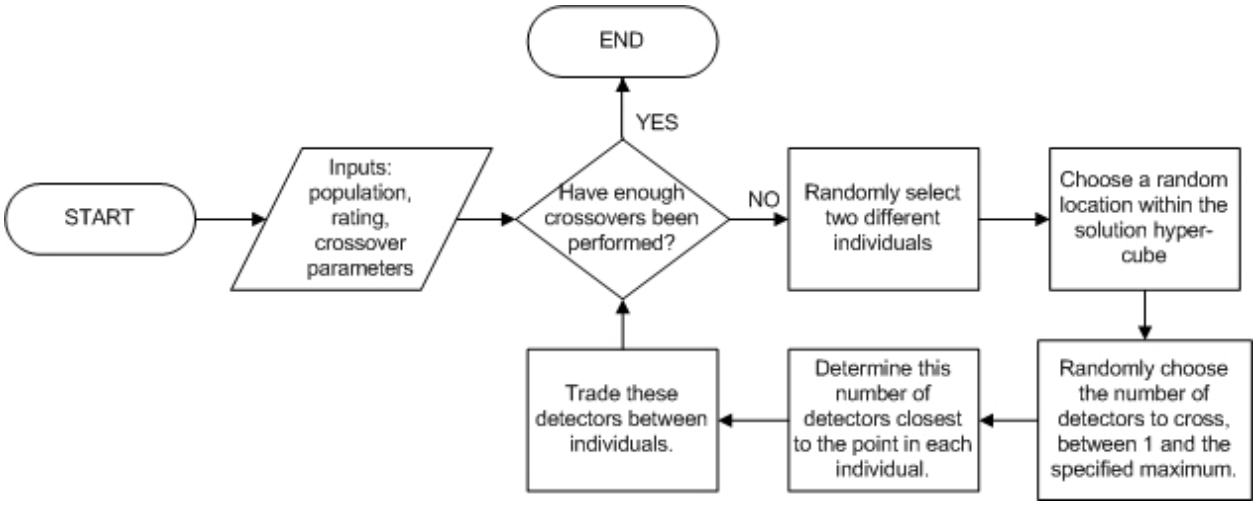

Figure 4.7—Flowchart of Crossover Genetic Operator (53) (54)

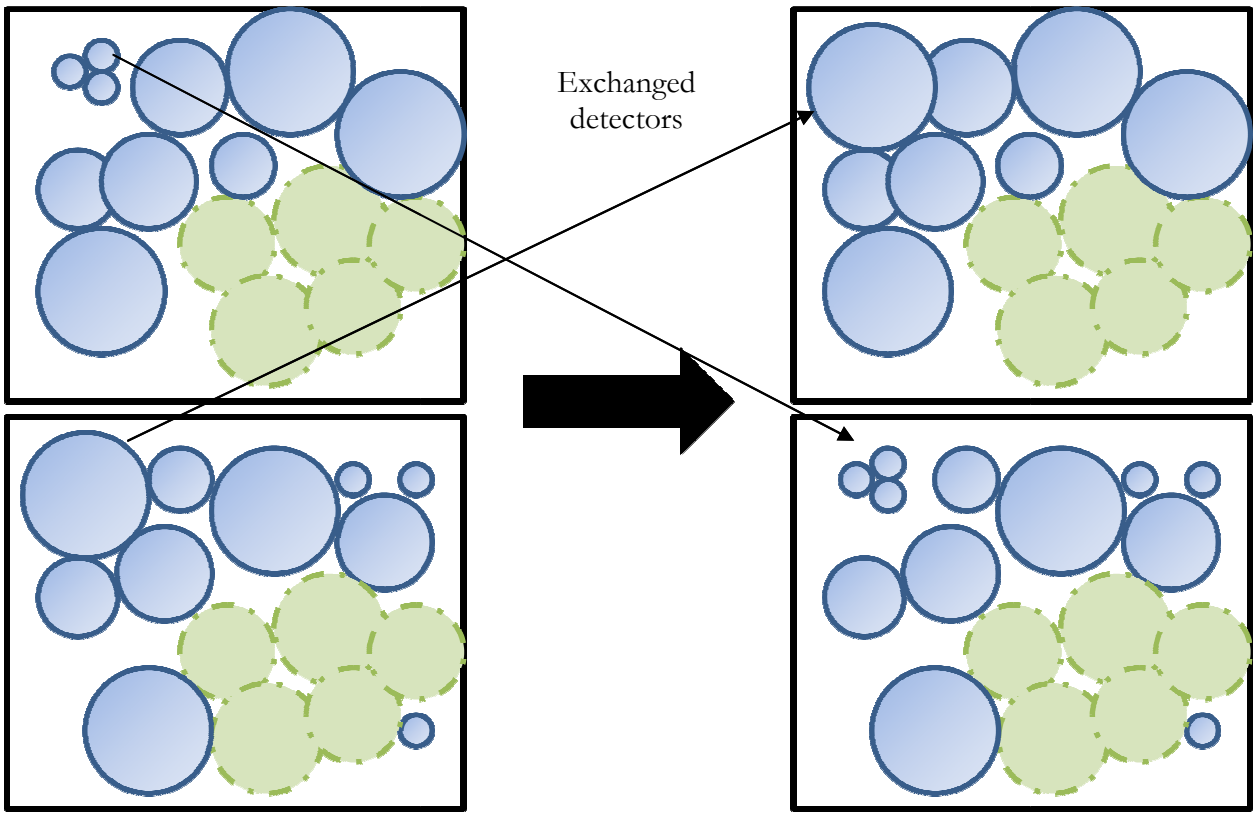

Figure 4.8—Diagram of the Crossover Genetic Operator (53)

4.4.3.3 Gene Addition

The gene addition genetic operator 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. A number of new points are generated randomly in the solution space. These are tested to determine which will be valid new detectors, and a number of them are added to the detector set. Since new detectors are added at the same time, they are not permitted to overlap existing detectors or the self, but new detectors may overlap each other. In practice however, this happens rarely. Note that the maximum number of detectors specified in the detector generation algorithm is absolute. Therefore, in order to be able to add new detectors, space must be available in the detector array to store new detectors. The user also specifies the maximum number of detectors to be added to a set at any given time, since adding more detectors may negatively affect the performance index of the individual. A flowchart of this genetic operator is given below in Figure 4.9.

Three variations exist for this operator. The user specifies weights to determine the likelihood of favoring larger detectors over smaller detector, and vice versa. Large detectors are preferable for covering the solution space with fewer detectors. Smaller detectors are better at the failure identification phase, where a smaller detector is less likely to be activated by more than one type of failure. An additional variation is available by giving both of these weights as 0. If the weights are 0, detectors will be added at random, without respect to the size of the new detectors.

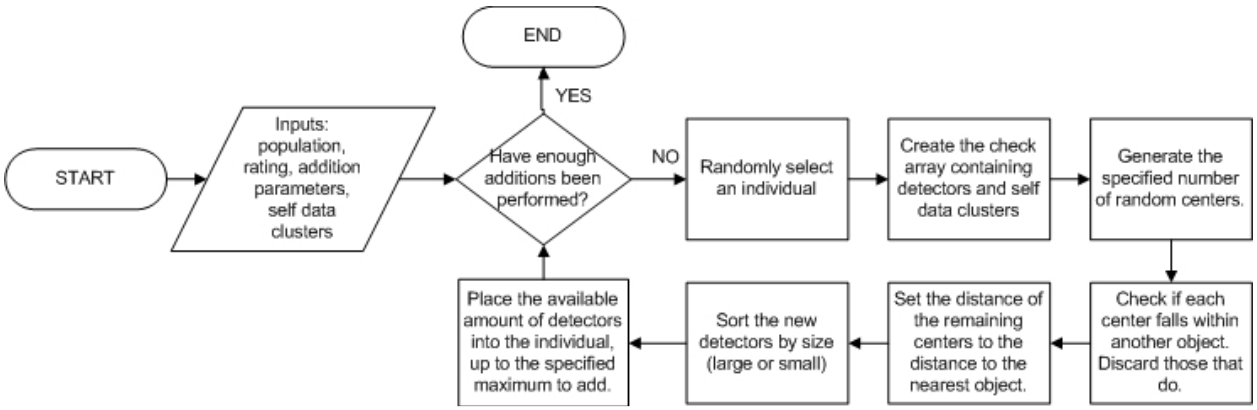

Figure 4.9—Flowchart of the Gene Addition Genetic Operator (53) (54)

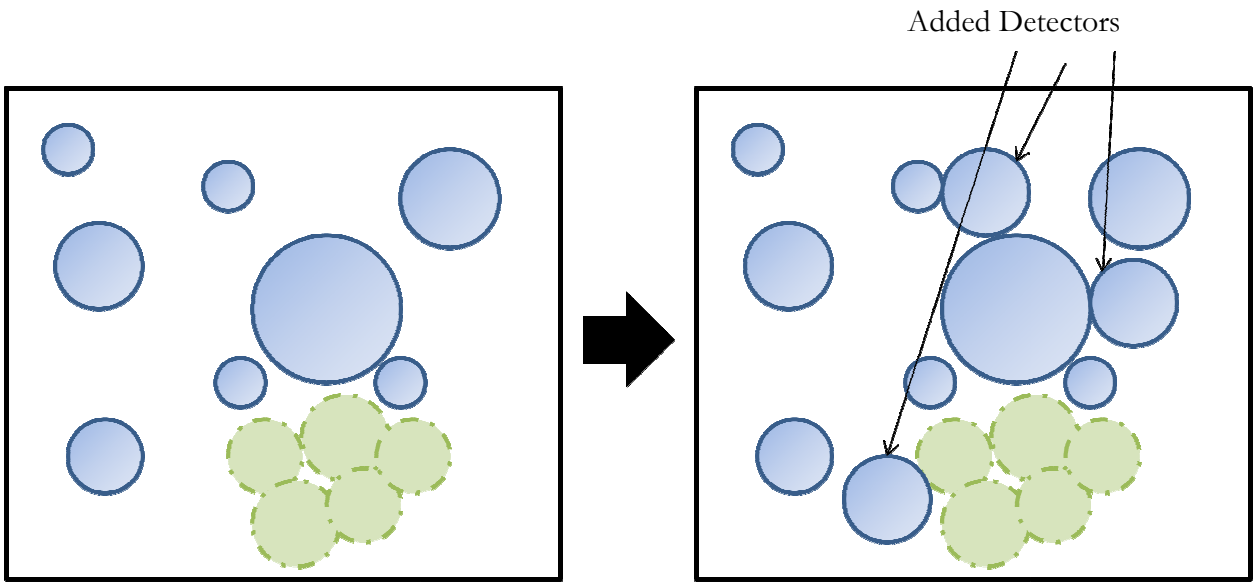

Figure 4.10—Diagram of the Gene Addition Genetic Operator (53)

4.4.3.3.1 Addition of Hyper-Spheres

Addition of hyper-spheres is the simplest case of gene addition. Since each detector is defined by only one radius, it is a simple matter to generate a number of points within the solution space, eliminate any and all that fall within existing entities, and assign a radius to each of the new centers with respect to the closest object. These new candidate detectors are then sorted according to their size and added to the detector set based on the number of available slots in the individual's detector array and the maximum number to be added as defined by the user. Size is favored based on the weights specified by the user. For a particular individual, all detectors added in a single instance of the addition operator will favor either large or small, or add randomly.

4.4.3.3.2 Addition of Hyper-Ellipsoids and Hyper-Rotational Ellipsoids

Addition of hyper-ellipsoids and hyper-rotational-ellipsoids is similar to adding hyper-spheres, in that the new detectors that are added will start as hyper-spheres. Like other detectors in the hyper-ellipsoid and hyper-rotational-ellipsoids individuals, new detectors will be capable of being altered and becoming literal hyper-ellipsoids or hyper-rotational-ellipsoids. Assigning the radius to the new centers for this operator is more difficult than for the hyper-spheres. It is not possible to tell in hyper-space the distance to a hyper-ellipsoid or hyper-rotational-ellipsoid, only whether a point falls within one. Therefore, the radius is assigned using an application of the bisection method, which is described above in Section 3.4.3. The accuracy required is specified by the user, such that the radius of the new detector is correct to this accuracy. The radius is assigned in such a way that overlapping is not permitted with the self or another detector.

Once the values have been assigned, the new detectors are sorted according to size and added to the detector set in the same manner as described for hyper-spheres. Note that random addition of detectors requires less calculation, and therefore significantly speeds up the FDGO algorithm.

4.4.3.3.3 Addition of Hyper-Rectangles

Addition of hyper-rectangles is the most complicated instance of the gene addition genetic operator. This is partially due to the nature of hyper-rectangles, and partially due to the fact that the hyper-rectangles are added with varying lengths in each dimension, unlike hyper-spheres, hyper-ellipsoids and hyper-rotational-ellipsoids, which are added with the same length for all dimensions. When determining whether an overlap has occurred for a hyper-rectangle, overlap can occur in all dimensions but one. If all dimensions overlap, then the rectangle exhibits overlap, however if even one dimension does not exhibit overlap, the hyper-rectangle is acceptable.

In order to add hyper-rectangles, random centers are generated in the same way as for the other cases. Once centers that fall within other objects have been eliminated, the side length for each dimension must be set. The side length to each object is calculated. Ideally, the rectangle should be assigned the largest possible side length in a dimension for which overlap does not occur. Determining the side length for each dimension, therefore, depends on the values in all other dimensions. For this reason, all possible combinations of the measured side lengths must be evaluated, until the largest acceptable combination is found. This is assigned to the particular candidate center. Since the side lengths for each dimension vary, the candidate hyper-rectangle detectors are ranked according to area, rather than the measurement of a given dimension. Area is found by multiplying together the side length of the rectangle in each dimension. The new detectors are then added according to the same method as given above.

4.4.3.4 Gene Removal

The gene removal genetic operator 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 detectors with the greatest overlap are removed. The number of detectors that may be removed from a single individual in a single instance of this genetic operator is specified by the user, along with a threshold which determines the amount of overlapping a detector must have before it will be removed. A flowchart of this simple genetic operator is shown in Figure 4.11, along with a diagram of the process in Figure 4.12. This genetic operator must be used with caution. Removing detectors may significantly decrease the coverage of the non-self.

This genetic operator is performed similarly for each of the different hyper-shapes. For hyper-spheres, overlap is calculated using equations 11, 12, and 13 above. Since a simple metric such as this does not exist for the other hyper-shapes, overlap among hyper-ellipsoids, hyper-rotational-ellipsoids, and hyper-rectangles is calculated using the Monte Carlo method.

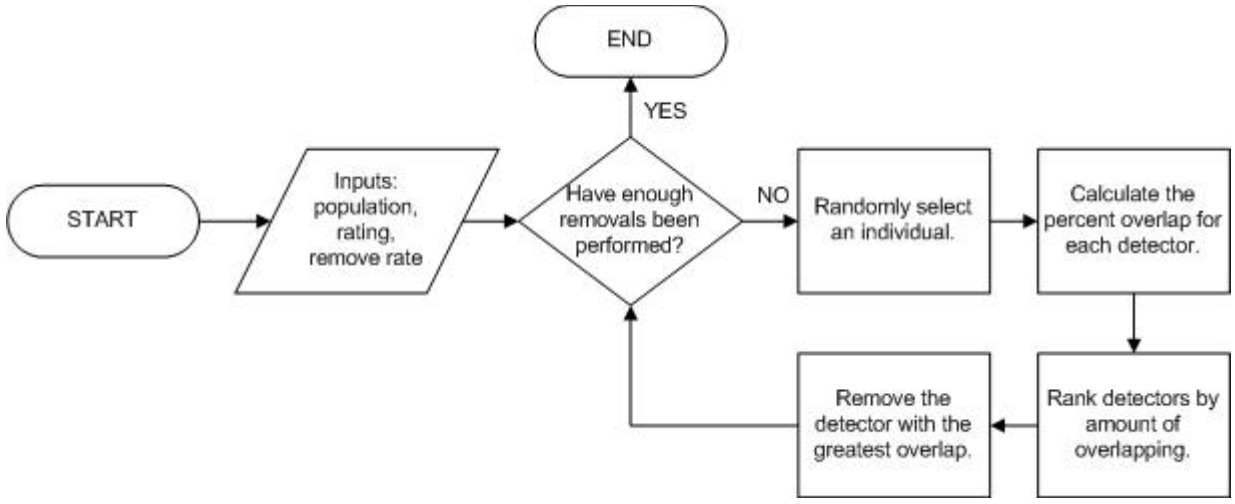

Figure 4.11—Flowchart of Detector Gene Removal Genetic Operator (53) (54)

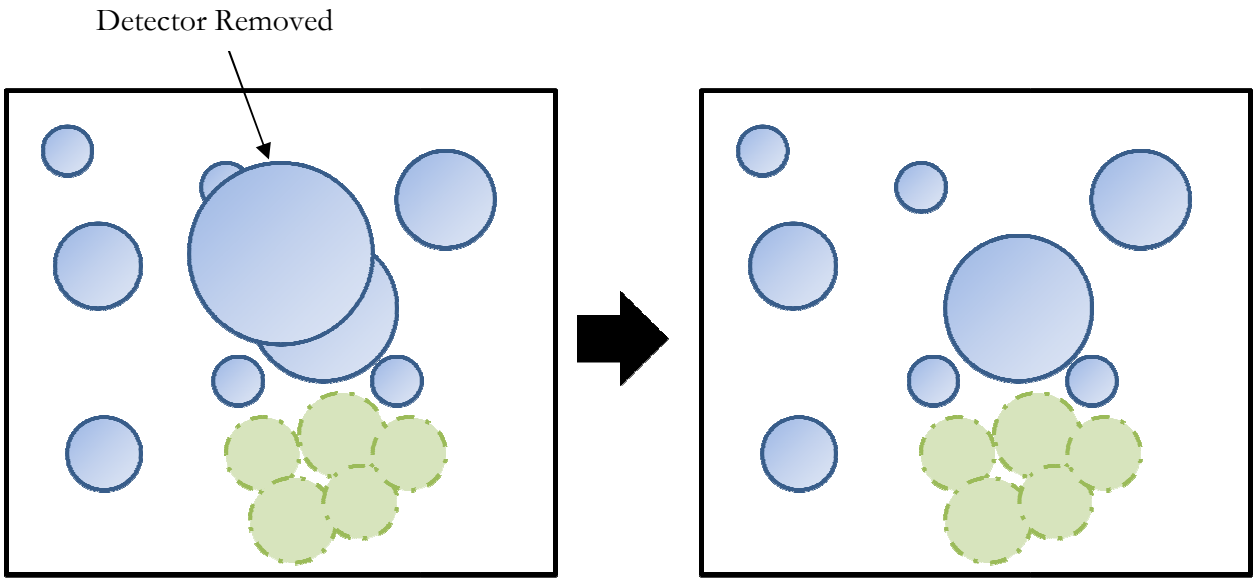

Figure 4.12—Diagram of the Gene Removal Genetic Operator (53)

4.4.4 Rating the Population

The "fitness" of the individuals is evaluated based on the following criteria:

a. number of detectors in the set
b. percentage of non-self that is covered by detectors
c. 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 21, 22, and 23.

$$PI_{coveragei} = \frac{1}{U_{coverage} - L_{coverage}} coverage_i - \frac{L_{coverage}}{U_{coverage} - L_{coverage}} \quad (21)$$

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

$$PI_{overlappingi} = \frac{1}{U_{overlapping} - L_{overlapping}} overlapping_i + \frac{U_{overlapping}}{U_{overlapping} - L_{overlapping}} + 1 \quad (22)$$

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

$$PI_{numberi} = \frac{1}{U_{number} - L_{number}} number_i - \frac{U_{number}}{U_{number} - L_{number}} + 1 \quad (23)$$

where $PI_{numberi}$ 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.

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 is 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.

4.4.5 Selecting the New Population

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 (64) is the method used in this application. Each individual in the population has a performance index $PI_{individual}$ computed as:

$$PI_{individual} = W_{overlap} * PI_{overlap} + W_{number} * PI_{number} + W_{coverage} * PI_{coverage} \quad (24)$$

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

$$TF = \sum_{i=1}^{N} PI_i \quad (25)$$

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:

$$p_i = \frac{PI\_individual}{TF} \quad (26)$$

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

$$q_i = \sum_{i=1}^{j} p_j \quad (27)$$

Since the population size is maintained constant throughout the algorithm, the population in each generation can only contain the same number of individuals as in the initial population. 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 4.13.

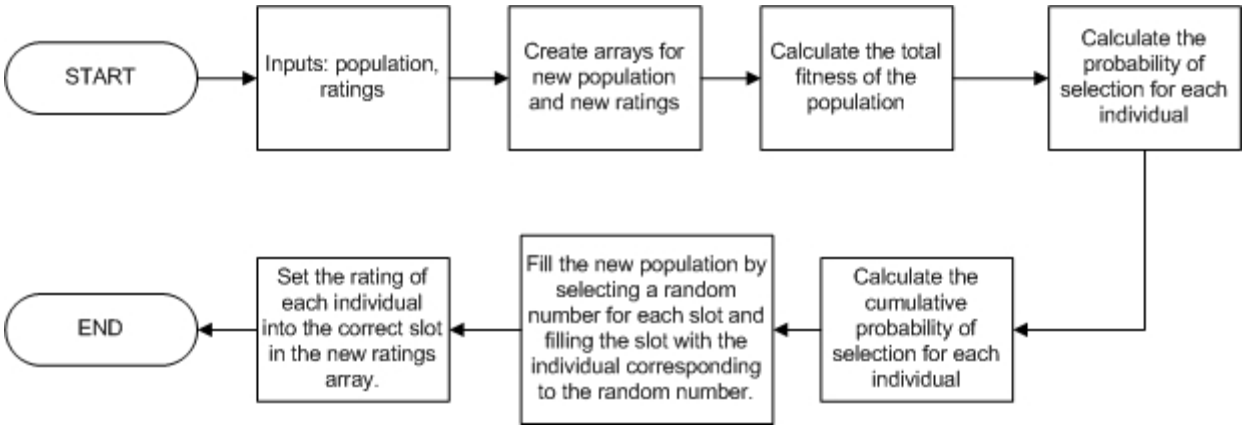

Figure 4.13—Flowchart of the Roulette Wheel Selection Algorithm (53) (54)

Elitist strategy is used after the new population has been generated to ensure that good solutions are not lost before a better solution is reached. In order to perform elitist strategy, one individual in the newly generated population is replaced at random with a copy of the best individual from the previous generation.

5 Description of Interactive Utility: West Virginia University Immunity-Based Failure Detector Optimization and Testing

5.1 Compatibility

This program was designed using MATLAB R2008a, and is intended to work with this version or newer. Some features that have been built into the program, such as multi-threading, may not be available when using previous versions of MATLAB, and could cause errors. This can be avoided by using the compiled Windows version of the program. If compatibility with Linux is desired, the MATLAB version of the program must be used in the Linux MATLAB environment.

5.2 Getting Started

This program is intended to perform a genetic algorithm to optimize detectors for aircraft failures. This description should instruct an unfamiliar user in the operation of this design tool for its intended purposes. It will step through each section from preparing and clustering data, to performing the genetic algorithm. It will describe all options in detail.

5.2.1 Accessing the Help Guide

A complete help guide is provided within the program, which may be accessed from the 'Help' menu. Click on the 'Help' menu button at the top of the figure, and select 'Load Help File', the only option. This will load an additional figure so that the user can simultaneously view the Help File and the IFDOT Utility. Below in Figure 5.1 is the opening menu of the IFDOT Utility.

5.2.2 Data Needs

In order to begin, the user will need at least one data file containing flight test data under normal conditions. Additional normal flight data and failure data will also be useful to this analysis. The IFDOT tool may also be applied to non-aircraft systems requiring failure detection. This will require collection of data regarding system values in order to define the self. Verification requires the ability to simulate a failure and collect failure data.

5.3 Processing Data

Several options are available for processing data. Begin by clicking on the menu labeled 'File', selecting 'Data Processing' then selecting 'Load Raw Data'. This will open the load file panel, as shown below in Figure 5.2. Before proceeding to process data, a data file must be loaded here.

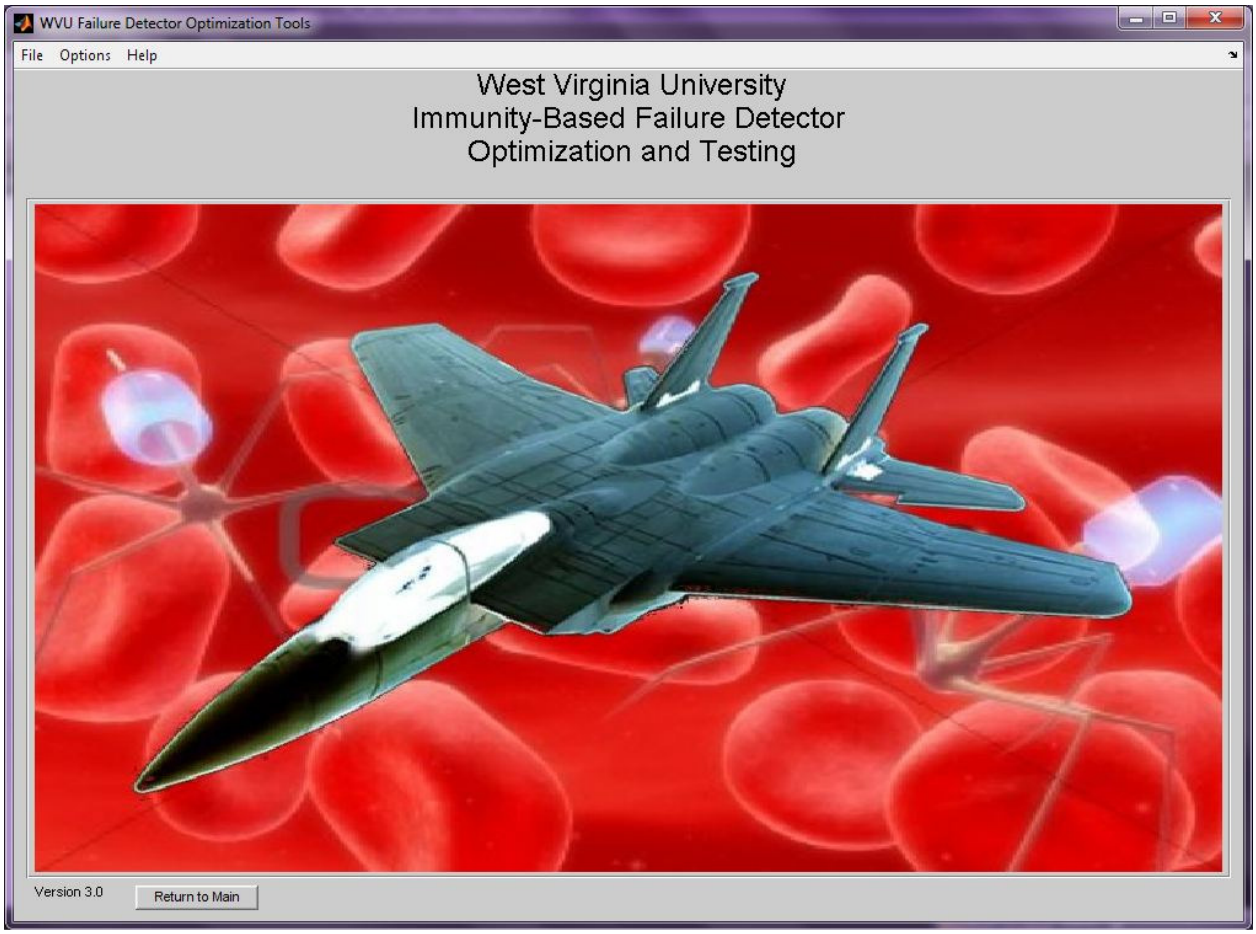

Figure 5.1—Opening Screen of the IFDOT Utility

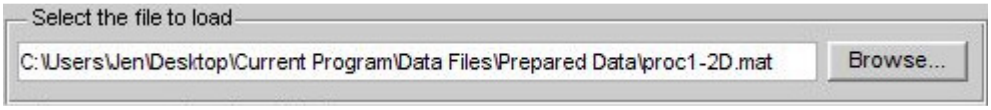

Figure 5.2—File Loading Panel

The raw data file should contain only the identifiers the user desires to use in the detector set, with each identifier in a column, and the data saved to the variable name 'sensors'. The most recently loaded raw data file will automatically be loaded as the default choice. This occurs for all data type. If no file has been loaded before or the data stored in the last known file path is invalid, the display box will remain blank until a valid file is chosen. When a file is chosen that is invalid, an error message will appear and continuing will be disabled until an appropriate file is loaded.

Once an appropriate file has been loaded, it will be possible to continue. Go to the 'File' menu, select 'Data Processing', and click 'Process Raw Data', which is now enabled. This will bring up the menu shown below in Figure 5.3.

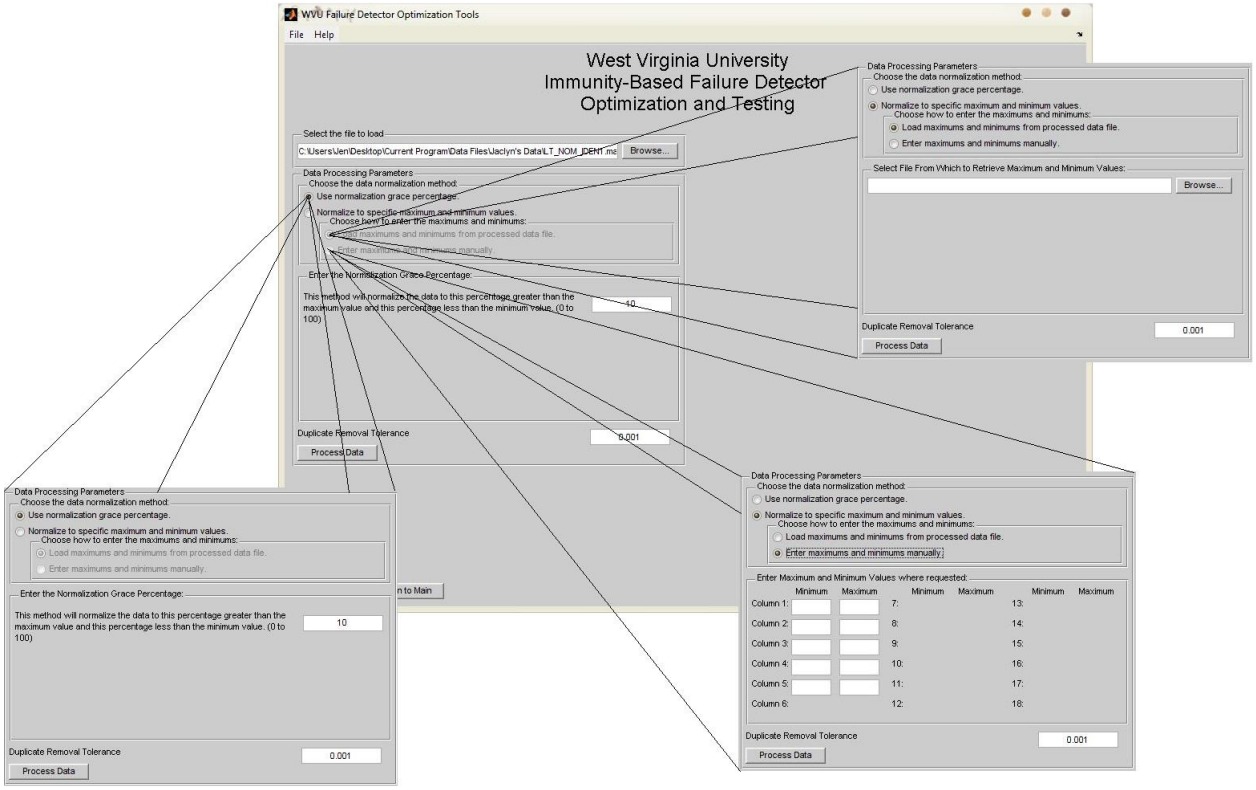

Figure 5.3—Data Processing Menu with Options

There are two options for normalizing data. The default method uses a percentage margin related to the values in the data. This means that there will be a margin the size of the percentage chosen left around the "edges" of the normalized data. This is the most basic method of normalization. The recommended percentage is 0.10 (10%). This method is shown below in Figure 5.4.

The second normalization method involves specifying the normalization limits for each identifier in the data. This is used when two sets of raw data need to be combined, for instance, to make a more complete self, or to prepare failure data for comparison with a detector set. In each of these cases, in order to obtain valid results, the data will need to be normalized using the same limits. The default method for this choice is to load a previously normalized set, and retrieve maximum and minimum values from this file. This is shown below in Figure 5.5. Note that the file loaded should contain the same dimensions as the file currently being normalized. If this is not the case, an error message will be displayed and continuing will not be possible until a correct file is chosen, or a different method is chosen.

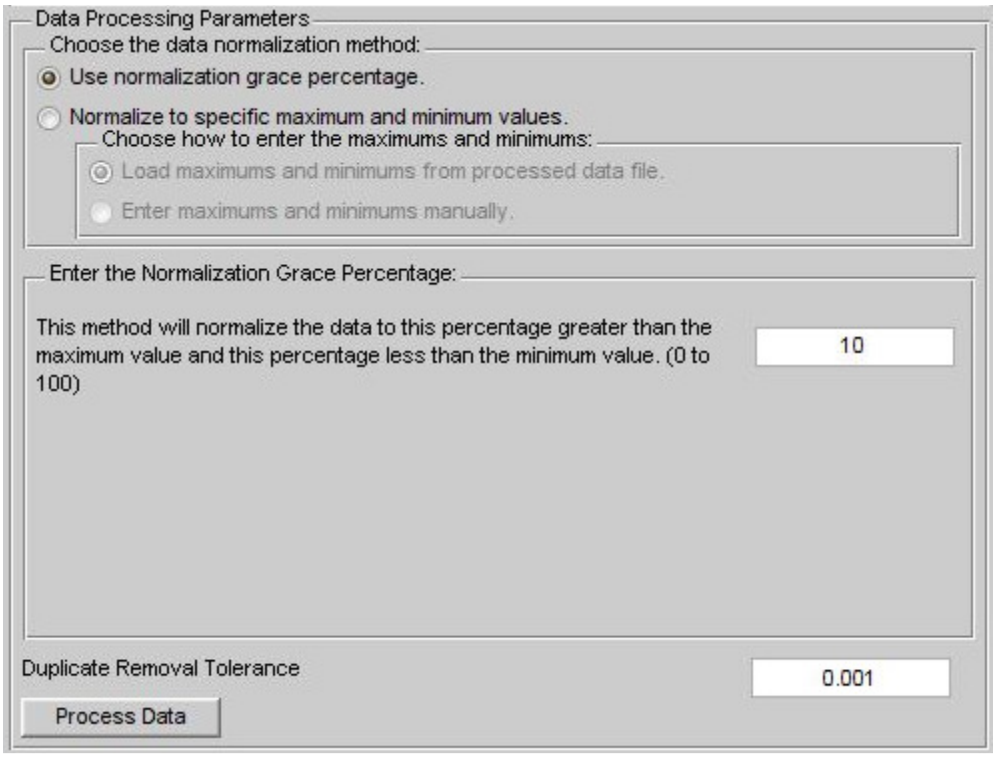

Figure 5.4—Processing Menu for Margin Normalization

The other possibility for normalizing data by specifying the normalization limits is to manually specify the limits. This is shown in Figure 5.6. This is the trickiest method to use, being that the normalization maximums and minimums must be specified including the grace percentage used in a different normalization file if they are to be combined validly. These values can be found by loading the desired normalized file separately in MATLAB and getting the values saved as 'normmaximums' and 'normminimums', then entering the values in the boxes. Otherwise, these values should be the maximum and minimum values the program should consider for each dimension.

Duplicates will be removed from the data during the process, in order to remove redundancy and speed up future algorithm processes. This is performed automatically after normalizing. The user will need to select a tolerance for this process. This tolerance is the maximum distance that may separate two points, which will be declared identical. The recommended value for the duplicate removal tolerance is 0.001 or smaller. Choosing a tolerance that is too large could lead to significant loss of data, and poor detection results. Regardless of the normalization type chosen, duplicate removal will be performed in the same manner.

Once all methods have been properly applied and all needed parameters appropriately selected, click on the 'Process Data' button to perform the processing on the data. While the data is processing, the program will be disabled. This process may take several minutes depending on the data. When the process is complete, the user will be prompted to save the file. This data file will now be ready for clustering.

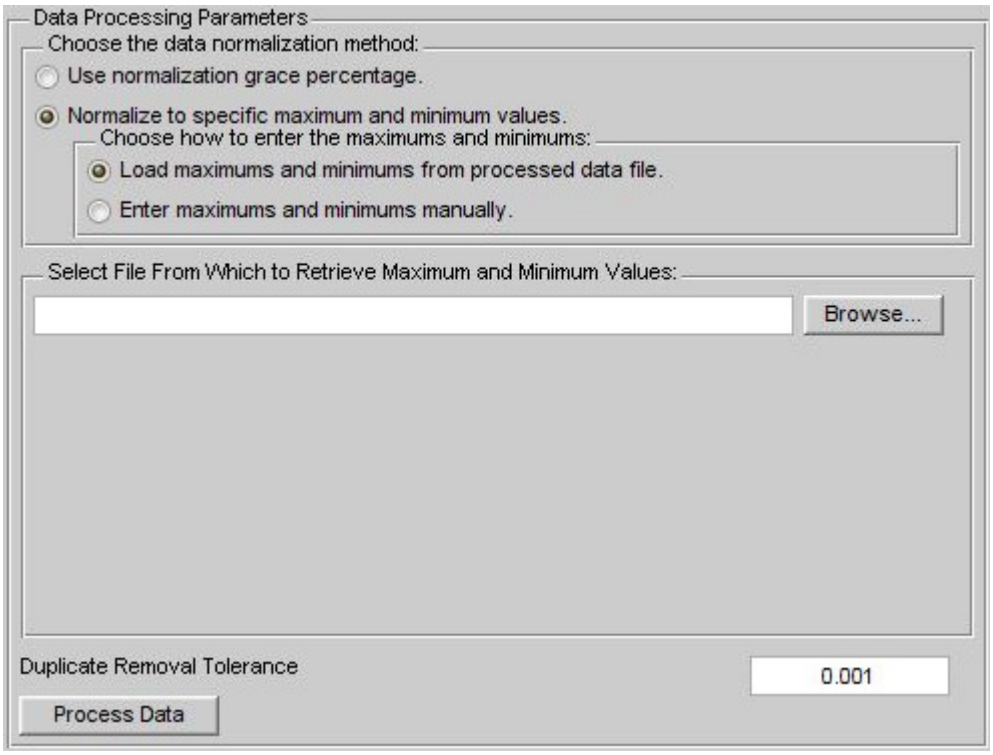

Figure 5.5—Data Processing Using File-Specified Normalization Limits

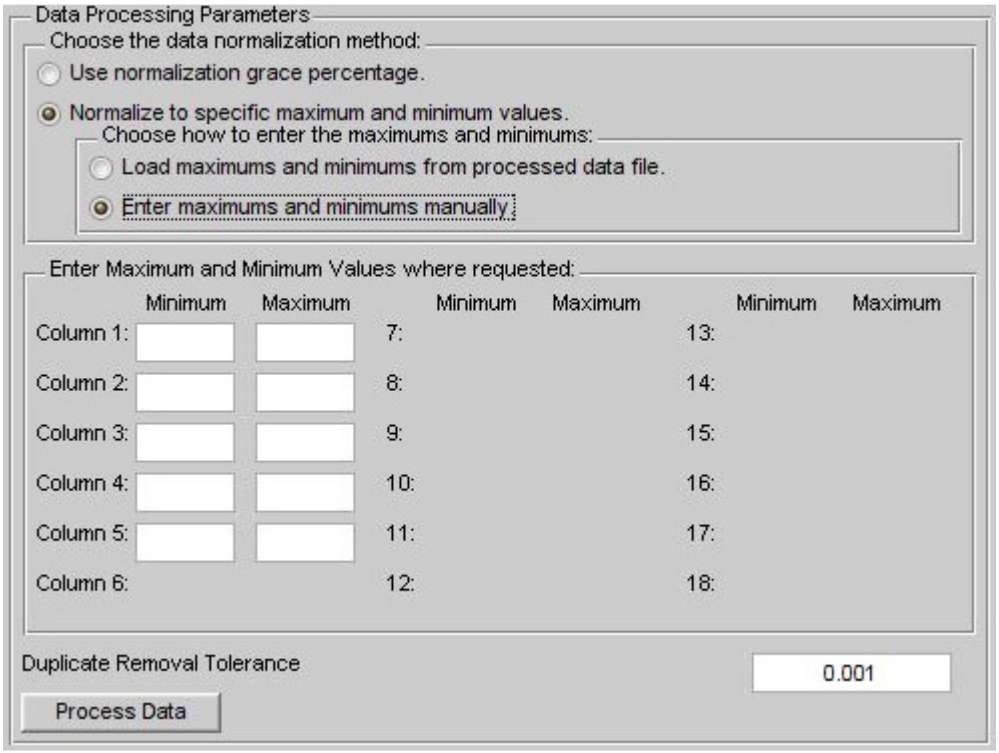

Figure 5.6—Data Processing Using Manually Specified Normalization Limits

5.4 Clustering Data

In order to begin clustering data, click on the 'File' menu, select 'Data Clustering', and click 'Load Processed Data'. As before, an appropriate file must be selected before the program can continue. This time the file must contain normalized data saved with the variable name 'selfdata'. This is the file type produced by the processing section of this program. Once an appropriate file has been chosen, click the 'File' menu, select 'Data Clustering' and click on 'Cluster Processed Data'.

Several options are available for clustering data. Data can be clustered using 2 shapes, sphere or rectangle. Spherical clusters are used for ellipsoid and rotational ellipsoid detectors as well as for spherical detectors. Hyper-rectangles can only be used in conjunction with hyper-rectangular detectors later. Hyper-spheres can be clustered using two different methods. Number-Imposed Clustering is generally the quicker option; however it does not limit the empty space within the clusters. Space-Optimized Clustering generally results in better clusters, but takes considerably longer to compute. For clustering using hyper-rectangles, only one method is available, which is nearly equivalent to the hyper-spheres Number-Imposed Clustering. Confidence percentage and permitted error are used to calculate the coverage and overlapping present in the clustered sets. All clustering methods will return the final number of clusters, their overlap and their coverage of the total hyper-cube. These values are useful to the user when determining values for the genetic algorithm.

The parameters that must be entered for the Number-Imposed Clustering Method are listed in Table 5.1 below, with an accompanying description of each of the parameters effects on the clustering algorithm. A figure of the interface for clustering is shown in Figure 5.7 below, followed by a figure of the Number-Imposed Clustering Menu in Figure 5.8.

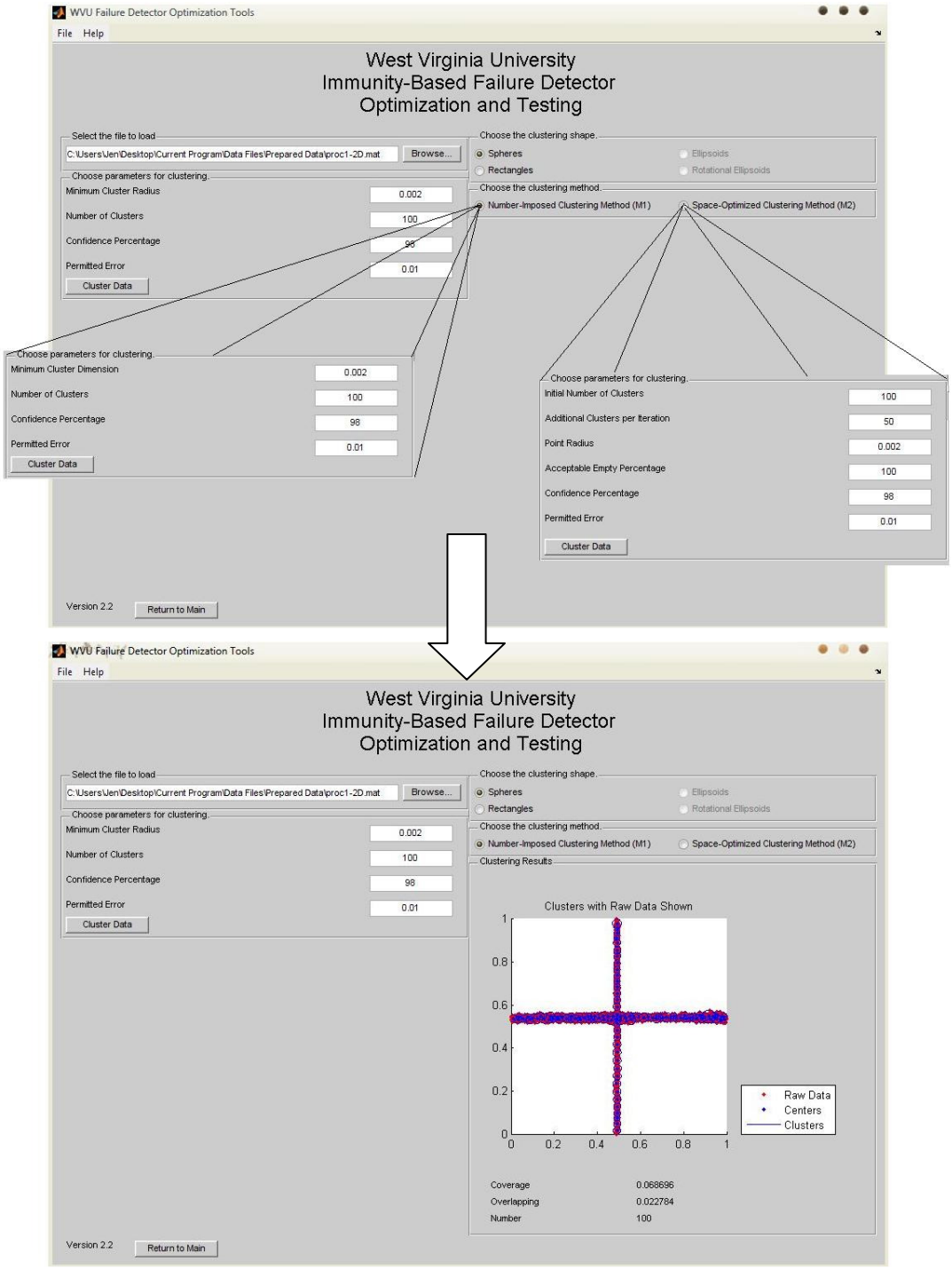

Figure 5.7—Clustering Method Menus

Table 5.1—Number-Imposed Cluster Method Parameters

Parameter	Description
Minimum Cluster Radius	This is the distance around each self point that can be reasonably assumed to belong to the self. One way of determining this for a data set is to look at the average distance between a data point and the next data point in the time history.
Maximum Number of Clusters	This is the maximum desired number of clusters. This limit will usually be met, making it effectively the desired number of clusters. A larger desired number of clusters leaves less "empty space" or space not covered by a data point (calculated using the radius above), while a smaller number of clusters leaves more empty space but likely incorporates parts of the self set for which explicit data does not exist. This value must be balanced in order to produce a self set that covers all possible normal condition occurrences, but does not incorporate failure occurrences. A self set that does not cover all normal occurrences will ultimately result in a detector set that produces a high number of false alarms, while covering abnormal conditions within the self will results in a detector set that produces lower detection rates.
Confidence Percentage	This value is used in the Monte Carlo volume estimation algorithm, which estimates the coverage and overlapping among the clusters, and later, among the detectors. This value is collected here only, and reused throughout the evolutionary algorithm. This value determines the confidence in the solution provided. The default value is 98, but 99 is recommended and a value closer to 100 will produce more reliable answer. Increasing this value also drastically increases the computation time needed for the Monte Carlo algorithm to run.
Permitted Error	This value is used in the Monte Carlo volume estimation algorithm, which estimates the coverage and overlapping among the clusters, and later, among the detectors. This value is collected here only, and reused throughout the evolutionary algorithm. This value determines the approximate accuracy of the solution returned by the Monte Carlo algorithm. The default for this value is 0.01, but 0.001 is recommended. Smaller values produce more accurate results, however, a smaller number will drastically increase the computation time of the Monte Carlo algorithm.

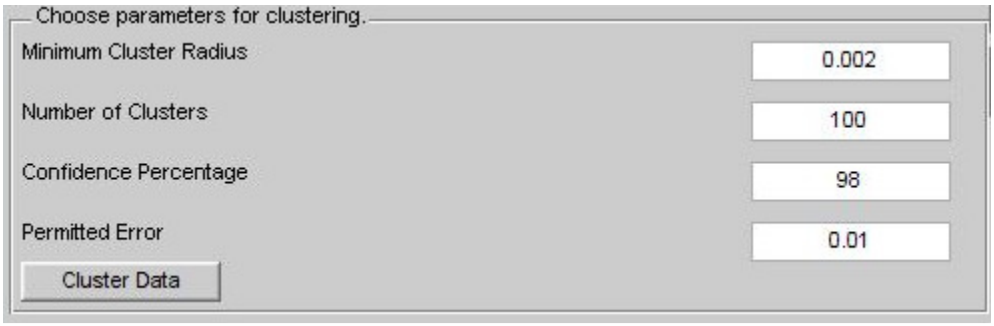

Figure 5.8—Number-Imposed Clustering Method

Clustering Method 2 is the more complicated and more time-consuming of the clustering methods. This method also limits the empty space within the clusters, in addition to generating the self clusters set. Several inputs are needed. These are described below in Table 5.2 in detail. The menu for this method is shown in Figure 5.9.

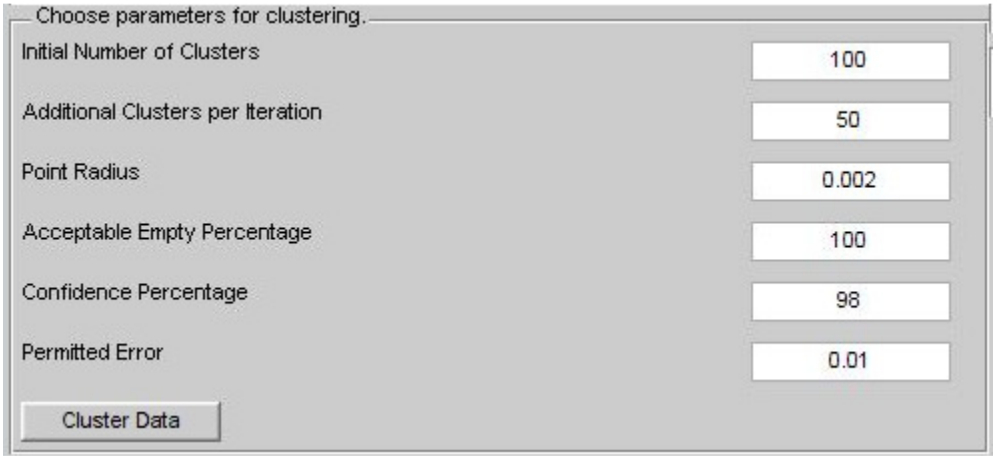

Figure 5.9—Hyper-Spheres Space-Limiting Clustering Method

Shown in Table 5.3 below are the parameters needed for the Rectangle Clustering Method, including a detailed description. Figure 5.10 below shows the utility menu for this method.

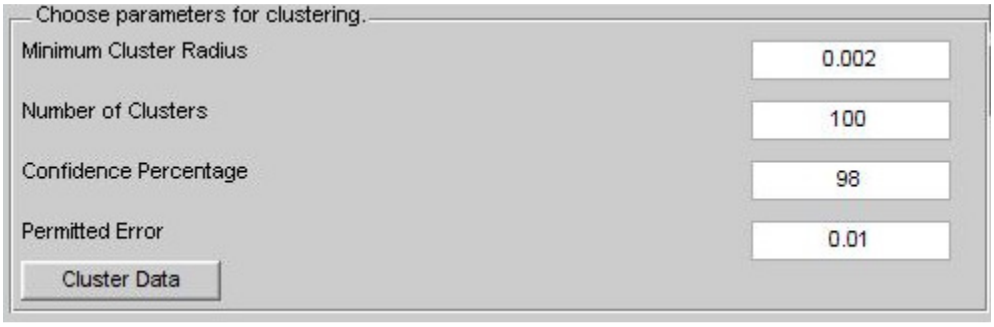

Figure 5.10—Hyper-Rectangles Clustering Method

Once these have been selected, click the 'Cluster Data' button to begin the clustering process. When the clustering begins a progress bar will appear. Once the clustering is complete, the user will be prompted to save the file.

If two files need to be merged, this can be done by clicking the 'File' menu, selecting 'Data Clustering', and clicking on either 'Merge Processed Data' or 'Merge Clustered Data'. This will open two file loading panels. Select appropriate data files with the same number of dimensions, to continue. Then click the 'Merge Processed Data' buttons or 'Merge Processed Data'. When this process is complete, the user will be prompted to save the new file.

Table 5.2—Space-Limiting Clustering Method

Parameter	Description
Initial Number of Clusters	This acts as a minimum desired number of clusters. The algorithm will begin by generating this number of clusters, and determine whether the desired empty percentage is met.
Cluster Increase Step	In each iteration in which the empty percentage is not obtained, a new set of clusters will be generated, with more clusters than the previous set. This is the number of clusters by which to increase the size of the set at each iteration.
Point Radius	This is the distance around each self point that can be reasonably assumed to belong to the self. One way of determining this for a data set is to look at the average distance between a data point and the next data point in the time history. This is used to determine the occupied space within a cluster. This is compared to the total size of the cluster to determine the empty percentage.
Empty Percentage	This is the desired maximum percentage of empty space within each detector. A value of 100 percent generates will not limit the empty space within the clusters.
Confidence Percentage	This value is used in the Monte Carlo volume estimation algorithm, which estimates the coverage and overlapping among the clusters, and later, among the detectors. This value is collected here only, and reused throughout the evolutionary algorithm. This value determines the confidence in the solution provided. The default value is 98, but 99 is recommended and a value closer to 100 will produce more reliable answer. Increasing this value also drastically increases the computation time needed for the Monte Carlo algorithm to run.
Permitted Error	This value is used in the Monte Carlo volume estimation algorithm, which estimates the coverage and overlapping among the clusters, and later, among the detectors. This value is collected here only, and reused throughout the evolutionary algorithm. This value determines the approximate accuracy of the solution returned by the Monte Carlo algorithm. The default for this value is 0.01, but 0.001 is recommended. Smaller values produce more accurate results, however, a smaller number will drastically increase the computation time of the Monte Carlo algorithm.

Table 5.3—Rectangle Clustering Method

Parameter	Description
Minimum Cluster Dimension	This is the distance in each dimension from each self point that can be reasonably assumed to belong to the self. One way of determining this for a data set is to look at the average distance between a data point and the next data point in the time history, in each of the corresponding dimensions.
Maximum Number of Clusters	This is the maximum desired number of clusters. This limit will usually be met, making it effectively the desired number of clusters. A larger desired number of clusters leaves less "empty space" or space not covered by a data point (calculated using the radius above), while a smaller number of clusters leaves more empty space but likely incorporates parts of the self set for which explicit data does not exist. This value must be balanced in order to produce a self set that covers all possible normal condition occurrences, but does not incorporate failure occurrences. A self set that does not cover all normal occurrences will ultimately result in a detector set that produces a high number of false alarms, while covering abnormal conditions within the self will results in a detector set that produces lower detection rates.
Confidence Percentage	This value is used in the Monte Carlo volume estimation algorithm, which estimates the coverage and overlapping among the clusters, and later, among the detectors. This value is collected here only, and reused throughout the evolutionary algorithm. This value determines the confidence in the solution provided. The default value is 98, but 99 is recommended and a value closer to 100 will produce more reliable answer. Increasing this value also drastically increases the computation time needed for the Monte Carlo algorithm to run.
Permitted Error	This value is used in the Monte Carlo volume estimation algorithm, which estimates the coverage and overlapping among the clusters, and later, among the detectors. This value is collected here only, and reused throughout the evolutionary algorithm. This value determines the approximate accuracy of the solution returned by the Monte Carlo algorithm. The default for this value is 0.01, but 0.001 is recommended. Smaller values produce more accurate results, however, a smaller number will drastically increase the computation time of the Monte Carlo algorithm.

5.5 Generating Detectors and Performing Optimization

This section is intended to cover performing the genetic algorithm optimization. This segment of the program is equipped with a multi-threading feature, due to the computationally-intense nature of this algorithm. The program makes use of the maximum number of cores available by default. If multiple-cores are not available, the effects of this feature will be null. If multiple cores are available and multithreading is not desired, this feature may be turn off by clicking

on the Options menu, deselecting 'Use Multithreading Where Applicable', and clicking 'Save'. Parallel computation is also implemented for increasing the computation speed of the algorithm. This is discussed in Section 5.7.2.

To begin the genetic algorithm, first click on the 'File' menu, select 'Detector Optimization', then select the 'Negative Selection' option, and click on 'Load Clustered Data'. This will bring up the load file panel as in previous tasks. Select an appropriate clustered data file to continue.

Next, click on the 'File' menu, select 'Detector Optimization', then select over 'Negative Selection' and click on 'Perform Optimization'. This will load the menu shown below in Figure 5.11. This is the main menu for the genetic algorithm. Several parameters must be selected here in order to perform the genetic algorithm. The genetic algorithm may be performed using several different detector shapes. These are hyper-spheres, hyper-ellipsoids, hyper-rotational ellipsoids, and hyper-rectangles. Note that the clustered data file will determine which of these parameters is available. If the data was clustered using hyper-spherical clusters, hyper-spheres, hyper-ellipsoids, and hyper-rotational-ellipsoids will be available. If the data was clustered using hyper-rectangle clusters, only hyper-rectangles will be available. If the clusters were not created using this program, it will be up to the user to select an appropriate shape. Selecting an incorrect shape will cause errors.

If hyper-ellipsoid or hyper-rotational-ellipsoid detectors are chosen, the mutation parameters will be different from above and will appear as shown above in the option on the lower right in Figure 5.11. In addition, if Enhanced NSA-R with Variable Detectors is chosen, the detector generation parameters will look as shown in Figure 5.11 on the lower left.

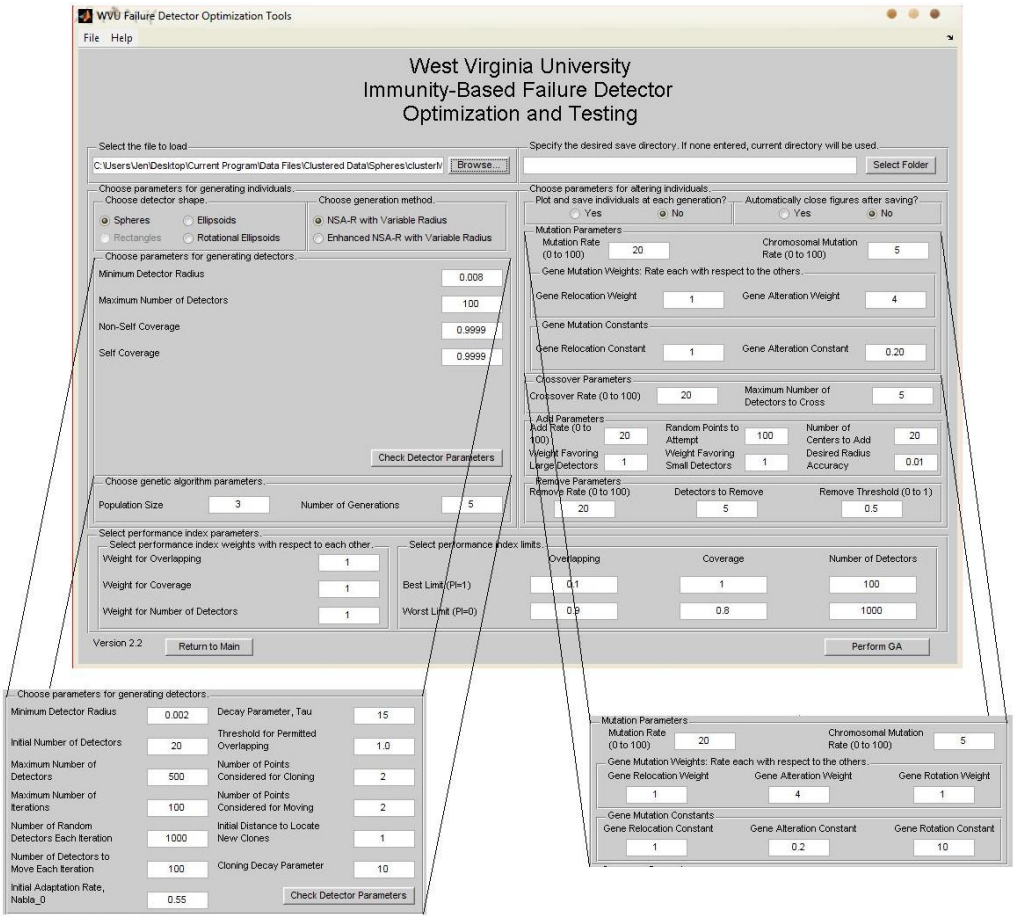

Figure 5.11—Detector Optimization Main Menu with Algorithm Options

The parameters for each method will now be discussed, beginning with Phase I, detector generation. Three methods exist for detector generation. Two different methods are capable of producing spherical detectors. Another method is intended to produce rectangular detectors. The simple method of generating hyper-spheres, called NSA-R with Variable Detectors, contains four input parameters. These are discussed in Table 5.4 below. The menu for this method is shown in Figure 5.12.

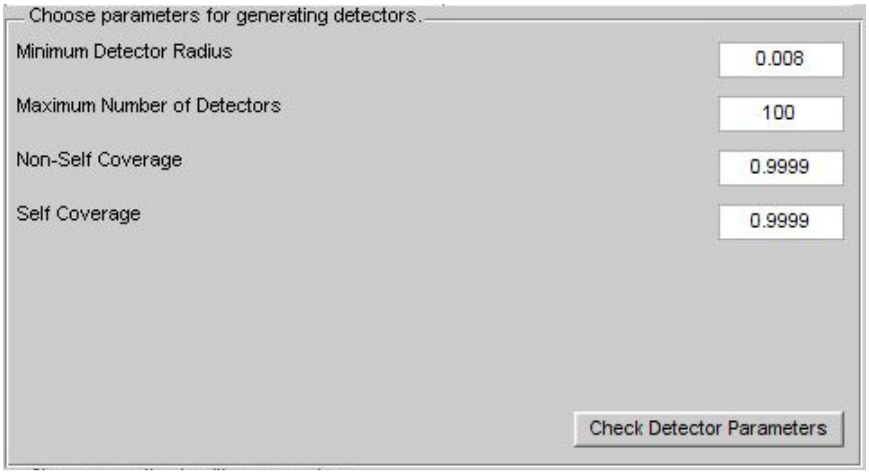

Figure 5.12—NSA-R Detector Generation Method

Table 5.4—NSA-R Detector Generation Parameters for Hyper-Spheres

Parameter	Description
Minimum Detector Radius	Places a lower limit on the size of the detectors.
Maximum Number of Detectors	Specifies the maximum desired number of detectors in an individual. This also specifies the maximum number of detectors that can be in any individual at any time throughout the algorithm. This is used as a stopping criterion in the detector generation algorithm.
Non-Self Coverage	An approximate coverage determined based on the number of centers attempted that have fallen within existing objects. This is used as a stopping criterion in the detector generation algorithm. This value should approach 1, although increasing this value can significantly increase computing time.
Self-Coverage	An approximate coverage determined based on the number of attempted new detectors that had radii smaller than the desired radius. This is used as a stopping criterion for the detector generation algorithm. This value should approach 1, although increasing this value can significantly increase computing time.

The parameters for generating rectangular detectors are similar to the previous method for generating hyper-spheres, with two exceptions. Where the hyper-spheres method requests a minimum radius, the hyper-rectangles method requests a minimum semi-side length. This dimension is the minimum distance from the center to the edge of a hyper-rectangular detector in

any one dimension. Also, a decay parameter is needed for changing the size of the detectors while they are being generated. These parameters are shown in Table 5.5.

Table 5.5—Generation of Rectangle Detectors Parameters

Parameter	Description
Minimum Detector Dimension	Places a lower limit on the size of the detectors. This dimension is measured from the center to the edge of the detector in each dimension.
Maximum Number of Detectors	Specifies the maximum desired number of detectors in an individual. This also specifies the maximum number of detectors that can be in any individual at any time throughout the algorithm. This is used as a stopping criterion in the detector generation algorithm.
Non-Self Coverage	An approximate coverage determined based on the number of centers attempted that have fallen within existing objects. This is used as a stopping criterion in the detector generation algorithm.
Tau	An decay parameter used to determine the size of detectors as they are generated. This value should increase for higher numbers of detectors, from approximately 135 for 200 detectors to 150 for 500 detectors.

Thirteen parameters are required for the second detector generation method for hyper-spheres, called Enhanced NSA-R with Variable Radius. These are described in Table 5.6 below, followed by Figure 5.13 of the menu for this method.

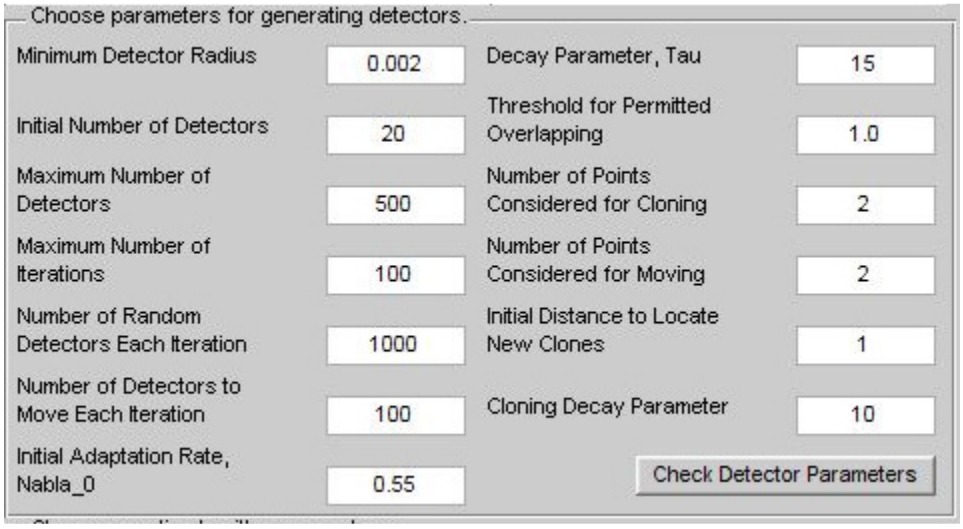

Figure 5.13—Enhanced NSA-R Detector Generation Method

Table 5.6—Enhanced NSA-R Parameters for Hyper-Spheres

Parameter	Description
Minimum Detector Radius	Places a lower limit on the size of the detectors.
Initial Number of Detectors	The initial number of detectors generated by the algorithm.
Maximum Number of Detectors	A stopping criterion which specifies the largest number of detectors that may be generated for an individual. This also specifies the maximum number of detectors that can be in any individual at any time throughout the algorithm.
Maximum Number of Iterations	A stopping criterion used to determine the largest number of iterations which may be performed in order to produce the desired number of acceptable detectors.
Number of Random Detectors Each Iteration	This specifies the number of new candidate centers that are generated at random each iteration, with the purpose of generating new acceptable detectors.
Number of Detectors to Move Each Iteration	Each iteration, a number of unacceptable detectors are moved in an attempt to make them acceptable.
Initial Adaptation Rate	Specifies the maximum distance a detector may be moved.
Decay Parameters, Tau	Specifies how many times and how far a detector may be moved before it is rejected.
Threshold for Permitted Overlapping	Specifies the amount of overlapping a detector may exhibit and be acceptable.
Number of Points Considered for Cloning	Number of acceptable detectors considered for creating clone detectors.
Number of Points Considered for Moving	The number of nearest points considered when moving a detector.
Initial Distance to Locate New Clones	Clones are initially generated a specified distance from the original.
Cloning Decay Parameter	Determines how many times and how far a clone may be moved before it is rejected.

Phase II requires a considerable number of 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 parameters.

The performance index parameters consist of 9 values. These are described in Table 5.7 below, followed by Figure 5.14 of the menu for the performance index. For the performance index, three weights must be entered to determine the weights of the three grading criteria. These should be chosen with respect to each other. This means that if all three have the same weight, they will be equally weighted. However, for instance, if coverage has a weight of 2 and number and overlap have a weight of 1, the performance index will be composed 50% from the coverage rating, and 25% each from the number and overlap ratings.

Table 5.7—Performance Index Parameters

Parameter	Description
Weight for Overlapping	This weight specifies the relative importance of overlapping with respect to the other performance index criteria.
Weight for Coverage	This weight specifies the relative importance of coverage with respect to the other performance index criteria.
Weight for Number of Detectors	This weight specifies the relative importance of number of detectors with respect to the other performance index criteria.
Best Limit for Overlapping	Expected most desirable value for overlapping, ideally should approach zero.
Worst Limit for Overlapping	Expected least desirable value for overlapping, ideally should approach one.
Best Limit for Coverage	Expected most desirable value for coverage, ideally should approach one.
Worst Limit for Coverage	Expected least desirable value for coverage, ideally should approach zero.
Best Limit for Number of Detectors	Expected most desirable value for number of detectors, ideally should be small with respect to coverage achieved.
Worst Limit for Number of Detectors	Expected least desirable value for number of detectors, ideally should choose the largest number of detectors allowable in the detector set.

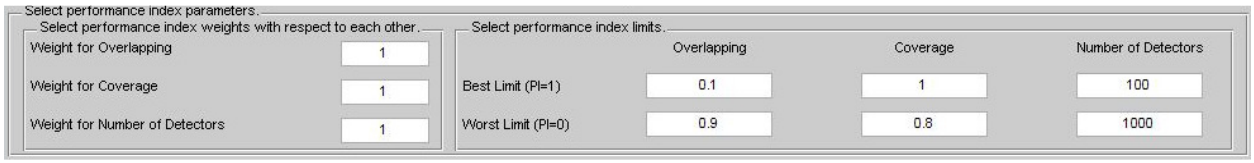

Figure 5.14—Performance Index Parameters

The crossover parameters consist of only 2 values. The crossover rate is the number of individuals with respect to the size of the population that should undergo crossover in each generation. The number of detectors to cross is the maximum number of detectors that can be traded between two sets of detectors in a single crossover instance. An example of the menu for crossover parameters is given in Figure 5.15 below.

Figure 5.15—Crossover Parameters

The addition parameters consist of 6 values. These are described in Table 5.8 below, followed by Figure 5.16 of the menu for this genetic operator.

Table 5.8—Gene Addition Parameters

Parameter	Description
Add Rate	Probability of selection for an individual to undergo this genetic operator in a specific generation.
Number of Random Points	The number of random centers generated in effort of finding new acceptable detectors.
Number of Centers to Add	The maximum number of new detectors that may be added to an individual during one instance of the genetic operator.
Weight Favoring Large Detectors	Weighting factor used to determine probability of adding large detectors rather than small detectors.
Weight Favoring Large Detectors	Weighting factor used to determine probability of adding small detectors rather than large detectors.
Desired Radius Accuracy	Desired accuracy achieved by the bisection algorithm used to determine the radius of newly added hyper-ellipsoids or hyper-rotational-ellipsoids (only applies to hyper-ellipsoids and hyper-rotational-ellipsoids). Smaller desired accuracy may result in increased calculation time.

Selecting both of the weighting factors as 0 will cause the algorithm to choose new detectors randomly, rather than according to the size. This significantly increases the calculation speed of the algorithm, and should be used in most situations.

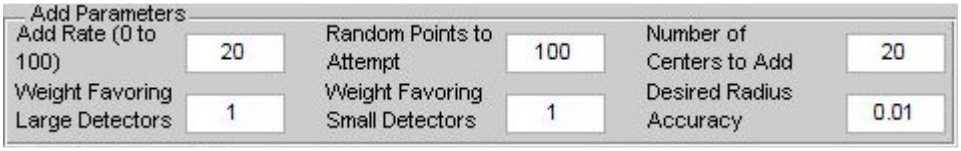

Figure 5.16—Gene Addition Parameters

The remove parameters consist of 3 values. Remove rate refers to the probability that an individual will undergo this operation in a single generation. Maximum number to remove is the maximum number of detectors that can be removed from a single individual in a given generation. Overlapping threshold is the percentage area of a detector which determines whether a detector exhibits enough overlap that it should be removed. This eliminates the possibility of removing desirable detectors and needlessly reducing coverage. Figure 5.17 shows the parameters menu for this operator.

Figure 5.17—Gene Removal Parameters

The genetic algorithm properties consist of 2 values. Population size is the number of individuals that will make up the entire population. Number of generations is the number of iterations the algorithm should perform before returning the results. Figure 5.18 shows this menu.

Figure 5.18—Genetic Algorithm Parameters

Mutation parameters, the trickiest, depend on the shape chosen. For hyper-spheres, the mutation parameters will consist of 6 parameters. For hyper-ellipsoids, 8 values make up the mutation parameters. The parameters for each of these options are given in Table 5.9 and Table 5.10 below, respectively. The menus accompanying these methods are included in Figure 5.19 and Figure 5.20 below, respectively.

Gene relocation weight and gene alteration weight are weights that work the same as the performance index weights, to determine the likelihood of a particular type of mutation occurring. Gene relocation is the moving of the center of a detector. Gene alteration is the changing of a detector's radius. For the case of hyper-ellipsoids and hyper-rotational-ellipsoids, an additional mutation type exists, called gene rotation. Gene rotation is the rotation of a detector about a certain axis. For each of the mutation types, a constant is requested to determine the maximum amount of alteration the detector can undergo at once. The gene relocation constant is the distance in multiples of the detector radius the center can be moved in one direction at a time. The gene alteration constant is the distance in multiples of the detector radius that the radius can be changed by at one time. The gene rotation constant, which is only requested for hyper-ellipsoids or hyper-rotational-ellipsoids, is the maximum number of degrees a detector can be rotated at one time.

Table 5.9—Mutation Parameters for Hyper-Spheres and Hyper-Rectangles

Parameter	Description
Mutation Rate	The probability of selection for a specific individual in a given generation.
Chromosomal Mutation Rate	The probability of selection for a specific detector within a selected individual in a given generation.
Gene Relocation Weight	A weighting factor used to determine the likelihood of performing gene relocation.
Gene Alteration Weight	A weighting factor used to determine the likelihood of performing gene alteration.
Gene Relocation Constant	A value used, with the radius of the detector, to determine the maximum distance the center can be moved at one time.
Gene Alteration Constant	A value used, with the radius of the detector, to determine the maximum length the radius can be changed at one time.

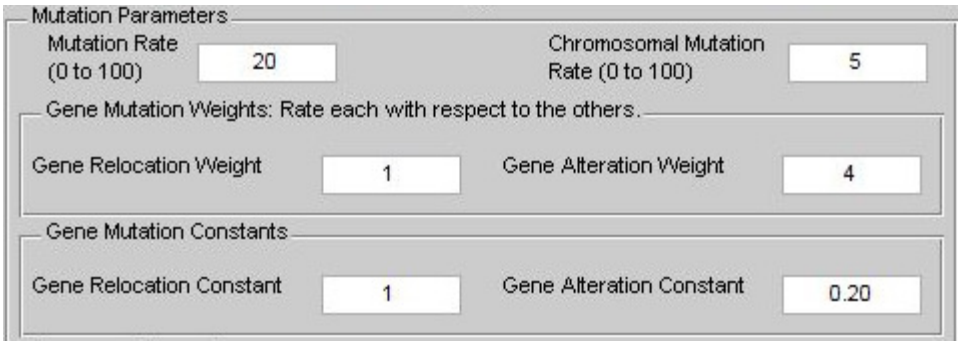

Figure 5.19—Mutation Parameters for Hyper-Spheres and Hyper-Rectangles

Table 5.10—Mutation Parameters for Hyper-Ellipsoids and Hyper-Rotational-Ellipsoids

Parameter	Description
Mutation Rate	The probability of selection for a specific individual in a given generation.
Chromosomal Mutation Rate	The probability of selection for a specific detector within a selected individual in a given generation.
Gene Relocation Weight	A weighting factor used to determine the likelihood of performing gene relocation.
Gene Alteration Weight	A weighting factor used to determine the likelihood of performing gene alteration.
Gene Rotation Weight	A weighting factor used to determine the likelihood of performing gene rotation.
Gene Relocation Constant	A value used, with the radius of the detector, to determine the maximum distance the center can be moved at one time.
Gene Alteration Constant	A value used, with the radius of the detector, to determine the maximum length the radius can be changed at one time.
Gene Rotation Constant	The largest number of degrees a detector can be rotated in a plane at one time.

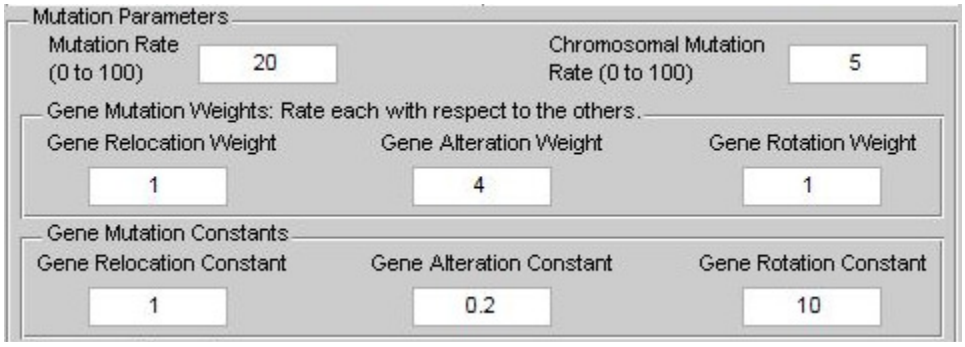

Figure 5.20—Mutation Parameters for Hyper-Ellipsoids and Hyper-Rotational-Ellipsoids

Once all of these many parameters are chosen, click the 'Perform GA' button. This will begin the genetic algorithm, and a progress bar will appear. Note that this may take a considerable amount of time, depending on the number of generations, population size, and computational

power utilized. When the algorithm is finished, the user will be prompted to save the results. Then the Results display will appear. If the data is 2 dimensional, the optimal detector will also be plotted.

5.6 Failure Testing

In order to perform detector testing, applicable failure data must be available. In order to be consistent with the detector set, the data must be measured/simulated from the same aircraft, contain the same identifiers as the detectors, and be normalized to the same limits as the cluster data. The user must ensure that these requirements are met.

To begin testing, click 'File' → 'Detector Testing' → 'Run Detection'. This will bring up the testing menu. This is shown below, with example results, in Figure 5.21. In the first browse box, select the file containing the detectors that are to be tested. These should be saved to variable name 'optdetector'. Beneath this, select the appropriate detector shape. In the second browse box, select the file containing the data for comparison. This should be saved with the variable name 'dataN'. Below, select whether the data contains failures or not. Then, select the sampling rate, activation window, time of failure, and size of point radius. The values of these parameters are discussed in Table 5.11 below.

Table 5.11—Detection Testing Parameters

Parameter	Description
Sampling Rate	The collection rate of the data in Hz, typically 50Hz.
Sampling Window	The number of contiguous samples that will be compared to determine if a failure has occurred, typically 50 samples, or 1 sec at 50 Hz.
Time of Failure	Time the failure was introduced during the data collection. This is needed to distinguish between detection rate and false alarms.
Point Radius	Determines the radius of a point, for comparison with the detectors.

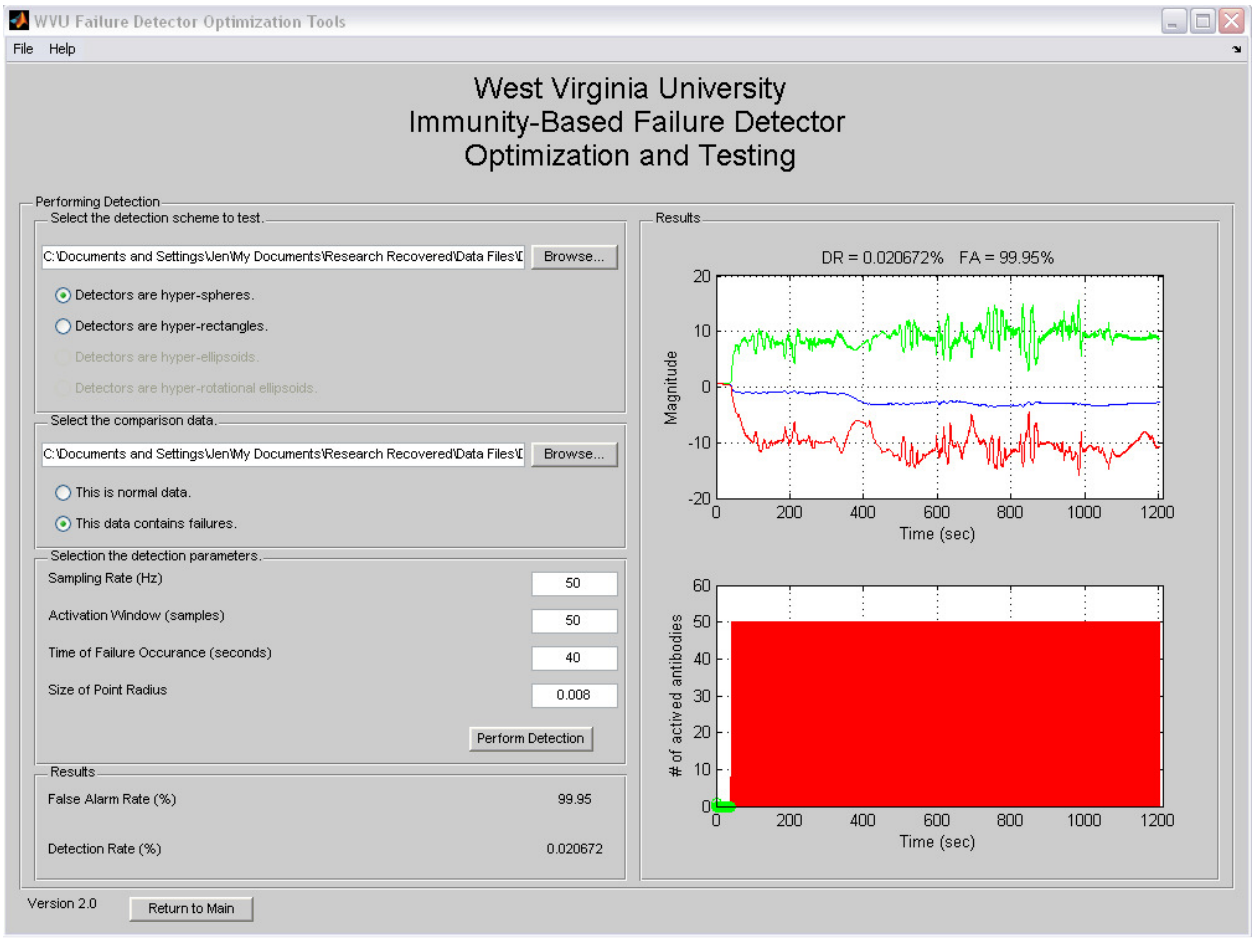

Figure 5.21—Testing Menu with Results

5.7 Continuing Optimization and Other Features

5.7.1 Continuing Optimization

Sometimes it is desirable to continue optimizing where the algorithm left off. Note that continuing with different performance index values than used in the previous optimization will result in inconsistencies if all generations (previous and continued) are plotted together.

Beginning from a previous trial is similar to running an initial trial of the genetic algorithm, except that the initial population is obtained from the previous trial rather than generated. Thus the population size is fixed and the detector generation parameters are not needed. To run the genetic algorithm continuing from a previous trial, click on the 'File' menu, select 'Detector Optimization', then select 'Negative Selection', and click 'Load Previous Trial Data'. This will cause the file loading panel to appear. Once an appropriate file has been loaded, the user may continue by clicking 'File', 'Detector Optimization', 'Negative Selection', and 'Continue Previous Optimization.' This will bring up a menu with all the necessary parameters. These are the same parameters as in the initial trial, except the ones not needed listed above. The default for these parameters will be set to the parameters used in the trial for which the data was loaded. Change these parameters as desired, and click 'Continue GA' to run the trial. When the algorithm is complete, the results will be displayed as before.

5.7.2 Options Menu and Parallel Computation

This menu allows the user to set some options that are static to the program. These include whether or not to use multi-threading capabilities when available, whether the files should be saved in a format compatible with MATLAB Version 6, what Monte Carlo parameters are to be used in the genetic algorithm, and whether or not the program is to be run such that parallel computation is available.

To access this menu, click on the 'Options' menu, then choose 'Select Options'. This loads the window shown below in Figure 5.22. The default values that appear in this window are the most recently used values for these parameters, as saved in the file 'options.mat'. If this file does not exist, the program chooses defaults to multi-threading on, and all other options off.

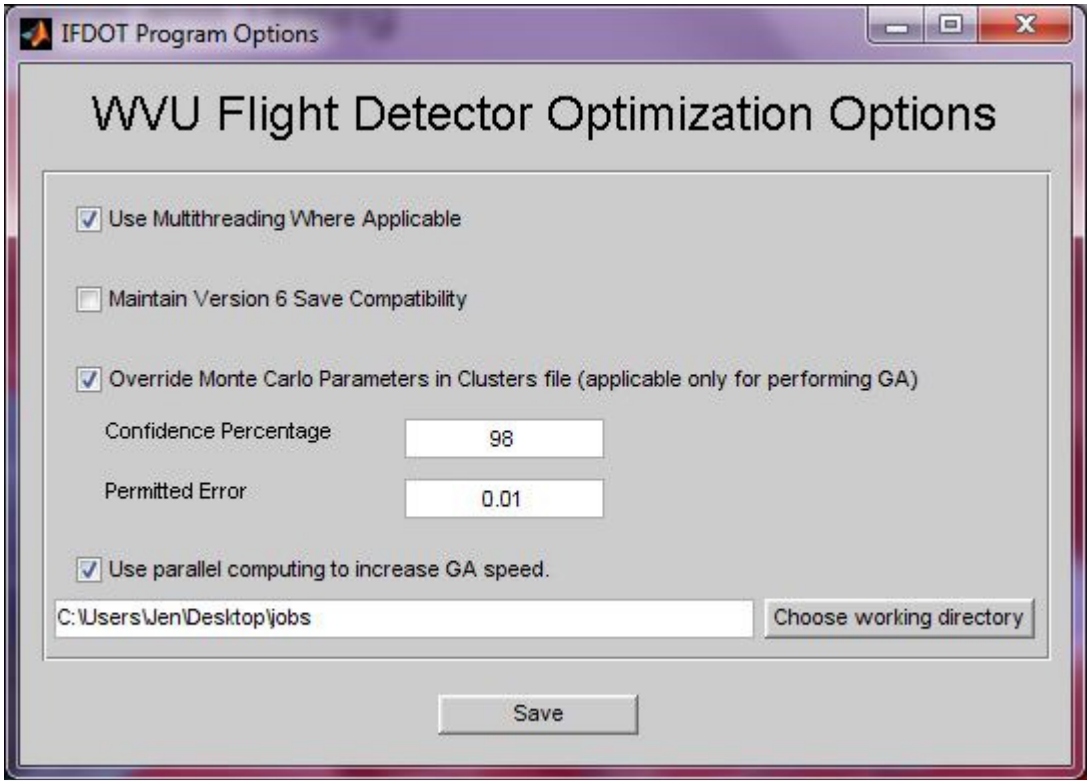

Figure 5.22—Options Menu

Each of these options serves a different purpose within the program. 'Use Multithreading Where Applicable' allows the program to calculate some portions of the algorithm using up to 4 available threads on the same machine. This option has no effect if parallel computation is used. 'Maintain Version 6 Compatibility' allows the program to save all pertinent data files in a format that is compatible with the older MATLAB Version 6. This is useful in the event that data collection or fault detection is performed using an older version of MATLAB. 'Override Monte Carlo Parameters in Clusters file' allows the using to specify Monte Carlo Volume Estimation Parameters different from those contained in the clustered data file. Generally, the parameters used in the clustered data file will also be accurate enough for the genetic algorithm; however, this option is made available in the event that this is not the case.

The option 'Use parallel computing to increase GA speed' is intended to increase the computation time of the genetic algorithm. If this method is selected, only one machine is needed to calculate the genetic algorithm. However, additional computers may be used as 'slave' machines

to calculate and return the work on an individual. To each this menu, choose File→Detector Optimization→Negative Selection→Run GA as Slave. This loads the menu shown below in Figure 5.23.

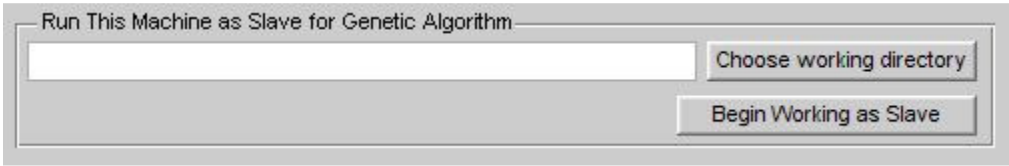

Figure 5.23—Slave Calculation Menu

These slave machines work entirely independently from the main computer, and need only to be able to remotely access the working directory of the main computer to perform calculations. If access is denied, an error will appear. Otherwise, a message will appear that the machine is in use. This message will disappear once the genetic algorithm completes.

The slave machines may be added or removed from the calculation pool at any time without affecting the genetic algorithm in terms of any parameter except speed. Due to the nature of multithreading, true multi-threading is not possible for parallel computation. However, if multiple threads are available on a single machine, these may be manually started by using multiple instances of the slave machine software on the same machine. Note that each instance of the software should be opened using a different instance of MATLAB, and each instance of MATLAB should use a different Current Directory, so as to avoid incorrect overwriting of files.

5.7.3 Displaying Results

It may be useful to revisit data from an optimization. If this is desired, click on 'File'→ 'Detector Optimization' → 'Negative Selection' → 'Load Previous Trial Data to Continue Optimization'. This will bring up the load file menu. Load an optimization file. When an appropriate file is loaded, the menu option 'Review Results' will become available. Click 'File'→ 'Detector Optimization' → 'Negative Selection' → 'Review Results' to load the results in the same manner that they appear at the end of the trial. See Figure 5.24 below for an example of the results. Note that this is a 2-D example. The results will appear somewhat differently in higher dimensions, as in Figure 5.25.

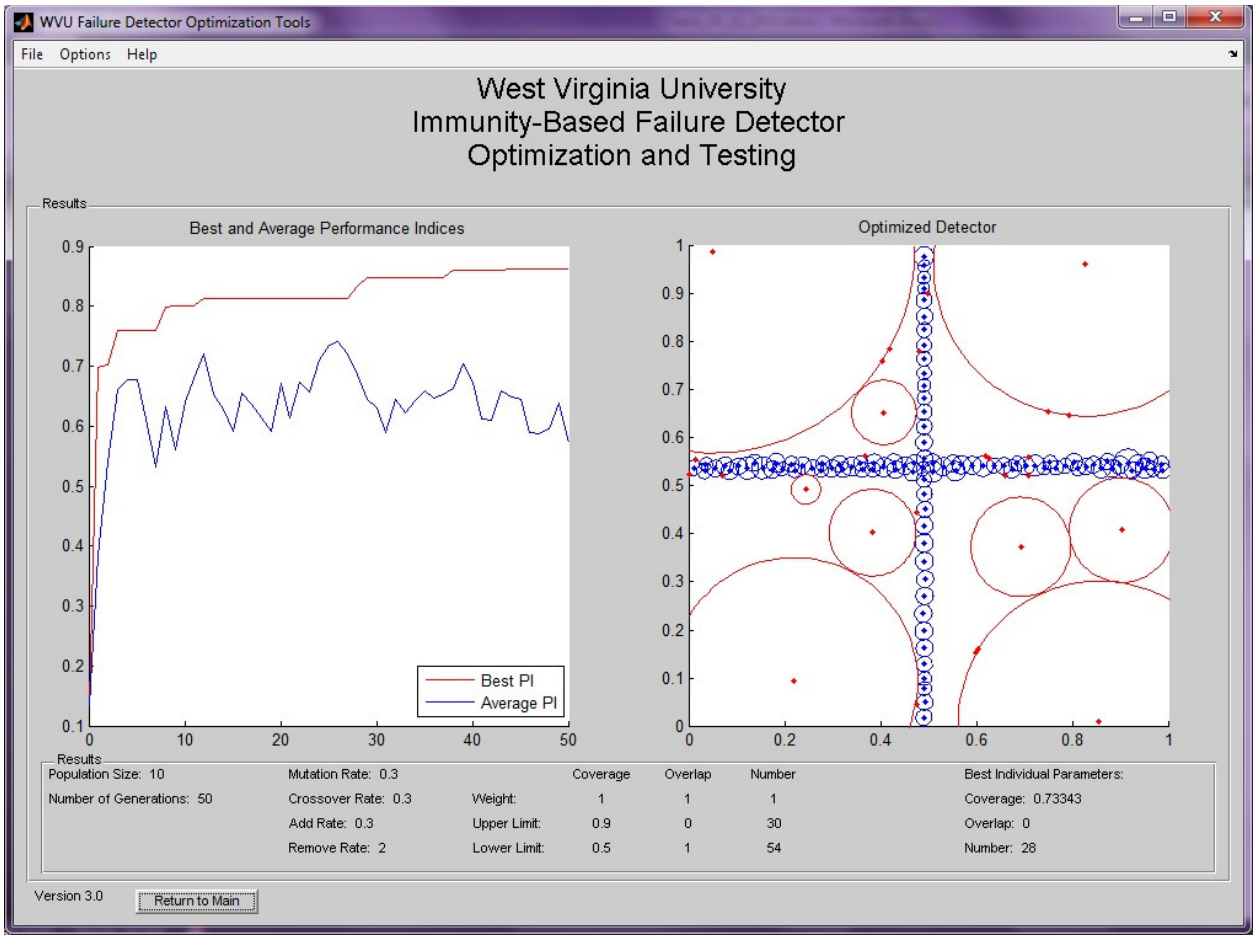

Figure 5.24—Results Display for 2-Dimensional Data Trial

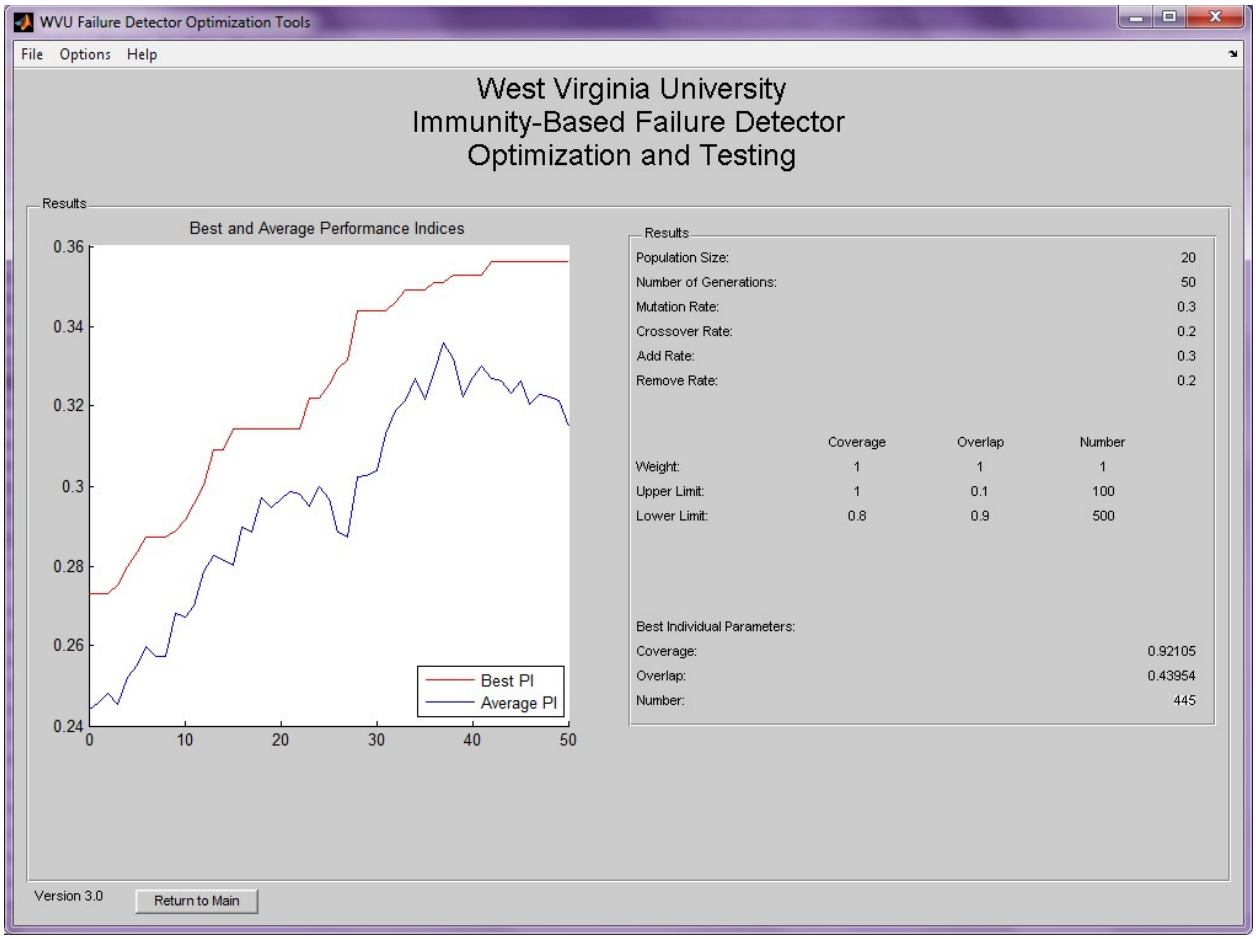

Figure 5.25—Results Display for Higher-Dimensional Data Trial

5.7.4 Positive Selection Detector Generation

The positive selection algorithm functions in the same way as negative selection clustering, as described in Section 5.4, except that the data array will be saved to the name 'optdetector', rather than 'selfdataclusters' as for clustering. In order to create a positive selection detector set, click on the 'File' menu, select 'Detector Optimization', 'Positive Selection', and 'Load Processed Data'. Load a data file, then go to the 'Create Positive Selection Detectors' and follow the instructions in the data clustering section.

5.7.5 Negative Selection Detector Generation

The user is capable of generating a set of detectors using the same detector generation methods as are used to generate the initial population for the FDGO algorithm. This may be useful for testing the detection performance of a set of identifiers for a particular type of failure, without optimizing the set first. This "quick" detector generation can save time: there is no point in optimizing a detector set only to discover that the identifiers are not adequate for a desired type of failure.

In order to generate such a set of detectors, click on the 'File' menu, select Detector Optimization → Negative Selection → Load Clustered Data, and load a valid clustered self set. Then select 'File' → 'Detector Optimization' → 'Negative Selection' → 'Create Detectors (Phase I

only)'. This will allow the user to specify the parameters for and generate a detector set without having to go through the optimization process. The user is also able to save the detector set, unlike running a sample detector set when optimizing.

5.7.6 Data Merging

It is useful to combine data files into single files, for instance, when new data is collected. Merging functions are provided to accommodate these needs. Be aware that in all cases, the program will determine whether the files are numerically compatible, but it is up to the user to ensure that the files are fundamentally compatible.

5.7.6.1 Raw Data:

If order to merge raw data, two data files containing raw data saved to the variable 'sensors', with the same number of columns, must be available. Click 'File'→ 'Data Processing' → 'Merge Raw' to bring up the merge raw data menu. This consists of two load file menu boxes. When the appropriate files are loaded, the button marked 'Merge Raw Data' will be enabled. Note that the button will not become active until two files containing appropriate, compatible data are loaded. Clicking the 'Merge Raw Data' button will merge the two data files together, to be saved as a single file.

5.7.6.2 Processed Data:

Merging processed data is more involved than merging raw data, but no more difficult for the user. Click 'File' → 'Data Clustering' → 'Merge Processed Data' to load the merge processed data menu. Again, two files are requested before the 'Merge Processed Data' button will be enabled. These files must contain processed data saved to the variable 'selfdata'. These files need only be compatible with respect to type and number of parameters. The files will undergo several processes. The maximums and minimums for each file will be found, compared, and the overall max and min values will be used to normalize the two sets compatibly, and eliminate new possible duplicates. The results of this process are equivalent, though not identical, to merging two raw data files, then normalizing the combined data. A different number of data points may result from this process than from combining the raw files, due to multiple instances of duplicate removal.

5.7.6.3 Clustered Data:

Merging clusters is useful if two compatible data sets are available. In this case, click 'File' → 'Data Clustering' → 'Merge Clustered Data'. This will bring up the load clustered files menu. Choose two files that have been clustered using the same shape and contain the same number of parameters. When two compatible files are loaded, the 'Merge Clusters' button will become activated. Click the button to merge the clusters, then save the resulting cluster set. Merging clusters can be done for both sphere and rectangle clusters.

6 Results Yielded Using the West Virginia University Immunity-Based Failure Detector Optimization and Testing Utility

The IFDOT Utility has essentially been designed to allow the user to maximum amount of flexibility and customizability in the design and generation of the self and non-self of an immunity-based detector set. Due to the expansive nature and open-ended design of the IFDOT Utility, full exploration of the design environment's capabilities cannot be investigated here. Instead, the focus of these results will be to first prove the functionality and validity of the optimization methods, then illustrate the potential detection improvements possible by utilizing this optimization utility. Several uses of this utility have already been made during the course of this research effort (12) (13) (50) (51) (53) (54) (81) (82) (85) (86). Comparison of the four available detector shapes will be made, for 2-D, 3-D, and 6-D cases. The 2-D case is intended to compare the relative performance of each of the shapes with respect to coverage, overlapping, and number of detectors. The 3-D case is intended to compare the relative performance of each of the shapes with respect to failure detection and false alarms. The 6-D case is primarily intended to illustrate the capability of the Utility to handle high-dimensional situations, being that the use of many identifiers may be necessary to detect many types of failures. Although calculation times will be included and discussed, these are only to be used as a general comparison, not an exact comparison, since it was necessary to use several different computers, of various speeds, to calculate the optimizations.

Three trials were performed for each of the four detector shapes, each with duration of 50 generations. Full results for all trials, with shapes compared side-by-side, are located in Appendix A. Plots of the best solutions achieved in each of these trials as well as the best and average performance index experienced throughout the trial is presented for each of these optimization trials. Three trials of equal length were chosen for comparing each of the shapes. Since allowing the algorithm to converge would take significantly more generations and convergence itself is somewhat subjective, this was determined to be a more accurate approach for direct comparison of the shapes. In addition, the optimization parameters for each of the shapes were kept as similar as possible, to enhance direct comparison among the shapes. These parameters were intentionally kept generic, as the purpose of these results is to compare each of the shapes, rather than explore the influence of the optimization parameters on the performance of the algorithm.

It should be noted that although this utility was designed for the generation and optimization of detectors for aircraft failure, and although the data results presented here utilize flight data obtained from the WVU 6-degree-of-freedom simulator, this utility is not limited to aircraft applications. The algorithms implemented in the IFDOT Utility utilize normalized data, and are therefore independent of the identifiers contained in the data. This utility may be applied to any system for which failure detection is desired, and for which time-histories of necessary identifiers are measureable.

6.1 Explanation of Failure Detection and Identification Scheme

In order to create immunity-based detectors, normal condition data must be available for the system. To completely and accurately define the self, time-history data is collected at a rate of 50 Hz from the Motus 6-degree-of-freedom flight simulator, implementing the WVU IFCS F-15 research aircraft model, for a wide variety of parameters. In order to define the self over a large range of the flight envelope, 9 points are defined within the flight envelope at varying altitudes and Mach numbers. An additional 4 points within this range are defined for the purpose of determining the detection performance of a set of detectors. These points are shown in Figure 6.1 below.

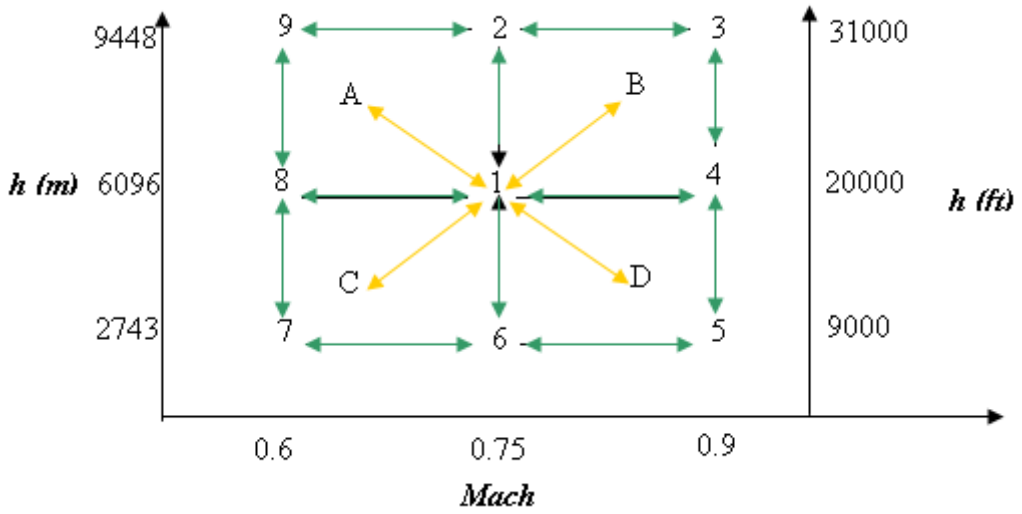

Figure 6.1—Definition of Flight Envelope Points (50)

The normal condition time history defines the self for each of these points. In order to define the self over a large range of the flight envelope, the time-histories for each of these points 1-9 are combined, duplicate points are removed, and the data values are normalized from 0 to 1.

The creation of a successful detector set begins with the choice of adequate parameters to detect the various sensor, actuator, structural, and propulsion failures. A hierarchical multi-self strategy (50) (51) has been proposed to increase detection rate, reduce false alarms, and reduce on-line computational requirements. This method polls the detection result of several smaller-dimensional detector sets, rather than using one high-dimensional set.

In order to test the performance of a set of detectors, abnormal condition data containing only one type of failure is collected at each of the points in the flight envelope at a rate of 50 Hz, and compared against the detector set. A detector is activated when an abnormal data point falls within one of the detectors in the set. A failure is declared when a detector is activated for a continuous 50 samples, or 1 second, and the failure is declared continuously until a detector has not been activated for the 50 previous samples.

6.2 2-Dimensional Example

The self for each of the 2-dimensional trials was defined by 100 clusters using data for roll-rate and pitch-rate since these parameters are uncoupled and therefore form a characteristic 'cross' pattern. The same self clusters are used for hyper-spheres, hyper-ellipsoids, and hyper-rotational-ellipsoids. For these shapes, 100 hyper-spheres were used to cluster the self data. Due to the characteristic of the shape, the same self clusters could not be used for hyper-rectangles. For this shape, the self data was clustered using 100 hyper-rectangles.

6.2.1 Hyper-Spheres

Three trials of 50 generation each were performed for 2-dimensional spherical detectors. The optimization parameters used for these trials are given below in Table 6.1. The results of each of these trials, including time to calculate are given in Table 6.2. The best individuals resulting from each of these trials and the performance indices throughout these trials are given in Figure 6.2 through Figure 6.7 below. The coverage, overlapping, and number of detectors for the best

individual from each trial as well as the time needed to calculate these trials is given in Table below. Trials 1 and 3 were calculated using a 2.2GHz Intel Core 2 Duo and 4 GB of RAM with multithreading on. Trial 2 was calculated using a 2.8GHz Intel Core 2 Duo and 6 GB of RAM with multithreading off.

Table 6.1—2-D Sphere Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Relocation Constant	1
Gene Alteration Constant	0.20
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100
Number of Detectors Worst Limit	500

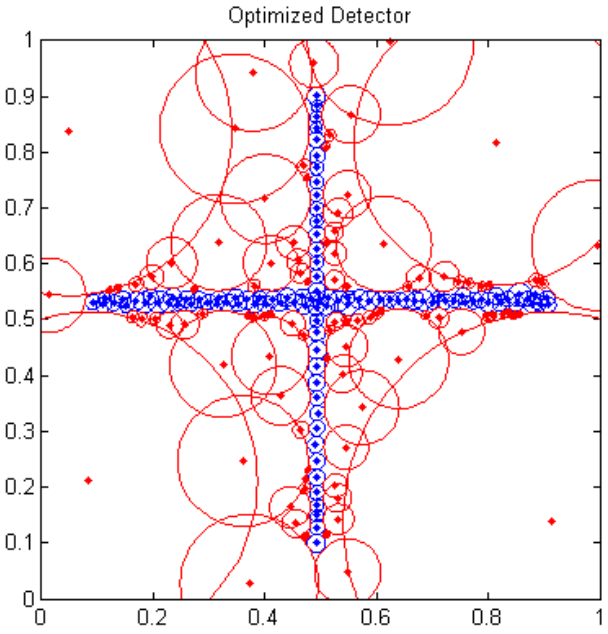

Figure 6.2—Best Individual in 2-D Hyper-Spheres Trial 1

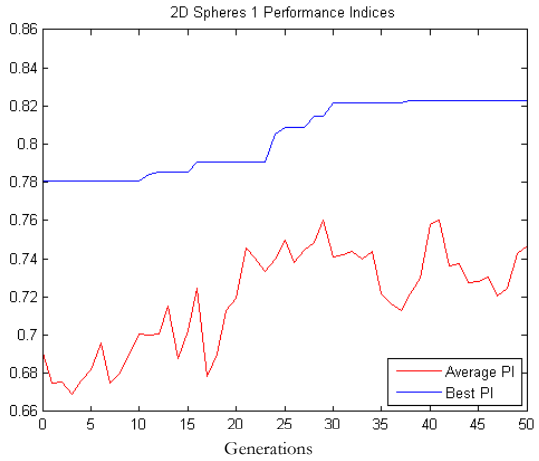

Figure 6.3—Performance Indices for 2-D Hyper-Spheres Trial 1

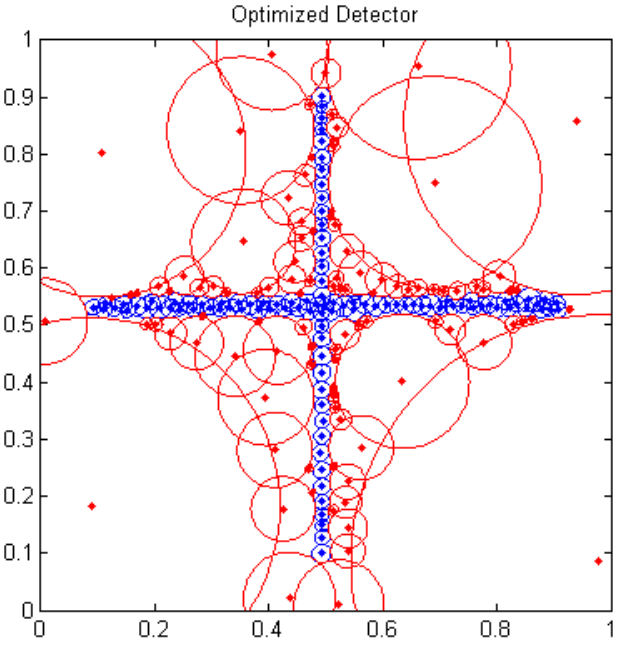

Figure 6.4—Best Individual in 2-D Hyper-Spheres Trial 2

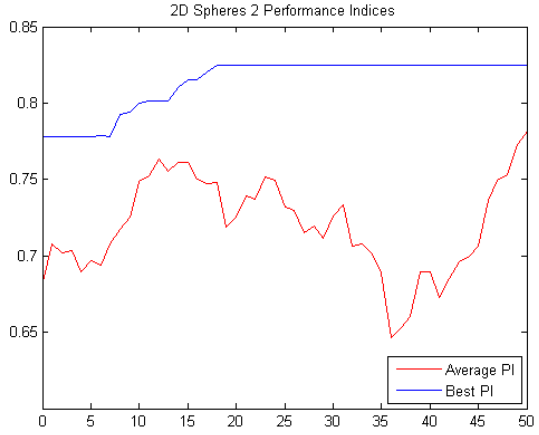

Figure 6.5—Performance Indices for 2-D Hyper-Spheres Trial 2

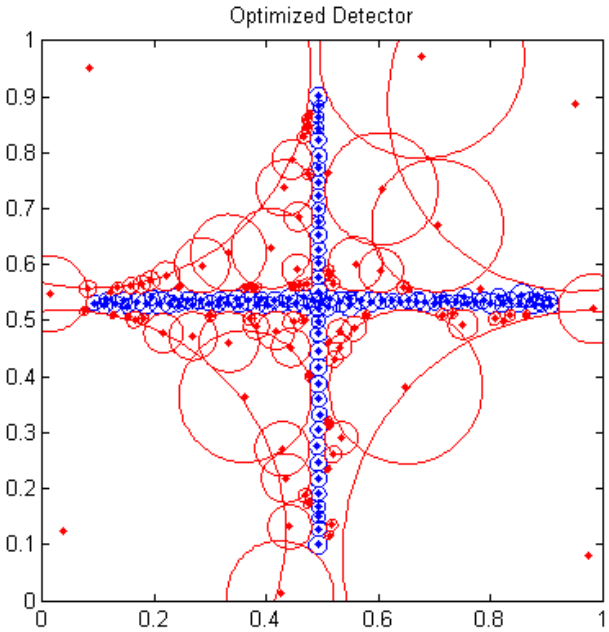

Figure 6.6—Best Individual in 2-D Hyper-Spheres Trial 3

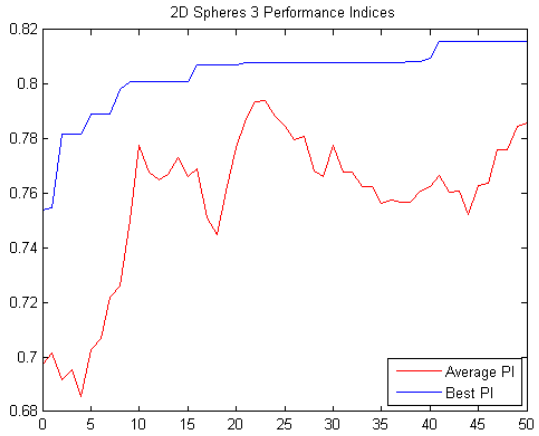

Figure 6.7—Performance Indices for 2-D Hyper-Spheres Trial 3

Table 6.2—2-D Hyper-Sphere Results

	Coverage %	Overlapping %	Number	Time (minutes)
Trial 1	97.62	22.97	97	334.92
Trial 2	97.76	22.62	102	680.24
Trial 3	96.58	20.49	89	291.78
Average	97.32	22.03	96	435.68

6.2.2 Hyper-Ellipsoids

Three trials of 50 generation each were performed for 2-dimensional hyper-ellipsoidal detectors. The optimization parameters used for these trials are given below in Table 6.3. The results of each of these trials, including time to calculate are given in Table 6.4. The best individuals resulting from each of these trials and the performance indices throughout these trials are given in Figure 6.8 through Figure 6.13 below. It should be noted that in these performance index plots, occasionally the performance index of the best individual shows a decrease, contrary to the purpose of elitist selection. This is not due to a fault in the elitist selection strategy. Rather, some of these trials were completed in segments and reassembled. Each time the trial is restarted, using continued optimization in the utility, the population is re-rated using the Monte Carlo Volume Estimation algorithm. Being that this is a numerical method, it does not return exactly the same values each time it is run. In these instances, when the trials were continued from where they left off, the best individual, which is the same set of detectors, happened to get rated slightly lower than it had been rated before. This never occurs within trials run consecutively. Note that in the best individuals, it can be observed that few of the detectors have been modified to for literal hyper-ellipsoids. This is due to the fact that only the mutation genetic operator is capable of altering the detectors in this way. If more variation of the detector shapes is desired, it may be necessary to use more aggressive

mutation parameters, or allow the algorithm to run for a larger number of generations. The coverage, overlapping, and number of detectors for the best individual from each trial as well as the time needed to calculate these trials is given in Table below. Trial 1 was calculated using a 2.13GHz Intel Core 2 Duo and 6 GB of RAM with multithreading off. Trials 2 and 3 were calculated using a 2.2GHz Intel Core 2 Duo and 2 GB of RAM with multithreading on.

Table 6.3—2-D Ellipsoid Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Rotation Weight	2
Gene Relocation Constant	1
Gene Alteration Constant	0.20
Gene Rotation Constant	10
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Accuracy	0.001
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100
Number of Detectors Worst Limit	500

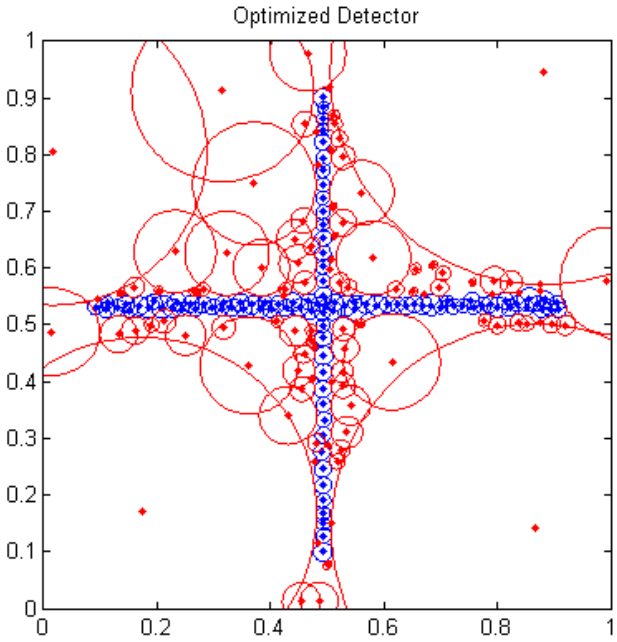

Figure 6.8—Best Individual in 2-D Hyper-Ellipsoids Trial 1

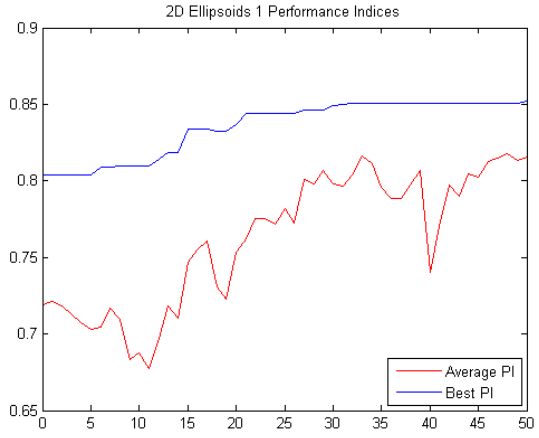

Figure 6.9—Performance Indices for 2-D Hyper-Ellipsoids Trial 1

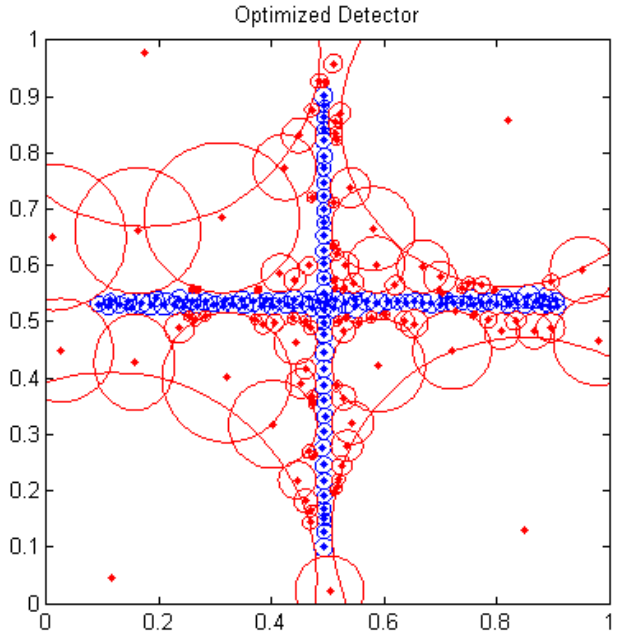

Figure 6.10—Best Individual in 2-D Hyper-Ellipsoids Trial 2

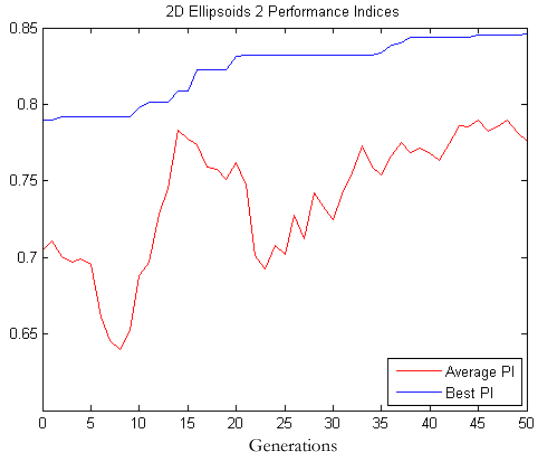

Figure 6.11—Performance Indices for 2-D Hyper-Ellipsoids Trial 2

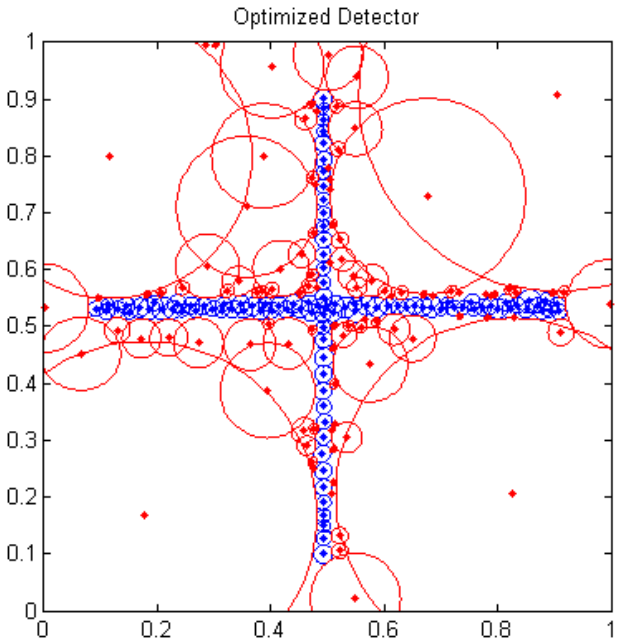

Figure 6.12—Best Individual in 2-D Hyper-Ellipsoids Trial 3

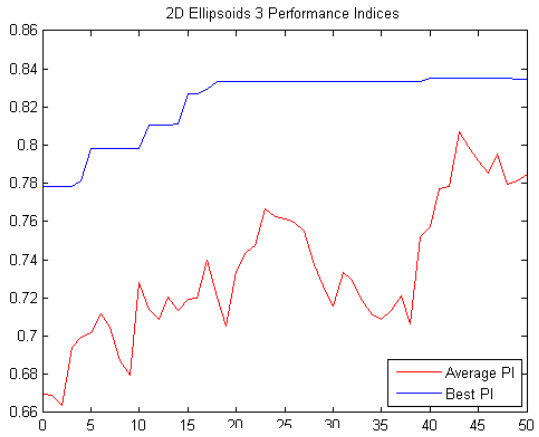

Figure 6.13—Performance Indices for 2-D Hyper-Ellipsoids Trial 3

Table 6.4—2-D Hyper-Ellipsoid Results

	Coverage %	Overlapping %	Number	Time (minutes)
Trial 1	96.77	12.17	102	25648
Trial 2	97.30	16.17	98	8291.5
Trial 3	97.29	18.32	103	8093.9
Average	97.12	15.55	101	14011

6.2.3 Hyper-Rotational-Ellipsoids

Three trials of 50 generation each were performed for 2-dimensional hyper-rotational-ellipsoidal detectors. Since this is a 2-dimensional trial, these shapes are physically equivalent to hyper-ellipsoids, although they are represented somewhat differently. The optimization parameters used for these trials are given below in Table 6.5. The results of each of these trials, including time to calculate are given in Table 6.6. The best individuals resulting from each of these trials and the performance indices throughout these trials are given in Figure 6.14 through Figure 6.19 below. It should be noted that in these performance index plots, occasionally the performance index of the best individual shows a decrease, contrary to the purpose of elitist selection. This is not due to a fault in the elitist selection strategy. Rather, some of these trials were completed in segments and reassembled. Each time the trial is restarted, using continued optimization in the utility, the population is re-rated using the Monte Carlo Volume Estimation algorithm. Being that this is a numerical method, it does not return exactly the same values each time it is run. In these instances, when the trials were continued from where they left off, the best individual, which is the same set of detectors, happened to get rated slightly lower than it had been rated before. This never occurs within trials run consecutively. Note that in the best individuals, it can be observed that few of the detectors have been modified to for literal hyper-ellipsoids. This is due to the fact that only the

mutation genetic operator is capable of altering the detectors in this way. If more variation of the detector shapes is desired, it may be necessary to use more aggressive mutation parameters, or allow the algorithm to run for a larger number of generations. The coverage, overlapping, and number of detectors for the best individual from each trial as well as the time needed to calculate these trials is given in Table below. Trial 1 was calculated using a 2.2GHz Intel Core 2 Duo and 4 GB of RAM with multithreading on. Trial 2 was calculated using a 2.8GHz Intel Core 2 Duo and 6 GB of RAM with multithreading off. Trial 3 was calculated using a 2.2GHz Intel Core 2 Duo and 2 GB of RAM with multithreading on.

Table 6.5—2-D Rotational Ellipsoid Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Rotation Weight	2
Gene Relocation Constant	1
Gene Alteration Constant	0.20
Gene Rotation Constant	10
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Accuracy	0.001
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100
Number of Detectors Worst Limit	500

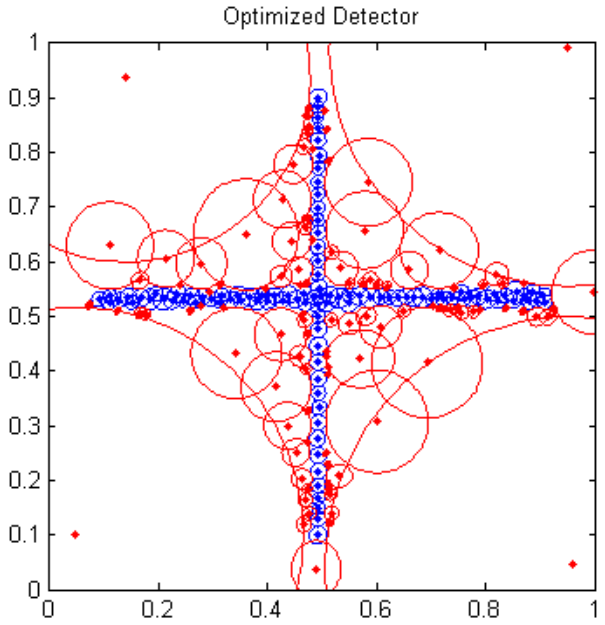

Figure 6.14—Best Individual in 2-D Hyper-Rotational-Ellipsoids Trial 1

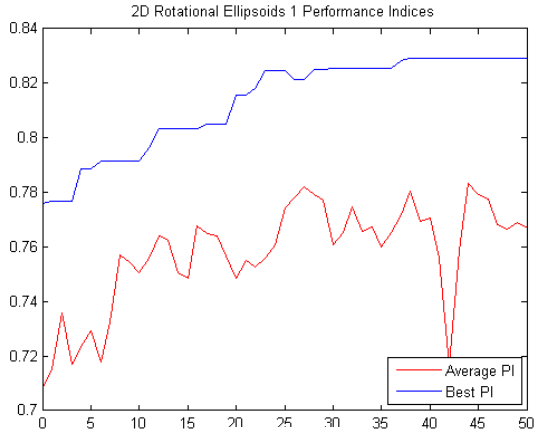

Figure 6.15—Performance Indices for 2-D Hyper-Rotational-Ellipsoids Trial 1

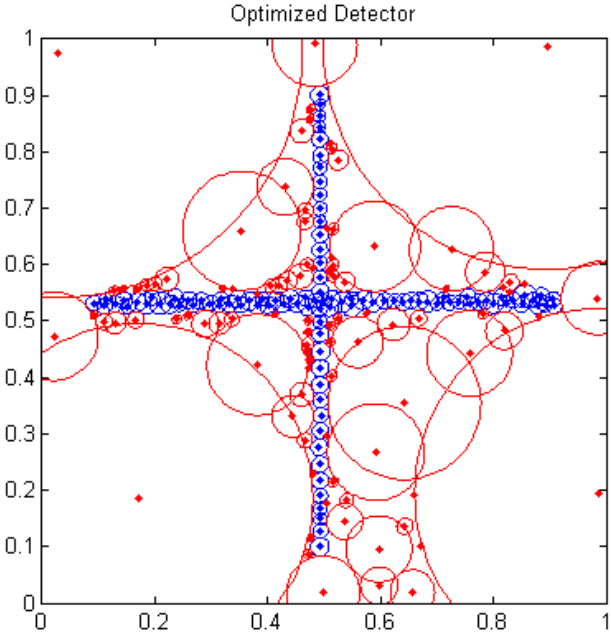

Figure 6.16—Best Individual in 2-D Hyper-Rotational-Ellipsoids Trial 2

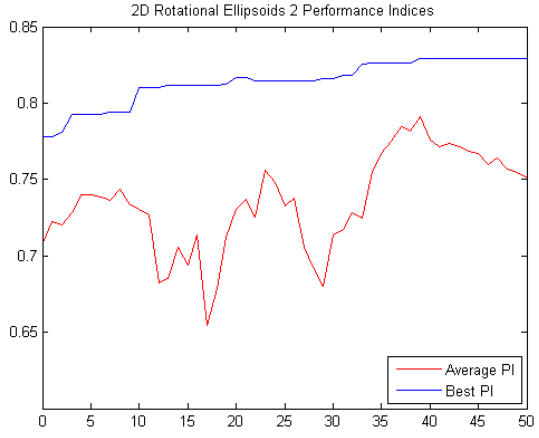

Figure 6.17—Performance Indices for 2-D Hyper-Rotational-Ellipsoids Trial 2

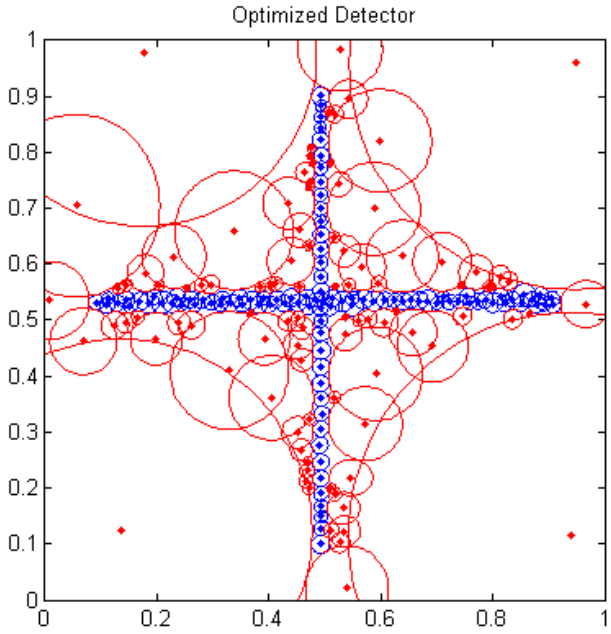

Figure 6.18—Best Individual in 2-D Hyper-Rotational-Ellipsoids Trial 3

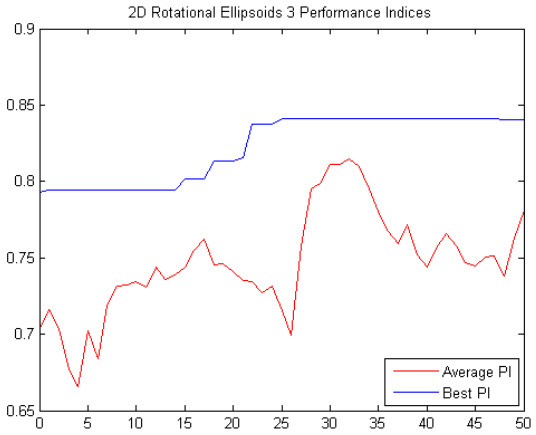

Figure 6.19—Performance Indices for 2-D Hyper-Rotational-Ellipsoids Trial 3

Table 6.6—2-D Hyper-Rotational-Ellipsoid Results

	Coverage %	Overlapping %	Number	Time (minutes)
Trial 1	96.05	13.62	108	15826
Trial 2	95.05	11.31	96	20944
Trial 3	97.56	18.44	97	8400.9
Average	96.22	14.46	100	15057

6.2.4 Hyper-Rectangles

Three trials of 50 generation each were performed for 2-dimensional hyper-rectangular detectors. The optimization parameters used for these trials are given below in Table 6.3. The results of each of these trials, including time to calculate are given in Table 6.4. The best individuals resulting from each of these trials and the performance indices throughout these trials are given in Figure 6.8 through Figure 6.13 below. The coverage, overlapping, and number of detectors for the best individual from each trial as well as the time needed to calculate these trials is given in Table below. Trial 1 was calculated using a 2.8GHz Intel Core 2 Duo and 4 GB of RAM with multithreading off. Trials 2 and 3 were calculated using a 2.2GHz Intel Core 2 Duo and 2 GB of RAM with multithreading on.

Table 6.7—2-D Rectangle Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Relocation Constant	1
Gene Alteration Constant	0.20
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100
Number of Detectors Worst Limit	500

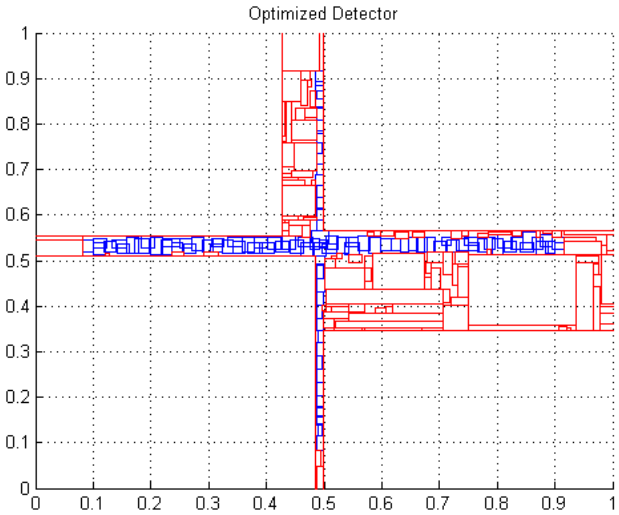

Figure 6.20—Best Individual in 2-D Hyper-Rectangles Trial 1

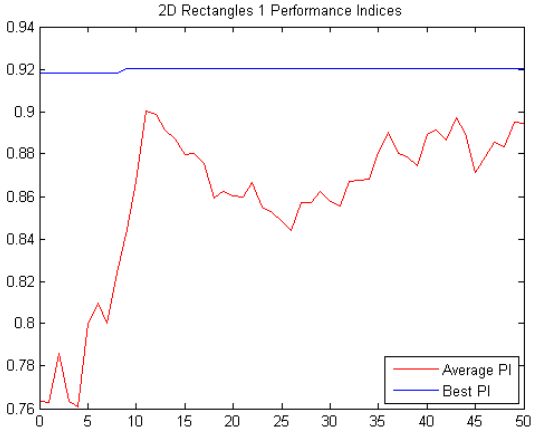

Figure 6.21—Performance Indices for 2-D Hyper-Rectangles Trial 1

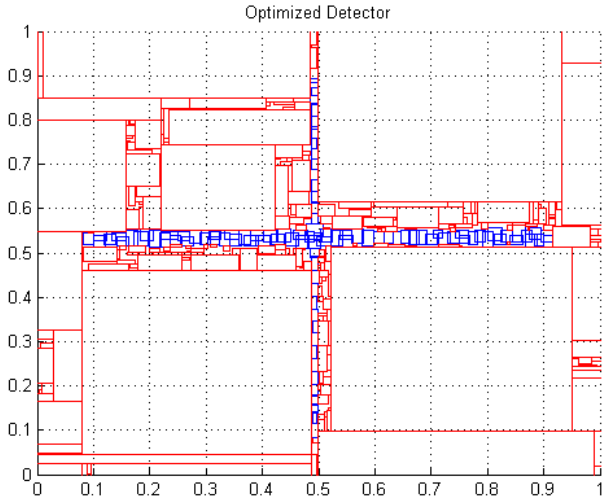

Figure 6.22—Best Individual in 2-D Hyper-Rectangles Trial 2

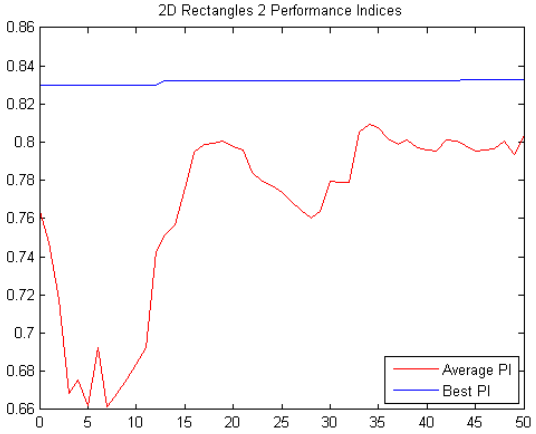

Figure 6.23—Performance Indices for 2-D Hyper-Rectangles Trial 2

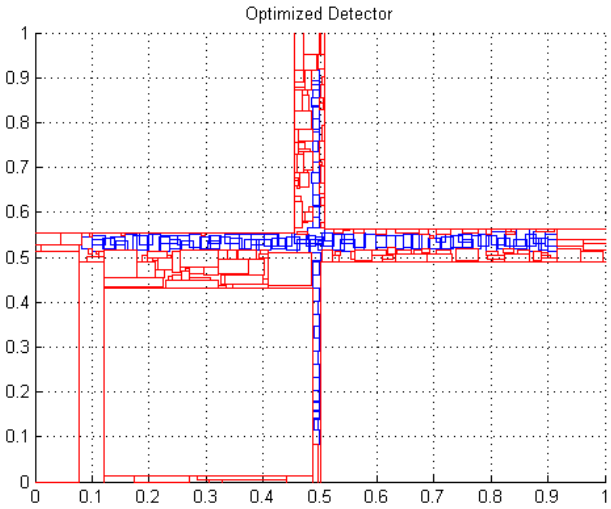

Figure 6.24—Best Individual in 2-D Hyper-Rectangles Trial 3

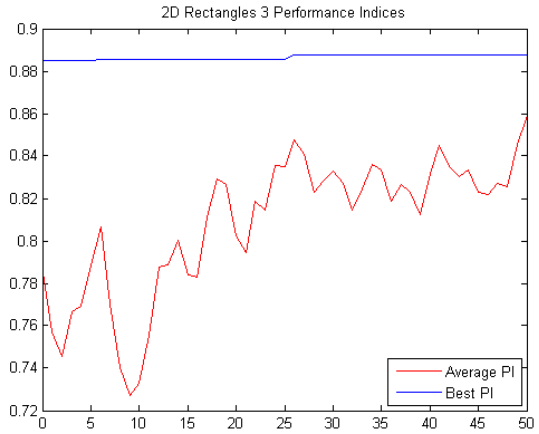

Figure 6.25—Performance Indices for 2-D Hyper-Rectangles Trial 3

Table 6.8—2-D Hyper-Rectangles Results

	Coverage %	Overlapping %	Number	Time (minutes)
Trial 1	97.93	0	104	488.45
Trial 2	97.73	0	205	186.54
Trial 3	97.91	0	143	170.43
Average	97.86	0	151	281.81

6.2.5 Shape Comparison

For the 2-dimensional trials, hyper-rectangles are the quickest to calculate and obtained no overlapping while maintaining higher average coverage than the other shapes. However, the cost is that the hyper-rectangles required approximately 50% more detectors to achieve this performance than the other shapes.

Hyper-spheres were also quick to calculate, taking approximately one-third the calculation time of the 2-dimensional hyper-ellipsoids or hyper-rotational-ellipsoids. In addition, hyper-spheres achieved the second best coverage, nearly that obtained by the hyper-rectangles, while only needing an average of 96 detectors. However, the hyper-spheres obtained the highest overlapping of any shape, approximately one-third higher than that for hyper-ellipsoids or hyper-rotational-ellipsoids.

In 2-dimensions, hyper-ellipsoids and hyper-rotational-ellipsoids are physically the same. Average coverage, overlapping, and number of detectors for hyper-rotational-ellipsoids was slightly lower than that for hyper-ellipsoids, however, the results for each are very similar. These shapes achieved lower overlapping and similar number of detectors and coverage of the solution space of hyper-spheres, while taking significantly longer to calculate.

6.3 3-Dimensional Example with Detection Results

Due to time limitations imposed by the calculation times of some of the shapes, and due to the desire for consistency, these detector sets have undergone optimization for 50 generations, using a population size of 20 individuals. These optimized sets have not converged to the global optimized solution, thus potentially better results are possible. In addition, the detector sets used for the clustering comparison found in section 6.3.1 were not individuals in the initial population of these trials. For sphere, ellipsoid, and rotational ellipsoid detectors, the self is defined using hyper-spheres to cluster the self data, consisting of roll-, pitch-, and yaw-rate neural network estimates. For rectangle detectors, the same self data is clustered using hyper-rectangles.

6.3.1 Clustering of the Self

The number of clusters used to define the self can have a significant impact upon the performance of the detector sets generated from them. When clustering is performed, some area is included in the self definition that has not been confirmed to be self. The number of clusters used can be as few as 1 or as many as the number of data points used to define the self region. Using too few clusters will include too much area of the solution space in the definition of the self, which should actually belong to the non-self region, lowering the overall detection rate. Using too many clusters with too little empty space, however, excludes some areas from the self definition which should actually belong to the self. This induces a high number of false alarms. Each of these situations is undesirable for producing an effective detector set.

In order to determine the appropriate number of clusters to use, several trials were performed and compared using 500, 2000, and 5000 clusters to define the self for this set of identifiers over the full flight envelope. Since optimization was not desired, three detector sets were generated using only Phase 1 detector generation for each of the varying sets of clusters. The average results of these tests can be seen below in Table 6.9. For all detection results except for data listed as nominal, the number if the table represents detection rate. For nominal data sets, the number represents rate of false alarms. The full results have been included in Table A-1 through Table A-3 in Appendix A.

Table 6.9—3-D Cluster Comparison Results

Failure	Type	Location	Magnitude	Envelope	500 Clusters	2000 Clusters	5000 Clusters
					Average	Average	Average
Actuator	aileron	left	5 deg	123	99.6901	98.2217	99.8591
			8 deg		97.3680	88.6537	99.8895
				187	97.0270	99.0515	99.9581
		right		165	94.9428	99.7752	78.5384
					94.6657	99.7465	78.4237
				123	93.9443	99.7909	93.3521
	rudder	left	8 deg	145	85.8053	88.7225	89.6260
				123	44.3588	55.6500	61.3476
		right		167	46.3305	48.0334	55.0064
				123	43.8885	49.7818	57.9506
	stabilator	left	2 deg	123	98.8065	90.1800	99.9134

			8 deg		99.7700	98.6483	99.9701
				145	99.7114	98.1649	99.9666
		right		189	91.5663	99.9672	96.1968
				123	91.2982	99.6217	93.5932
Sensors	LFDB	r	3 deg	165	31.9329	32.2342	25.5986
				123	85.5207	89.4497	90.6592
		p	10 deg	167	2.8810	13.1274	21.0116
				123	4.3998	8.4633	9.8990
		q		187	0.6211	1.6506	3.7003
				123	6.9244	13.5703	21.0387
	LSB	r	3 deg	167	7.4095	9.4846	12.1730
				123	77.8160	85.9596	84.4357
		p	5 deg		2.9554	6.5922	7.4326
			10 deg	145	5.2651	9.1546	12.6752
				123	13.2313	18.5033	20.5612
		q		189	13.0588	20.7536	28.9663
				123	1.7156	4.5480	10.2534
Structural	Wing	left	15%	167	92.6890	97.0155	99.6777
			35%	123	96.2721	99.3355	99.3294
		right			97.3429	97.6027	99.9047
Engine	Left		1%	167	2.8908	7.3410	8.8903
			10%	123	36.2581	38.5549	42.9825
					3.7761	7.7107	14.6056
	Right		1%		15.8538	20.7849	23.4692
			10%		1.6640	2.8995	4.3847
			1%	187	10.9421	14.3553	24.4388
Nominal	1A				0.5853	0.2906	1.4199
	1B				0.3030	0.5048	0.8395
	1C				0.7211	1.0933	1.4191
	1D_1				0.0541	0.0569	0.2008
	1D_2				0.0000	0.0110	0.0531
	4				0.0000	0.0000	0.0687
	12				0.0000	0.0000	0.6612

As seen in the table above, no clearly better solution is found using a higher number of clusters to define the self. For this reason, 500 clusters are chosen to define the self to reduce computational load of the optimization algorithm. For consistency, 500 clusters will also be used to define the self using hyper-rectangles, and to define the self for the 6D cases.

6.3.2 Hyper-Spheres

Hyper-spheres are the simplest shape to use in the detector optimization algorithm, since the radius is constant for all dimensions. This in turn allows them to also be calculated the quickest. The average calculation time of the 3D sphere trials was 2607.75 minutes, or about 44 hours. The optimization parameters for these trials are given in Table 6.10. The calculation times, coverage, overlapping, and number of detectors are shown in Table 6.11. These results were run using Intel Pentium 4 processors at 2.4GHz, which support only single-thread applications. The detection results of these four trials are located below in Table 6.12. For all detection results except for data listed as nominal, the number if the table represents detection rate. For nominal data sets, the number represents rate of false alarms. Figure 6.26 through Figure 6.29 show the performance index of the population throughout each trial.

Table 6.10—3-D Hyper-Spheres Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Rotation Weight	2
Gene Alteration Constant	0.20
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Accuracy	0.001
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100
Number of Detectors Worst Limit	500

Table 6.11—Performance Parameters for 3-D Hyper-Spheres

	Coverage %	Overlapping %	Number of Detectors	Time (minutes)
Trial 1	92.105	43.954	445	2715.6
Trial 2	91.848	40.301	476	2632.2
Trial 3	92.499	44.821	471	2518.3
Trial 4	92.786	48.155	435	2565.0
Average	92.310	44.308	457	2607.8

Table 6.12—3-D Hyper-Spheres Detection Results after Optimization

Failure	Type	Location	Magnitude	Envelope	Hyper-Spheres				
					Trial 1	Trial 2	Trial 3	Trial 4	Average
Actuator	aileron	left	5 deg	123	99.2949	99.9443	97.9425	99.1547	99.0841
			8 deg		99.9294	99.7776	99.9508	96.6325	99.0726
				187	99.3418	99.9554	98.9732	96.9804	98.8127
		right		165	99.0266	99.0884	99.7727	95.6296	98.3793
					98.4405	99.1531	99.8393	95.0912	98.1310
				123	99.6611	98.5203	99.84	95.2662	98.3219
	rudder	left	8 deg	145	86.048	85.8129	85.3886	85.0495	85.5748
				123	43.0428	45.2908	49.9754	40.9999	44.8272
		right		167	48.2743	49.3586	40.6949	47.4805	46.4521
				123	38.3939	32.6606	44.9385	42.8311	39.7060
	stabilator	left	2 deg	123	85.1851	97.2371	90.8451	97.6386	92.7265
			8 deg		99.969	98.8492	99.969	99.9673	99.6886
				145	99.9653	99.9672	99.9653	99.9634	99.9653
		right		189	95.1128	99.9647	99.9647	99.9647	98.7517
				123	94.875	93.841	99.2114	99.8642	96.9479
Sensors	LFDB	r	3 deg	165	28.7776	31.4133	31.5839	30.5924	30.5918
				123	82.8107	82.9812	85.5003	84.5769	83.9673
		p	10 deg	167	3.1761	2.6186	5.7173	1.1496	3.1654
				123	5.4637	5.6127	4.055	5.5348	5.1666
		q		187	0.31722	0.048803	0	0.20741	0.1434
				123	4.0997	5.7533	5.6278	4.6176	5.0246
	LSB	r	3 deg	167	6.5438	7.6643	7.5249	7.1066	7.2099
				123	78.413	72.9417	75.5589	71.0334	74.4868
		p	5 deg		4.547	4.0498	3.3215	2.1881	3.5266
			10 deg	145	5.2783	5.146	4.6219	4.8098	4.9640
				123	14.7675	14.2896	11.8674	12.4536	13.3445
		q		189	8.4048	11.0526	12.6558	10.7433	10.7141
				123	0.6049	1.0591	0.74951	0.71692	0.7826

Structural	Wing	left	15%	167	99.7521	99.6586	99.9771	99.9771	99.8412
			35%	123	98.5251	98.5214	99.8882	99.8956	99.2076
		right			99.7524	87.2759	99.919	94.5213	95.3672
Engine	Left		1%	167	9.3677	6.0338	10.1375	4.584	7.5308
			10%	123	36.7297	36.5343	36.0772	35.7643	36.2764
					5.0642	3.0959	6.5	6.0702	5.1826
	Right		1%		18.9159	14.1244	17.3858	16.751	16.7943
			10%		1.5002	1.5452	1.7615	1.7286	1.6339
			1%	187	4.3127	10.7625	10.0828	6.6248	7.9457
Nominal	1A				0.033301	0.012109	0.069629	0.21494	0.0825
	1B				0.38056	0.31185	0.4942	0.30392	0.3726
	1C				0.73898	1.10148	0.92907	1.0496	0.9548
	1D_1				0.005791	0	0	0	0.0014
	1D_2				0	0	0	0	0.0000
	4				0	0	0	0	0.0000
	12				0	0	0	0	0.0000

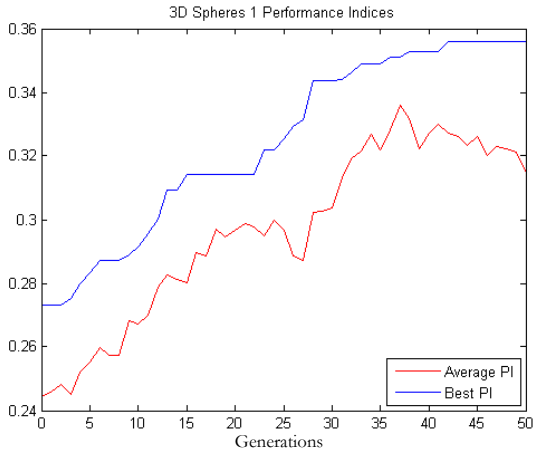

Figure 6.26—Performance Indices for 3-D Hyper-Spheres Trial 1

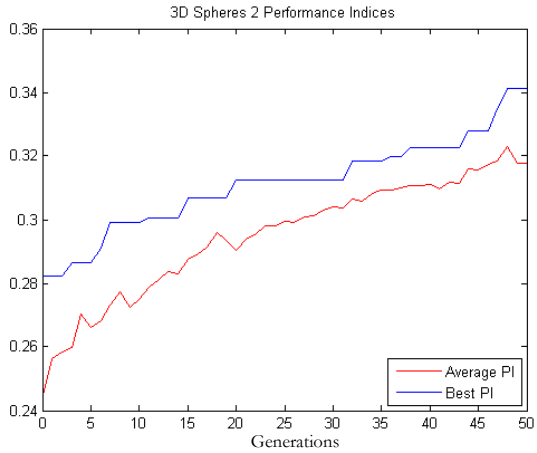

Figure 6.27—Performance Indices for 3-D Hyper-Spheres Trial 2

Figure 6.28—Performance Indices for 3-D Hyper-Spheres Trial 3

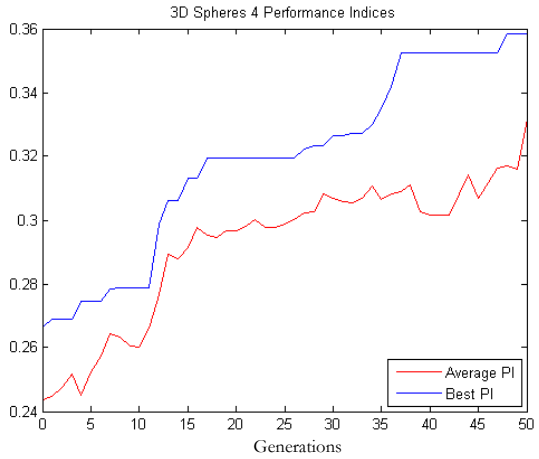

Figure 6.29—Performance Indices for 3-D Hyper-Spheres Trial 4

6.3.3 Hyper-Ellipsoids

Hyper-ellipsoids are a more complicated and more time-consuming shape to use in the detector optimization algorithm, since the semi-axis length may vary for each dimension. The average calculation time of the 3D ellipsoid trials was 52938 minutes, or about 37 days. The optimization parameters used for these trials are given in Table 6.13 The calculation times, coverage, overlapping, and number of detectors are shown in Table 6.14. Trials 1 and 3 were run using a 3.33GHz Intel Core i7 supporting 4 threads per trial. Trial 2 was calculated using a 2.0GHz Intel Core 2 Duo supporting 2 threads. The detection results of these three trials are located below in Table 6.15. For all detection results except for data listed as nominal, the number if the table represents detection rate. For nominal data sets, the number represents rate of false alarms. Figure 6.30 through Figure 6.32 show the performance index of the population throughout each trial. It should be noted that in these performance index plots, occasionally the performance index of the best individual shows a decrease, contrary to the purpose of elitist selection. This is not due to a fault in the elitist selection strategy. Rather, some of these trials were completed in segments and reassembled. Each time the trial is restarted, using continued optimization in the utility, the population is re-rated using the Monte Carlo Volume Estimation algorithm. Being that this is a numerical method, it does not return exactly the same values each time it is run. In these instances, when the trials were continued from where they left off, the best individual, which is the same set of detectors, happened to get rated slightly lower than it had been rated before. This never occurs within trials run consecutively.

Table 6.13—3-D Hyper-Ellipsoids Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Rotation Weight	2
Gene Relocation Constant	1
Gene Alteration Constant	0.20
Gene Rotation Constant	10
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Accuracy	0.001
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100
Number of Detectors Worst Limit	500

Table 6.14—Performance Parameters for 3-D Hyper-Ellipsoids

	Coverage %	Overlapping %	Number of Detectors	Time (minutes)
Trial 1	92.945	56.111	476	61534
Trial 2	93.47	52.52	461	76870
Trial 3	93.44	51.47	446	20411
Average	93.284	53.37	461	52938

Table 6.15—3D Hyper-Ellipsoids Detection Results after Optimization

Failure	Type	Location	Magnitude	Envelope	Hyper-Ellipsoids			
					Trial 1	Trial 2	Trial 3	Average
Actuator	aileron	left	5 deg	123	99.9462	99.9443	99.7887	99.8931
			8 deg		99.9551	99.9530	99.8076	99.9052
				187	99.9575	99.9554	99.4795	99.7975
		right		165	98.5201	99.8987	99.7702	99.3963
					98.8196	99.8177	99.8153	99.4842
				123	99.7622	99.8653	99.9537	99.8604
	rudder	left	8 deg	145	86.3164	87.2379	85.7401	86.4315
				123	47.8538	52.7156	47.6913	49.4202
		right		167	41.6099	52.9917	48.7674	47.7897
				123	39.1616	46.0749	40.6153	41.9506
	stabilator	left	2 deg	123	99.8923	99.9085	98.9565	99.5858
			8 deg		99.9690	99.9690	99.8226	99.9202
				145	99.9653	99.9653	99.9653	99.9653
		right		189	99.9628	86.7990	99.9052	95.5557
				123	99.9652	90.6587	99.5648	96.7296
Sensors	LFDB	r	3 deg	165	31.0651	32.3310	31.6577	31.6846
				123	82.0751	87.6956	86.2397	85.3368
		p	10 deg	167	0.8240	2.2706	2.0021	1.6989
				123	5.3282	6.5811	3.7316	5.2136
		q		187	0.5124	0.3591	0.0000	0.2905
				123	8.5329	8.4368	7.0774	8.0157
	LSB	r	3 deg	167	6.4417	7.1937	6.8625	6.8326
				123	73.0277	80.2227	81.1957	78.1487
		p	5 deg		3.3249	4.1067	2.6619	3.3645
			10 deg	145	5.8687	5.7760	5.9587	5.8678
				123	12.8115	13.9375	11.2695	12.6728
		q		189	16.8095	16.2050	13.5007	15.5051
				123	3.0449	2.5703	1.6294	2.4149
Structural	Wing	left	15%	167	99.9771	99.9771	99.9771	99.9771
			35%	123	99.8662	99.4265	99.9011	99.7313
		right			99.9801	99.9801	87.5021	95.8208
Engine	Left		1%	167	5.4482	4.9021	4.9130	5.0878
			10%	123	35.6808	37.1147	35.0815	35.9590
					4.8189	5.7944	3.8300	4.8144
	Right		1%		16.4758	20.2448	15.6649	17.4618
			10%		1.6525	2.2841	2.0158	1.9841
			1%	187	3.6648	6.1724	10.5124	6.7832

Nominal	1A	0.6206	0.0121	1.7559	0.7962
	1B	0.4863	0.3832	0.0793	0.3163
	1C	0.6613	0.7765	0.9773	0.8050
	1D_1	0.0000	0.0405	0.0521	0.0309
	1D_2	0.0000	0.0000	0.1564	0.0521
	4	0.0000	0.0000	0.0000	0.0000
	12	0.7055	0.0000	0.0000	0.2352

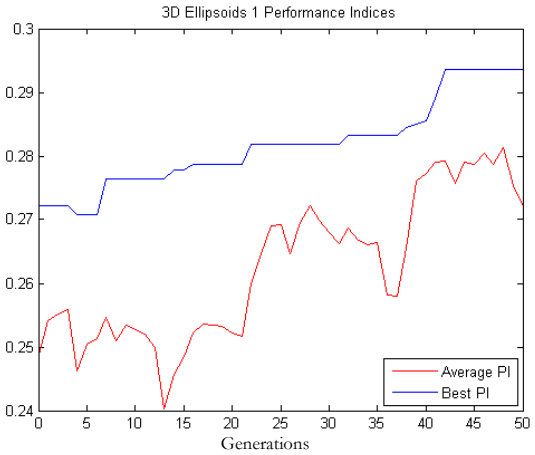

Figure 6.30—Performance Indices for 3-D Hyper-Ellipsoids Trial 1

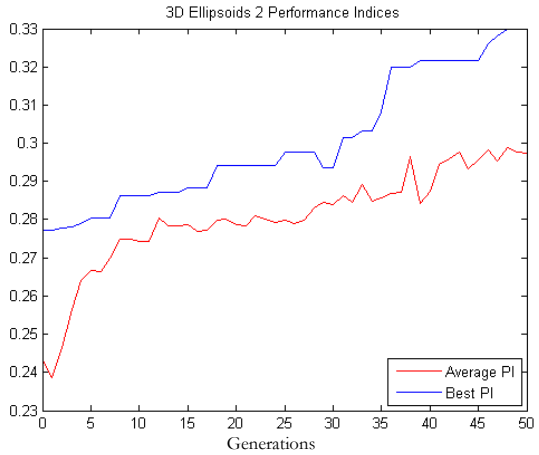

Figure 6.31—Performance Indices for 3-D Hyper-Ellipsoids Trial 2

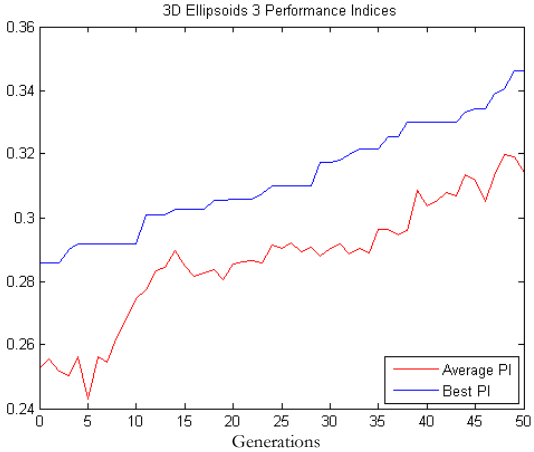

Figure 6.32—Performance Indices for 3-D Hyper-Ellipsoids Trial 3

6.3.4 Hyper-Rotational-Ellipsoids

Hyper-rotational-ellipsoids are more complicated than hyper-spheres and similar to hyper-ellipsoids, since the semi-axis ellipsoids, since the semi-axis length may vary for 1 dimension. The optimization parameters for these trials are located in these trials are located in Table 6.16. The average calculation time of the 3-D rotational ellipsoid trials was 23625 minutes, or about 16 days. The calculation times, coverage, overlapping, and number of detectors are shown in Table

6.17. These results were run using a 3.33 GHz Intel Core i7 with 12 GB of RAM, supporting 4 threads per trial. The detection results of these three trials are located below in

Table 6.18. For all detection results except for data listed as nominal, the number if the table represents detection rate. For nominal data sets, the number represents rate of false alarms. Figure 6.33 through Figure 6.35 show the performance index of the population throughout each trial.

Table 6.16—3-D Hyper-Rotational-Ellipsoids Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Rotation Weight	2
Gene Relocation Constant	1
Gene Alteration Constant	0.20
Gene Rotation Constant	10
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Accuracy	0.001
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100
Number of Detectors Worst Limit	500

Table 6.17—Performance Parameters for 3-D Hyper-Rotational-Ellipsoids

	Coverage %	Overlapping %	Number of Detectors	Time (minutes)
Trial 1	93.32	58.24	471	24392
Trial 2	92.96	53.28	471	25294
Trial 3	92.22	50.42	440	21190
Average	92.84	53.98	461	23625

Table 6.18—3D Hyper-Rotational-Ellipsoids Detection Results after Optimization

Failure	Type	Location	Magnitude	Envelope	Hyper-Rotational-Ellipsoids			
					Trial 1	Trial 2	Trial 3	Average
Actuator	aileron	left	5 deg	123	99.9424	99.9039	99.9443	99.9302
			8 deg		99.9530	99.9551	98.7578	99.5553
				187	99.9554	99.5301	99.6112	99.6989
		right		165	99.6467	99.5158	99.9333	99.6986
					99.6641	99.4578	99.8728	99.6649
				123	99.3749	99.9516	99.9516	99.7594
	rudder	left	8 deg	145	87.4917	87.1131	86.7740	87.1263
				123	46.7265	50.7630	46.1735	47.8877
		right		167	44.3613	49.7326	45.7527	46.6155
				123	40.0870	41.0258	40.8947	40.6692
	stabilator	left	2 deg	123	99.9055	92.1350	98.9270	96.9892
			8 deg		99.9707	99.9707	99.9707	99.9707
				145	99.9672	99.9672	99.9672	99.9672
		right		189	99.9666	85.8215	97.9169	94.5683
				123	99.2480	81.7004	97.9284	92.9589
Sensors	LFDB	r	3 deg	165	32.4878	31.7269	31.0743	31.7630
				123	86.6363	87.2550	86.6803	86.8572
		p	10 deg	167	2.4151	1.9919	4.0062	2.8044
				123	3.1763	5.6618	5.0150	4.6177
		q		187	0.0000	0.3747	0.1760	0.1836
				123	9.2509	6.7440	7.1932	7.7294
	LSB	r	3 deg	167	6.6733	6.8127	7.5274	7.0045
				123	83.1523	80.3611	79.8746	81.1293
		p	5 deg		3.4605	3.9828	3.5191	3.6541
			10 deg	145	5.8554	6.1890	5.4398	5.8281
				123	11.7938	14.6398	13.0823	13.1720
		q		189	15.8604	16.0018	13.2638	15.0420
				123	3.0347	2.1956	2.5031	2.5778

Structural	Wing	left	15%	167	96.8248	99.9771	99.9771	98.9263
			35%	123	88.0760	99.7050	99.6995	95.8268
		right			99.9801	99.9801	99.9312	99.9638
Engine	Left		1%	167	3.2944	2.0968	3.7089	3.0334
			10%	123	36.3769	35.8667	36.2669	36.1702
					4.5831	9.1794	4.8626	6.2084
	Right		1%		17.6966	18.8774	16.1798	17.5846
			10%		1.5815	2.2062	1.3566	1.7148
			1%	187	4.6947	4.9652	9.1711	6.2770
Nominal	1A				0.0696	0.0272	0.0000	0.0323
	1B				0.3806	0.7532	0.3092	0.4810
	1C				0.6533	0.4953	0.5917	0.5801
	1D_1				0.0000	0.0782	0.0000	0.0261
	1D_2				0.0082	0.0000	0.0000	0.0027
	4				0.0000	0.0000	0.0000	0.0000
	12				0.0000	0.0000	0.0000	0.0000

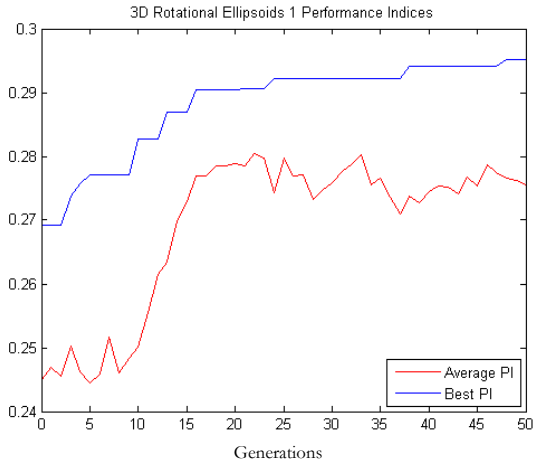

Figure 6.33—Performance Indices for 3-D Hyper-Rotational-Ellipsoids Trial 1

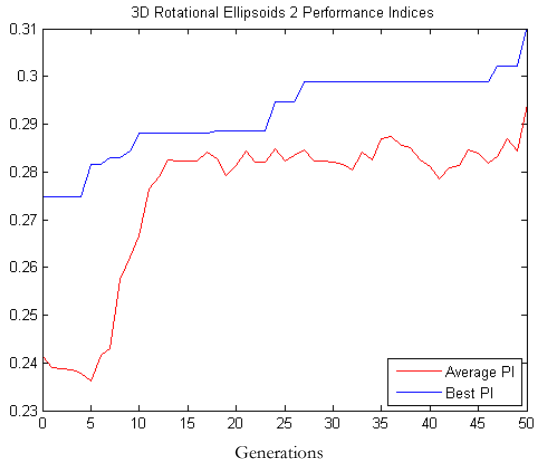

Figure 6.34—Performance Indices for 3-D Hyper-Rotational-Ellipsoids Trial 2

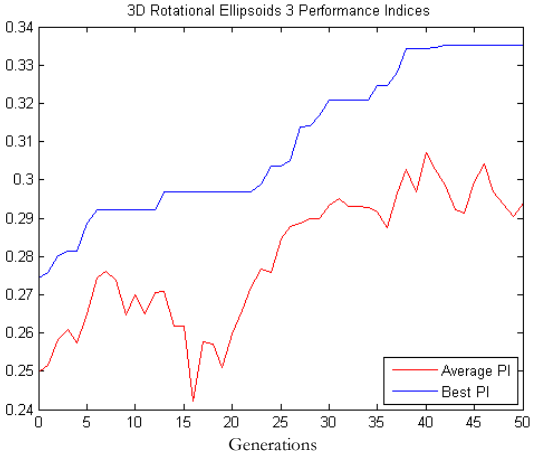

Figure 6.35—Performance Indices for 3-D Hyper-Rotational-Ellipsoids Trial 3

6.3.5 Hyper-Rectangles

Hyper-rectangles are the most time-consuming shape to calculate, since the semi-side length may vary for all dimensions. The average calculation time of the 3D rectangle trials was 47413 minutes, or about 33 hours. The optimization parameters used for these trials are located in Table 6.19. The calculation times, coverage, overlapping, and number of detectors are shown in Table 6.20. These results were run using a 3.6GHz Intel i7 with 6 GB of RAM, supporting 4 threads per trial. The results of these three trials are located below in Table 6.21. For all detection results except for data listed as nominal, the number if the table represents detection rate. For nominal data sets, the number represents rate of false alarms. Figure 6.36 through Figure 6.38 show the performance index of the population throughout each trial.

Table 6.19—3-D Hyper-Rectangles Optimization Parameters

Parameter	Value
Minimum Detector Radius	0.005
Maximum Number of Detectors	500
Non-Self Coverage	0.9999
Self Coverage	0.9999
Population Size	20
Number of Generations	50
Mutation Rate	30%
Chromosomal Mutation Rate	5%
Gene Relocation Weight	1
Gene Alteration Weight	2
Gene Rotation Weight	2
Gene Alteration Constant	0.20
Crossover Rate	20%
Maximum Number of Detectors to Cross	5
Add Rate	30%
Random Points to Attempt	2000
Number of Centers to Add	20
Weight Favoring Large Detectors	0
Weight Favoring Smaller Detectors	0
Accuracy	0.001
Remove Rate	20%
Detectors to Remove	5
Remove Threshold	0.5
Performance Index Weight for Overlapping	1
Performance Index Weight for Coverage	1
Performance Index Weight for Number of Detectors	1
Overlapping Best Limit	0.1
Overlapping Worst Limit	0.9
Coverage Best Limit	1
Coverage Worst Limit	0.8
Number of Detectors Best Limit	100

Number of Detectors Worst Limit	500

Table 6.20—Performance Parameters for 3-D Hyper-Rectangles

	Coverage %	Overlapping %	Number of Detectors	Time (minutes)
Trial 1	74.69	0.0236	491	38906
Trial 2	73.421	0.3345	480	53525
Trial 3	74.80	0.0695	493	49807
Average	74.31	0.1425	488	47413

Table 6.21—3D Hyper-Rectangles Detection Results after Optimization

Failure	Type	Location	Magnitude	Envelope	Hyper-Rectangles			
					Trial 1	Trial 2	Trial 3	Average
Actuator	aileron	left	5 deg	123	16.7522	60.3058	55.2495	44.1025
			8 deg		55.6789	93.0853	1.0883	49.9508
				187	36.4562	88.1625	21.0057	48.5415
		right		165	88.5911	74.6350	11.8981	58.3747
					87.5360	71.8498	13.0374	57.4744
				123	96.2386	62.8428	45.5429	68.2081
	rudder	left	8 deg	145	53.8942	68.2664	62.0028	61.3878
				123	29.8766	36.6517	33.2371	33.2551
		right		167	43.8975	17.3589	28.0115	29.7560
				123	29.4415	24.7800	36.6380	30.2865
	stabilator	left	2 deg	123	69.4654	58.5896	37.0226	55.0259
			8 deg		99.9655	36.7173	14.0933	50.2587
				145	99.9614	76.2299	8.6379	61.6097
		right		189	5.9055	84.4780	52.2280	47.5372
				123	23.2418	68.5259	60.7827	50.8501
Sensors	LFDB	r	3 deg	165	3.6387	5.6679	18.4449	9.2505
				123	55.0602	58.8341	71.2094	61.7012
		p	10 deg	167	3.3124	26.8510	1.3327	10.4987
				123	6.5574	6.8503	1.4053	4.9377
		q		187	0.5333	0.1848	0.1900	0.3027
				123	9.6040	6.5243	5.8573	7.3285
	LSB	r	3 deg	167	4.1160	6.2799	6.4467	5.6142
				123	48.8949	44.0653	51.3400	48.1001
		p	5 deg		3.8438	5.1513	0.5860	3.1937
			10 deg	145	4.8177	9.3920	3.0336	5.7478
				123	11.9119	13.3222	6.5876	10.6072
		q		189	15.0278	11.5793	10.5294	12.3788
				123	4.2302	4.0429	1.4664	3.2465

Structural	Wing	left	15%	167	84.8580	60.1140	70.0250	71.6657
			35%	123	86.3043	51.1057	76.0476	71.1525
		right			61.2914	79.3688	51.4434	64.0345
Engine	Left		1%	167	1.8731	2.6188	3.8493	2.7804
			10%	123	14.7602	26.1366	23.2005	21.3658
					3.8281	5.4350	3.0427	4.1019
	Right		1%		9.8760	7.8842	13.1226	10.2943
			10%		0.8236	0.6818	2.4190	1.3081
			1%	187	6.3770	13.7157	6.4998	8.8642
Nominal	1A				0.0000	0.0333	0.1423	0.0585
	1B				0.0000	0.0529	0.4519	0.1683
	1C				1.2182	0.5408	0.1205	0.6265
	1D_1				0.1274	0.0319	0.9411	0.3668
	1D_2				0.0000	0.0110	0.2168	0.0759
	4				0.0000	0.0000	0.0000	0.0000
	12				0.8307	0.0000	0.4294	0.4200

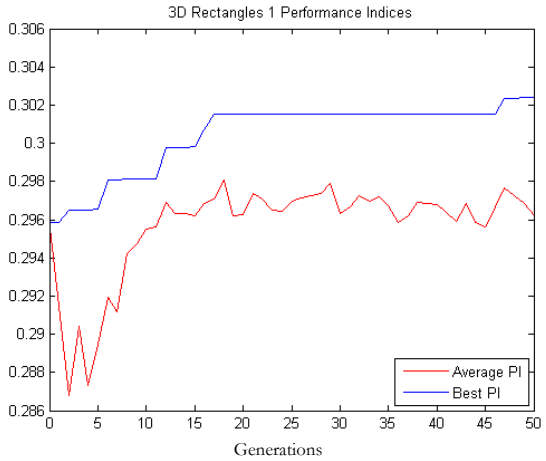

Figure 6.36—Performance Indices for 3-D Hyper-Rectangles Trial 1

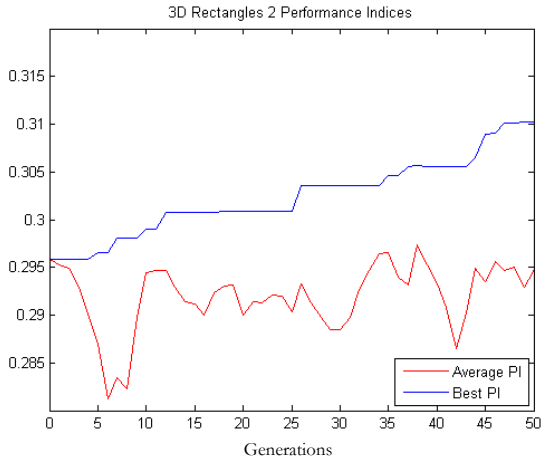

Figure 6.37—Performance Indices for 3-D Hyper-Rectangles Trial 2

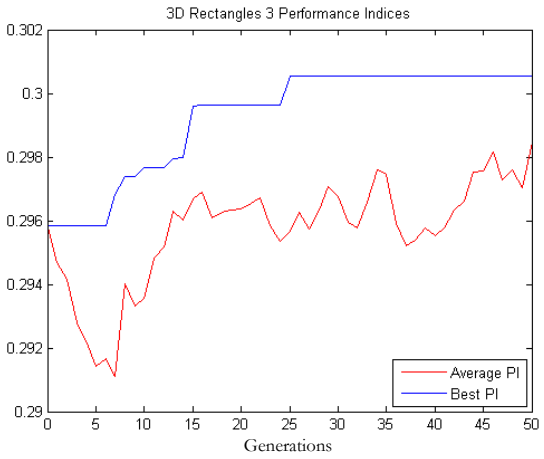

Figure 6.38—Performance Indices for 3-D Hyper-Rectangles Trial 3

6.3.6 Comparison of Results Among Shapes

All detector shapes exhibited similar or better detection performance to the Phase I detectors, with the exception of hyper-rectangles. In some cases, the average detection rate exhibited by a detector shape for a particular failure was lower than that of the Phase I sets. However, only for hyper-rectangles was this difference significant. For the hyper-spheres, hyper-ellipsoids, and hyper-rotational-ellipsoids, it is likely that detection decreased in some cases due to the fact that the performance index weights were set such that improvement in coverage, overlapping, or number of detectors was equal. Better coverage, and thus potentially better detection, could likely be achieved using performance index weights that favor improvements in coverage of the solution space.

Comparing the detection results of the hyper-spheres, hyper-ellipsoids, and hyper-rotational-ellipsoids, it can be observed that hyper-ellipsoids tended to achieve the best detection performance of these shapes, though this is not always the case. In some cases hyper-rotational-ellipsoids achieved better detection than hyper-ellipsoids. Generally, hyper-spheres performed lower than the hyper-ellipsoids or hyper-rotational-ellipsoids. However, hyper-spheres were the quickest to calculate among all four shapes used in the GA. However, hyper-rectangles were the quickest for calculating detection performance, taking about half the time of hyper-spheres.

The hyper-rectangles trials exhibited sporadic detection. The nature of the inconsistency of the detection results indicates that poor coverage is the cause, rather than ineffectiveness of the identifiers or the detector shape. As seen in the 2-D results section, hyper-rectangles required significantly more detectors to achieve adequate coverage. Since these trials were limited to 500 detectors, hyper-rectangles could likely achieve better detection results if more detectors were allowed in the set. This is supported by the slow performance index increase observed in hyper-rectangle trials. This could mean that an adequate set of hyper-rectangle detectors may need a higher number of detectors than an adequate set of hyper-spheres, hyper-ellipsoids, or hyper-rotational-ellipsoids, which may cost more to run online in the detection scheme, though this would require investigation.

6.4 6-Dimensional Example with Detection Results

The purpose of this trial is to illustrate the ability of this design environment to cope with high-dimensionality, a key issue for the AIS. Due to computational time constraints, only one trial per shape is presented. These results are presented in Table 6.22 below. For all detection results except for data listed as nominal, the number if the table represents detection rate. For nominal data sets, the number represents rate of false alarms. The identifiers used to define this self are the roll-, pitch-, and yaw-rate neural network estimates and DQEE (52) for roll-, pitch-, and yaw-rate, which are parameters derived from the sensor-based neural network roll-, pitch-, and yaw-rate estimates.

Numerical results for optimization of hyper-rectangles were not available due to the extremely high calculation time of this shape in 6 dimensions. Due to this fact, multithreading of the addition genetic operator for hyper-rectangles, the most computationally intense algorithm function in higher dimensions, was implemented to help reduce the computational time of this shape. Prior to implementation of this function, a single generation of 6-D hyper-rectangles took greater than 3 weeks to calculate. Unfortunately, actual calculation time is not known due to a power failure. Regardless, without significant computational power and time available, this shape is not

recommended for optimization in higher dimensions. Due to other characteristics of hyper-rectangles, however, this shape may still be viable for use in failure detection online, possibly without optimization of the detectors beforehand. The results presented in Table 6.22 for hyper-rectangles are for a Phase I, non-optimized detector set. These are intended only to give a performance basis for this shape in higher dimensions.

Table 6.22—6-D Trial Detection

Failure	Type	Location	Magnitude	Envelope	Hyper-Spheres	Hyper-Ellipsoids	Hyper-Rotational-Ellipsoids	Hyper-Rectangles
Actuator	aileron	left	5 deg	123	25.2954	90.2484	45.0502	8.5355
			8 deg		76.2284	52.5037	99.2944	72.3241
				187	47.4695	76.1670	72.7971	3.8945
		right		165	11.9154	4.3457	8.0070	1.5070
					9.4866	10.1812	7.2028	0.7557
				123	64.9755	86.3039	85.0349	1.2939
	rudder	left	8 deg	145	25.0016	34.0575	32.6595	17.9378
				123	18.978	44.5013	43.0640	4.7800
		right		167	14.5393	32.3509	31.3266	6.4516
				123	51.0922	49.2517	85.0891	36.8499
	stabilator	left	2 deg	123	99.9449	26.2347	98.3600	1.3196
			8 deg		99.6646	52.8048	97.0083	1.4014
				145	86.5128	40.2070	32.9487	3.0605
		right		189	84.1533	47.4845	46.4609	1.1733
				123	79.0832	73.1339	79.1916	36.8137
Sensors	LFDB	r	3 deg	165	98.8314	99.4387	99.6628	89.3757
				123	33.7748	35.4046	31.4411	14.3686
		p	10 deg	167	75.1587	75.2925	76.2339	28.5559
				123	24.7725	23.5856	23.5507	12.8351
		q		187	48.4827	46.8860	47.8138	23.2527
				123	74.4099	73.9841	75.8964	59.9626
	LSB	r	3 deg	167	98.7732	98.8529	99.3688	78.0753
				123	78.6295	78.4487	78.3985	22.9174
		p	5 deg		57.0056	56.6932	56.1214	28.2262
			10 deg	145	79.3356	79.5833	79.0512	20.4709
				123	75.3036	76.9191	77.7623	43.6270
		q		189	32.465	30.3917	30.9925	13.0634
				123	21.42	14.8235	36.0490	0.3032
Structural	Wing	left	15%	167	33.8115	27.7176	20.8945	1.6233
			35%	123	0.11229	0.3124	0.0653	0.1266
		right			76.5737	87.4287	78.6368	13.4057
Engine	Left		1%	167	0.037286	3.0224	0.0548	0.0243

		10%	123	7.5394	32.3293	14.8190	1.9688
				0.55719	3.6683	1.6469	0.3746
	Right	1%		0.031075	0.2087	0.8302	0.2930
		10%		0.070944	0.1021	0.2405	0.0986
		1%	187	1.6642	3.2351	2.9850	1.7324
Nominal	1A			0.0000	0.0000	0.0000	0.0000
	1B			0.0000	0.0000	0.0000	0.0000
	1C			0.0000	0.0000	0.0000	0.0000
	1D_1			0.0000	0.0000	0.0000	0.0000
	1D_2			0.0000	0.0000	0.0000	0.0027
	4			0.0000	0.0000	0.0000	0.0000
	12			0.0000	0.0000	0.0000	0.0000

Table 6.23—6-D Calculation Time Results

	Calculation Time (minutes)	Detection Time (seconds)
Hyper-Spheres	1370.8	7389
Hyper-Ellipsoids	23845	43171
Hyper-Rotational-Ellipsoids	24658	40752
Hyper-Rectangles	---	2129

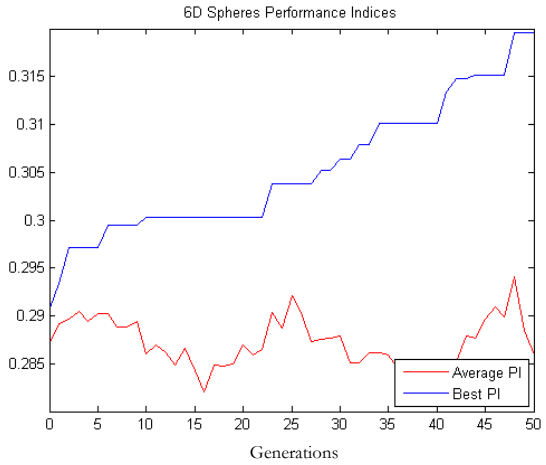

Figure 6.39—Performance Indices for 6-D Hyper-Spheres Trial

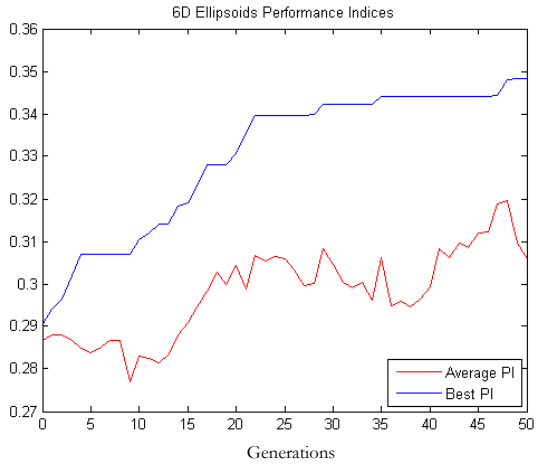

Figure 6.40—Performance Indices for 6-D Hyper-Ellipsoids Trial

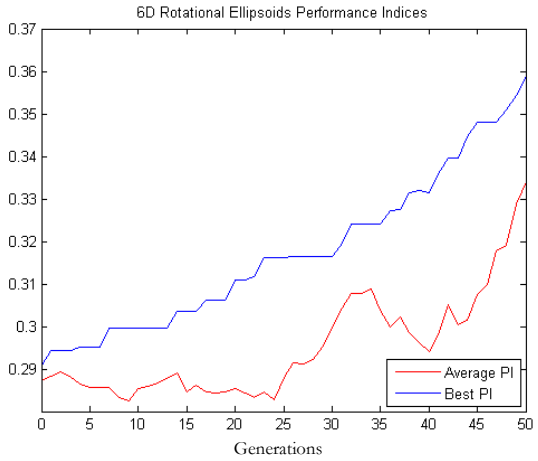

Figure 6.41—Performance Indices for 6-D Hyper-Rotational-Ellipsoids Trial

The detection performance for each of these shapes was sporadic and not adequate. The detection was particularly poor for hyper-rectangles, similar to the 2-D and 3-D cases. This indicates that more detectors are needed to achieve adequate coverage of the solution space in 6 dimensions, for each of the shapes. The calculation times for this trial illustrate the computational requirements of each shape. As seen previously in the 2-D and 3-D cases, hyper-spheres took the least time to calculate using the genetic algorithm, followed by hyper-ellipsoids and hyper-rotational-ellipsoids, which had nearly equivalent calculation times of over 17 times that of hyper-spheres. The performance indices for each of the shapes show steady increases in the performance index of the best individual. This indicates that more generations of the genetic algorithm could help the detector sets to achieve better detection performance without the allowance of more detectors.

7 Conclusions and Recommendations

An evolutionary algorithm for the generation and optimization of immunity-based failure detectors has been successfully designed, implemented, and demonstrated. Results obtained through the use of this utility illustrate the strengths and weaknesses of each of the detector shapes implemented.

Hyper-spheres performed well in all areas. Though this shape did not always provide the best detection performance, it took significantly less time than the other shapes to optimize through the use of the genetic algorithm, and took significantly less time to determine detection performance than hyper-ellipsoids or hyper-rotational-ellipsoids.

Hyper-ellipsoids and hyper-rotational-ellipsoids tended to take equivalent calculation times, both for optimization and detection. However, hyper-ellipsoids tended to perform better than hyper-rotational-ellipsoids with respect to failure detection. Both shapes took significantly longer to calculate than hyper-spheres, regardless of number of dimensions. These shapes do not seem to suffer the same effects of high dimensionality that limits the use of hyper-rectangles.

Hyper-rectangle detectors take the longest to calculate in higher dimensions. Their calculation time is also highly susceptible to the effects of dimensionality. In addition, hyper-rectangles need a significantly higher number of detectors for adequate coverage of the solution space. However, hyper-rectangles exhibit the lowest overlapping among the detector shapes. In addition, detection results were the quickest to run for hyper-rectangles, meaning that this shape would likely be a good choice for running online in a detection scheme.

Further exploration of the optimization parameters is needed to fully define the capabilities and possible detection improvements available through the use of the IFDOT Utility. Comparison of online calculation costs for optimized detector sets for each detector shape should be performed to determine the feasibility of running each shape online, the number of detectors that may feasibly be included in a detector set intended to run online for each shape, and the coverage of the solution space that may be achieved utilizing the most detectors feasible for online calculation for each shape. Integration of all four detector shapes into a structured genetic algorithm should be investigated, to take advantage of the strengths given by each shape.

Bibliography

1. **Totah, J, KrishnaKumar, K and Viken, S.** Stability, Maneuverability, and Safe Landing in the Presence of Adverse Conditions. *Integrated Resilient Aircraft Control Technical Plan, Aviation Safety Program, Aeronautics Research Mission Directorate, NASA.* [Online] 2007. http://www.aeronautics.nasa.gov/nra_pdf/irac_tech_plan_c1.pdf.

2. **Srivastava, A N, Mah, R W and Meyer, C.** Automated detection, diagnosis, prognosis to enable mitigation of adverse events during flight. *Integrated Vehicle Health Management Technical Plan, Aviation Safety Program, Aeronautics Research Mission Directorate, NASA.* [Online] 2008. http://www.aeronautics.nasa.gov/nra_pdf/ivhm_tech_plan_c1.pdf.

3. **Young, S D and Quon, L.** Integrated Intelligent Flight Deck. *Integrated Intelligent Flight Deck Technical Plan, Aviation Safety Program, Aeronautics Research Mission Directorate, NASA.* [Online] 2007. http://www.aeronautics.nasa.gov/nra_pdf/iifd_tech_plan_c1.pdf.

4. **Young, R and Rohn, D.** Aircraft Aging and Durability Project. *Aircraft Aging and Durability Project Technical Plan, Aviation Safety Program, Aeronautics Research Mission Directorate, NASA.* [Online] 2007. http://www.aeronautics.nasa.gov/nra_pdf/aad_tech_plan_c1.pdf.

5. *NASA's Aviation Safety Program.* **White, J.** Reno, Nevada : s.n., January 9-12, 2006. 44th Annual AIAA Aerospace Sciences Meeting.

6. *Fault Detection for the Aircraft Distribution Systems Using Impedance Estimation.* **Zhou, Qian, Sumner, M and Thomas, D.** York : s.n., April 2-4, 2008. 4th IET Conference on Power Electronics, Machines and Drives, 2008. pp. 666-670.

7. *A Diagnostic Approach for Electro-Mechanical Actuators in Aerospace Systems.* **Balaban, E, et al.** Big Sky, MT : IEEE, March 7-14, 2009. Aerospace Conference, 2009.

8. *Application of Fault Detection and Isolation to a Boeing 747-100/200 Aircrafrt.* **Marcos, A, Ganguli, S and Balas, G.** Monterey, California : AIAA, August 2002. Proceedings of the AIAA Guidance, Navigation and Control Conference. pp. Paper 02-4944.

9. *Linear Parameter varying Control Synthesis for Actuator Failure, Based on Estimated Parameter.* **Shin, J Y, Wu, N E and Belcastro, C.** Monterey, California : AIAA, August 2002. Proceedings of the AIAA Guidance, Navigation, and Control Conference. pp. Paper 02-4546.

10. *Artificial Immune System Approaches for Aerospace Applications.* **KrishnaKumar, K.** Reno, Nevada : AIAA-2003-0457, 2003. Proceedings of the 41st Aerspace Sciences Meeting and Exhibit.

11. *Negative Selection Algorithm for Aircraft Fault Detection.* **Dasgupta, D, et al.** [ed.] G. Nicosia et al. (Eds.). Catania, Sicily, Italy : s.n., September 13-16, 2004. Proceedings from the 3rd International Conference on Artificial Immune Systems. pp. 1-13.

12. *Integrated Framework for Aircraft Sub-System Failure Detection, Identification, and Evaluation Based on the Artificial Immune System Paradigm.* **Perhinschi, M G, Moncayo, H and Davis, J.** s.l. : submitted to AIAA Journal of Aircraft, May 2009.

13. *Evolutionary Algorithm for Artificial Immune System-Based Failure Detector Generation and Optimization.* **Perhinschi, M G, Moncayo, H and Davis, J.** Chicago, IL : AIAA, August 2009. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

14. *Self-nonself discrimination in a computer.* **Forrest, S, et al.** Los Alamitos, California : IEEE Compter Society Press, 1994. Proceedings of the IEEE Symposium on Research in Security and Privacy. pp. 202-212.

15. **Holland, J H.** *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor, Michigan : University of Michigan Press, 1975. revised and republished in 1992.

16. **Davis, L.** *Handbook of Genetic Algorithms.* New York : Van Nostrand Reinhold, 1991. pp. 1-22.

17. *Active Aircraft Fault Detection and Isolation.* **Glavaski, S and Elgersma, M.** 2001. Proceedings of the IEEE Systems Readiness Technology Conference. pp. 692-705.
18. *A New Approach to Robust Fault Detection and Identification.* **Saif, M and Guan, Y.** July 1993. IEEE Transactions on Aerospace and Electronic Systems. Vol. 29, Issue 3, pp. 685-695.
19. *EKF Based Surface Fault Detection and Reconfiguration in Aircraft Control Systems.* **Caliskan, F and Hajiyev, Ch. M.** Chicago, IL : IEEE, 2000. Proceedings of the American Control Conference, 2000.Vol. 2, pp. 1220-1224.
20. *Detection of Sensor Abrupt Faults in Aircraft Control Systems.* **Samara, P A, et al.** s.l. : IEEE, 2003. Proceedings of the 2003 IEEE Conference on Control Applications. Vol. 2, pp. 1366-1371.
21. *Neural Network Based Fault Detection and Identification for Fighter Control Surface Failure.* **Zhengdao, Zhang and Weihua, Zhang.** June 17-19, 2009. Chinese Control and Decision Conference 2009.
22. *Comparison of Fault Detection Techniques: Problem and Solution.* **Lou, S J, Budman, H and Duever, T A.** Anchorage, Alaska : s.n., 2002. Proceedings of the American Control Conference. pp. 4513-4518.
23. *A Fault Tolerant Flight Control System for Sensor and Actuator Failures Using Neural Networks.* **Napolitano, M R, Younghawn, A and Seanor, B.** No. 2, 2000, Aricraft Design, Vol. Vol. 3.
24. **Gonzalez, F A and Dasgupta, D.** *Anomaly Detection Using Real-Valued Negative Selection.* Netherlands : Kluwer Academic Publishers, 2003.
25. **Dasgupta, D.** An Overview of Artificial Immune Systems and Their Application. *Artificial Immune Systems and Their Applications.* s.l. : Springer-Verlag, 1999, pp. 3-18.
26. **Dasgupta, D.** Advances in Artificial Immune Systems. *IEEE Computational Intelligence Magazine.* November 2006.
27. *Revisiting Negative Selection Algorithms.* **Ji, Z and Dasgupta, D.** No. 2, July 2007, Evolutionary Computation Journal, Vol. 15.
28. **Linnemeyer, P A.** The Immune System--An Overview. *The Seattle Treatment Project.* [Online] October 2008. [Cited: June 28, 2009.] http://www.thebody.com/content/art1788.html.
29. **Ji, Z.** *Negative Selection Algorithms: From Thymus To V-Detector.* Memphis, Tennessee : s.n., August, 2006.
30. *Architecture for an Artificial Immune System.* **Hofmeyr, S A and Forrest, S.** [ed.] 473. No. 4, Winter 2000, MIT Press Journals, Evolutionary Computation, Vol. 8, p. 443. Posted Online March 13, 2006. http://www.mitpressjournals.org/doi/abs/10.1162/106365600568257.
31. *Artificial Immune System (AIS) Research in the Last Five Years.* **Ji, Z and Dasgupta, D.** Canberra, Australia : s.n., December 2003. Congress on Evolutionary Computation Conference (CEC).
32. *The Immune System.* **Jerne, N K.** 1, 1973, Scientific American, Vol. 229, pp. 52-60.
33. *Towards a Network Theory of the Immune System.* **Jerne, N K.** 1974, Ann. Immunol (Inst. Pasteur), Vol. 125C, pp. 373-389.
34. **Rowe, G W.** *The Theoretical Models of Biology.* 1st. s.l. : Oxford University Press, 1994.
35. *Data Mining Approaches for Intrusion Detection.* **Lee, W and Stolfo, S.** Berkeley, CA : USENIX Association, 1998. Proceedings of the 7th USENIX Security Symposium. pp. 79-94.
36. *Learning Using an Artifiicial Immune System.* **Hunt, J E and Cooke, D E.** s.l. : Journal of Network and Computer Applications, 1996, Vol. 19, pp. 189-212.
37. *Artificial Immune Systems: A Novel Paradigm to Pattern Recognition.* **De Castro, L and Timmis, J.** [ed.] J M Corchado, L Alonso and C Fyfe. University of Paisley, UK : SOCO-2002, 2002. Artificial Neural Networks in Pattern Recognition. pp. 67-84.
38. *An Immunity-Based Technique to Characterize Intrusions in Computer Networks.* **Dasgupta, D and Gonzalez, F.** s.l. : IEEE, June 2002. IEEE Transactions on Evolutionary Computation. Vol. 6, pp. 281-291.

39. *Immunity-Based Intrusion Detection System: A General Framwork.* **Dasgupta, D.** October 1999. Proceedings of the 22nd National Information Systems Security Conference (NISSC). pp. 147-160.
40. *Anomaly Detection in Multidimensional Data Using Negative Selection Algorithm.* **Dasgupta, D and Majumdar, N.** s.l. : ACM, 2002. Proceedings of the Congress on Evolutionary Computation, 2002. pp. 1039-1044.
41. *Negative Selection and Niching by and Artificial Immune System for Network Intrusion Detection.* **Kim, Jungwon and Bentley, Peter.** 1999. In Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference. pp. 149-158.
42. *Augmenting an Artificial Immune Network.* **Neal, M, Hunt, J and Timmis, J.** s.l. : IEEE, 1998. 1998 IEEE International Conference on Systems, Man, and Cybernetics. Vol. 4, pp. 3821-3826.
43. *Artificial Immune Systems in Industrial Applications.* **Dasgupta, D and Forrest, S.** Honolulu, HI : IEEE, 1999. Proceedings of the Second International Conference on Intelligent Processing and Maufacturing of Materials. Vol. Vol. 1, pp. 257-267.
44. *Study of Fault Diagnosis in Brushless Machines Based on Artificial Immune Algorithm.* **Tao, W, et al.** Montreal, Quebec, Canada : IEEE, 2006. IEEE Symposium on Industrial Electronics.
45. *Robot Error Detection Using an Artificial Immune System.* **Canham, R, Jackson, A H and Tyrrell, A.** s.l. : IEEE, 2003. Proceedings from the IEEE Dod Conference on Evolvable Hardware, 2003.
46. *****.** [Online] 2003. http://www.k-team.com.
47. *Robot Fault Tolerance Using an Embryonic Array.* **Jackson, A, Canham, R and Tyrrell, A.** s.l. : IEEE, 2003. Proceedings of the 2003 NASA/DoD Conference on Evolvable Machines.
48. *Research on Fault-Tolerant Controller for Mobile Robot Based on Artificial Immune Principle.* **Yang, B, Fan, S and Shi, M.** s.l. : IEEE, 2007. Third International Conference on Natural Computation.
49. *A Novel Artificial-Immune-Based Approach for System-Level Fault Diagnosis.* **Elhadef, M, Das, S and Nayak, A.** s.l. : Proceedings of the First International Conference on Availability, Reliability, and Security, 2006.
50. *Artificial Immune System-Based Aircraft Failure Detection and Identification Using an Integrated Hierarchial Multi-Self Strategy.* **Moncayo, H, Perhinschi, M G and Davis, J.** s.l. : accepted for publication in the AIAA Journal of Guidance, Control, and Dynamics, January 2010.
51. *Immunity-Based Aircraft Failure Detection and Identification Using an Integrated Hierarchial Multi-Self Strategy.* **Moncayo, H, Perhinschi, M G and Davis, J.** Chicago, IL : AIAA, August 2009. Proceedings of the AIAA Guidance, Navigation, and Control Conference.
52. **Moncayo, Hever Y.** *Immunity-Based Detection, Identification, and Evaluation of Aircraft Sub-System Failures.* Morgantown, WV : West Virginia University, 2009.
53. *Evolutionary Algorithm for Artificial Immune System-Based Failure Detector Generation and Optimization.* **Davis, J, Perhinschi, M G and Moncayo, H.** No. 2, s.l. : AIAA Journal of Guidance, Control, and Dynamics, March-April 2010, Vol. 33, pp. 302-320.
54. *Intergrated Framework for Aircraft Sub-System Failure Detection, Identification, and Evaluation Based on Artificial Immune System Paradigm.* **Davis, J, Perhinschi, M G and Moncayo, H.** Chicago, IL : AIAA, August 2009. Proceedings of the AIAA Guidance, Navigation, and Control Conference.
55. *Is Negative Selection Appropriate for Anomaly Detection?* **Stibor, T, et al.** Washington, DC, USA : ACM Press, 25-29 June 2005. Proceedings of the 2005 Conference on Genetic and Evolutionary Computation. Vol. 1, pp. 321-328.
56. *A Comparative Study of Real-Valued Negative Selection to Statistical Anomaly Detection Techniques.* **Stibor, T, Timmis, J and Eckert, C.** 2005. Proceedings of the International Conference on Artificial Immune Systems. pp. 262-275.
57. *Applicability Issues of the Real-Valued Negative Selection Algorithms.* **Ji, Z and Dasgupta, D.** Seattle, Washington : ACM, 2006. Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. pp. 111-118.

58. *Immunity-Based Aircraft Fault Detection System.* **Dagupta, D, et al.** Chicago, IL : AIAA, September 20-24, 2004. Proceedings of the American Institute of Aeronautics and Astronautics Conference.
59. *The Curse of Dimensionality in Data Mining and Time Series Prediction.* **Verleysen, M and François, D.** Berlin : © Springer-Verlag, 2005, IWANN 2005, LNCS 3512, pp. 758 – 770.
60. *A Neural Network Approach for the Reduction of the Dimensionality of Slowly Time-Varying Electomagnetic Inverse Problems.* **Morabito, F C and Coccorese, E.** May 2006. IEEE Transaction on Magnetics. Vols. Vol. 32, Issue 3, Part 1, pp. 1306-1309.
61. *Dimensionality Reduction and Feature Extraction Applications in Identifying Computer Users.* **Bleha, S A and Obaidat, M S.** March-April 1991. IEEE Transactions on Systems, Man, and Cybernetics. Vol. 21, Issue 2, pp. 452-456.
62. *Nonlinear Methods For Clustering and Dimensionality Reduction.* **Eghbalnia, H, Assadi, A and Carew, J.** s.l. : IJCNN '99, July 10-16, 1999. International Joint Conference on Neural Networks. Vol. 2, pp. 1004-1009.
63. **Goldberg, D E.** Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA : Addison-Wesley, 1989, pp. 1-24.
64. **Michalewicz, Z.** Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Germany : Springer-Verlag, 1992, pp. 13-42.
65. *Control System Optimization Using Genetic Algorithms.* **KrishnaKumar, K and Goldberg, D E.** Number 3, s.l. : Journal of Guidance, Control, and Dynamics, 1992.
66. *An Immune Fault Detection System with Automatic Detector Generation by Genetic Algorithms.* **Amaral, J L M, et al.** Rio de Janeiro : s.n., October 20-24, 2007. Seventh International Conference on Intelligent Systems Design and Applications. pp. 283-288.
67. *An Evolutionary Approach to Generate Fuzzy Anomaly (Attack) Signatures.* **Gonzalez, F, et al.** s.l. : IEEE, June 18-20, 2003. IEEE Systems, Man, and Cybernetics Information Assurance Workshop, 2003. pp. 251-259.
68. *An Evolutionary Algorithm to Generate Hyper-Ellipsoid Detectors for Negative Selection.* **Shapiro, J M, Lamont, G B and Peterson, G L.** s.l. : ACM Press, 2005. Proceedings of the 2005 Conference on Evolutionary Computation. pp. 337-344.
69. *Genetic Algorithms-based Detector Generation in Negative Selection Algorithm.* **Gao, X Z, Ovaska, S J and Wang, X.** Logan, UT : IEEE, 2006. IEEE Mountain Workshop on Adaptive and Learning Systems. pp. 133-137.
70. *A General Framework for Evolving Multi-Shaped Detectors in Negative Selection.* **Balachandran, S, et al.** Honolulu, HI : IEEE, 2007. IEEE Symposium on Foundations of Computational Intelligence. pp. 401-408.
71. **Balachandran, Sankalp.** *Multi-Shaped Detector Generation Using Real-Valued Representation For Anomaly Detection.* Memphis, TN : University of Memphis, December 2005.
72. **Dasgupta, D and McGregor, D R.** *sGA: A Structured Genetic Algorithm.* 1992.
73. *****.** Euclidean Distance. *NIST.* [Online] http://www.itl.nist.gov/div897/sqg/dads/HTML/euclidndstnc.html.
74. ***. Mahalanobis Distance. *AI ACCESS.* [Online] http://www.aiaccess.net/English/Glossaries/GlosMod/e_gm_mahalanobis.htm.
75. *Estimating the Detector Coverage in a Negative Selection Algorithm.* **Ji, Z and Dasgupta, D.** Washington, DC : ACM, 2005. Proceedings of the 2005 Conference on Genetic and Evolutionary Computation.
76. *A Randomized Real-Valued Negative Selection Algorithm.* **Gonzalez, F, Dasgupta, D and Nino, L.** UK : ICARIS, 2003, 2003. Second International Conference on Artificial Immune System.

77. *****.** Introduction to Monte Carlo Methods. *Computational Science Education Project.* [Online] 1995. http://www.ipp.mpg.de/~rfs/comas/Helsinki/helsinki04/CompScience/csep/csep1.phy.ornl.gov/mc/mc.html.
78. *Using the Triangle Inequality to Accelerate k-Means.* **Elkan, C.** Washington, DC : IEEE, 2003. Proceedings of the 12th International Conference on Machine Learning.
79. *Integrated System for Immunity-Based Failure Detection, Identification, and Evaluation.* **Perhinschi, M G, Moncayo, H and Davis, J.** McLean, VA : NASA, November 2009. poster within the Integrated Resilient Aircraft Control Project, NASA Aviation Safety Program, Technical Conference.
80. *Augmented Negative Selection Algorithm with Variable-Coverage Detectors.* **Ji, Z and Dasgupta, D.** s.l. : CEC2004, June 19-23, 2004. Congress on Evolutionary Computation, 2004. Vol. Vol. 1, pp. 1081-1088.
81. *Artificial Immune System-Based Aircraft Failure Detection and Identfication Over Extended Flight Envelope.* **Moncayo, H, Perhinschi, M G and Davis, J.** s.l. : The Aeronautical Journal, December 2009.
82. *Artificial Immune System-Based Aircraft Failure Evaluation over Extended Flight Envelope.* **Moncayo, H, Perhinschi, M G and Davis, J.** Toronto, Canada : AIAA, August 2010. accepted for publication in the Proceedings of the AIAA Guidance, Navigation, and Control Conference.
83. *V-Detector: An Efficient Negative Selection Algorithm with "Probably Adequate" Coverage.* **Ji, Z and Dasgupta, D.** No. 10, s.l. : Information Sciences, April 29, 2009, Vol. 179, pp. 1390-1406.
84. *Real-Valued Negative Selection Algorithm with Variable-Sized Detectors.* **Ji, Z and Dasgupta, D.** s.l. : Springer-Verlag, 2004. Genetic and Evolutionary Computation Conference, 2004. . pp. 287-298.
85. *Development of a Detection Scheme for Aircraft Engine Failures Based on the Artificial Immune System Paradigm.* **Perhinschi, M G, et al.** Toronto, Canada : AIAA, August 2010. accepted for publication in the Proceedings of the AIAA Guidance, Navigation, and Control Conference.
86. *In-Flight Actuator Failure Detection and Identification for a Reduced Size UAV Using Artificial Immune System Approach.* **Sanchez, S P, et al.** Chicago, IL : AIAA, August 2009. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

Appendices

A Additional Results Figures and Tables

Table A.1—Full Clustering Comparison Results—500 Clusters, Phase I only

Failure	Type	Location	Magnitude	Envelope	500 Clusters			
					Trial 1	Trial 2	Trial 3	Average
Actuator	aileron	left	5 deg	123	99.7234	99.9404	99.4064	99.6901
			8 deg		99.9551	99.9508	92.198	97.3680
				187	99.8684	99.9534	91.2591	97.0270
		right		165	99.1477	85.7376	99.9432	94.9428
					98.6564	85.4679	99.8728	94.6657
				123	86.8972	96.4996	98.4361	93.9443
	rudder	left	8 deg	145	86.125	86.023	85.2679	85.8053
				123	42.6244	46.907	43.5449	44.3588
		right		167	42.8396	47.6894	48.4624	46.3305
				123	40.8363	47.3291	43.5	43.8885
	stabilator	left	2 deg	123	98.2304	98.5832	99.6059	98.8065
			8 deg		99.969	99.6451	99.696	99.7700
				145	99.9653	99.4737	99.6953	99.7114
		right		189	99.9657	99.9628	74.7705	91.5663
				123	99.4917	99.7476	74.6553	91.2982
Sensors	LFDB	r	3 deg	165	31.7523	31.7269	32.3195	31.9329
				123	85.5846	85.0769	85.9007	85.5207
		p	10 deg	167	1.6257	2.8139	4.2035	2.8810
				123	5.7193	5.2419	2.2383	4.3998
		q		187	0.31199	1.1538	0.39739	0.6211
				123	6.7773	8.476	5.5199	6.9244
	LSB	r	3 deg	167	7.8212	6.9646	7.4427	7.4095
				123	76.513	78.3794	78.5556	77.8160
		p	5 deg		3.7618	2.9465	2.158	2.9554
			10 deg	145	4.7251	5.8845	5.1857	5.2651
				123	12.767	14.16	12.767	13.2313
		q		189	13.331	15.0349	10.8104	13.0588
				123	2.218	2.0469	0.88189	1.7156
Structural	Wing	left	15%	167	84.1295	99.9771	93.9604	92.6890
			35%	123	90.9086	99.7911	98.1165	96.2721
		right			99.974	99.9801	92.0746	97.3429
Engine	Left		1%	167	3.0202	3.279	2.3732	2.8908
			10%	123	35.9919	36.9023	35.88	36.2581
					3.4344	4.4461	3.4478	3.7761
	Right		1%		14.2532	17.3503	15.9579	15.8538
			10%		1.5106	1.983	1.4985	1.6640

		1%	187	8.364	10.5193	13.9431	10.9421
Nominal	1A			0.95059	0.80225	0.003027	0.5853
	1B			0.31185	0.32506	0.27221	0.3030
	1C			0.53549	1.2289	0.39894	0.7211
	1D_1			0.16215	0	0	0.0541
	1D_2			0	0	0	0.0000
	4			0	0	0	0.0000
	12			0	0	0	0.0000

Table A.2—Full Clustering Comparison Results—2000 Clusters, Phase I only

Failure	Type	Location	Magnitude	Envelope	2000 Clusters			
					Trial 1	Trial 2	Trial 3	Average
Actuator	aileron	left	5 deg	123	94.7976	99.9443	99.9232	98.2217
			8 deg		66.0530	99.9530	99.9551	88.6537
				187	97.2416	99.9554	99.9575	99.0515
		right		165	99.7579	99.6195	99.9481	99.7752
					99.7673	99.5681	99.9040	99.7465
				123	99.6359	99.7811	99.9558	99.7909
	rudder	left	8 deg	145	89.1080	89.4762	87.5832	88.7225
				123	56.6897	55.1802	55.0801	55.6500
		right		167	53.0376	51.0738	39.9887	48.0334
				123	51.9013	51.4984	45.9457	49.7818
	stabilator	left	2 deg	123	84.6936	86.5016	99.3447	90.1800
			8 deg		99.9707	96.0017	99.9724	98.6483
				145	99.9672	94.5582	99.9692	98.1649
		right		189	99.9684	99.9666	99.9666	99.9672
				123	99.9704	99.0687	99.8259	99.6217
Sensors	LFDB	r	3 deg	165	32.3610	30.8968	33.4448	32.2342
				123	90.7645	89.2052	88.3795	89.4497
		p	10 deg	167	7.9737	16.7755	14.6331	13.1274
				123	7.7629	8.1218	9.5051	8.4633
		q		187	3.5608	0.4479	0.9429	1.6506
				123	17.2326	10.0375	13.4408	13.5703
	LSB	r	3 deg	167	9.6514	9.3103	9.4920	9.4846
				123	85.0795	86.9962	85.8030	85.9596
		p	5 deg		5.6419	5.9700	8.1648	6.5922
			10 deg	145	8.7593	8.9261	9.7784	9.1546
				123	16.2881	17.4741	21.7478	18.5033
		q		189	21.8789	18.6814	21.7004	20.7536
				123	5.5276	3.7231	4.3932	4.5480
Structural	Wing	left	15%	167	98.6403	99.6586	92.7475	97.0155
			35%	123	99.9121	99.1407	98.9538	99.3355
		right			94.3654	98.4611	99.9817	97.6027
Engine	Left		1%	167	6.9528	8.7338	6.3365	7.3410
			10%	123	38.1560	40.0508	37.4580	38.5549
					8.1202	8.0879	6.9240	7.7107
	Right		1%		22.2306	17.1713	22.9527	20.7849
			10%		3.1942	2.9329	2.5713	2.8995
			1%	187	10.6693	19.8450	12.5517	14.3553
Nominal	1A				0.1877	0.0848	0.5994	0.2906

	1B	0.3700	0.6052	0.5391	0.5048
	1C	1.2745	1.1085	0.8970	1.0933
	1D_1	0.1332	0.0290	0.0087	0.0569
	1D_2	0.0165	0.0000	0.0165	0.0110
	4	0.0000	0.0000	0.0000	0.0000
	12	0.0000	0.0000	0.0000	0.0000

Table A.3—Full Clustering Comparison Results—5000 Clusters, Phase I only

Failure	Type	Location	Magnitude	Envelope	5000 Clusters			
					Trial 1	Trial 2	Trial 3	Average
Actuator	aileron	left	5 deg	123	99.9520	99.9539	99.6715	99.8591
			8 deg		99.9572	99.9594	99.7520	99.8895
				187	99.9595	99.9615	99.9534	99.9581
		right		165	99.6986	39.9437	95.9730	78.5384
					98.4069	41.7394	95.1248	78.4237
				123	99.6611	82.2497	98.1456	93.3521
	rudder	left	8 deg	145	91.7291	88.3591	88.7897	89.6260
				123	60.5326	61.4121	62.0980	61.3476
		right		167	57.3915	53.8023	53.8253	55.0064
				123	59.9567	57.1841	56.7109	57.9506
	stabilator	left	2 deg	123	99.9233	99.9483	99.8686	99.9134
			8 deg		99.9707	99.9707	99.9690	99.9701
				145	99.9672	99.9672	99.9653	99.9666
		right		189	99.9591	89.0270	99.6042	96.1968
				123	95.7872	85.3475	99.6449	93.5932
Sensors	LFDB	r	3 deg	165	32.1189	13.1575	31.5194	25.5986
				123	90.5979	90.4580	90.9216	90.6592
		p	10 deg	167	15.4184	21.7034	25.9130	21.0116
				123	9.8945	9.3206	10.4820	9.8990
		q		187	1.2009	6.8114	3.0885	3.7003
				123	15.4181	24.3375	23.3606	21.0387
	LSB	r	3 deg	167	12.0767	12.0692	12.3730	12.1730
				123	87.5435	84.4755	81.2880	84.4357
		p	5 deg		8.4678	6.3333	7.4968	7.4326
			10 deg	145	12.7617	12.3355	12.9285	12.6752
				123	21.6743	19.6970	20.3123	20.5612
		q		189	22.6602	31.1852	33.0535	28.9663
				123	7.3179	12.3607	11.0817	10.2534
Structural	Wing	left	15%	167	99.8207	99.4164	99.7959	99.6777
			35%	123	99.9322	99.9139	98.1421	99.3294
		right			99.9817	99.9801	99.7524	99.9047
Engine	Left		1%	167	9.1681	10.6639	6.8388	8.8903
			10%	123	43.8063	42.4179	42.7233	42.9825
					12.3457	12.7242	18.7468	14.6056
	Right		1%		25.3278	22.8817	22.1981	23.4692
			10%		5.4437	3.6372	4.0732	4.3847
			1%	187	20.9658	26.4993	25.8514	24.4388
Nominal	1A				2.6763	1.1716	0.4117	1.4199

	1B	0.6448	0.8563	1.0175	0.8395
	1C	1.6038	1.5476	1.1058	1.4191
	1D_1	0.1824	0.1795	0.2403	0.2008
	1D_2	0.0027	0.0467	0.1098	0.0531
	4	0.0000	0.2061	0.0000	0.0687
	12	1.0812	0.0051	0.8972	0.6612

Table A.4—2-D Shape Results

		Coverage %	Overlap %	Number of Detectors
Hyper-Spheres	Trial 1	97.62	22.97	97
	Trial 2	97.76	22.62	102
	Trial 3	96.58	20.49	89
	Average	**97.32**	**22.03**	**96**
Hyper-Ellipsoids	Trial 1	96.77	12.17	102
	Trial 2	97.3	16.17	98
	Trial 3	97.29	18.32	103
	Average	**97.12**	**15.55**	**101**
Hyper-Rotational-Ellipsoids	Trial 1	96.05	13.62	108
	Trial 2	95.05	11.31	96
	Trial 3	97.56	18.44	97
	Average	**96.22**	**14.46**	**100**
Hyper-Rectangles	Trial 1	97.93	0	104
	Trial 2	97.73	0	205
	Trial 3	97.91	0	143
	Average	**97.86**	**0**	**151**

Table A.5—2-D Calculation Time Results

	Time (minutes)			
	Trial 1	**Trial 2**	**Trial 3**	**Average**
Hyper-Spheres	334.9181	680.3432	291.7751	435.6788
Hyper-Ellipsoids	25648	8291.5	8093.9	14011.13
Hyper-Rotational-Ellipsoids	15826	20944	8400.9	15056.97
Hyper-Rectangles	488.45	186.54	170.43	281.8067

Table A.6—3-D Shape Results

		Coverage %	Overlap %	Number of Detectors
Hyper-Spheres	Trial 1	92.105	43.954	445
	Trial 2	91.848	40.301	476
	Trial 3	92.499	44.821	471
	Trial 4	92.786	48.155	435
	Average	**92.310**	**44.308**	**457**
Hyper-Ellipsoids	Trial 1	92.945	56.111	476
	Trial 2	93.47	52.52	461
	Trial 3	93.44	51.47	446
	Average	**93.284**	**53.37**	**461**
Hyper-Rotational-Ellipsoids	Trial 1	93.32	58.24	471
	Trial 2	92.96	53.28	471
	Trial 3	92.22	50.42	440
	Average	**92.84**	**53.98**	**461**
Hyper-Rectangles	Trial 1	74.69	0.0236	491
	Trial 2	73.421	0.3345	480
	Trial 3	74.80	0.0695	493
	Average	**74.31**	**0.1425**	**488**

Table A.7—3-D Calculation Time Results

Calculation Time (min)					
	Trial 1	Trial 2	Trial 3	Trial 4	Average
Hyper-Spheres	2716	2632	2518	2565	2607.75
Hyper-Ellipsoids	61534	76870	20411	---	52938.33
Hyper-Rotational-Ellipsoids	24392	25294	21190	---	23625.33
Hyper-Rectangles	38906	53525	49807	---	47412.67

Table A.8—3-D Detection Time Results

Detection Time (seconds)					
	Trial 1	Trial 2	Trial 3	Trial 4	Average
Hyper-Spheres	6556	7100	7007	6580	6887.667
Hyper-Ellipsoids	46895	41995	41667	---	44281
Hyper-Rotational-Ellipsoids	42178	42300	41700	---	42059.33
Hyper-Rectangles	3530	3420	3446	---	3465.333

Table A.9—3-D Average Detection Results for Shape Comparison

Failure	Type	Location	Magnitude	Envelope	500 Clusters	Hyper-Spheres	Hyper-Ellipsoids	Hyper-Rotational-Ellipsoids	Hyper-Rectangles
					Average	Average	Average	Average	Average
Actuator	aileron	left	5 deg	123	99.6901	99.0841	99.8931	99.9302	44.1025
			8 deg		97.3680	99.0726	99.9052	99.5553	49.9508
				187	97.0270	98.8127	99.7975	99.6989	48.5415
		right		165	94.9428	98.3793	99.3963	99.6986	58.3747
					94.6657	98.1310	99.4842	99.6649	57.4744
				123	93.9443	98.3219	99.8604	99.7594	68.2081
	rudder	left	8 deg	145	85.8053	85.5748	86.4315	87.1263	61.3878
				123	44.3588	44.8272	49.4202	47.8877	33.2551
		right		167	46.3305	46.4521	47.7897	46.6155	29.7560
				123	43.8885	39.7060	41.9506	40.6692	30.2865
	stabilator	left	2 deg	123	98.8065	92.7265	99.5858	96.9892	55.0259
			8 deg		99.7700	99.6886	99.9202	99.9707	50.2587
				145	99.7114	99.9653	99.9653	99.9672	61.6097
		right		189	91.5663	98.7517	95.5557	94.5683	47.5372
				123	91.2982	96.9479	96.7296	92.9589	50.8501
Sensors	LFDB	r	3 deg	165	31.9329	30.5918	31.6846	31.7630	9.2505
				123	85.5207	83.9673	85.3368	86.8572	61.7012
		p	10 deg	167	2.8810	3.1654	1.6989	2.8044	10.4987
				123	4.3998	5.1666	5.2136	4.6177	4.9377
		q		187	0.6211	0.1434	0.2905	0.1836	0.3027
				123	6.9244	5.0246	8.0157	7.7294	7.3285
	LSB	r	3 deg	167	7.4095	7.2099	6.8326	7.0045	5.6142
				123	77.8160	74.4868	78.1487	81.1293	48.1001
		p	5 deg		2.9554	3.5266	3.3645	3.6541	3.1937
			10 deg	145	5.2651	4.9640	5.8678	5.8281	5.7478
				123	13.2313	13.3445	12.6728	13.1720	10.6072
		q		189	13.0588	10.7141	15.5051	15.0420	12.3788
				123	1.7156	0.7826	2.4149	2.5778	3.2465
Structural	Wing	left	15%	167	92.6890	99.8412	99.9771	98.9263	71.6657
			35%	123	96.2721	99.2076	99.7313	95.8268	71.1525
		right			97.3429	95.3672	95.8208	99.9638	64.0345
Engine	Left		1%	167	2.8908	7.5308	5.0878	3.0334	2.7804
			10%	123	36.2581	36.2764	35.9590	36.1702	21.3658
					3.7761	5.1826	4.8144	6.2084	4.1019
	Right		1%		15.8538	16.7943	17.4618	17.5846	10.2943
			10%		1.6640	1.6339	1.9841	1.7148	1.3081
			1%	187	10.9421	7.9457	6.7832	6.2770	8.8642

Nominal	1A	0.5853	0.0825	0.7962	0.0323	0.0585
	1B	0.3030	0.3726	0.3163	0.4810	0.1683
	1C	0.7211	0.9548	0.8050	0.5801	0.6265
	1D_1	0.0541	0.0014	0.0309	0.0261	0.3668
	1D_2	0.0000	0.0000	0.0521	0.0027	0.0759
	4	0.0000	0.0000	0.0000	0.0000	0.0000
	12	0.0000	0.0000	0.2352	0.0000	0.4200

B IFDOT Utility User's Guide

Table of Contents

Introduction

This guide is intended to address the various capabilities of the West Virginia University Immunity-Based Failure Detector Optimization and Testing (IFDOT) Utility. The guide is organized according to the order in which data must be processed for use in a failure detection control scheme. Several steps in this process contain a variety of options available to the User. Full exploration of the parameters available throughout the program is not given here, however complete explanation of each of the algorithms and the algorithms pertinent parameters is included. In addition, a detailed walkthrough for each of the algorithms is included. A demonstration directory has been included with the program, labeled 'Demo'. Each of the walkthroughs will center around these included files. Instructions are given throughout the walkthrough segments to allow the user to follow along with the guide utilizing the demo files, however, different files may be used as desired. In addition, extra demo files are included for additional practice with the IFDOT Utility.

The IFDOT Utility was designed using MATLAB version R2008a for Windows, and is intended to work with this version of MATLAB. As MATLAB is a versatile and dynamic language, compatibility cannot be guaranteed for past or future versions of MATLAB, although the program has been utilized with MATLAB versions 2007 through 2009. Linux compatibility is possible for comparable versions of MATLAB using UNIX-compatible MATLAB.

Before running the program, be sure to add the 'Ver. 3.0-WVU Failure Detector GA' and 'GA Functions V3.0' directories to path. To do this, open MATLAB, click 'File', and select 'Set Path'. This opens a dialog box within MATLAB. Select 'Add to Path' and navigate to the location of each of these directories, then click 'Save' and close the dialog box. Set the MATLAB Current Directory to the location in which you wish files to be saved. Changing the MATLAB Current Directory after running the program can cause the program to behave unexpectedly, depending on the files located in the new Current Directory, and is not recommended. To run the IFDOT Utility, run 'WVU_IBFDO_V3' from the MATLAB command window, or open 'WVU_IBFDO_V3.m' and click 'Run'. This loads the menu shown below in Figure B.1.

Upon running the IFDOT Utility for the first time in a particular Current Directory, it is recommended to set the program options. To do this, click on the 'Options' menu, and click 'Select Options'. This is shown in Figure B.2. Several options are available in this menu, shown in Figure B.3 below. These are discussed starting from the top and explained in more detail later as applicable. 'Use Multithreading Where Applicable' allows the program to open multiple processes to calculate the genetic algorithm more quickly. 'Maintain Version 6 Compatibility' makes the program save output files to a different version of '.mat' file which can be opened through MATLAB version 6. 'Override Monte Carlo Parameters in Clusters file' tells the program to use different parameters for Monte Carlo Volume Estimation in the genetic algorithm than those used in clustering. If this is selected, the parameters entered in the edit boxes will be used instead. Finally, 'Use parallel computing to increase GA speed' initiates a variation on genetic algorithm which allows multiple computers to participate in calculating the genetic algorithm to increase speed. Note that multithreading is not compatible with this variation and will simply not be used.

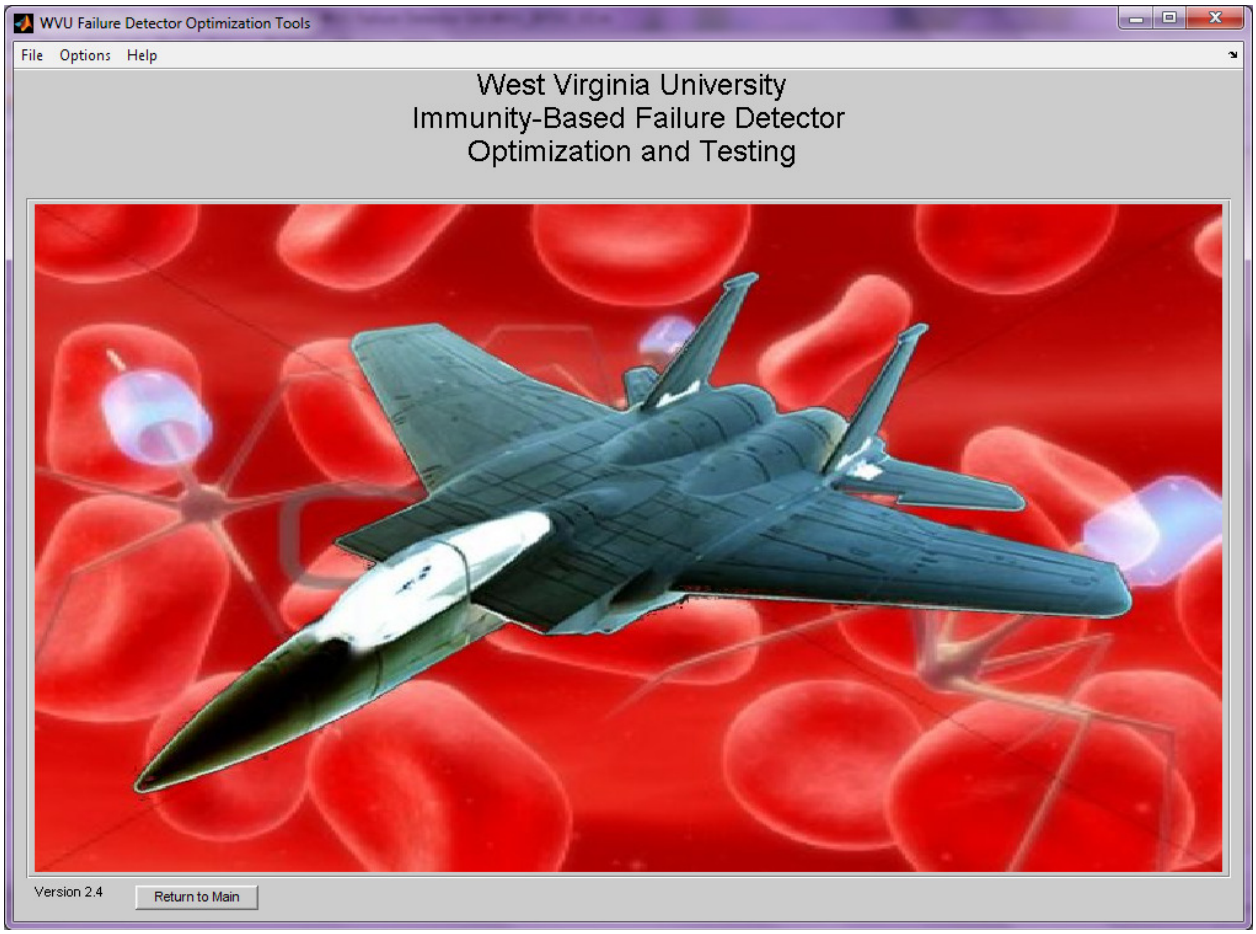

Figure B.1—IFDOT Main Menu

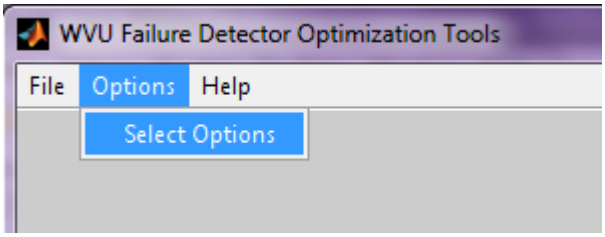

Figure B.2—Opening Options Menu

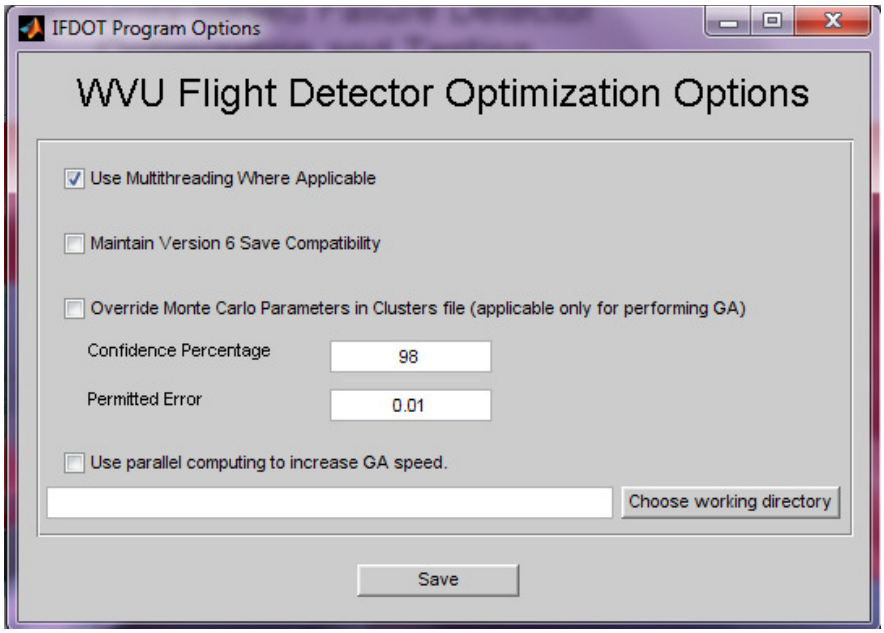

Figure B.3—Options Menu

This User's Guide is available through the IFDOT Utility by clicking the 'Help' menu and selecting 'Load Help File', as seen in Figure B.4. This loads the menu shown below in Figure B.5. This is opened as a separate window so that the user can follow along with the guide using the IFDOT Utility.

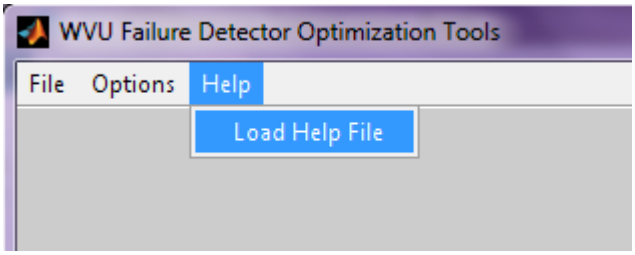

Figure B.4—Opening Help File

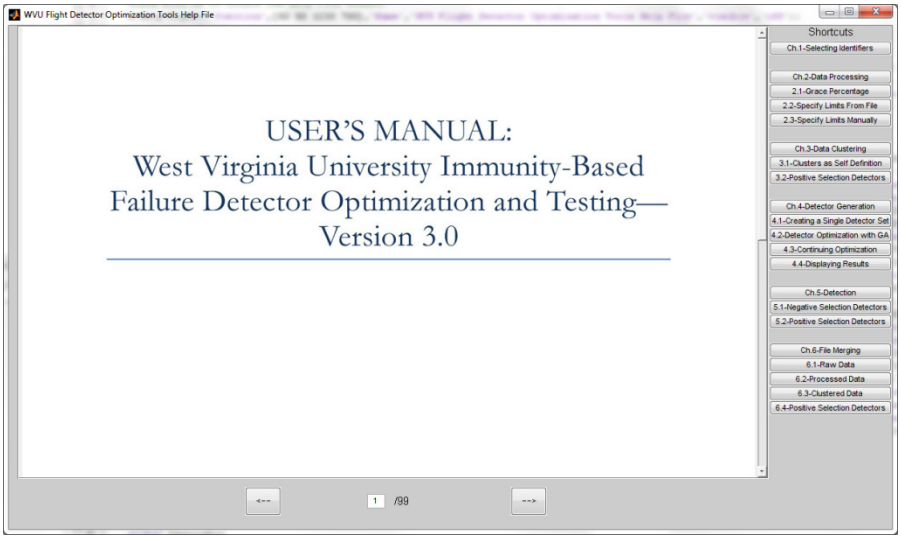

Figure B.5—Help File Menu

Chapter 1—Selecting Identifiers

The single most important step during the creation of immunity-based detectors is the selection of appropriate identifiers, capable of capturing the signature of various failures. This depends both on the nature of the system and the nature of the failures to be detected. A higher number of identifiers may be needed to determine the occurrence of a failiure, however, higher dimensionality is accompanied by increasingly high computational cost to generate and optimize the detectors, though the IFDOT Utility is compatible with an unlimited number of dimensions of the solution space.

Once the identifiers are chosen, data files are defined as a column of time history data for each of the identifiers. These data files may contain either normal data from the system with no failures present, or abnormal data collected from the system when a failure is present. Normal data is used to define the self for generation of negative-selection detectors, or for defining positive-selection detectors. Normal data files should be saved using the variable name 'sensors'. Abnormal data is used to determine the detection performance of a set of detectors for the particular failure contained in the abnormal data. Abnormal data files should be saved using the variable name 'dataN'.

Full raw data files are supplied with the 'Demo' directory, however, these are not directly used in the program. These files are truncated to 2-, 3-, and 6-dimensional data sets. The 2-dimensional data contains the roll-rate and pitch-rate as identifiers, or columns 4 and 5 in the original files. The 3-dimensional data set contains roll-, pitch-, and yaw-rate identifiers, or columns 4, 5, and 6 in the original files. The 6-dimensional data files contain velocity, angle of attack, sideslip angle, roll-rate, pitch-rate, and yaw-rate as identifiers, or columns 1-6 in the full data files.

Chapter 2—Data Processing

Processing a data file is done before clustering the data to reduce the computational load on the computer and to normalize the values of the identifiers to values between 0 and 1. Processing data involves normalizing the time-history data for each of the identifiers between values of 0 and 1, and eliminating duplicate data points. Specifying the normalization limits may be done in three ways. A grace percentage may be specified around the perimeter of the solution space, or the maximums and minimums may be specified either manually or from another processed data file.

2.1 Processing with Normalization Grace Percentage

To process data, a raw data file containing only the desired identifiers is loaded into the program. This is done by clicking on the 'File' menu, select 'Data Processing', then select 'Load Raw Data', as shown below in Figure B.6. This loads the menu shown in Figure B.7. Click on the Browse button, select the desired raw data file, and click open. This is shown in Figure B.8. To follow along with this guide, navigate to the 'Demo' directory, open the folder called '2-Truncated Raw Data', and select the file labeled 'selfdata1-2D.mat'.

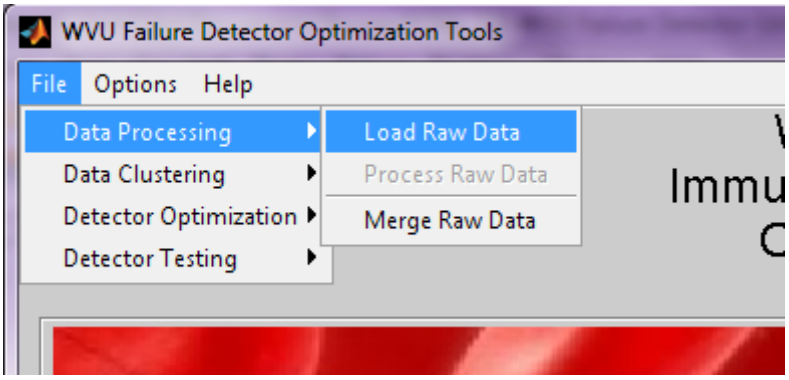

Figure B.6—Opening Load Raw Data Menu

Next, click on the 'File' menu, select 'Data Processing', then select 'Process Raw Data', as shown in Figure B.9. This loads the menu shown below in Figure B.10. This menu defaults to the normalization method which uses a grace percentage to define the normalization limits of the data. If this method is desired but the normalization menu is not visible, click on the radio button labeled 'Use normalization grace percentage'.

Figure B.7—Load Raw Data Menu

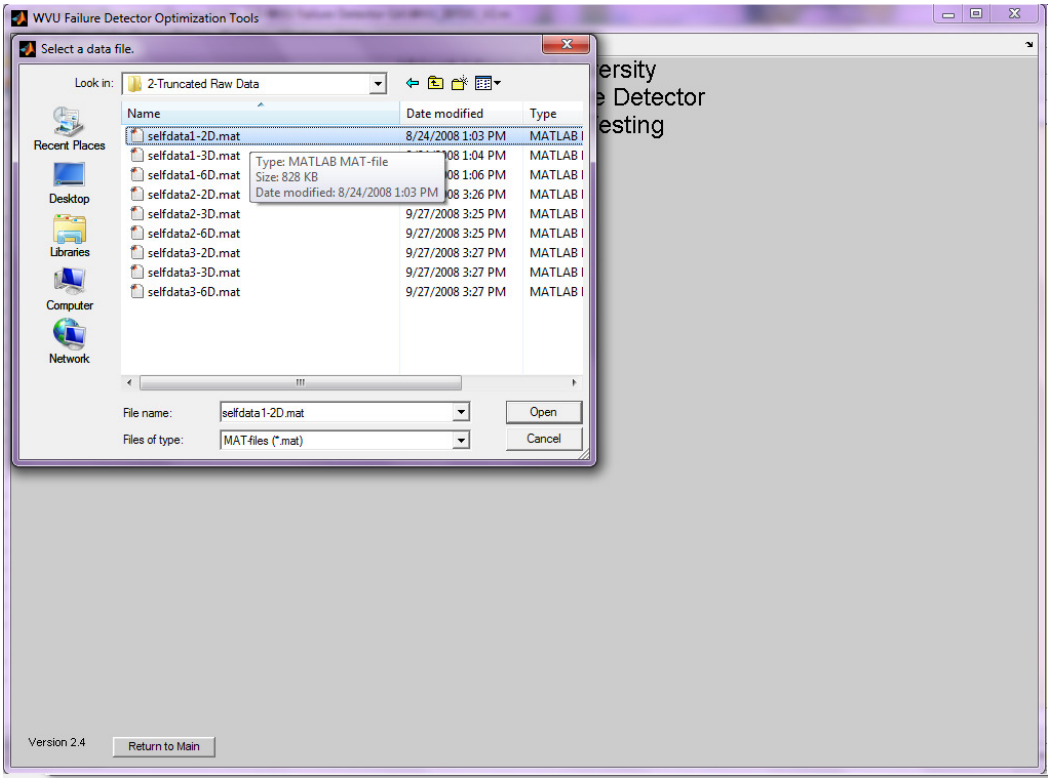

Figure B.8—Load Raw Data Browser

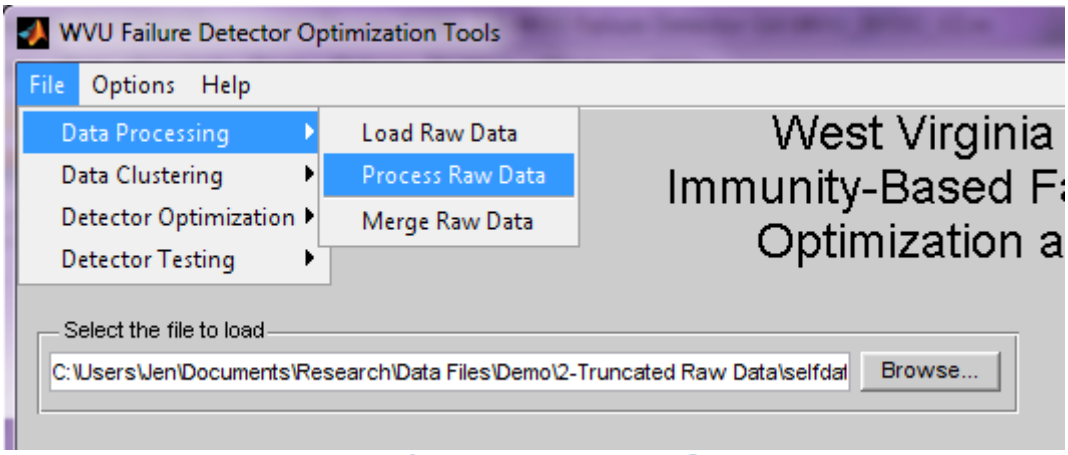

Figure B.9—Opening Process Raw Data Menu

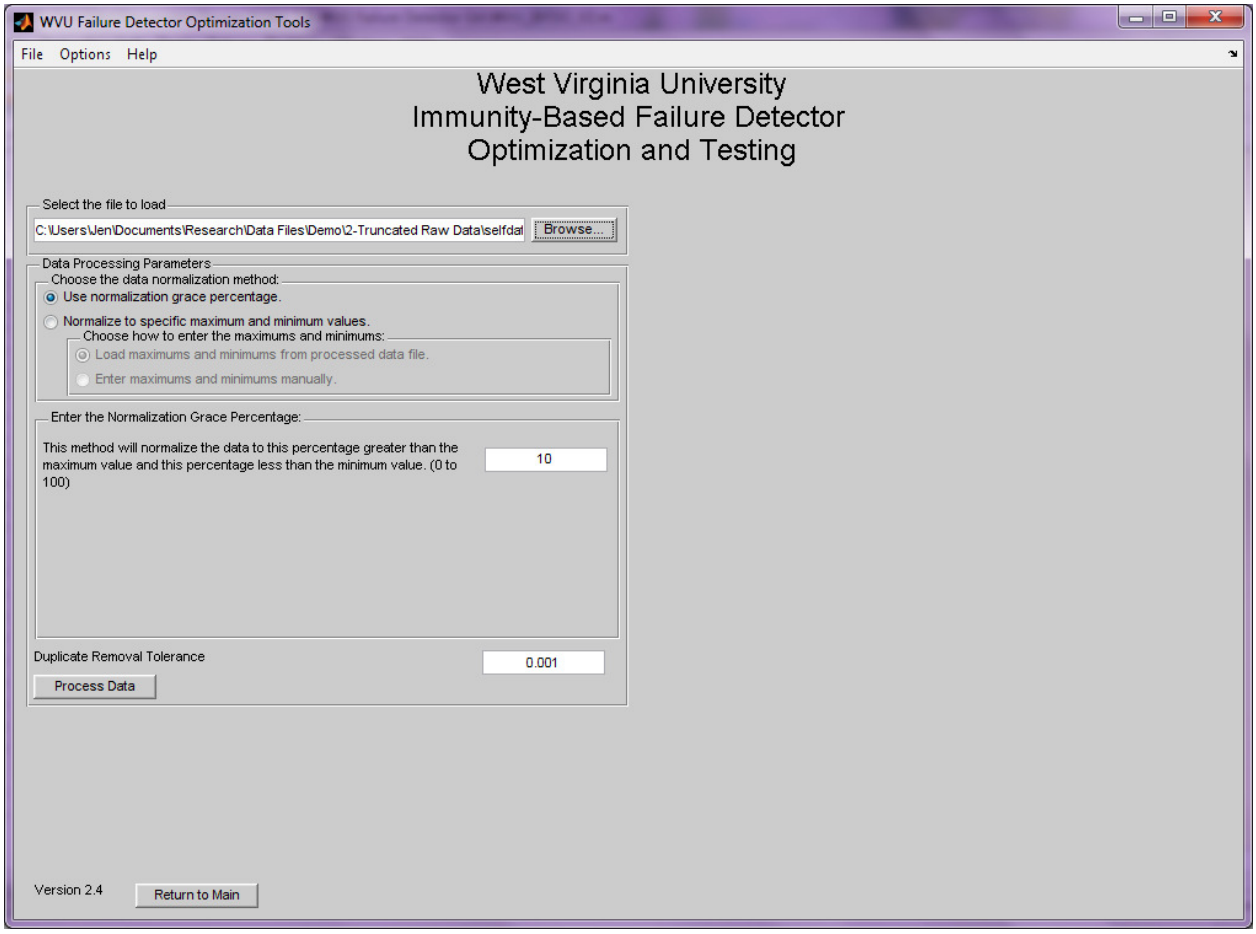

Figure B.10—Data Processing Menu for Normalization with a Grace Percentage

Two parameters are necessary for processing data using a grace percentage. The grace percentage specifies the normalized amount of space to leave around the edges of the solution space. The duplicate removal tolerance is used to specify the radius around a point in which if any other point falls, it is considered to be the same point and is removed from the data set. The default values for these parameters are a grace percentage of 10 and a duplicate removal tolerance of 0.001.

Specify these parameters as desired, then click the 'Process Data' button. This loads a progress bar, which is shown in Figure B.11. Data processing can take a considerable amount of time depending upon the size of the duplicate removal tolerance; smaller tolerance will take longer to process. Be patient. When data processing is complete, a save data dialog box will appear as shown in Figure B.12, beginning in the MATLAB Current Directory. Navigate to the desired save location, specify the desired file name for the processed data file and click 'Save'. The file name chosen for this file is 'procdata1.mat'. The data file is now ready to be clustered, or to create positive-selection detectors.

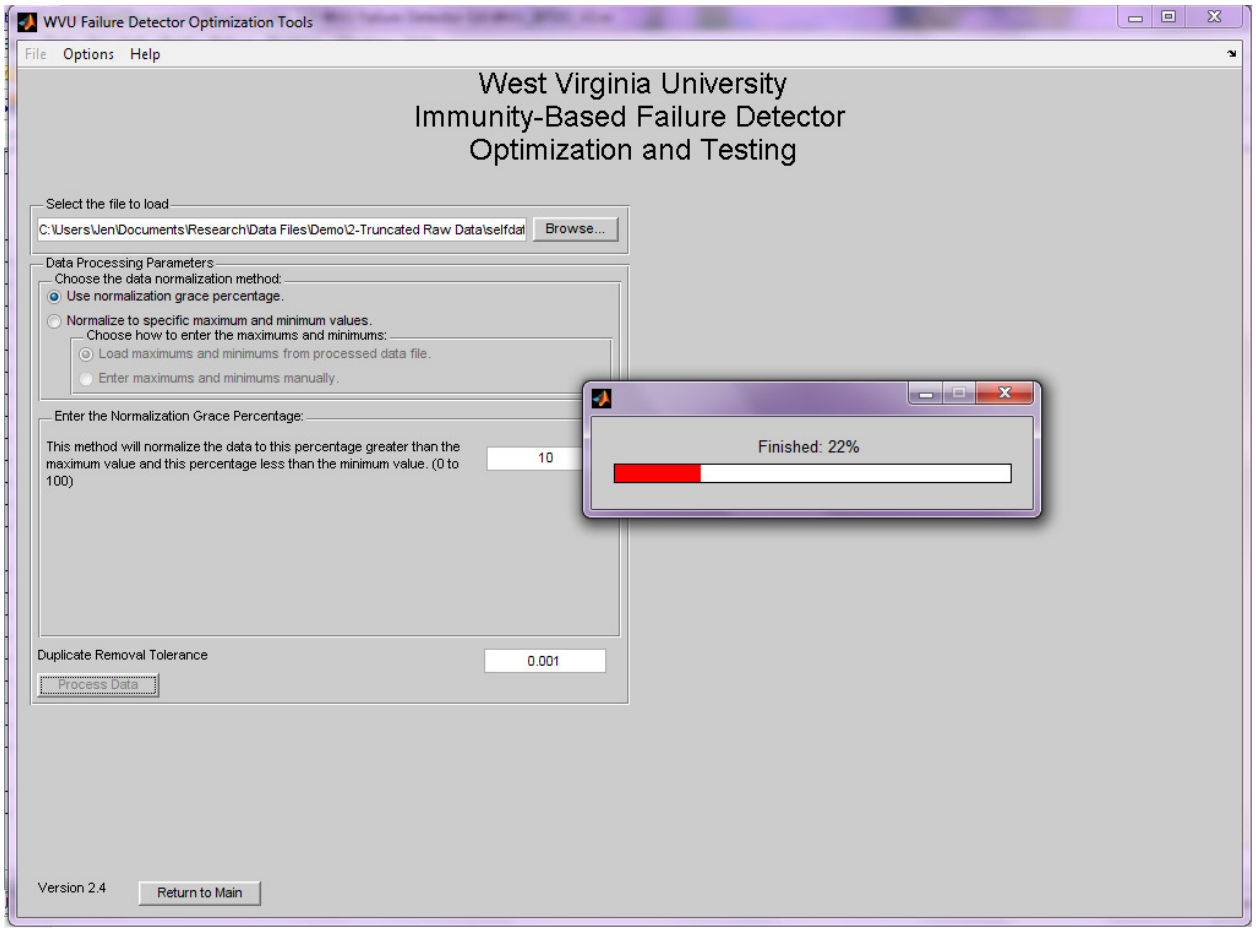

Figure B.11—Data Processing with Normalization Grace Percentage in Progress

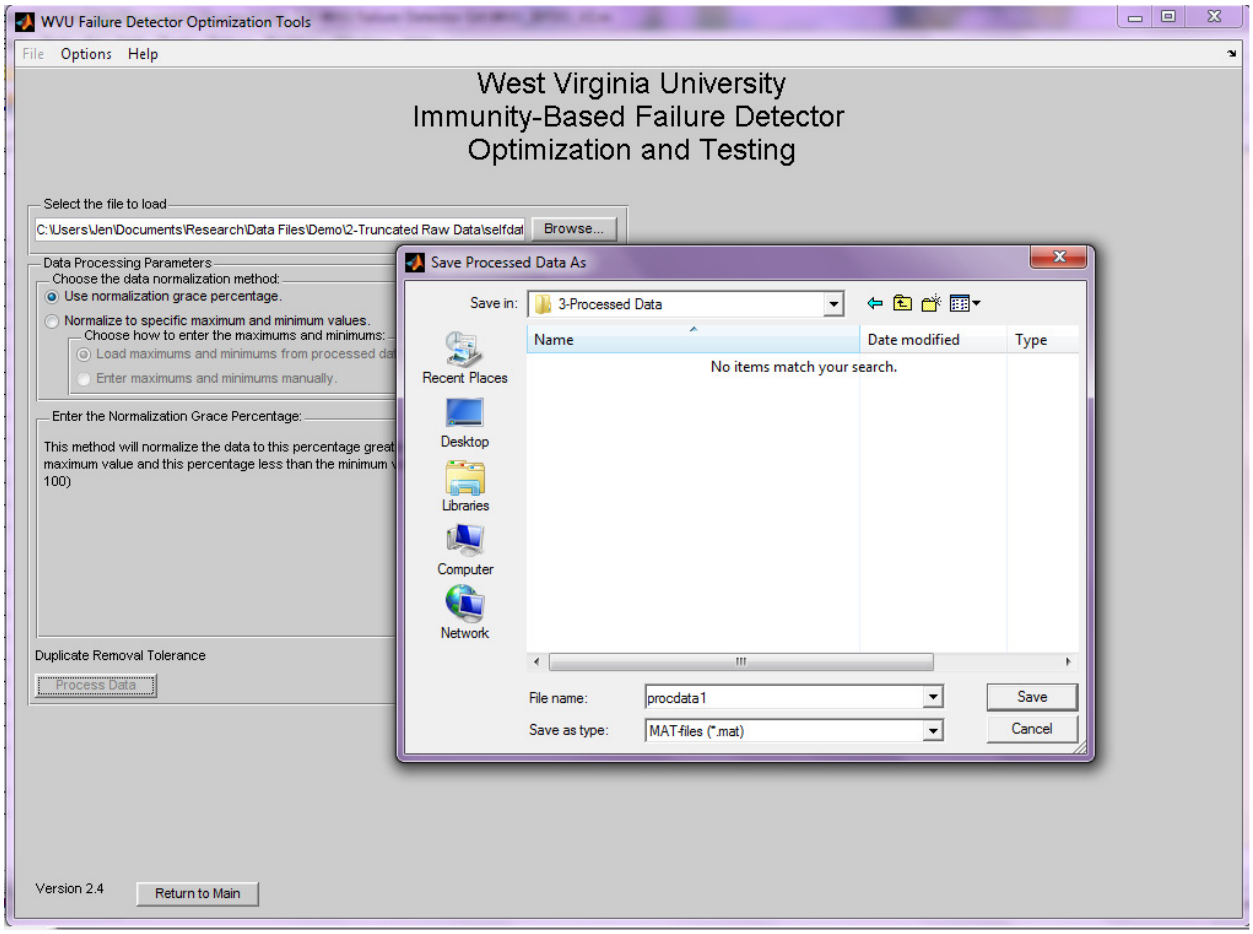

Figure B.12—Saving Data Processed with Grace Percentage Normalization

2.2 Processing with Normalization Limits Specified From a File

Processing raw data requires normalization maximum and minimum normalization limits to be specified. This can be done by loading the normalization limits from a processed data file. It should be noted that the processed data file must contain the same number of dimensions, with the same identifiers as the file currently being processed. This method is particularly useful if two sets of processed data need to be integrated. In addition, abnormal data files must also be normalized to the same limits as those defined for the self in order for the detection results to be valid.

To process data, a raw data file containing only the desired identifiers is loaded into the program. This is done by clicking on the 'File' menu, select 'Data Processing', then select 'Load Raw Data', as shown below in Figure B.13. This loads the menu shown in Figure B.14. Click on the Browse button, select the desired raw data file, and click open. This is shown in Figure B.15. To follow along with this guide, navigate to the 'Demo' directory, open the folder called '2-Truncated Raw Data', and select the file labeled 'selfdata2-2D.mat'.

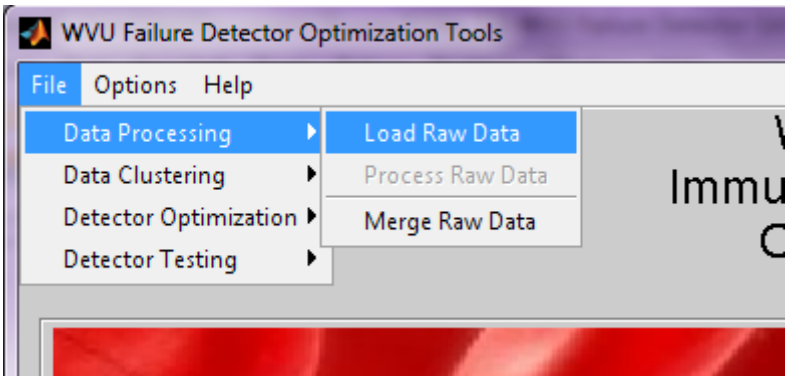

Figure B.13—Opening Load Raw Data Menu

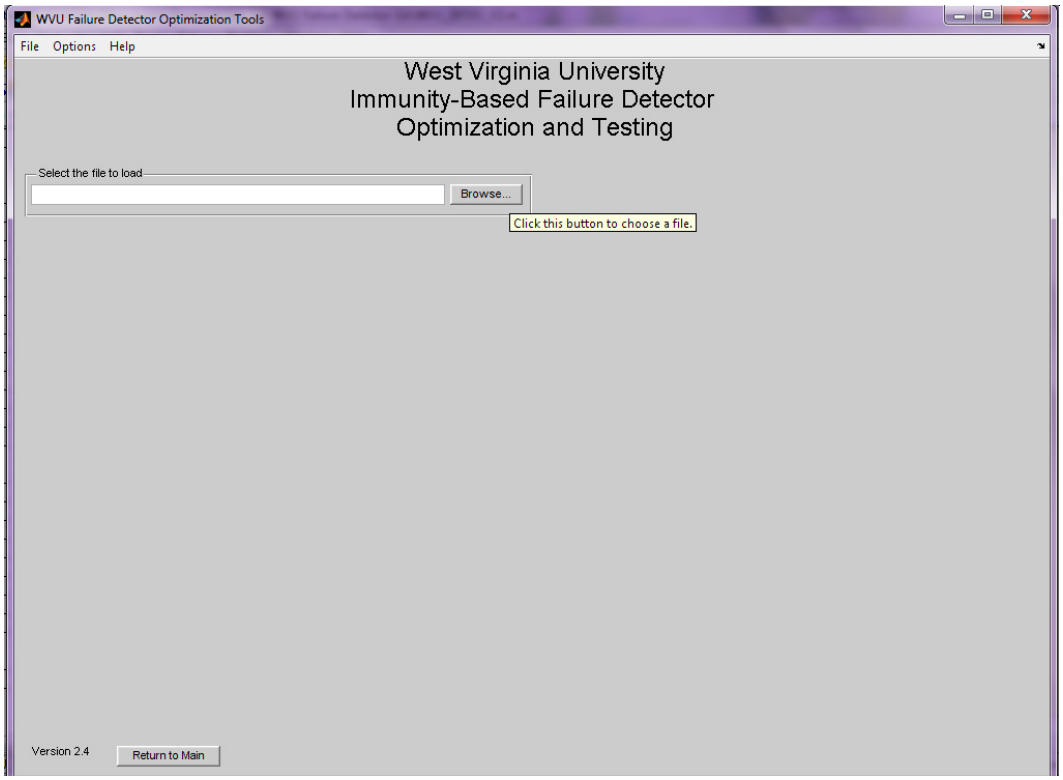

Figure B.14—Load Raw Data Menu

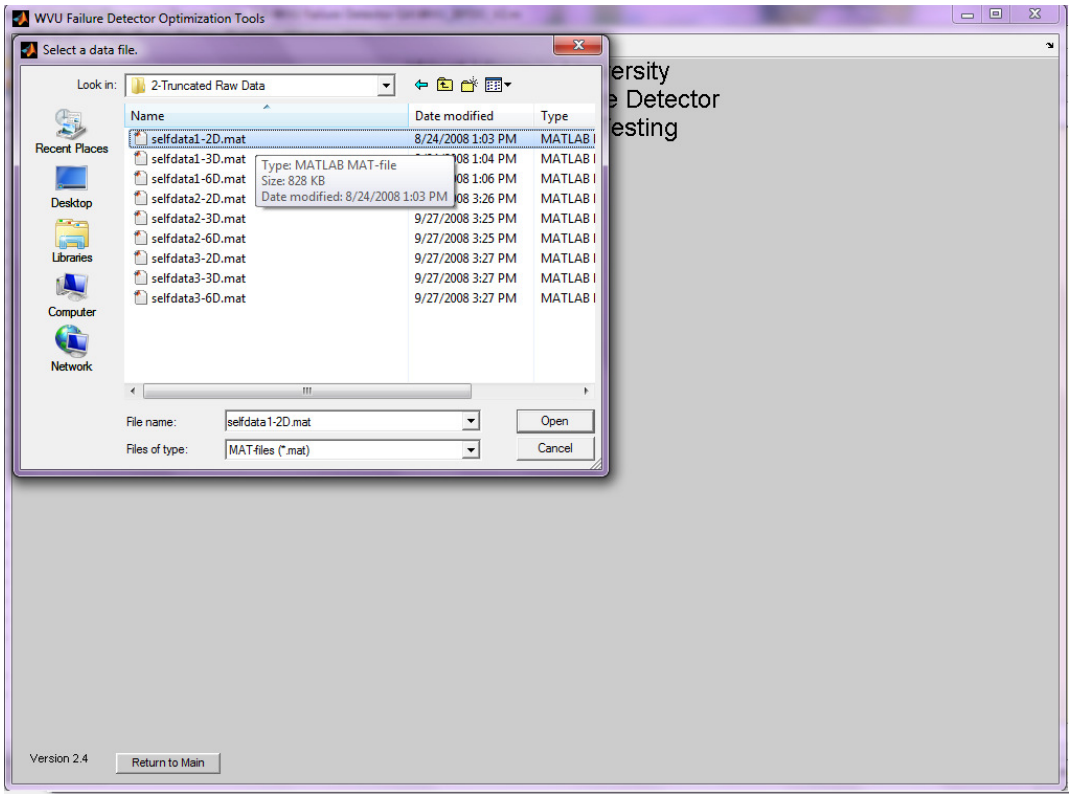

Figure B.15—Load Raw Data Browser

Next, click on the 'File' menu, select 'Data Processing', then select 'Process Raw Data', as shown in Figure B.16. This menu defaults to the normalization method which uses a grace percentage to define the normalization limits of the data. Open the normalization with limits from file menu by clicking on the radio button labeled 'Normalize to specific maximum and minimum values' and selecting the radio button labeled 'Load maximums and minimums from processed data file'. This will cause the menu shown in Figure B.17 to appear.

The normalization method for data processing with limits from file requires two parameters. These are the location of the processed data file from which to draw normalization limits and the duplicate removal tolerance. The duplicate removal tolerance is used to specify the radius around a point in which if any other point falls, it is considered to be the same point and is removed from the data set. The default duplicate removal tolerance is 0.001. Click on the 'Browse' button beneath the radio buttons to select a compatible processed data file from which to specify the normalization maximum and minimum limits. Then specify the duplicate removal tolerance, and click on the 'Process Data' button. This will load a progress bar, as shown in Figure B.18. To follow along with this guide, click on the Browse button beneath the radio buttons and navigate to the 'Demo' directory. Click on the folder labeled '3-Processed Data' and select the file called 'procdata1.mat'. Leave the duplicate removal tolerance as 0.001, the default.

When the data processing has completed a save data dialog box will appear as shown in Figure B.19, beginning in the MATLAB Current Directory. Navigate to the desired save location, specify the desired file name for the processed data file and click 'Save'. The file name chosen for this file is 'procdata2.mat'. The data file is now ready to be clustered, or to create positive-selection detectors.

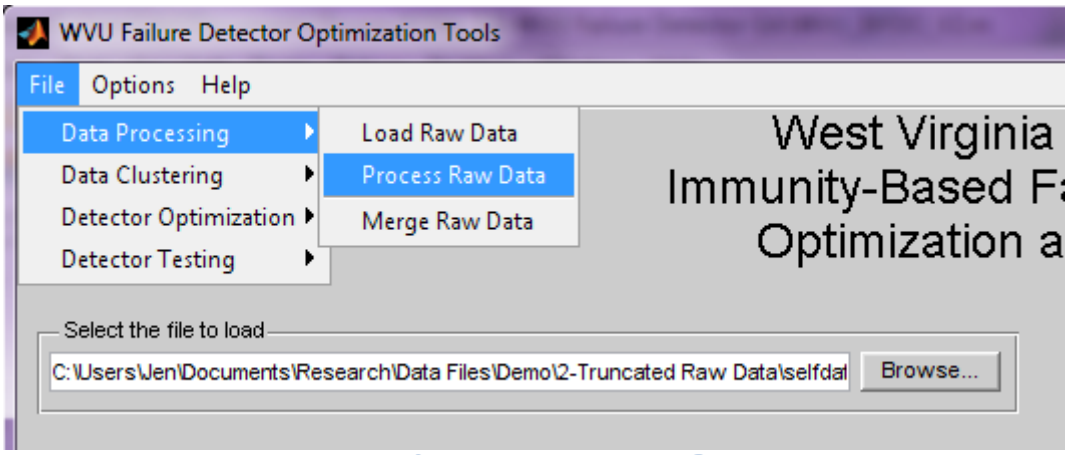

Figure B.16—Opening Process Raw Data Menu

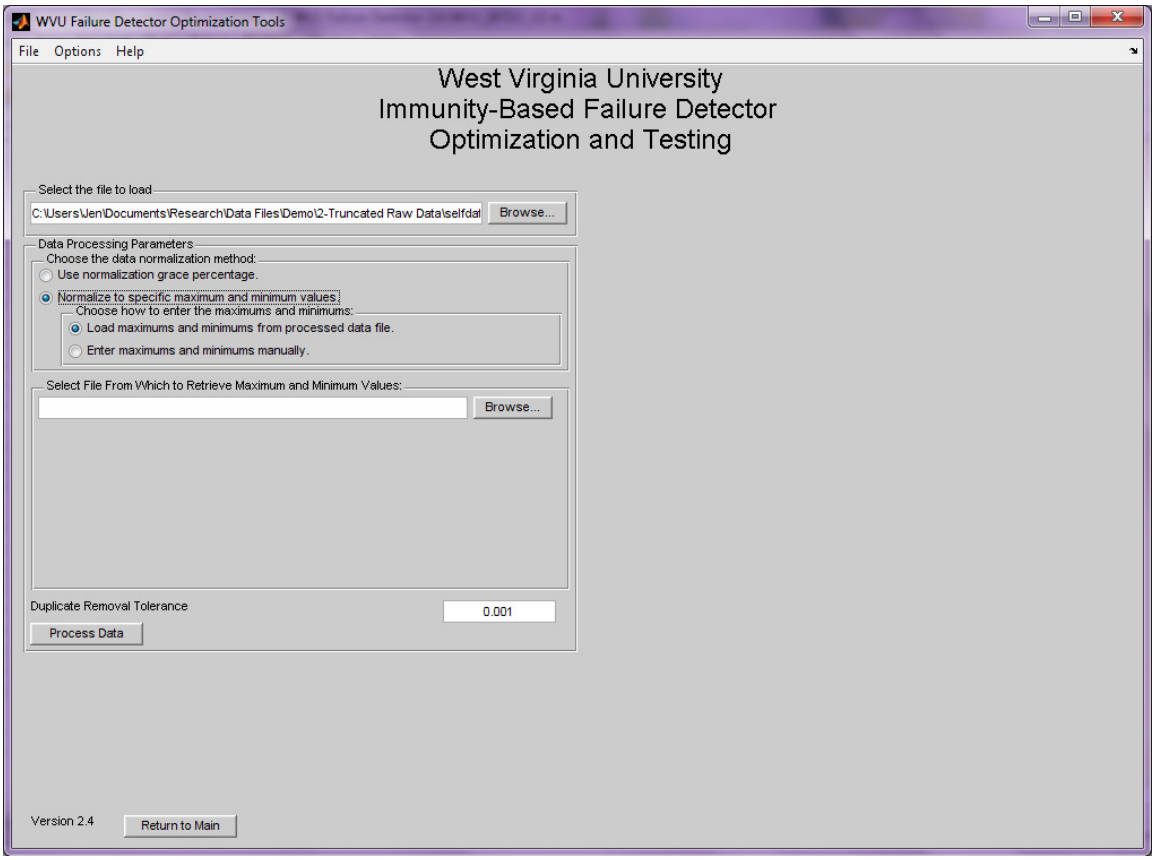

Figure B.17—Menu for Processing Data with Normalization Limits From a File

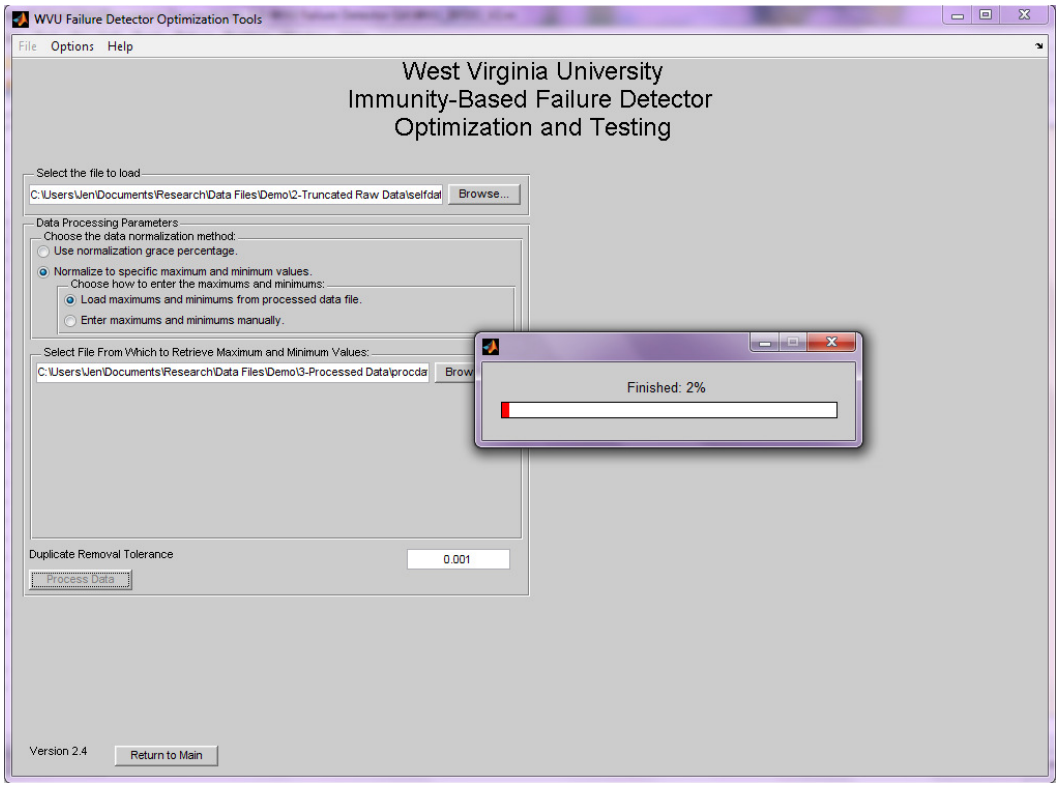

Figure B.18—Data Processing Using Normalization Limits From a File in Progress

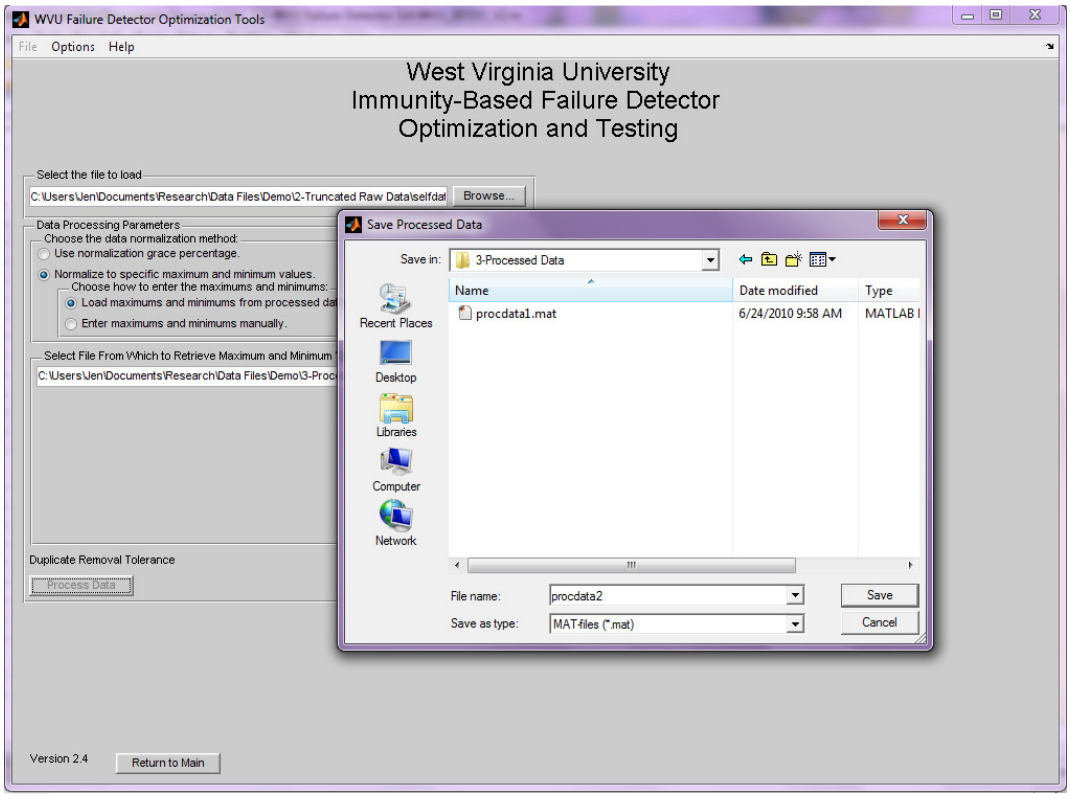

Figure B.19—Save Dialog for Data Processing with Normalization Limits From a File

2.3 Data Processing with Normalization Limits Specified Manually

Processing raw data requires normalization maximum and minimum normalization limits to be specified. This can be done manually entering the normalization maximums and minimums in the edit boxes provided. It should be noted that the number of edit boxes that appear is based on the number of dimensions in the raw data file that is loaded. If this number does not appear to be correct, check the raw data file that was loaded. Due to the need for these edit boxes, however, this processing method only applies to raw data files with 18 or fewer dimensions. If more dimensions than this are required, the second data processing method, in which normalization limits are taken from a file, must be used. This method of normalization is particularly useful if two sets of processed data need to be later integrated. In addition, abnormal data files must also be normalized to the same limits as those defined for the self in order for the detection results to be valid.

To process data, a raw data file containing only the desired identifiers is loaded into the program. This is done by clicking on the 'File' menu, select 'Data Processing', then select 'Load Raw Data', as shown below, in Figure B.20. This loads the menu shown in Figure B.21. Click on the Browse button, select the desired raw data file, and click open. This is shown in Figure B.22. To follow along with this guide, navigate to the 'Demo' directory, open the folder called '2-Truncated Raw Data', and select the file labeled 'selfdata3-2D.mat'.

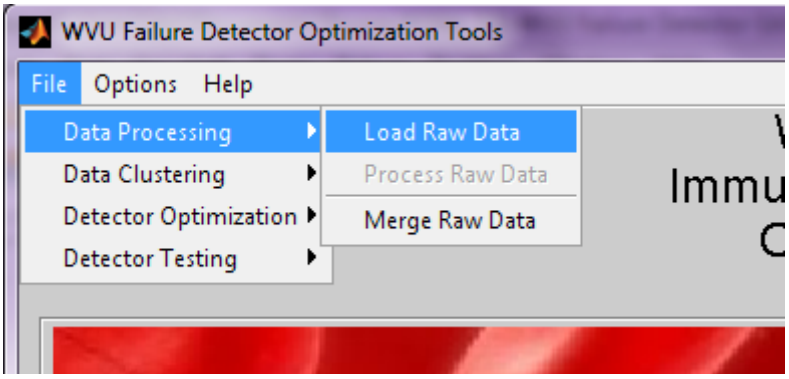

Figure B.20—Opening Load Raw Data Menu

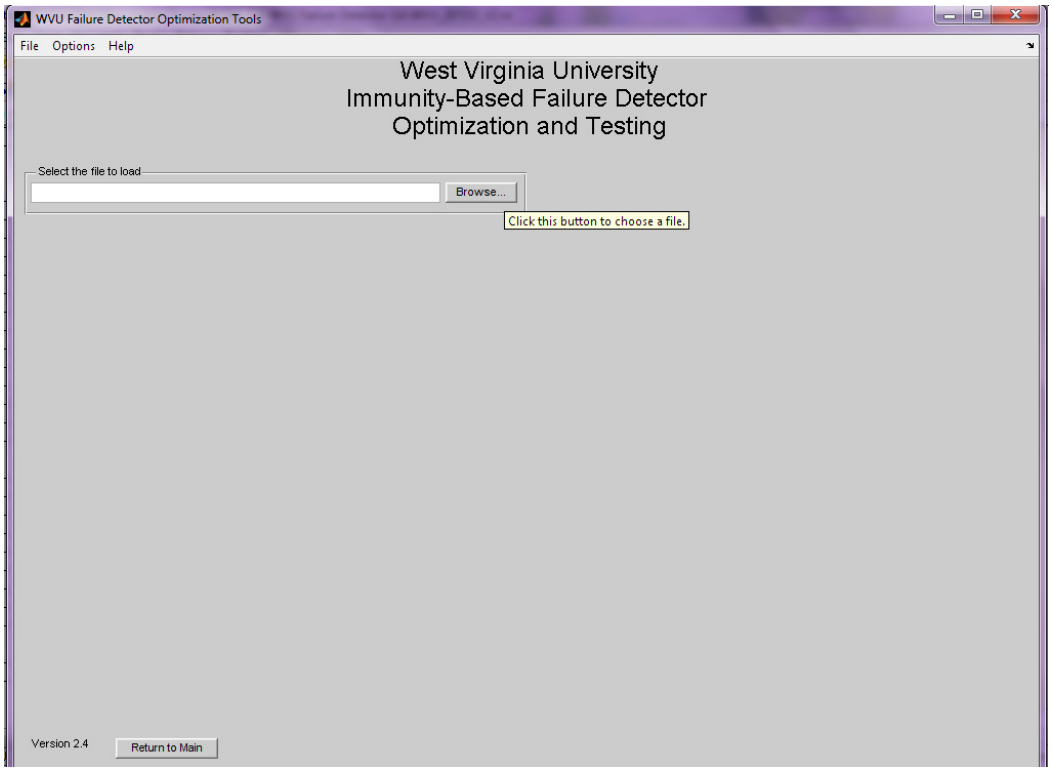

Figure B.21—Load Raw Data Menu

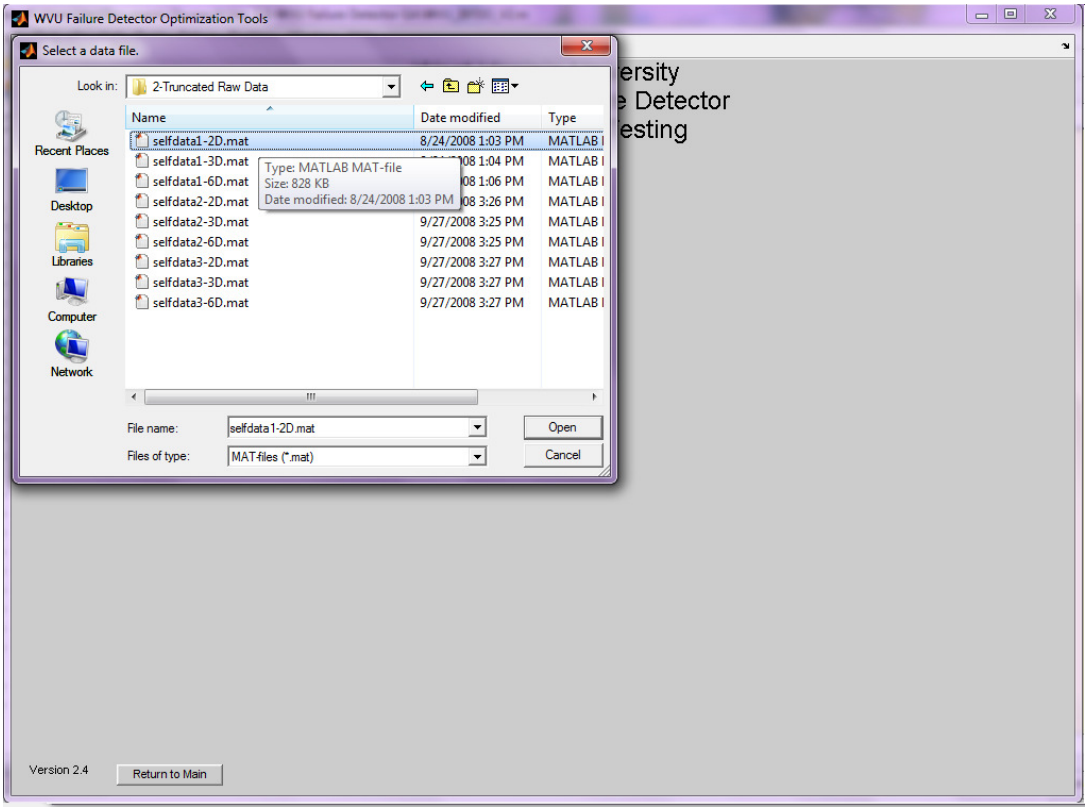

Figure B.22—Load Raw Data Browser

Next, click on the 'File' menu, select 'Data Processing', then select 'Process Raw Data', as shown in Figure B.23. This menu defaults to the normalization method which uses a grace percentage to define the normalization limits of the data. Open the normalization with limits specified manually menu by clicking on the radio button labeled 'Normalize to specific maximum and minimum values' and selecting the radio button labeled 'Enter maximums and minimums manually'. This will cause the menu shown in Figure B.24 to appear.

The normalization method for data processing with limits specified manually requires two parameters for each dimension in the data, in addition to the duplicate removal tolerance. These are the maximum value and minimum value to which each column of data should be normalized. The duplicate removal tolerance is used to specify the radius around a point in which if any other point falls, it is considered to be the same point and is removed from the data set. The default duplicate removal tolerance is 0.001. Enter the desired maximum and minimum values into the edit boxes provided. Then specify the duplicate removal tolerance, and click on the 'Process Data' button. This will load a progress bar, as shown in Figure B.25. To follow along with this guide enter -0.8356 as the minimum for column 1, 0.8603 as the maximum for column 1, -0.2190 as the minimum for column 2, 0.1915 as the maximum for column 2, and 0.001 (the default) as the duplicate removal tolerance.

When the data processing has completed a save data dialog box will appear as shown in Figure B.26, beginning in the MATLAB Current Directory. Navigate to the desired save location, specify the desired file name for the processed data file and click 'Save'. The file name chosen for this file is 'procdata3.mat'. The data file is now ready to be clustered, or to create positive-selection detectors.

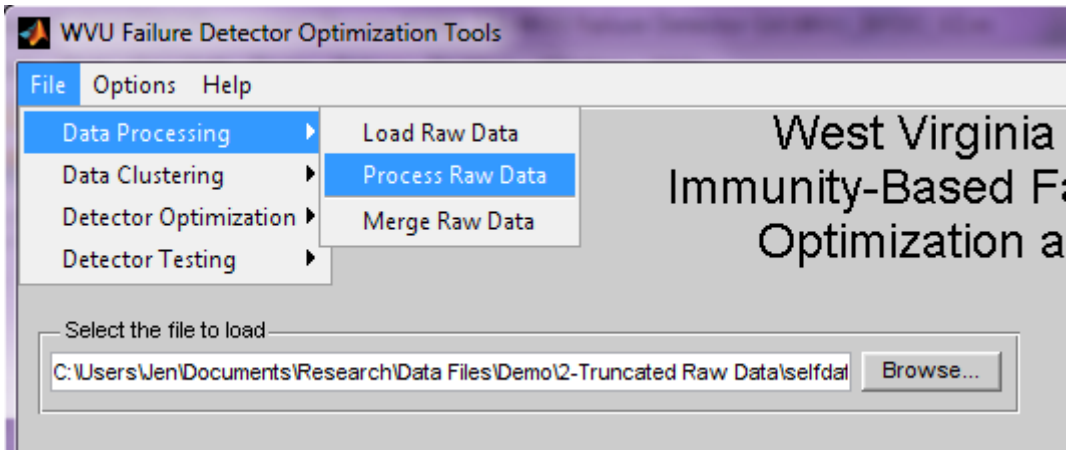

Figure B.23—Opening Process Raw Data Menu

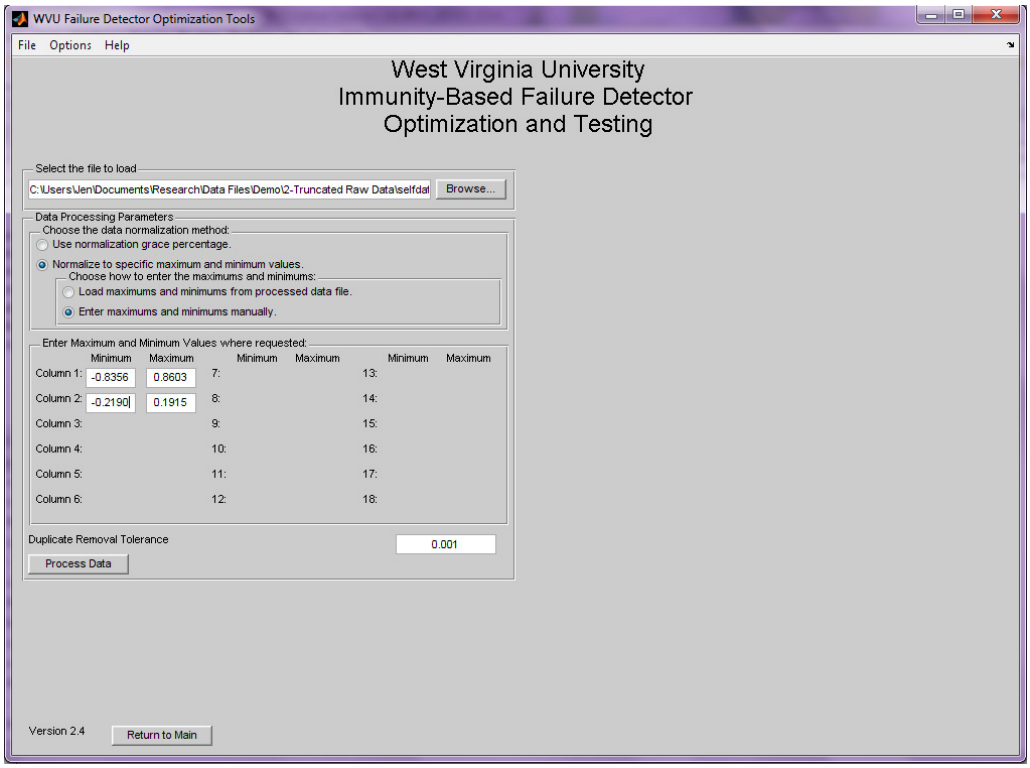

Figure B.24—Menu for Processing Data with Normalization Limits Specified Manually

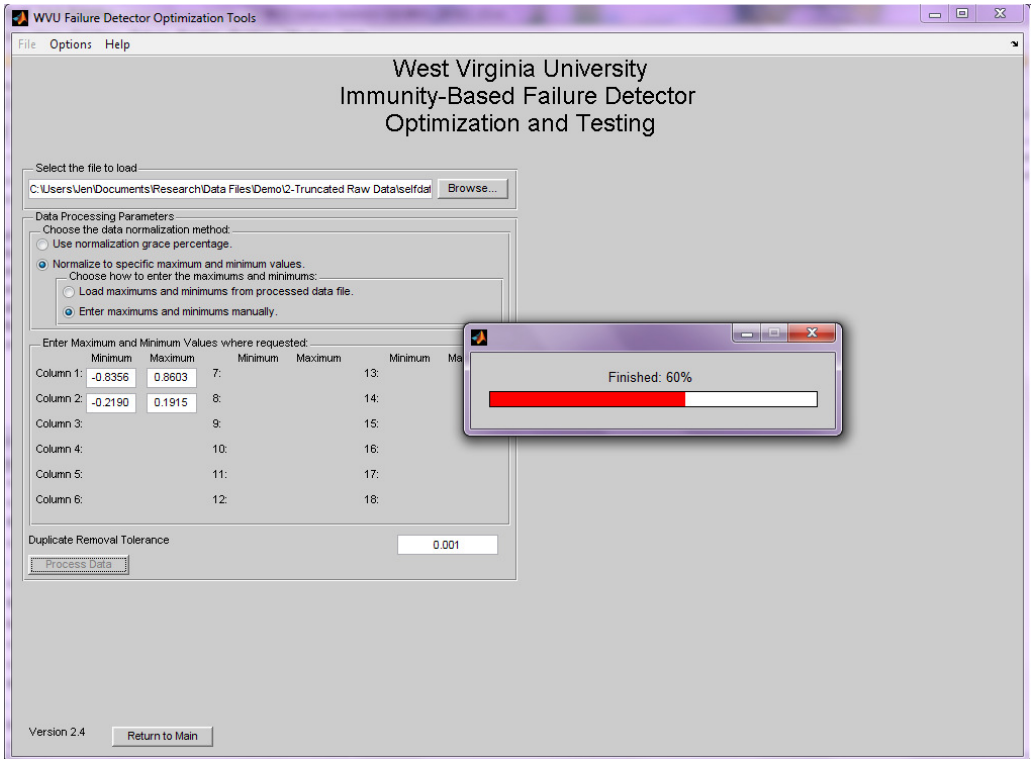

Figure B.25—Data Processing Using Normalization Limits Specified Manually in Progress

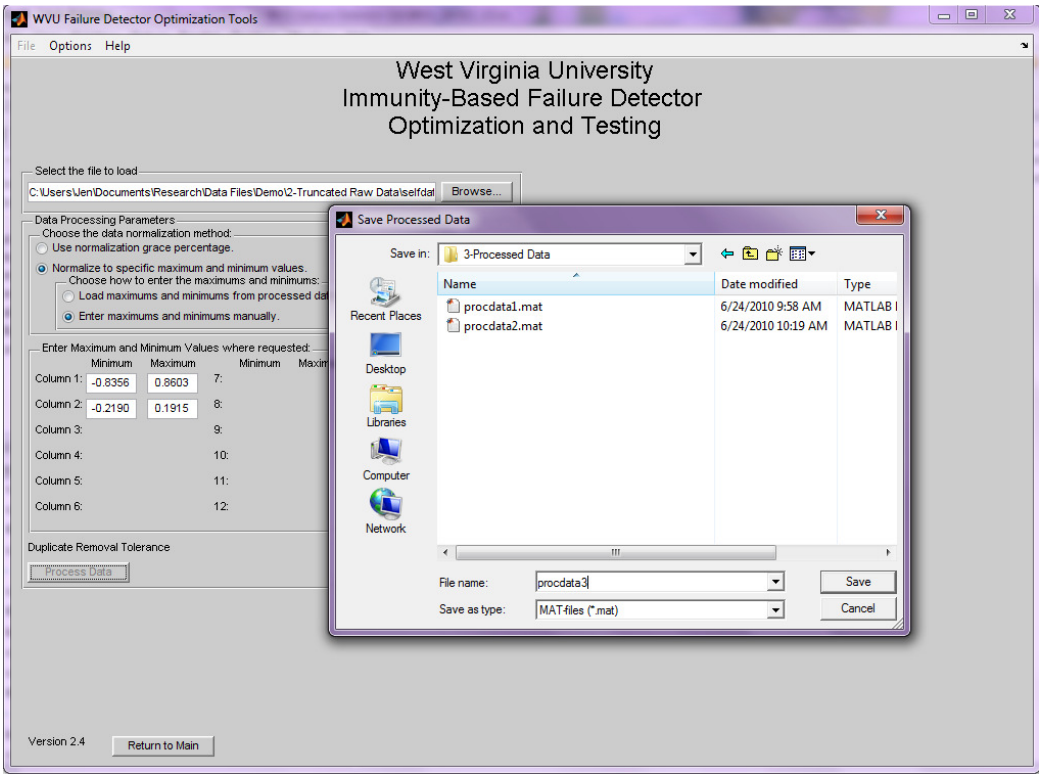

Figure B.26—Save Dialog for Data Processing with Normalization Limits Specified Manually

Chapter 3—Clustering Data

3.1 Cluster for Self Definition

Data clustering is performed on processed data to reduce the computational load imparted by the self, and to reduce the number of points ultimately needed to fully define the self. Processed data is composed of a large number of normalized time-history data points. However, this data is inherently discrete. Thus the points contained within the normal processed data do not completely define the self. In other words, there could still be points that belong to the self that are not included in the processed data set. Clustering is used as an approximation, as a way to include all points in the self that could possibly belong to the self. This is considered an approximation because the self is being defined from an incomplete set. Therefore, depending on the parameters used to generate the clusters, areas of the flight envelope that should belong to the self may be included in the non-self, or areas of the non-self could be included in the self. For this reason, clustering is a very important process in the production of effective failure detectors. Even with perfect coverage of the areas not covered by the clusters, if the clusters themselves are poorly constructed, (i.e. contain too much empty space or not enough empty space) the detectors will not give good fault detection results.

Clusters are defined as either hyper-spheres, containing a center and radius, or hyper-rectangles, containing a center and the distances from the center to the edge in each dimension. Hyper-sphere clusters are used in the generation of hyper-sphere detectors, and in the genetic algorithm for hyper-sphere, hyper-ellipsoid, and hyper-rotational ellipsoid detectors. Hyper-rectangles are used only in the generation and optimization of hyper-rectangle detectors.

3.1.1 Clustering with Hyper-Spheres Using Number-Imposed Clustering Method (M1)

Clustering method M1, which uses variable-sized clusters, uses an enhanced k-means algorithm to determine the location of cluster centers within the solution space, based on the locations of the processed data points. Each point in the processed data set is then assigned to the nearest cluster center. The radius of the cluster is then determined as the distance to the furthest point assigned to the center. This results in a cluster that contains empty space, or area that is not covered by processed data points. This algorithm is not concerned with reducing the amount of empty space in clusters, so the number of clusters in the set must be chosen carefully. Lower number of clusters results in more empty space, increasing the chances that points belonging to abnormal conditions are included in the self and potentially decreasing detection rate. Higher number of clusters reduces the empty space within the clusters, increasing the chances that normal points are excluded from the definition of the self and potentially producing a high number of false alarms.

In order to begin clustering, click on the 'File' menu, select 'Data Clustering', then 'Load Processed Data', as shown in Figure B.27. This will load the menu seen in Figure B.28. Click on the browse button to load a processed data file into the program for clustering, as in Figure B.29. To follow along with this guide, navigate to the 'Demo' directory, click on the folder labeled '3-Processed Data', and select the file labeled '2Dprocdata1.mat'.

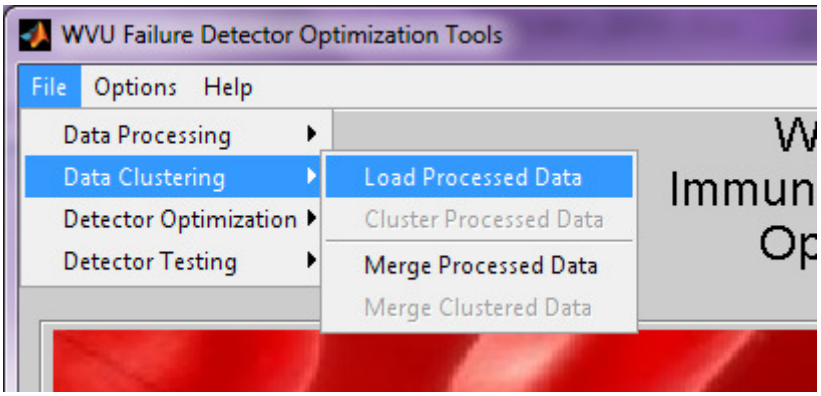

Figure B.27—Opening Load Processed Data Menu

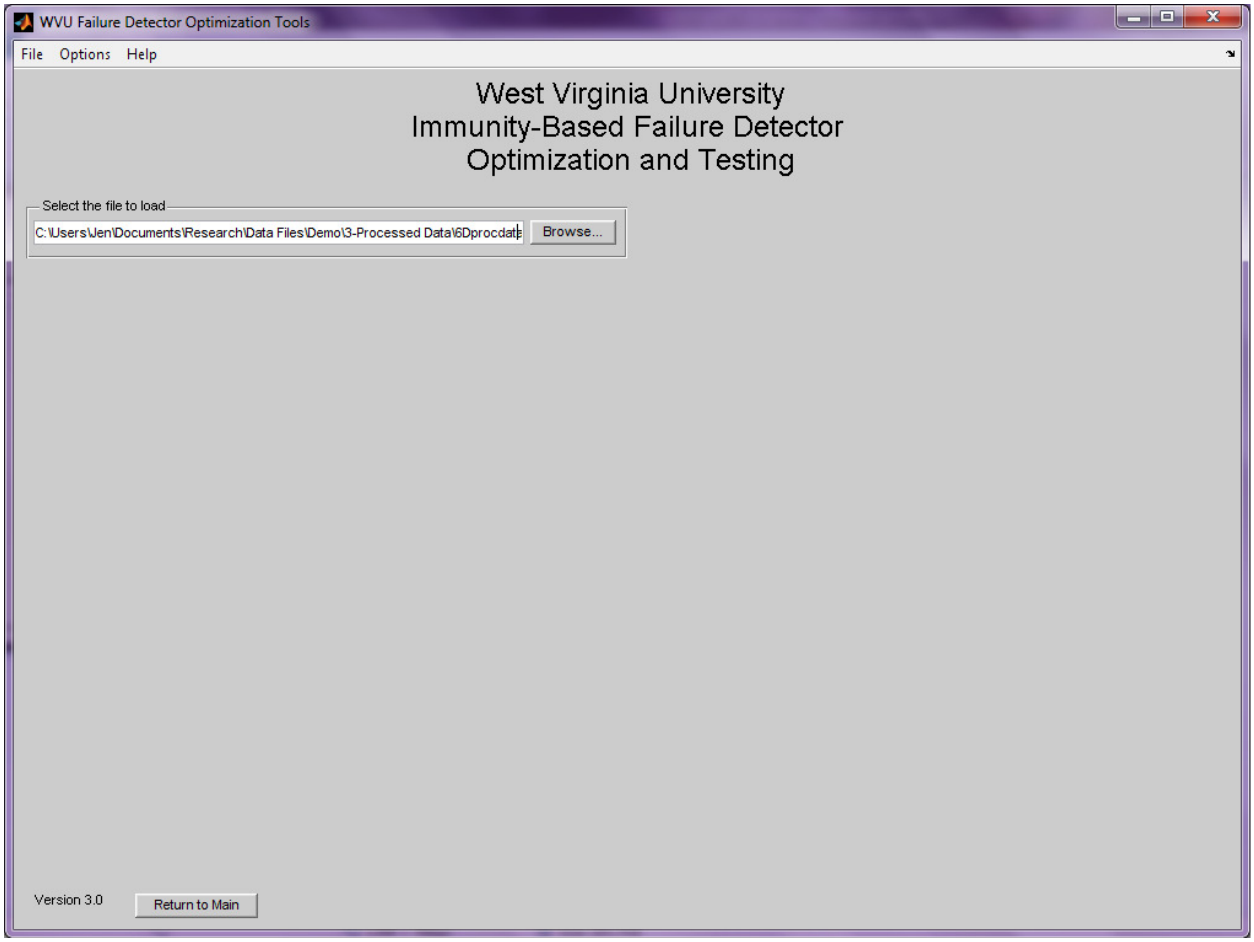

Figure B.28—Load Processed Data Menu

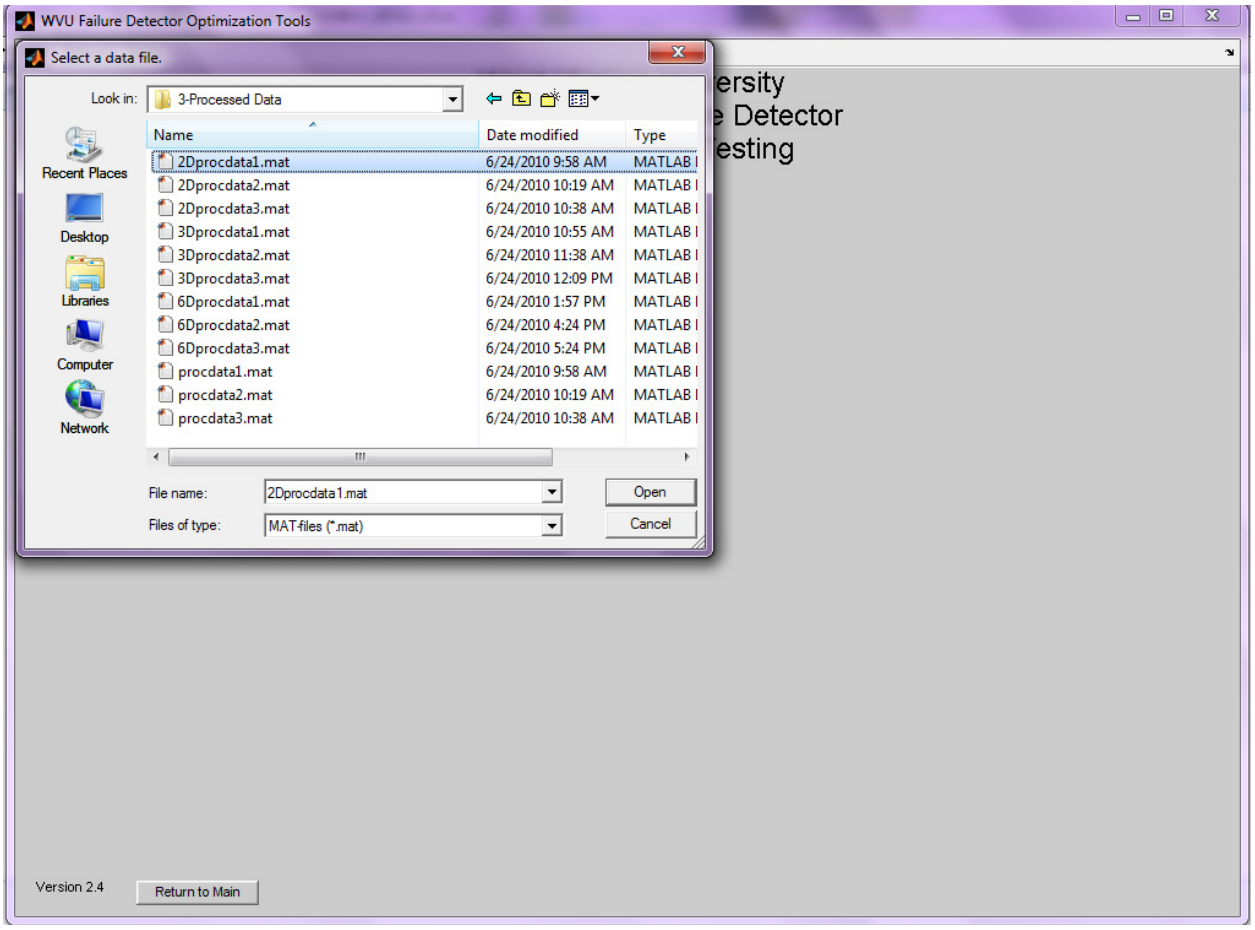

Figure B.29—Processed Data Browser Dialog

Once the processed data file has been selected, click on the 'File' menu, select 'Data Clustering' and then 'Cluster Processed Data'. This will load the menu seen in Figure B.30. This menu defaults to the M1 clustering method. If the correct menu is not shown, select the 'Hyper-spheres' radio button, then the 'Number-Imposed Clustering Method (M1)' radio button. This menu includes four parameters. These are the minimum cluster radius, desired number of clusters, confidence percentage, and permitted error. The minimum cluster radius is the smallest acceptable radius which may be assigned to a cluster. The desired number of clusters is the number of centers that will be generated by the k-means algorithm. The confidence percentage and permitted error are the Monte Carlo volume estimation parameters used to determine the accuracy desired when calculating the cover of the solution space and amount of overlapping present in the clustered set. Select these parameters and click the 'Cluster Data' button. To follow along with this guide, enter 0.002 as the minimum cluster radius, 100 as the desired number of clusters, 98 as the confidence percentage, and 0.01 as the permitted error and click the 'Cluster Data' button. This will load the menu seen in Figure B.31.

Once the clustering has completed, a save dialog will open, as shown in Figure B.32. Navigate to the desired save location, enter the desired name for the clustered data file and click 'Save'. The file name chosen for this clustered data file is '2Dclust1_M1.mat'. Then the clustering results will be displayed. If the data being clustered is 2-dimensional, the clusters will be plotted along with the self parameters, as in Figure B.33. Otherwise only the self parameters will appear, as shown in Figure B.34.

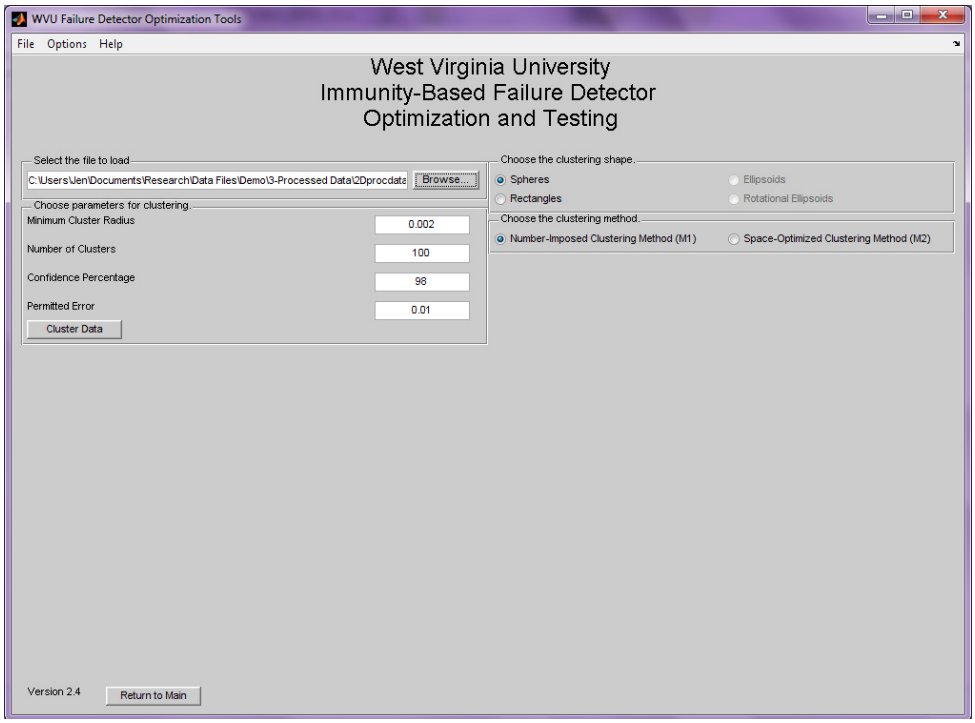

Figure B.30—Clustering M1 Menu

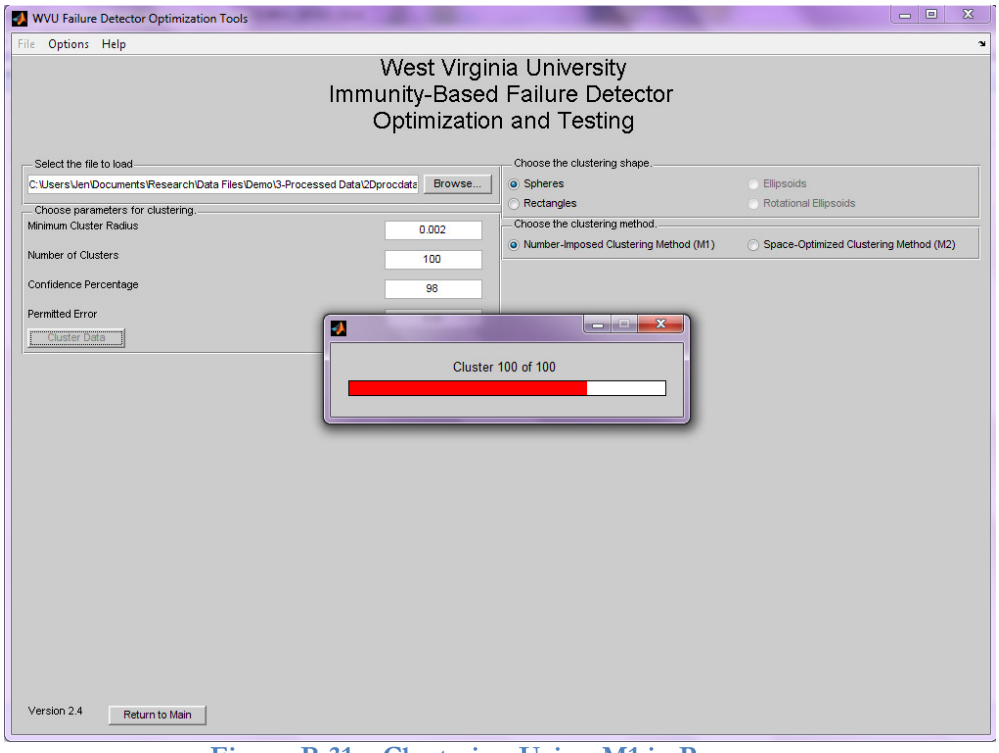

Figure B.31—Clustering Using M1 in Progress

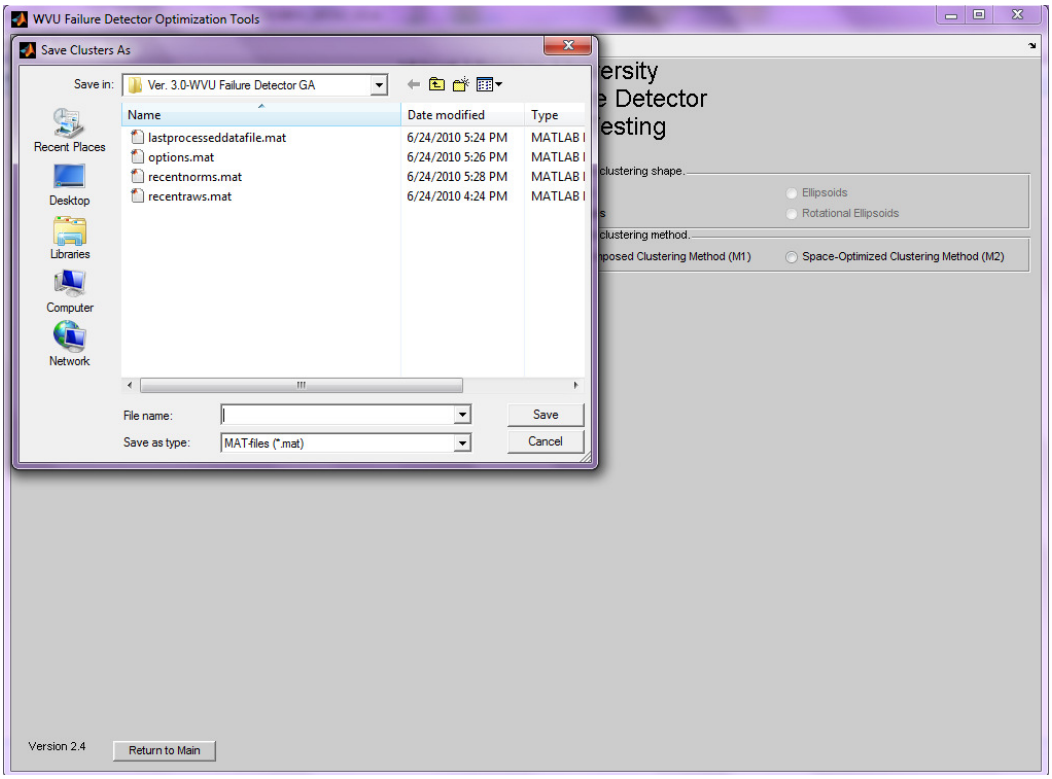

Figure B.32—Clustering M1 Save Dialog

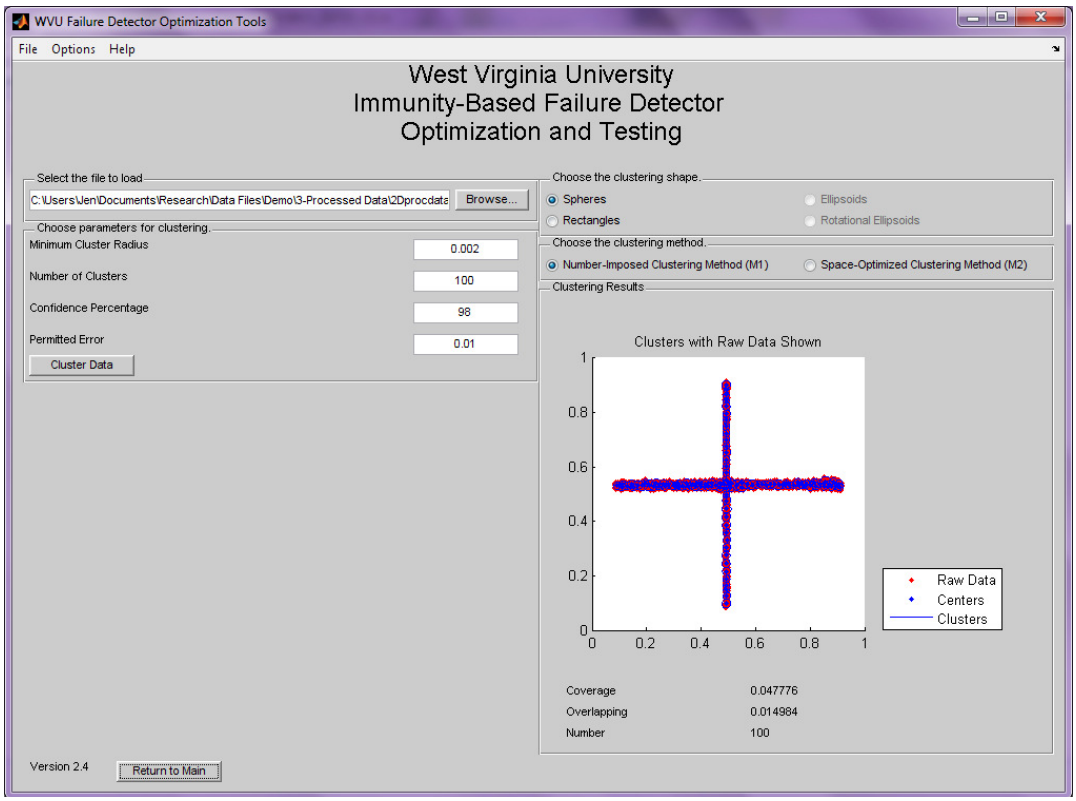

Figure B.33—Clustering M1 2-Dimensional Results

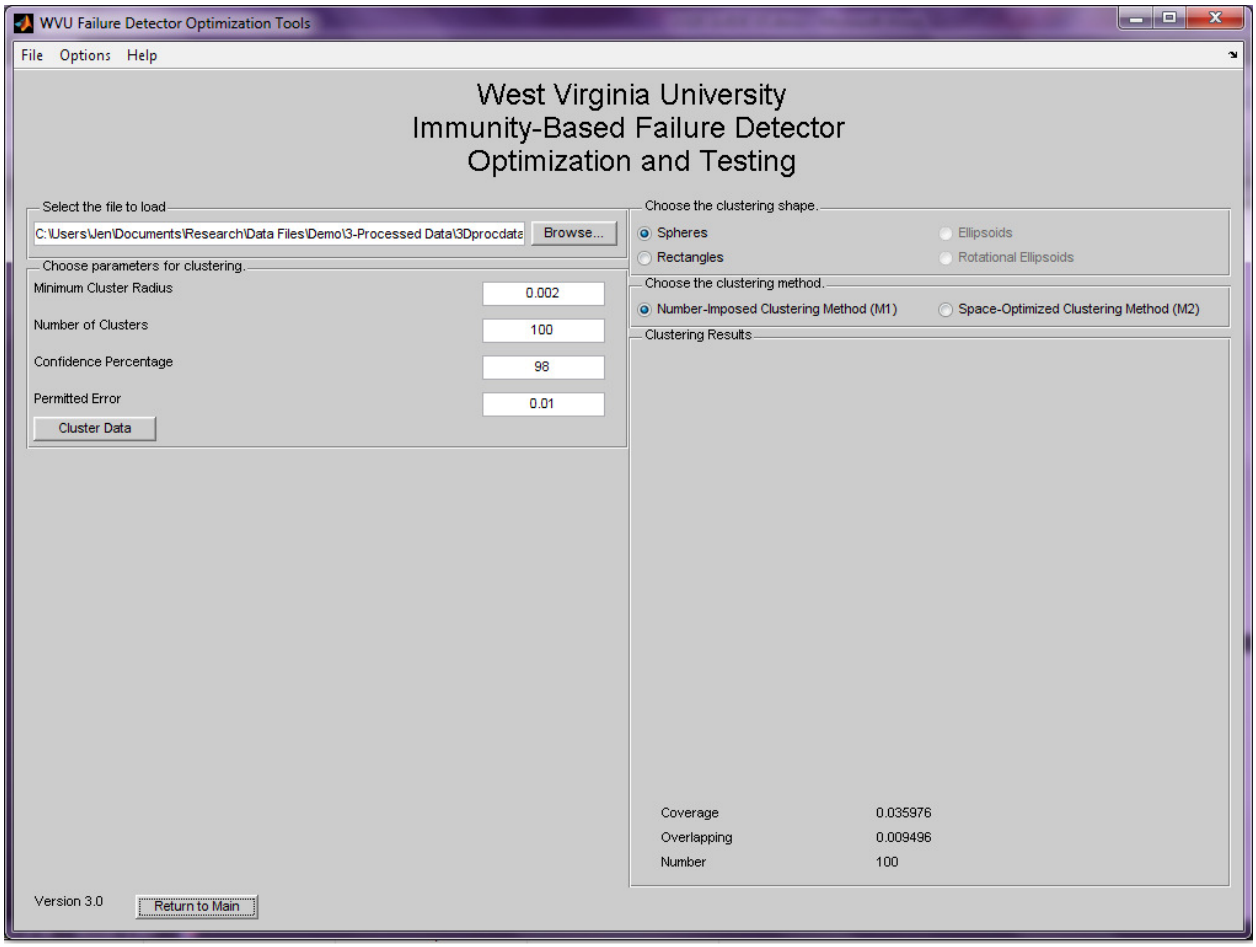

Figure B.34—Clustering M1 Higher-Dimensional Results

3.1.2 Clustering with Hyper-Spheres Using Space-Optimized Clustering Method (M2)

Clustering method M2, which uses variable-sized clusters, uses an enhanced k-means algorithm to determine the location of cluster centers within the solution space, based on the locations of the processed data points. Each point in the processed data set is then assigned to the nearest cluster center. The radius of the cluster is then determined as the distance to the furthest point assigned to the center. This results in a cluster that contains empty space, or area that is not covered by processed data points. Lower number of clusters results in more empty space, increasing the chances that points belonging to abnormal conditions are included in the self and potentially decreasing detection rate. Higher number of clusters reduces the empty space within the clusters, increasing the chances that normal points are excluded from the definition of the self and potentially producing a high number of false alarms. This algorithm attempts to reduce the empty space within the clusters to a specified threshold by iteratively increasing the desired number of clusters and then recalculating the set.

In order to begin clustering, click on the 'File' menu, select 'Data Clustering', then 'Load Processed Data', as shown in Figure B.35. This will load the menu seen in Figure B.36. Click on the browse button to load a processed data file into the program for clustering, as in Figure B.37. To

follow along with this guide, navigate to the 'Demo' directory, click on the folder labeled '3-Processed Data', and select the file labeled '2Dprocdata1.mat'.

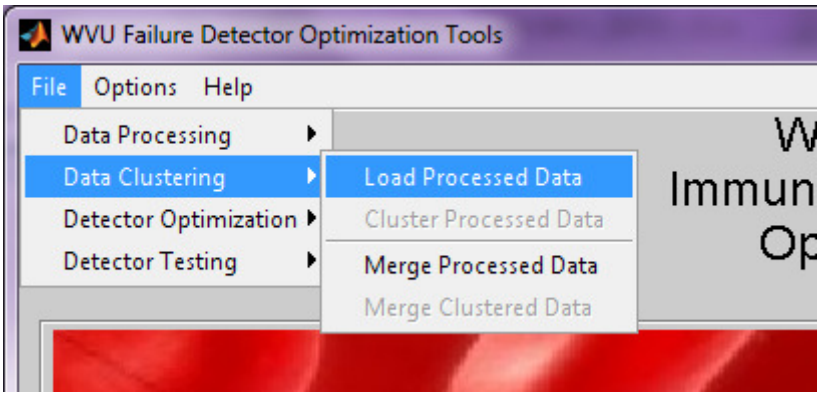

Figure B.35—Opening Load Processed Data Menu

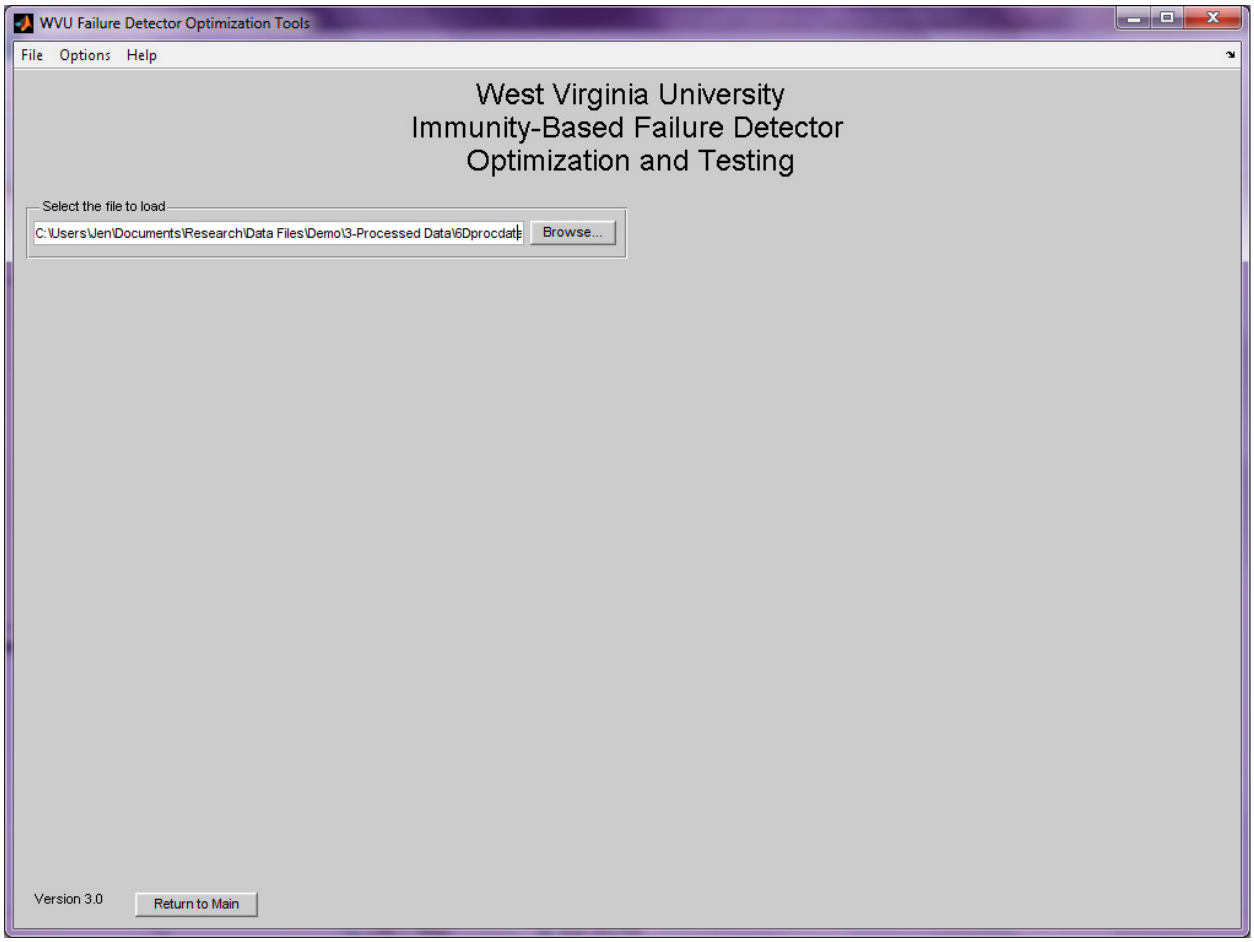

Figure B.36—Load Processed Data Menu

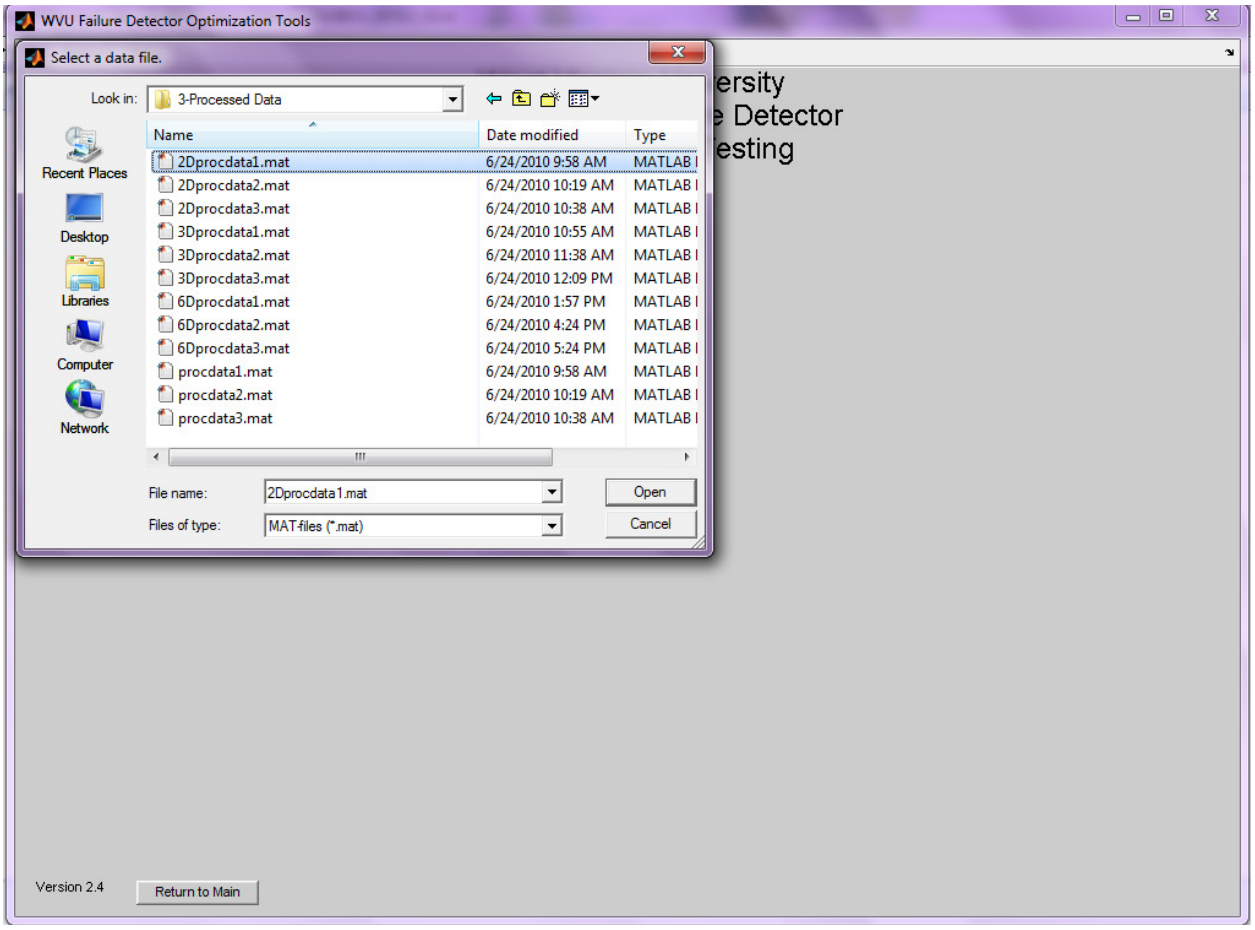

Figure B.37—Processed Data Browser Dialog

Once the processed data file has been selected, click on the 'File' menu, select 'Data Clustering' and then 'Cluster Processed Data'. This will load the menu seen in Figure B.38. This menu defaults to the M1 clustering method. To load the clustering method M2 menu, select the 'Hyper-spheres' radio button, then the 'Space-Optimized Clustering Method (M2)' radio button. This menu includes six parameters. These are the initial number of clusters, additional clusters per iteration, point radius, acceptable empty percentage, confidence percentage, and permitted error. The initial number of clusters is the minimum number of desired clusters in the self. The additional number of clusters per iteration is how many centers more centers to generate using the k-means algorithm with each iteration. The point radius is the radius around each of the processed data points that can be confidently considered part of the normal data set. The acceptable empty percentage is the amount of empty space that may exist within the clusters for the cluster set to be finalized. The confidence percentage and permitted error are the Monte Carlo volume estimation parameters used to determine the accuracy desired when calculating the cover of the solution space and amount of overlapping present in the clustered set. Select these parameters and click the 'Cluster Data' button. To follow along with this guide, enter 100 as the initial number of clusters, 50 as the additional clusters per iteration, 0.002 as the point radius, 100 as the acceptable empty percentage, 98 as the confidence percentage, and 0.01 as the permitted error and click the 'Cluster Data' button. This will load the menu seen in Figure B.39.

Once the clustering has completed, a save dialog will open, as shown in Figure B.40. Navigate to the desired save location, enter the desired name for the clustered data file and click

'Save'. The file name chosen for this clustered data file is '2Dclust1_M2.mat'. Then the clustering results will be displayed. If the data being clustered is 2-dimensional, the clusters will be plotted along with the self parameters, as in Figure B.41. Otherwise only the self parameters will appear, as shown in Figure B.42.

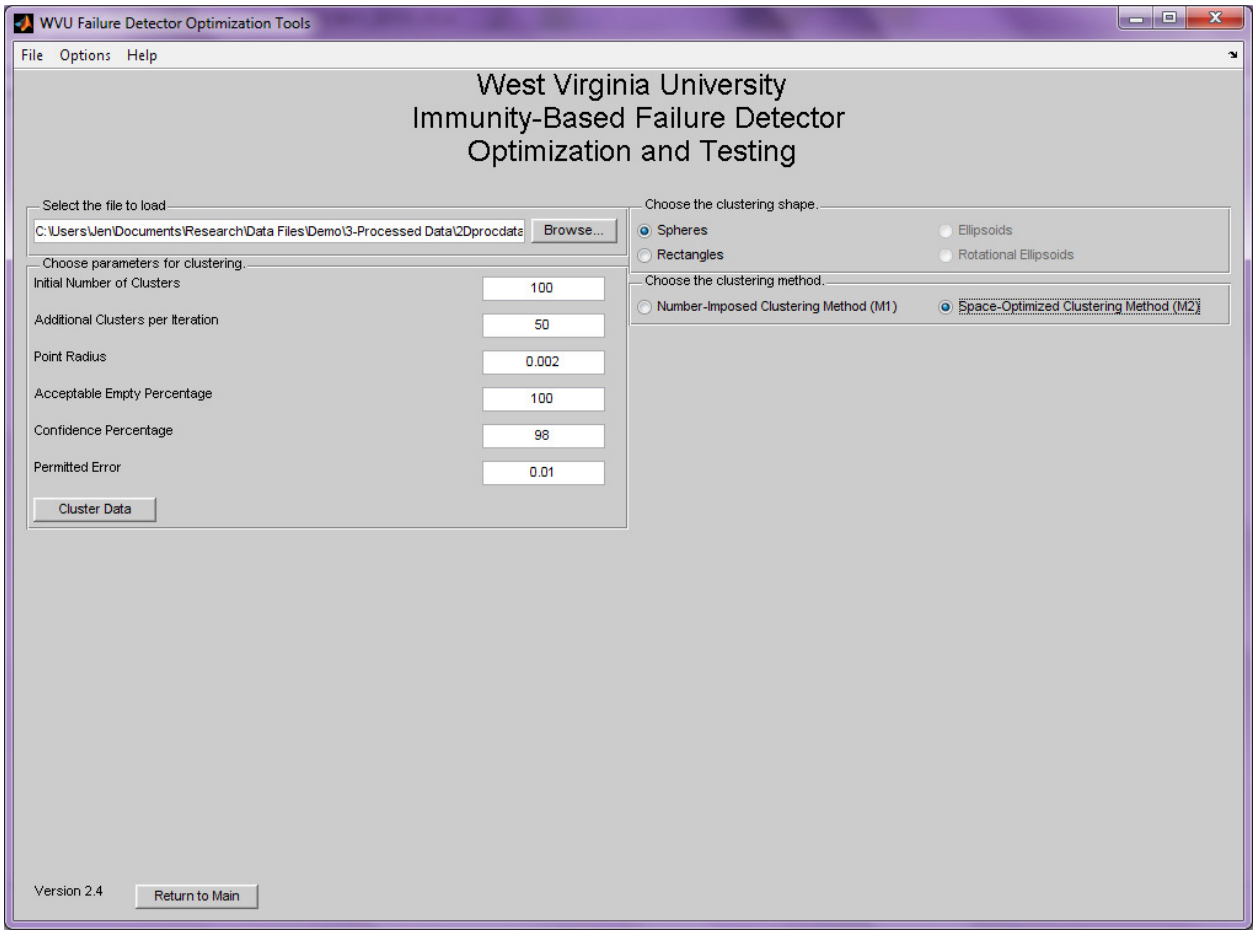

Figure B.38—Clustering M2 Menu

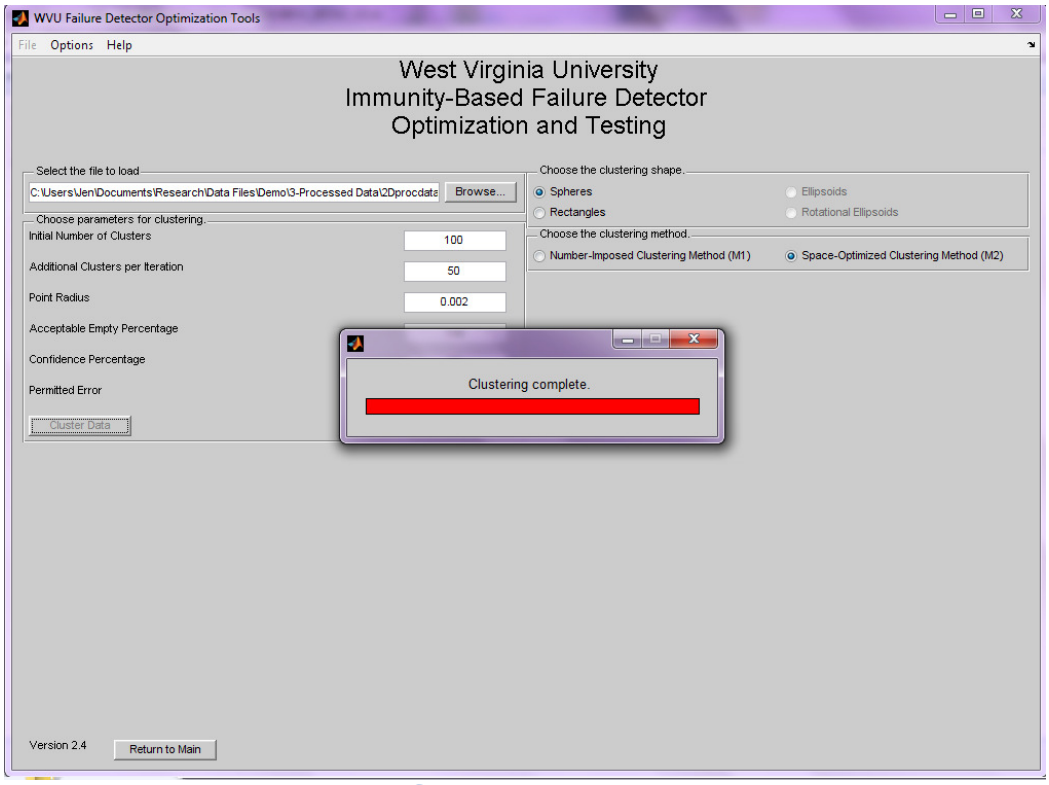

Figure B.39—Clustering Using M2 in Progress

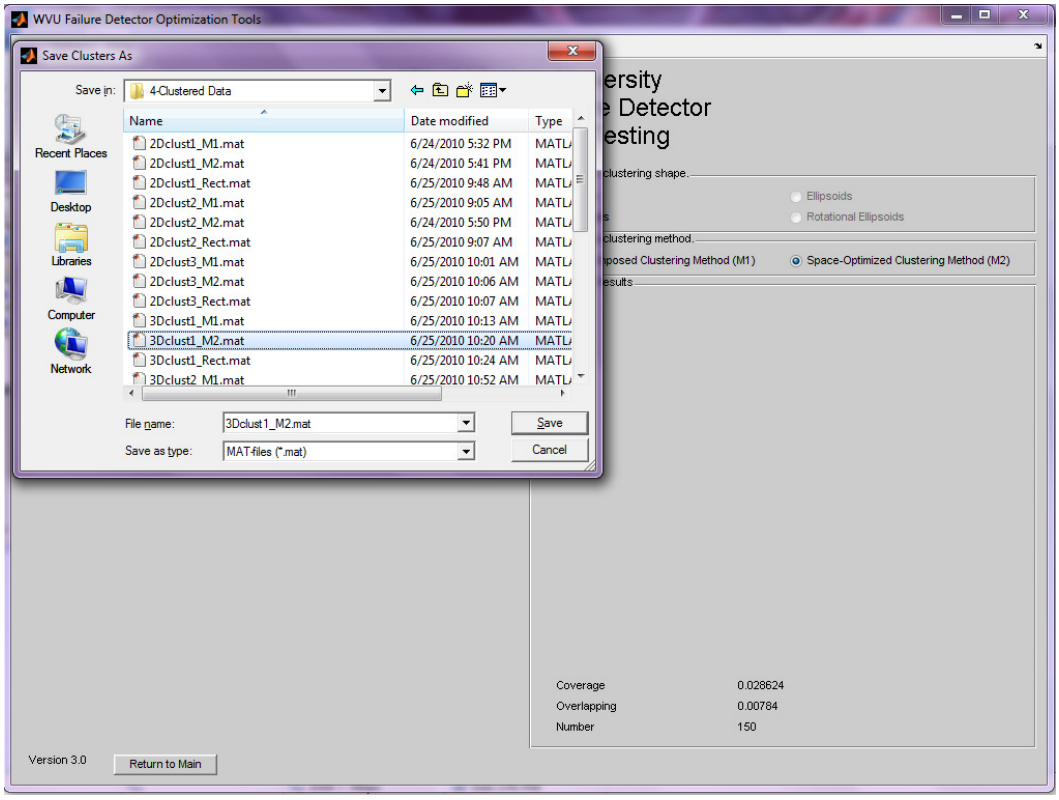

Figure B.40—Clustering M2 Save Dialog

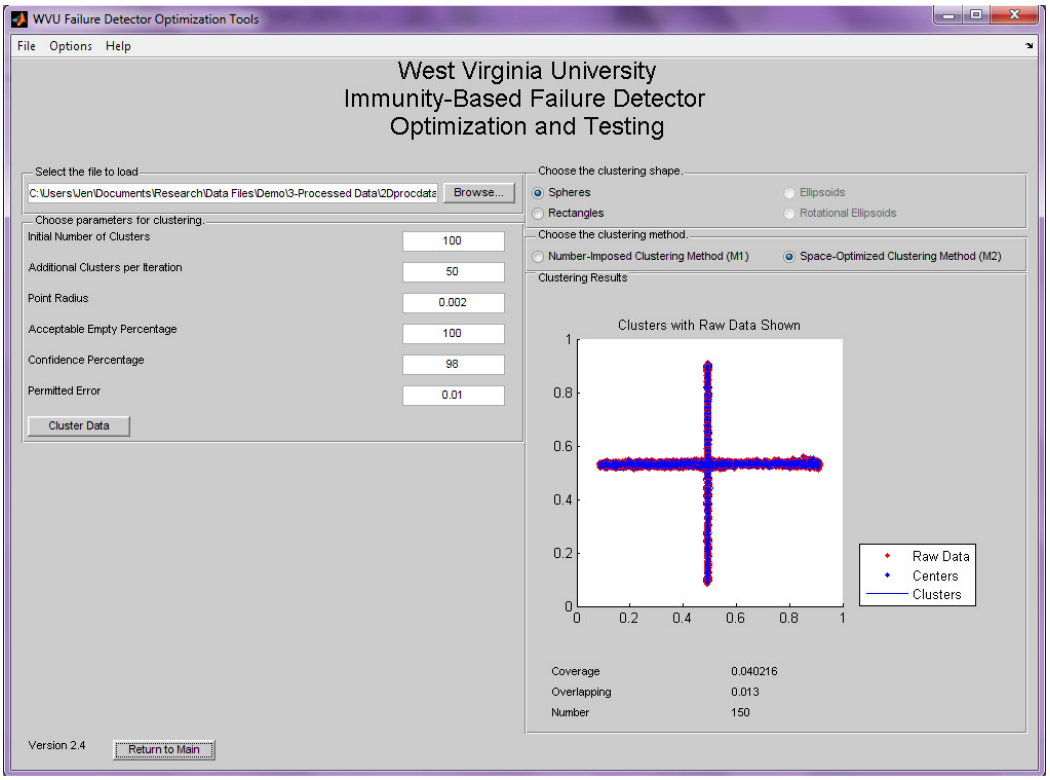

Figure B.41—Clustering M2 2-Dimensional Results

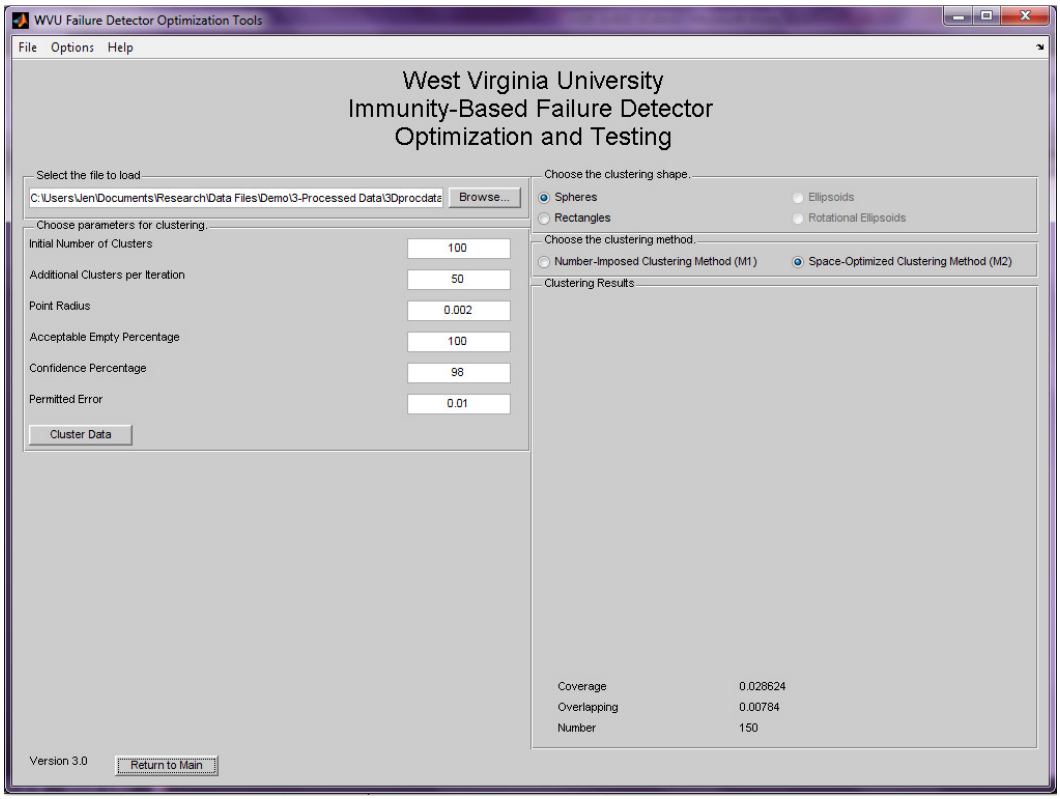

Figure B.42—Clustering M2 Higher-Dimensional Results

3.1.3 Clustering with Hyper-Rectangles

Clustering using hyper-rectangles, which uses variable-sized clusters, uses an enhanced k-means algorithm to determine the location of cluster centers within the solution space, based on the locations of the processed data points. Each point in the processed data set is then assigned to the nearest cluster center. The distance from the center to the edge of the cluster is then determined as the distance to the furthest point assigned to the center in each dimension. This results in a cluster that contains empty space, or area that is not covered by processed data points. This algorithm is not concerned with reducing the amount of empty space in clusters, so the number of clusters in the set must be chosen carefully. Lower number of clusters results in more empty space, increasing the chances that points belonging to abnormal conditions are included in the self and potentially decreasing detection rate. Higher number of clusters reduces the empty space within the clusters, increasing the chances that normal points are excluded from the definition of the self and potentially producing a high number of false alarms.

In order to begin clustering, click on the 'File' menu, select 'Data Clustering', then 'Load Processed Data', as shown in Figure B.43. This will load the menu seen in Figure B.44. Click on the browse button to load a processed data file into the program for clustering, as in Figure B.45. To follow along with this guide, navigate to the 'Demo' directory, click on the folder labeled '3-Processed Data', and select the file labeled '2Dprocdata1.mat'.

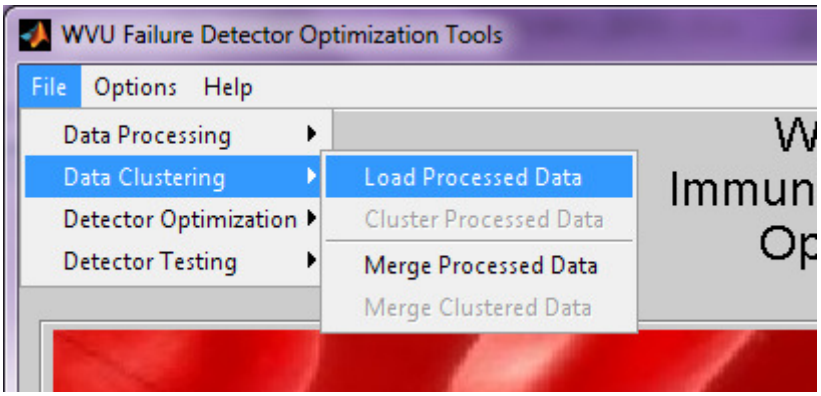

Figure B.43—Opening Load Processed Data Menu

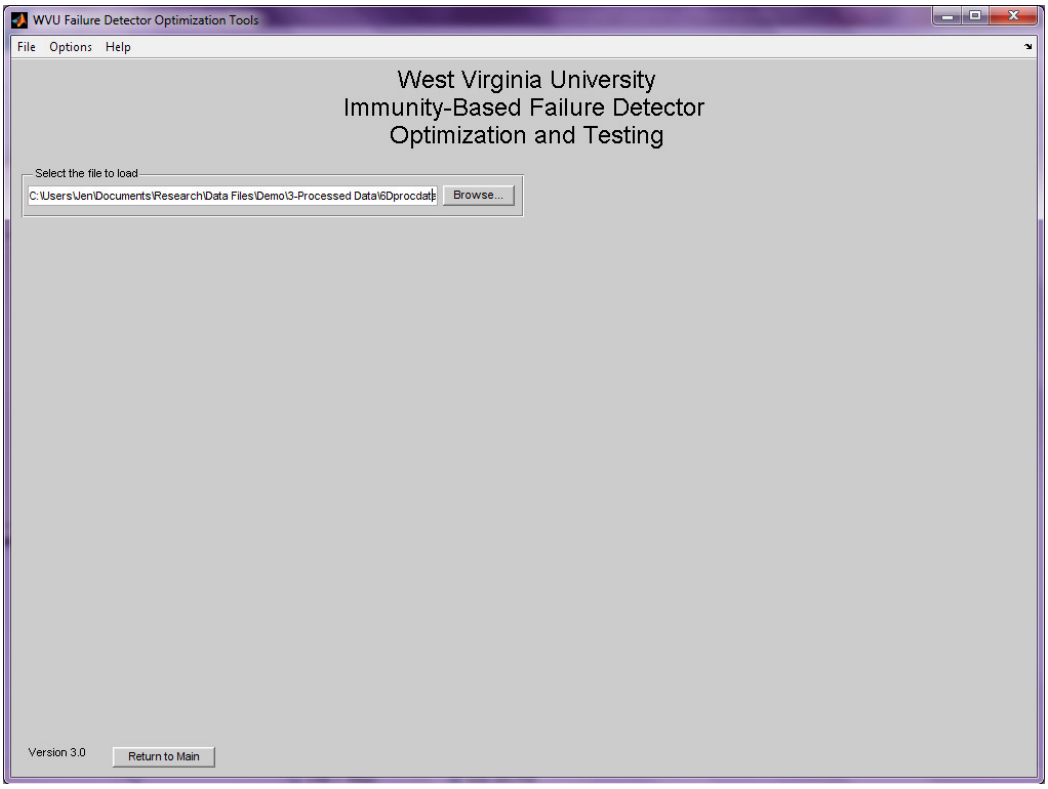

Figure B.44—Load Processed Data Menu

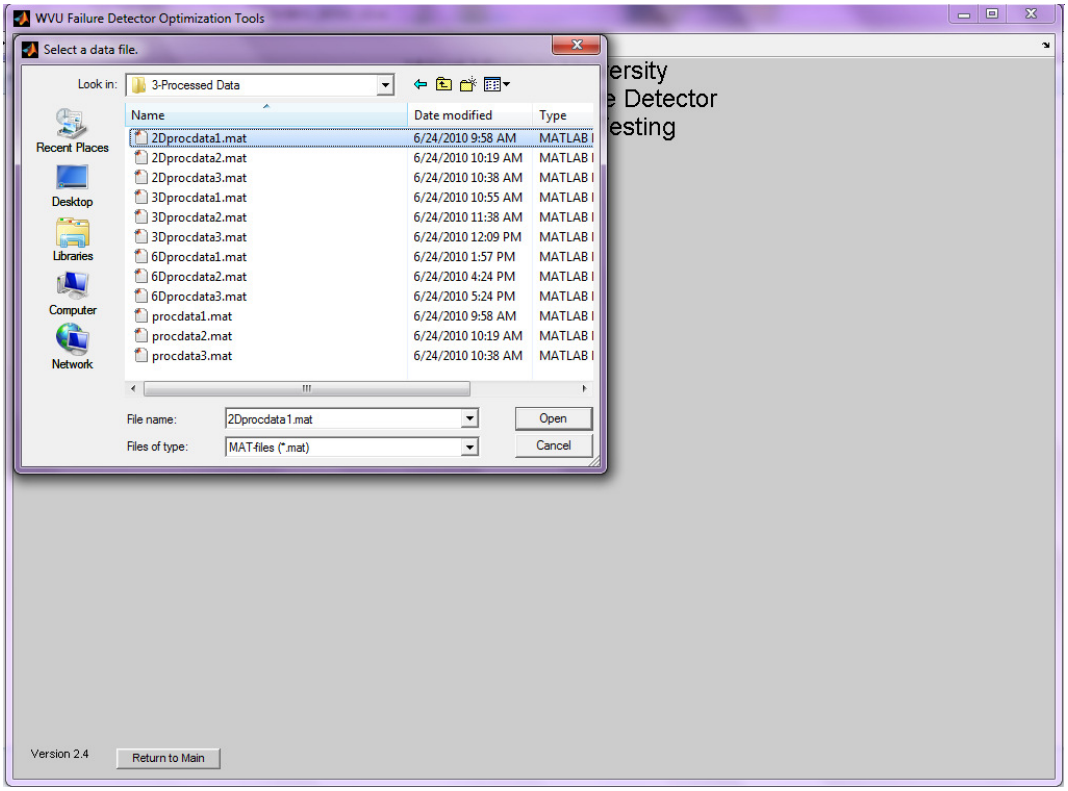

Figure B.45—Processed Data Browser Dialog

Once the processed data file has been selected, click on the 'File' menu, select 'Data Clustering' and then 'Cluster Processed Data'. This will load the menu seen in Figure B.46. This menu defaults to the M1 clustering method. To load the rectangle clustering menu select the 'Rectangles' radio button. This menu includes four parameters. These are the minimum cluster dimension, desired number of clusters, confidence percentage, and permitted error. The minimum cluster dimension is the smallest acceptable distance from the center to the edge in any dimension which may be assigned to a cluster. The desired number of clusters is the number of centers that will be generated by the k-means algorithm. The confidence percentage and permitted error are the Monte Carlo volume estimation parameters used to determine the accuracy desired when calculating the cover of the solution space and amount of overlapping present in the clustered set. Select these parameters and click the 'Cluster Data' button. To follow along with this guide, enter 0.002 as the minimum cluster dimension, 100 as the desired number of clusters, 98 as the confidence percentage, and 0.01 as the permitted error and click the 'Cluster Data' button. This will load the menu seen in Figure B.47.

Once the clustering has completed, a save dialog will open, as shown in Figure B.48. Navigate to the desired save location, enter the desired name for the clustered data file and click 'Save'. The file name chosen for this clustered data file is '2Dclust1_Rect.mat'. Then the clustering results will be displayed. If the data being clustered is 2-dimensional, the clusters will be plotted along with the self parameters, as in Figure B.49. Otherwise only the self parameters will appear, as shown in Figure B.50.

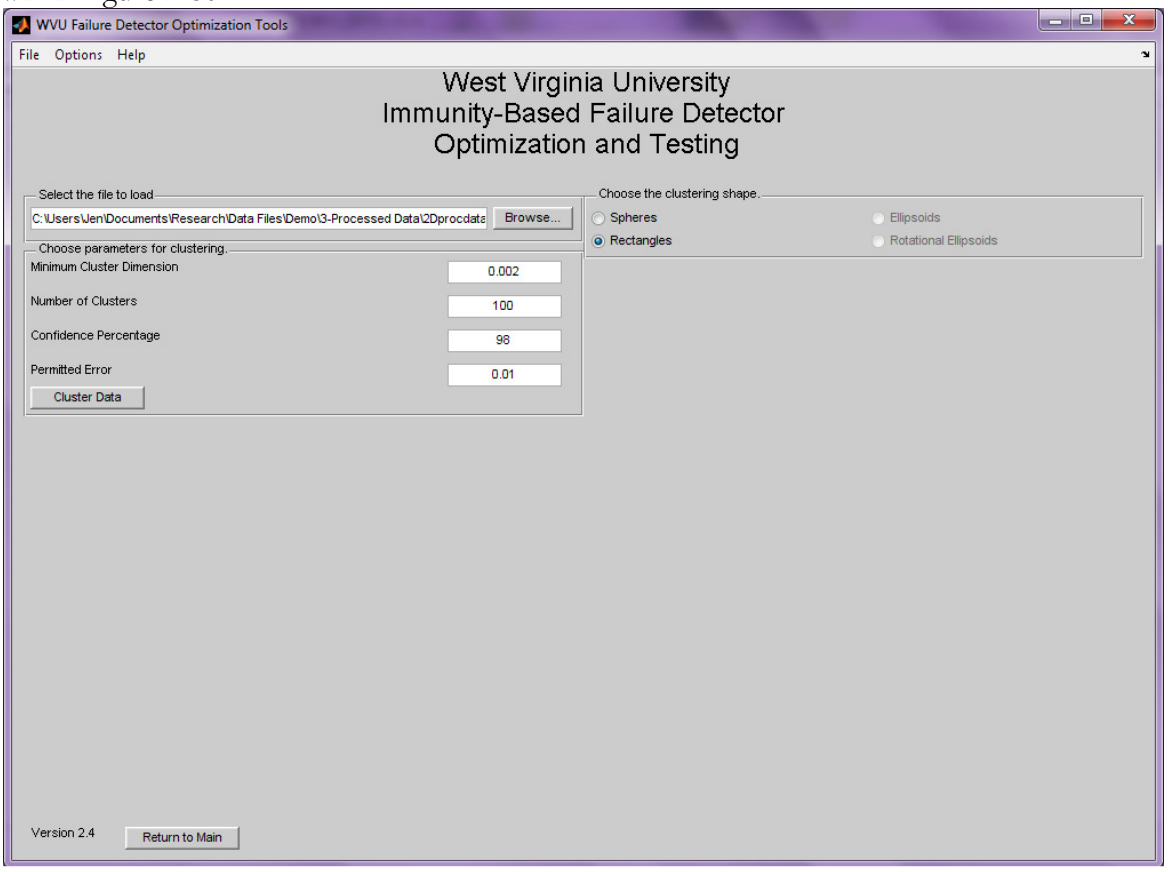

Figure B.46—Clustering Hyper-Rectangles Menu

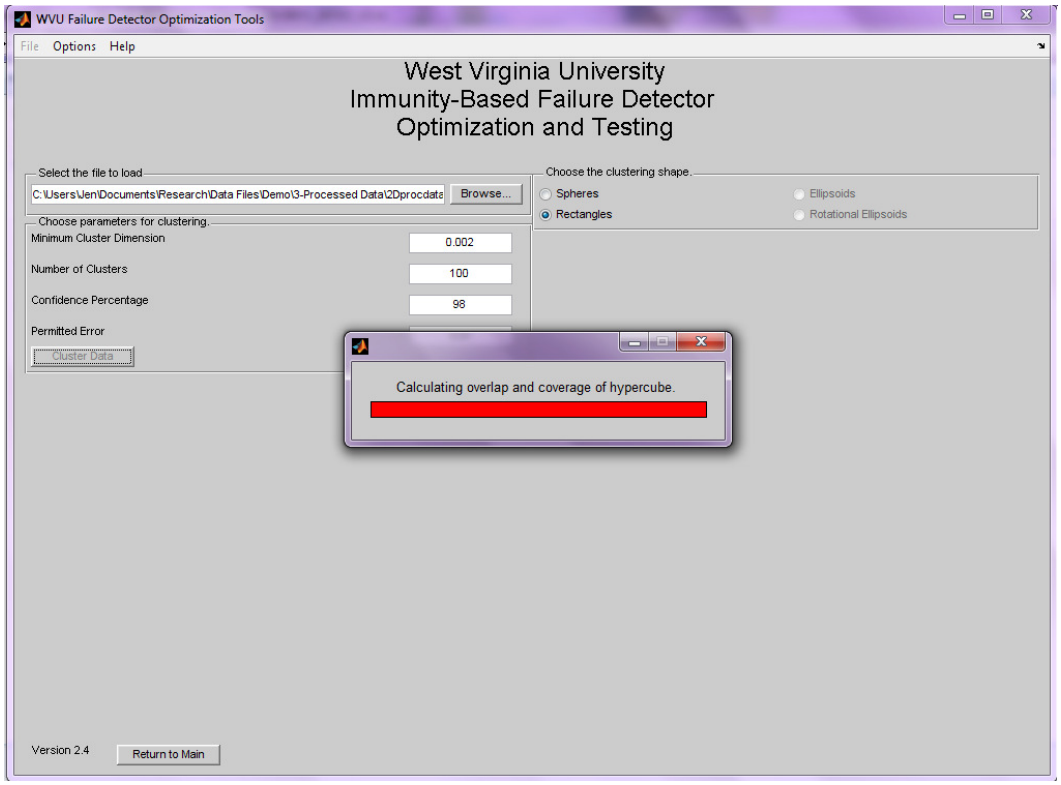

Figure B.47—Clustering Hyper-Rectangles in Progress

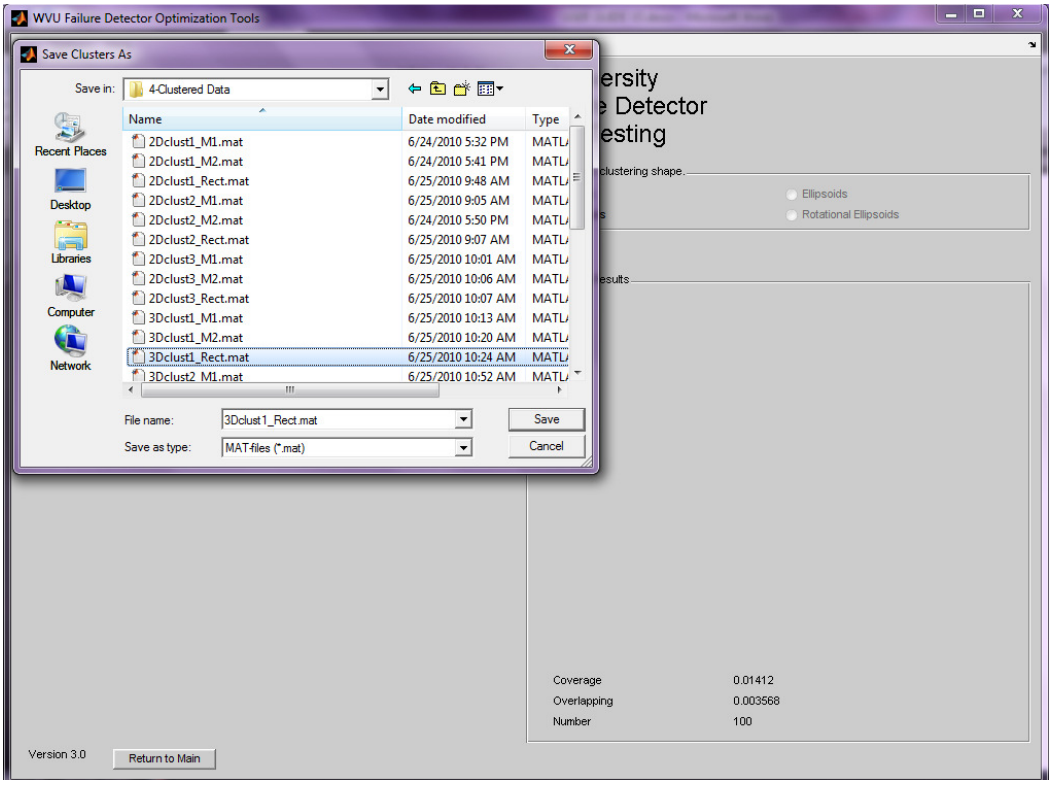

Figure B.48—Clustering Hyper-Rectangles Save Dialog

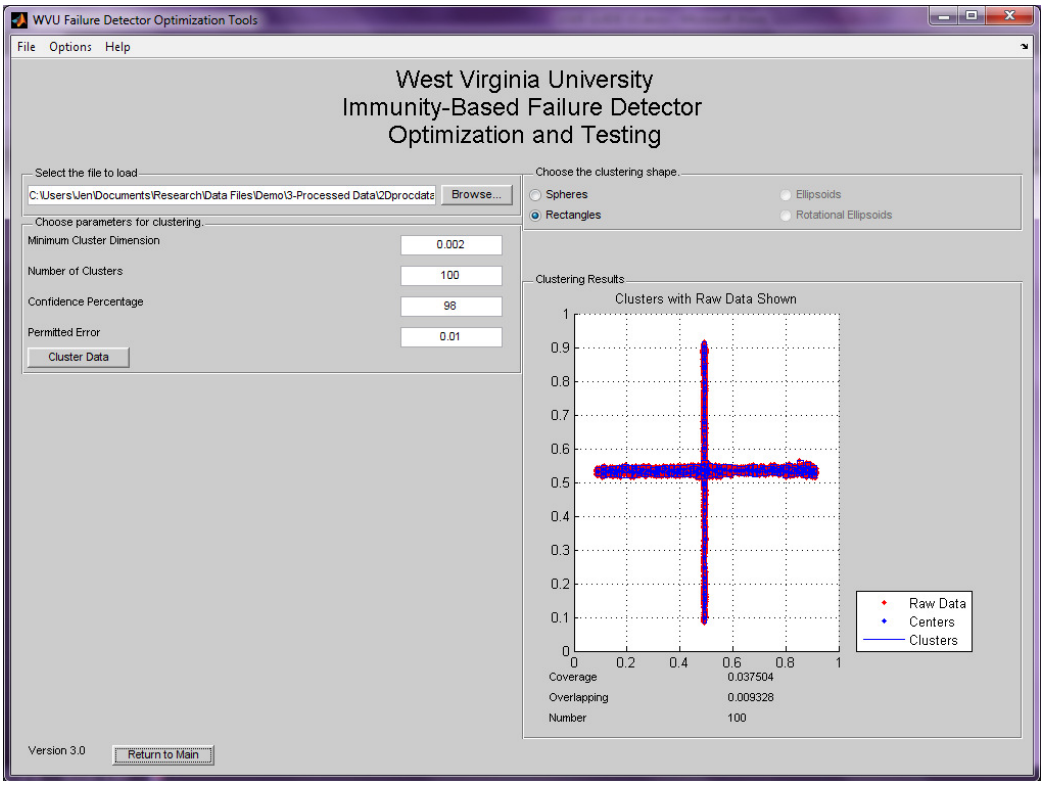

Figure B.49—Clustering Hyper-Rectangles 2-Dimensional Results

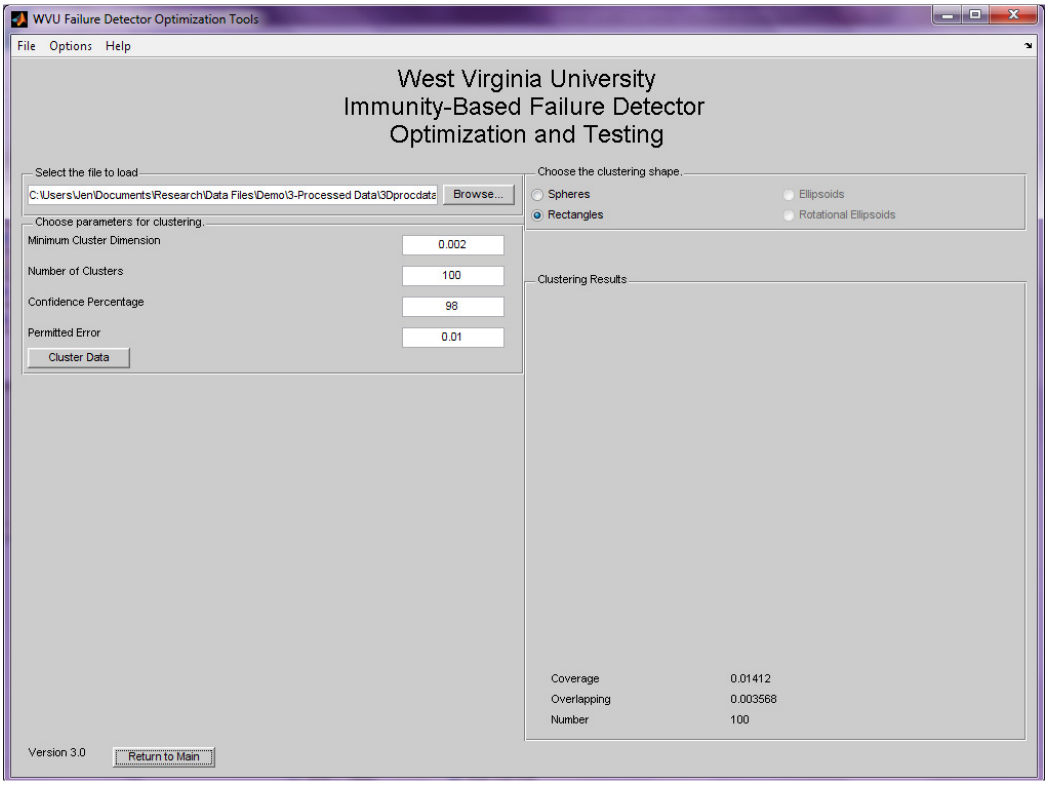

Figure B.50—Clustering Hyper-Rectangles Higher-Dimensional Results

3.2 Generation of Positive Selection Detectors

Positive selection detectors are generated in the same manner as clusters, since positive selection detectors are generated to cover the self, rather than the non-self. For this reason, 'clustering' is still used to describe the creation of positive selection detectors. Data clustering is performed on processed data to reduce the computational load imparted by the self, and to reduce the number of points ultimately needed to fully define the self. Processed data is composed of a large number of normalized time-history data points. However, this data is inherently discrete. Thus the points contained within the normal processed data do not completely define the self. In other words, there could still be points that belong to the self that are not included in the processed data set. Clustering is used as an approximation, as a way to include all points in the self that could possibly belong to the self. This is considered an approximation because the self is being defined from an incomplete set. Therefore, depending on the parameters used to generate the positive selection detectors, areas of the flight envelope that should belong to the self may be included in the non-self, or areas of the non-self could be included in the self. For this reason, care must be taken to produce effective failure detectors. Positive selection detectors are defined as either hyper-spheres, containing a center and radius, or hyper-rectangles, containing a center and the distances from the center to the edge in each dimension.

3.2.1 Positive Selection Hyper-Sphere Detector Generation Using Number-Imposed Clustering Method (M1)

Clustering method M1, which uses variable-sized clusters, uses an enhanced k-means algorithm to determine the location of cluster centers within the solution space, based on the locations of the processed data points. Each point in the processed data set is then assigned to the nearest cluster center. The radius of the cluster is then determined as the distance to the furthest point assigned to the center. This results in a cluster that contains empty space, or area that is not covered by processed data points. This algorithm is not concerned with reducing the amount of empty space in clusters, so the number of clusters in the set must be chosen carefully. Lower number of clusters results in more empty space, increasing the chances that points belonging to abnormal conditions are included in the self and potentially decreasing detection rate. Higher number of clusters reduces the empty space within the clusters, increasing the chances that normal points are excluded from the definition of the self and potentially producing a high number of false alarms.

In order to begin clustering, click on the 'File' menu, select 'Detector Optimization', then 'Positive Selection', then 'Load Processed Data' as shown in Figure B.51. This will load the menu seen in Figure B.52. Click on the browse button to load a processed data file into the program for clustering, as in Figure B.53. To follow along with this guide, navigate to the 'Demo' directory, click on the folder labeled '3-Processed Data', and select the file labeled '2Dprocdata1.mat'.

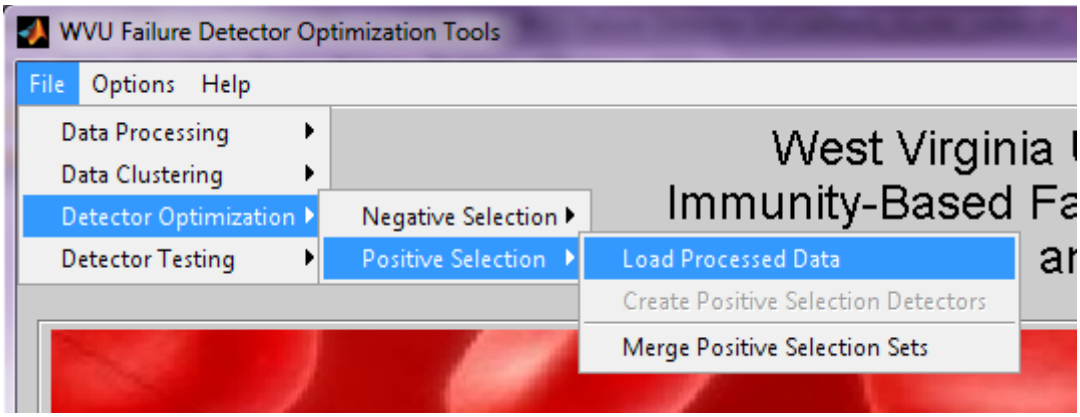

Figure B.51—Opening Load Processed Data Menu

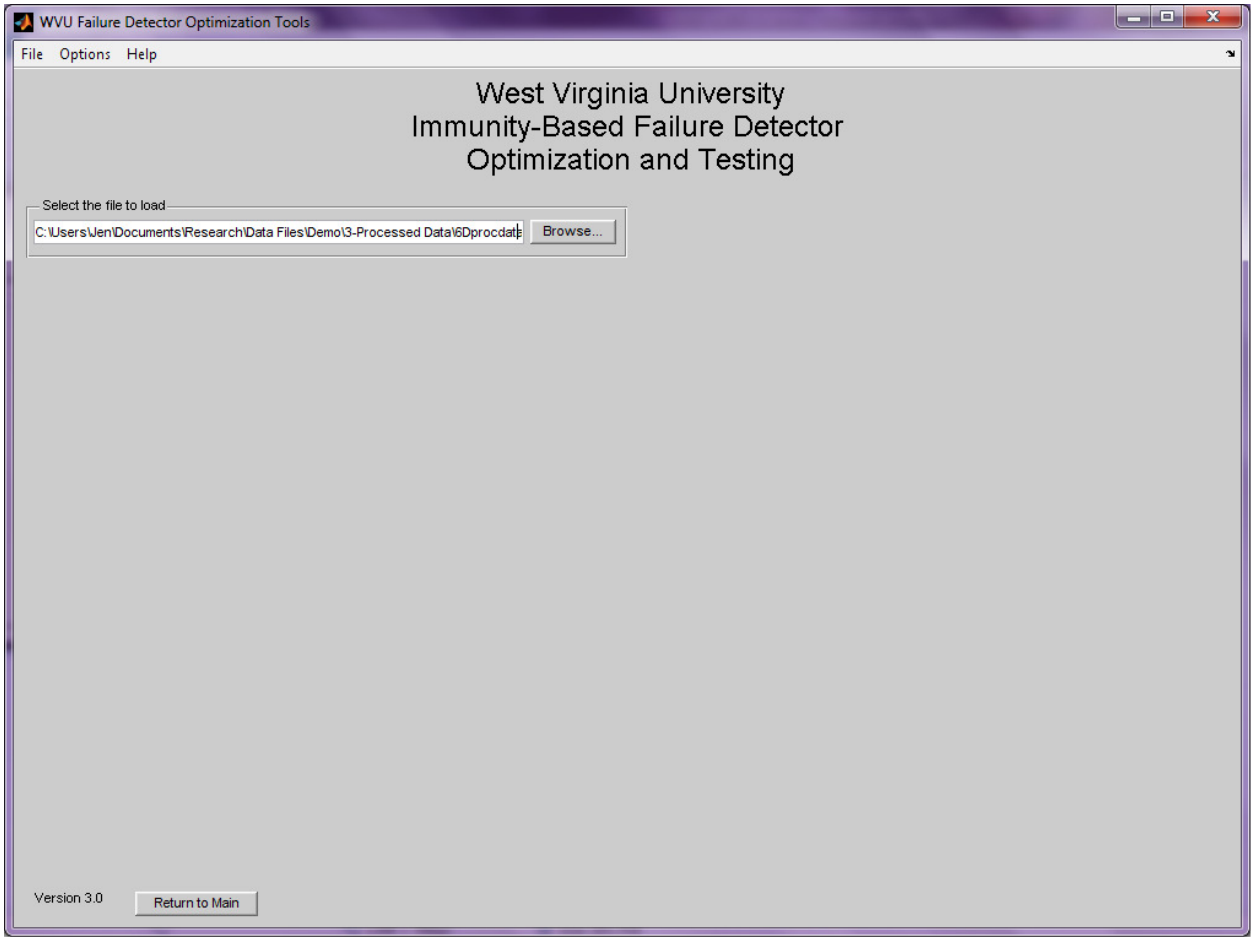

Figure B.52—Load Processed Data Menu

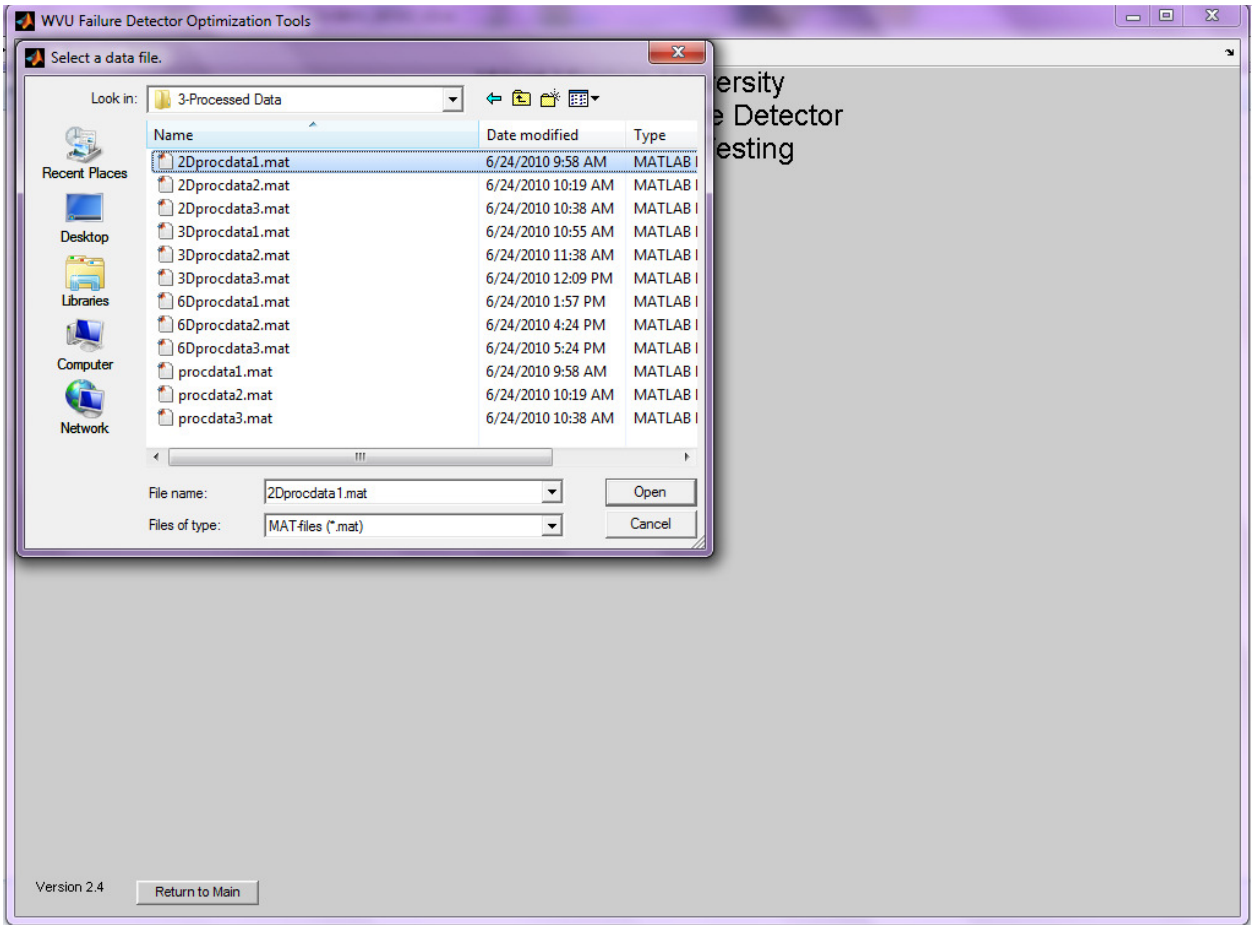

Figure B.53—Processed Data Browser Dialog

Once the processed data file has been selected, click on the 'File' menu, select 'Detector Optimization', then 'Positive Selection', then 'Create Positive Selection Detectors', as in Figure B.54. This menu defaults to the M1 clustering method, seen in Figure B.55. If the correct menu is not shown, select the 'Hyper-spheres' radio button, then the 'Number-Imposed Clustering Method (M1)' radio button. This menu includes four parameters. These are the minimum cluster radius, desired number of clusters, confidence percentage, and permitted error. The minimum cluster radius is the smallest acceptable radius which may be assigned to a cluster. The desired number of clusters is the number of centers that will be generated by the k-means algorithm. The confidence percentage and permitted error are the Monte Carlo volume estimation parameters used to determine the accuracy desired when calculating the cover of the solution space and amount of overlapping present in the clustered set. Select these parameters and click the 'Cluster Data' button. To follow along with this guide, enter 0.002 as the minimum cluster radius, 100 as the desired number of clusters, 98 as the confidence percentage, and 0.01 as the permitted error and click the 'Cluster Data' button. This will load the menu seen in Figure B.56.

Once the clustering has completed, a save dialog will open, as shown in Figure B.57. Navigate to the desired save location, enter the desired name for the clustered data file and click 'Save'. The file name chosen for this clustered data file is '2Dclust1_M1.mat'. Then the clustering results will be displayed. If the data being clustered is 2-dimensional, the clusters will be plotted along with the self parameters, as in Figure B.58. Otherwise only the self parameters will appear, as shown in Figure B.59.

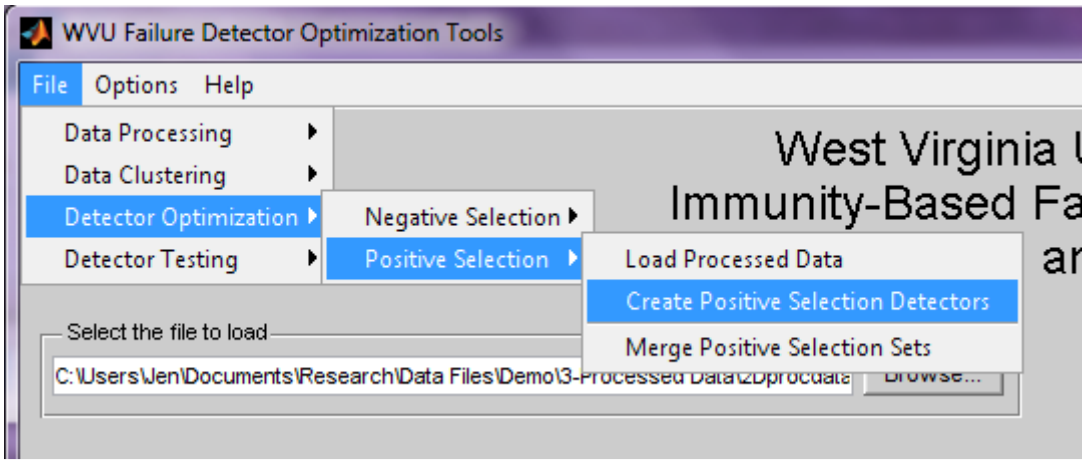

Figure B.54—Opening Positive Selection Detector Generation Menu

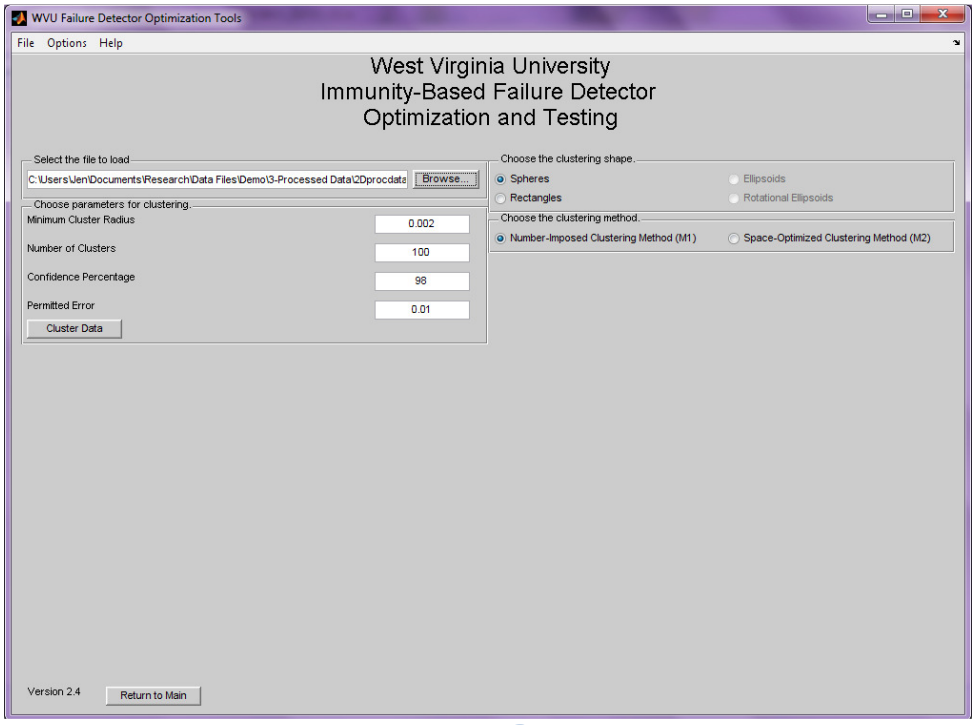

Figure B.55— Positive Detector M1 Menu

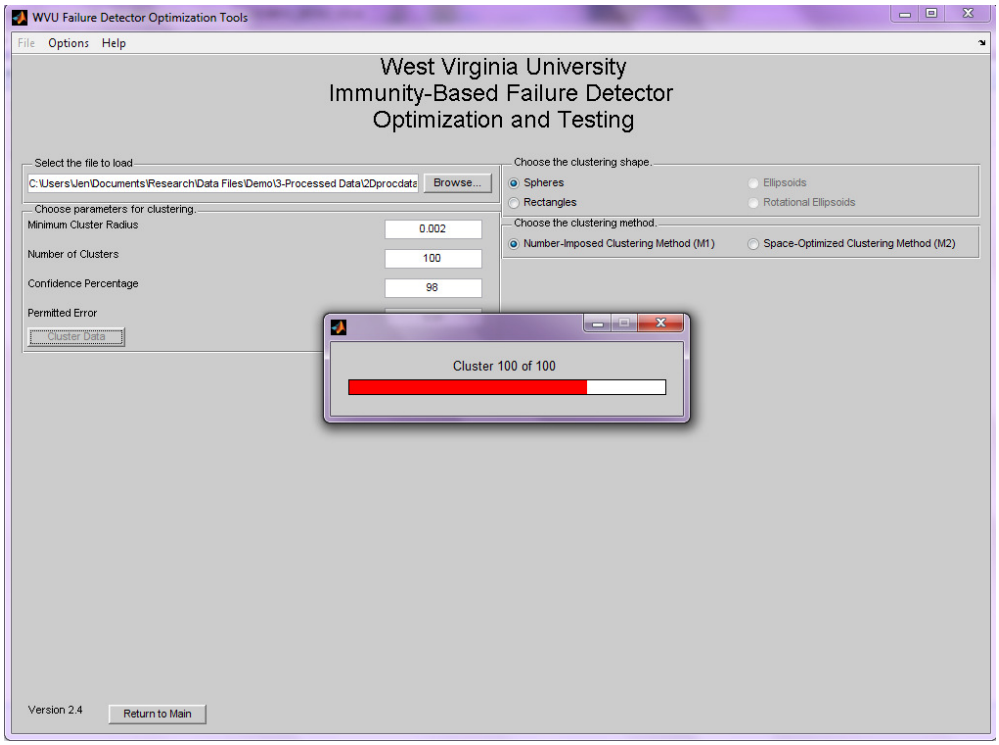

Figure B.56— Positive Detector Using M1 in Progress

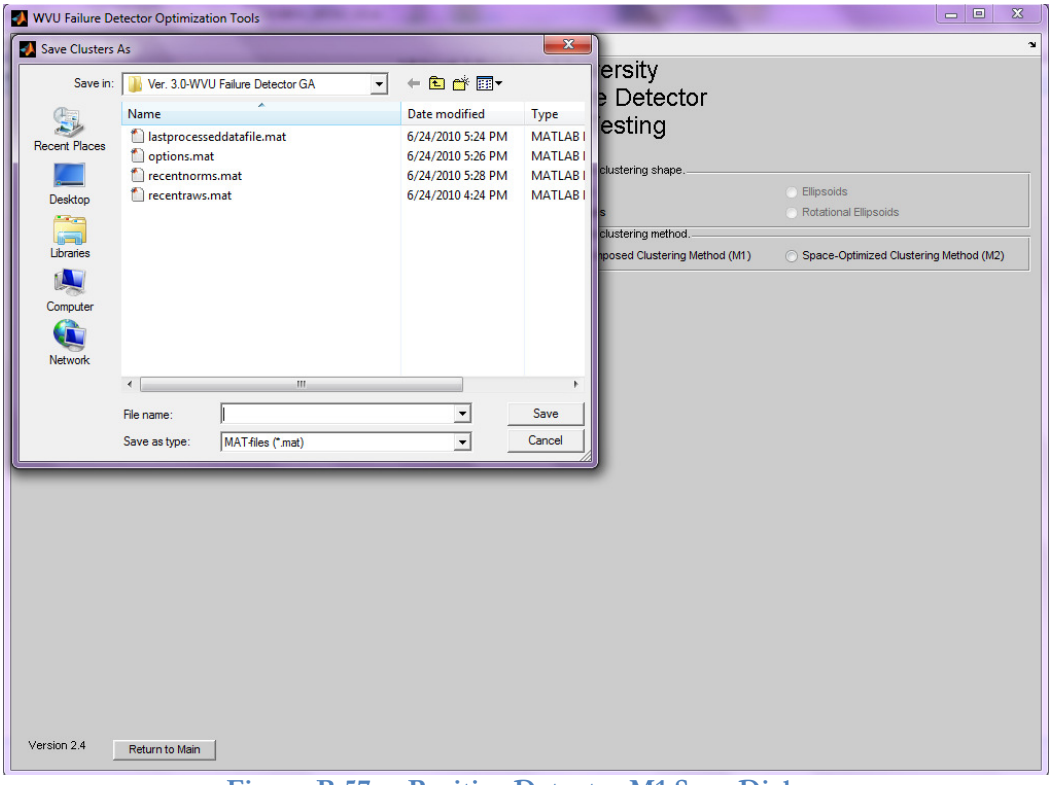

Figure B.57— Positive Detector M1 Save Dialog

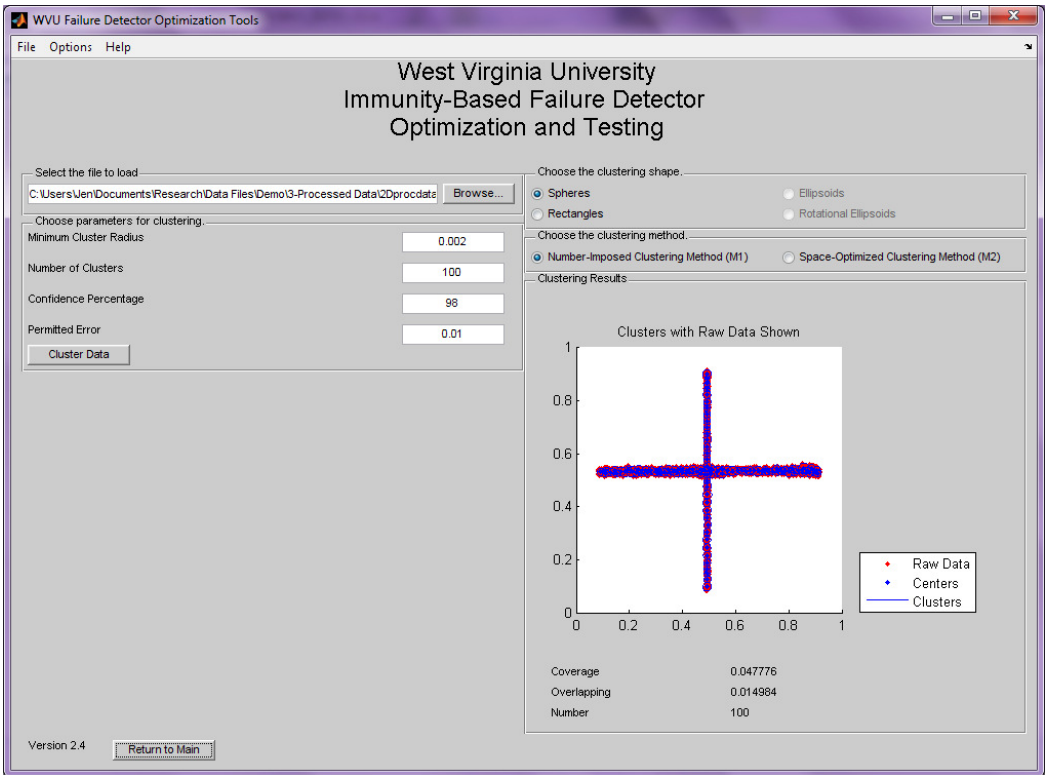

Figure B.58— Positive Detector M1 2-Dimensional Results

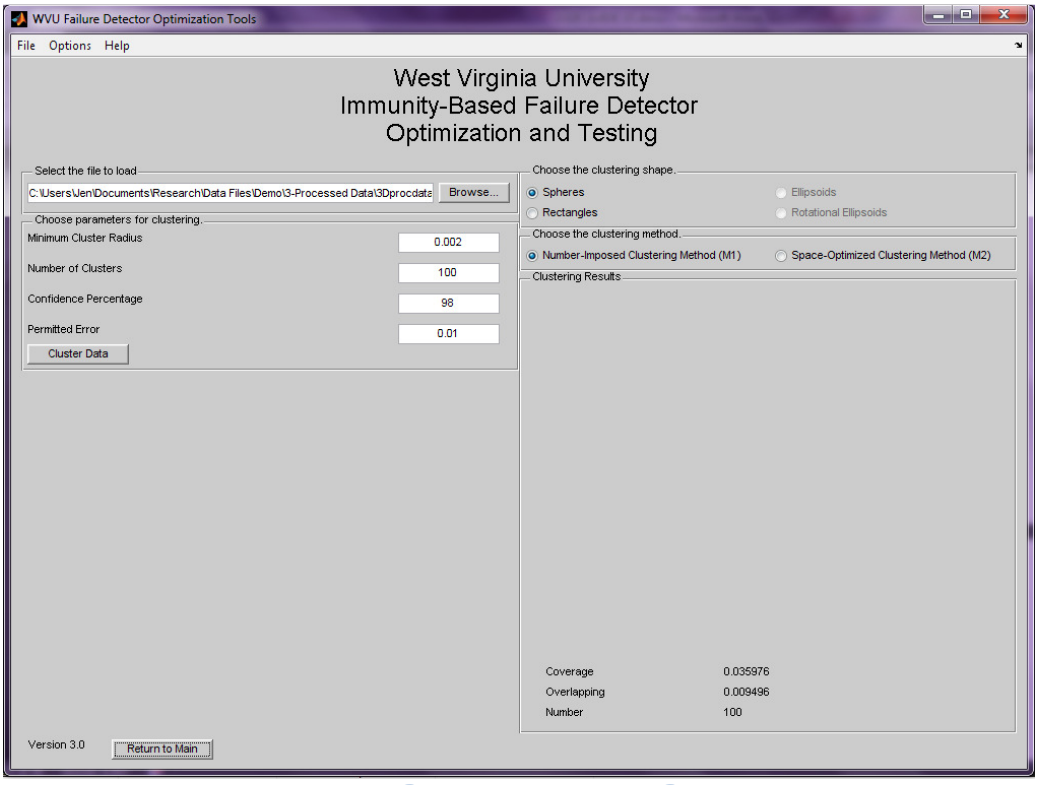

Figure B.59— Positive Detector M1 Higher-Dimensional Results

3.2.2 Positive Selection Hyper-Sphere Detector Generation Using Space-Optimized Clustering Method (M2)

Clustering method M2, which uses variable-sized clusters, uses an enhanced k-means algorithm to determine the location of cluster centers within the solution space, based on the locations of the processed data points. Each point in the processed data set is then assigned to the nearest cluster center. The radius of the cluster is then determined as the distance to the furthest point assigned to the center. This results in a cluster that contains empty space, or area that is not covered by processed data points. Lower number of clusters results in more empty space, increasing the chances that points belonging to abnormal conditions are included in the self and potentially decreasing detection rate. Higher number of clusters reduces the empty space within the clusters, increasing the chances that normal points are excluded from the definition of the self and potentially producing a high number of false alarms. This algorithm attempts to reduce the empty space within the clusters to a specified threshold by iteratively increasing the desired number of clusters and then recalculating the set.

In order to begin clustering, click on the 'File' menu, select 'Data Clustering', then 'Load Processed Data', as shown in Figure B.60. This will load the menu seen in Figure B.61. Click on the browse button to load a processed data file into the program for clustering, as in Figure B.62. To follow along with this guide, navigate to the 'Demo' directory, click on the folder labeled '3-Processed Data', and select the file labeled '2Dprocdata1.mat'.

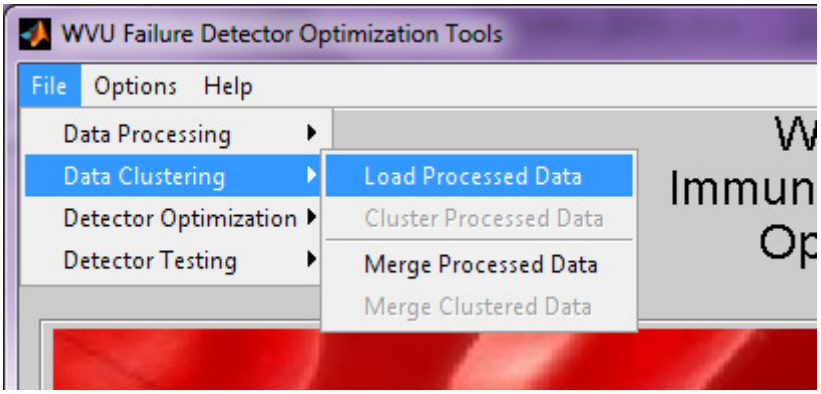

Figure B.60—Opening Load Processed Data Menu

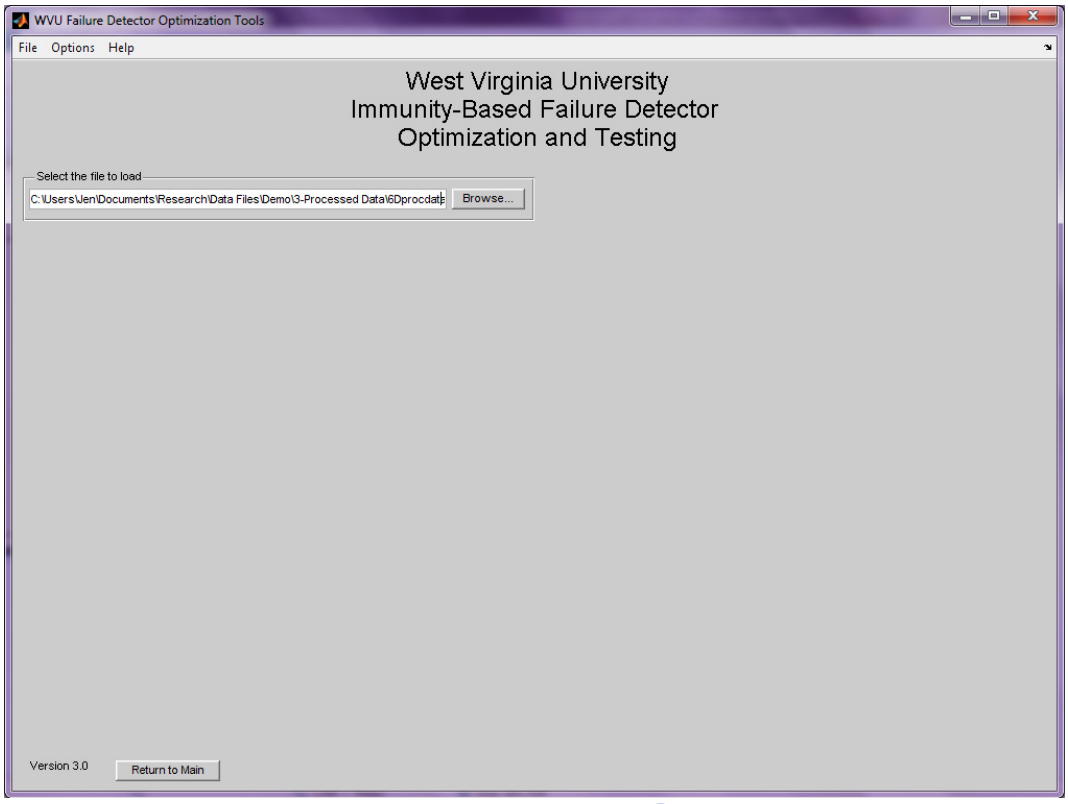

Figure B.61—Load Processed Data Menu

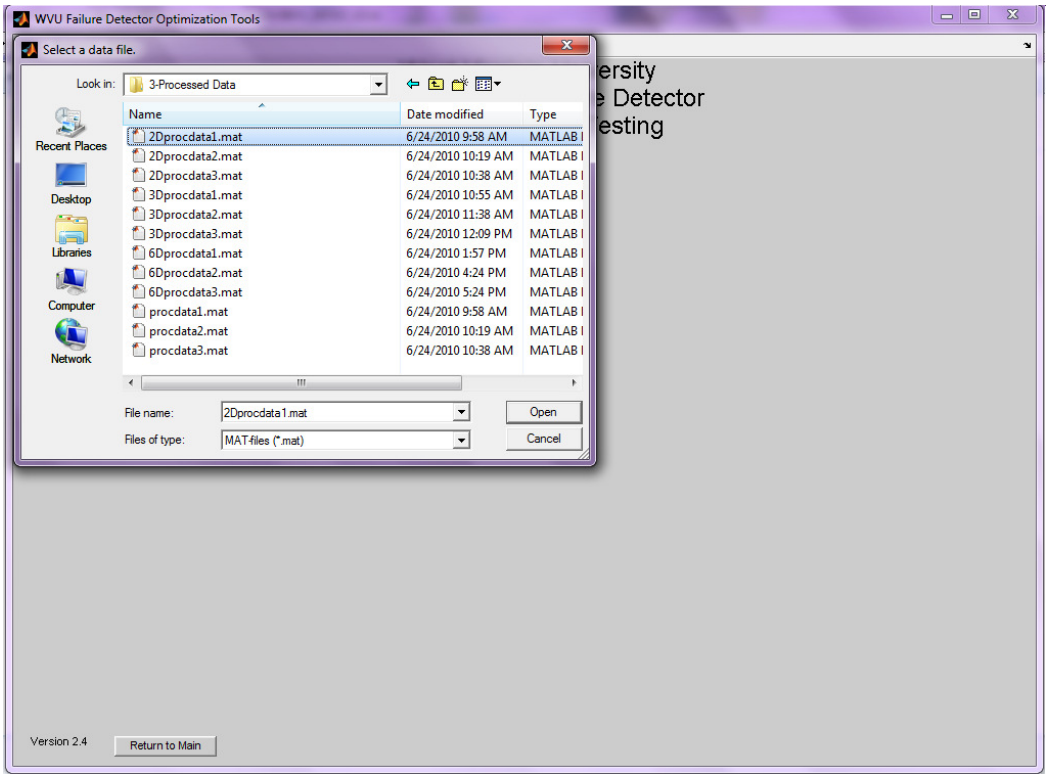

Figure B.62—Processed Data Browser Dialog

Once the processed data file has been selected, click on the 'File' menu, select 'Data Clustering' and then 'Cluster Processed Data'. This will load the menu seen in Figure B.63. This menu defaults to the M1 clustering method. To load the clustering method M2 menu, select the 'Hyper-spheres' radio button, then the 'Space-Optimized Clustering Method (M2)' radio button. This menu includes six parameters. These are the initial number of clusters, additional clusters per iteration, point radius, acceptable empty percentage, confidence percentage, and permitted error. The initial number of clusters is the minimum number of desired clusters in the self. The additional number of clusters per iteration is how many centers more centers to generate using the k-means algorithm with each iteration. The point radius is the radius around each of the processed data points that can be confidently considered part of the normal data set. The acceptable empty percentage is the amount of empty space that may exist within the clusters for the cluster set to be finalized. The confidence percentage and permitted error are the Monte Carlo volume estimation parameters used to determine the accuracy desired when calculating the cover of the solution space and amount of overlapping present in the clustered set. Select these parameters and click the 'Cluster Data' button. To follow along with this guide, enter 100 as the initial number of clusters, 50 as the additional clusters per iteration, 0.002 as the point radius, 100 as the acceptable empty percentage, 98 as the confidence percentage, and 0.01 as the permitted error and click the 'Cluster Data' button. This will load the menu seen in Figure B.64.

Once the clustering has completed, a save dialog will open, as shown in Figure B.65. Navigate to the desired save location, enter the desired name for the clustered data file and click 'Save'. The file name chosen for this clustered data file is '2Dclust1_M2.mat'. Then the clustering results will be displayed. If the data being clustered is 2-dimensional, the clusters will be plotted along with the self parameters, as in Figure B.66. Otherwise only the self parameters will appear, as shown in Figure B.67.

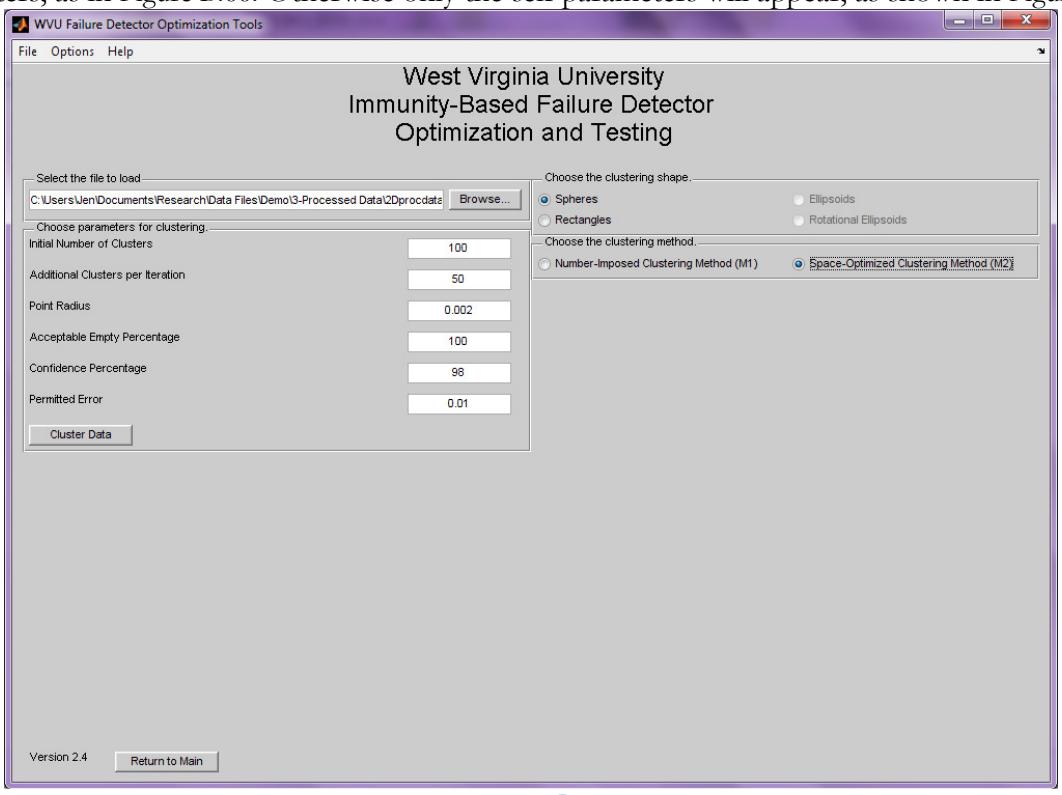

Figure B.63— Positive Detector M2 Menu

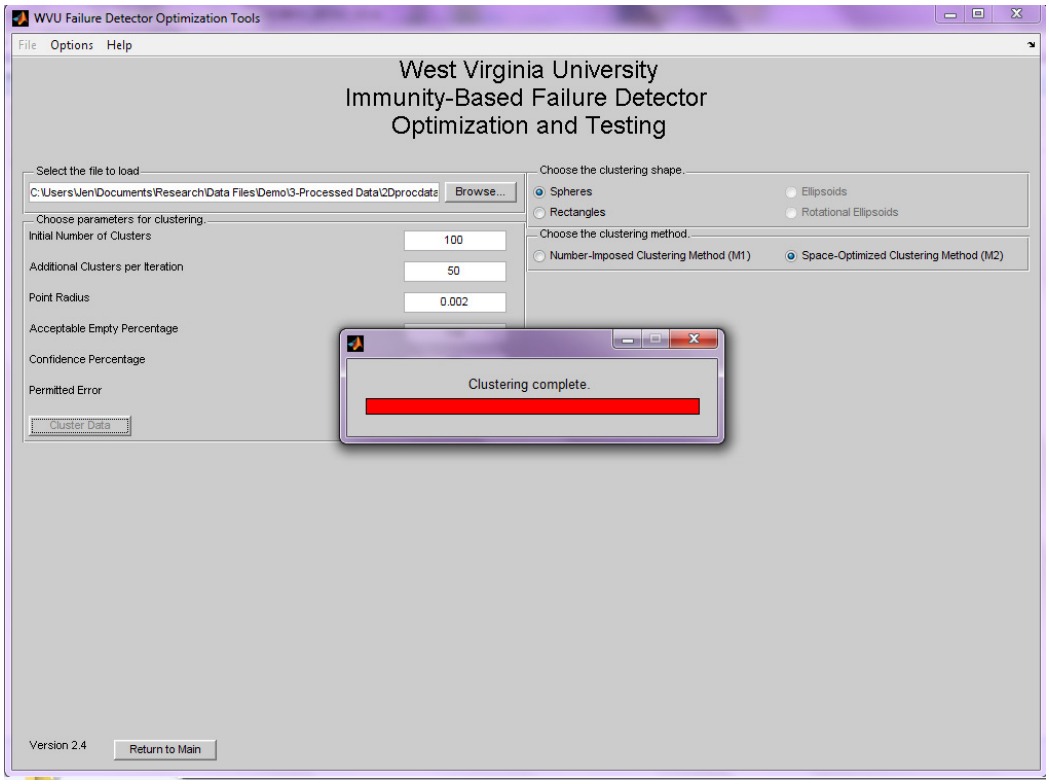

Figure B.64— Positive Detector Using M2 in Progress

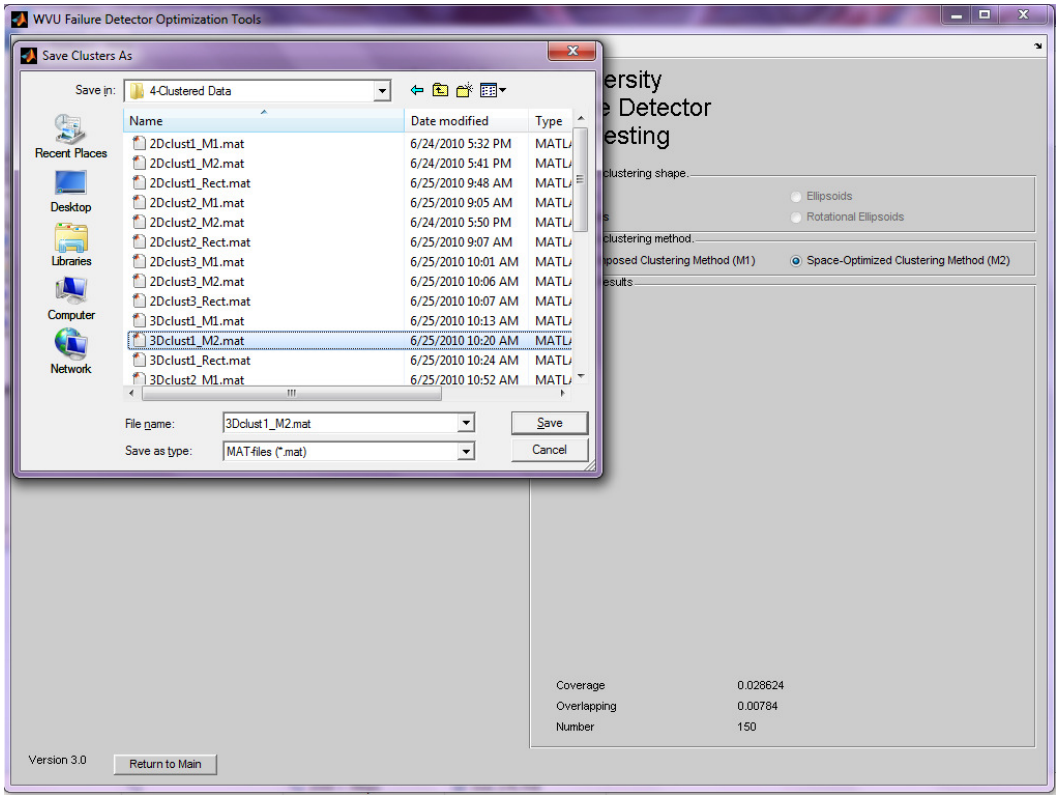

Figure B.65— Positive Detector M2 Save Dialog

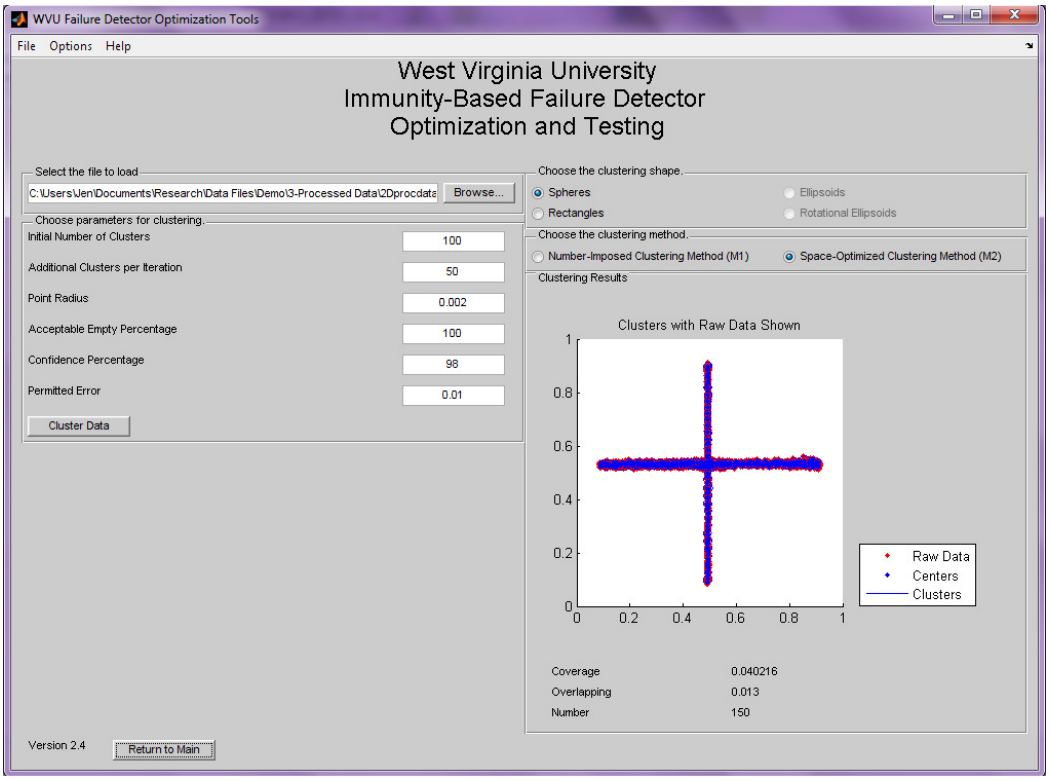

Figure B.66— Positive Detector M2 2-Dimensional Results

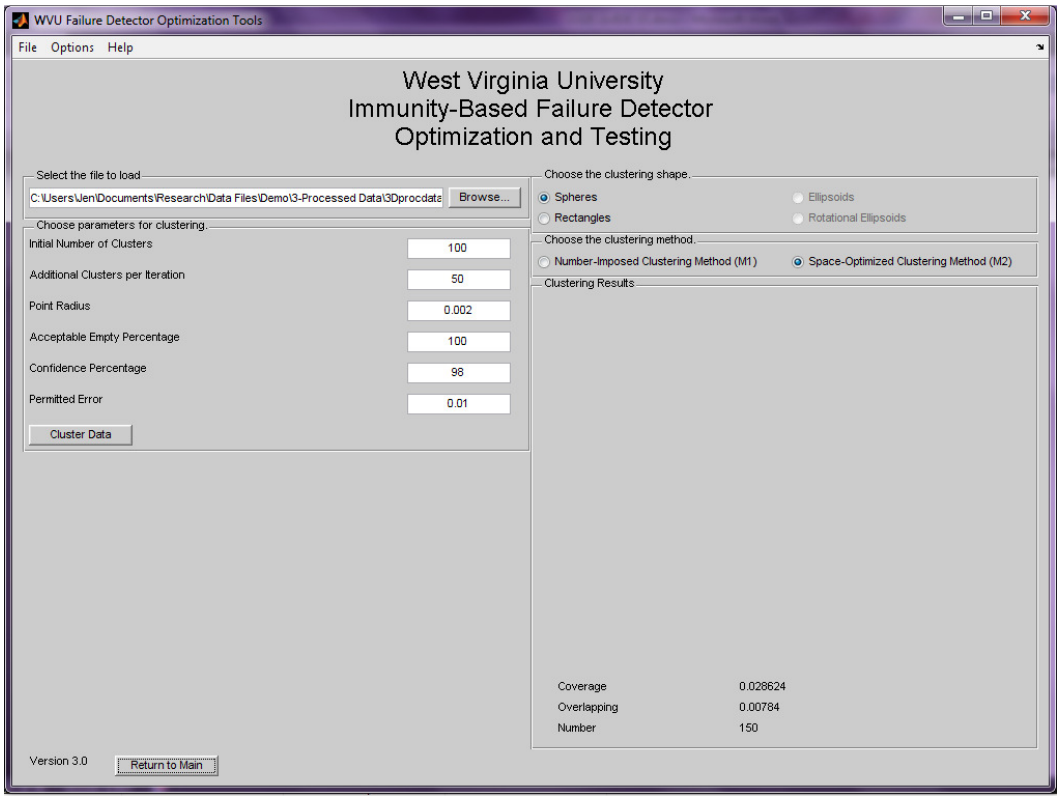

Figure B.67— Positive Detector M2 Higher-Dimensional Results

3.2.3 Positive Selection Hyper-Rectangle Detector Generation

Clustering using hyper-rectangles, which uses variable-sized clusters, uses an enhanced k-means algorithm to determine the location of cluster centers within the solution space, based on the locations of the processed data points. Each point in the processed data set is then assigned to the nearest cluster center. The distance from the center to the edge of the cluster is then determined as the distance to the furthest point assigned to the center in each dimension. This results in a cluster that contains empty space, or area that is not covered by processed data points. This algorithm is not concerned with reducing the amount of empty space in clusters, so the number of clusters in the set must be chosen carefully. Lower number of clusters results in more empty space, increasing the chances that points belonging to abnormal conditions are included in the self and potentially decreasing detection rate. Higher number of clusters reduces the empty space within the clusters, increasing the chances that normal points are excluded from the definition of the self and potentially producing a high number of false alarms.

In order to begin clustering, click on the 'File' menu, select 'Data Clustering', then 'Load Processed Data', as shown in Figure B.68. This will load the menu seen in Figure B.69. Click on the browse button to load a processed data file into the program for clustering, as in Figure B.70. To follow along with this guide, navigate to the 'Demo' directory, click on the folder labeled '3-Processed Data', and select the file labeled '2Dprocdata1.mat'.

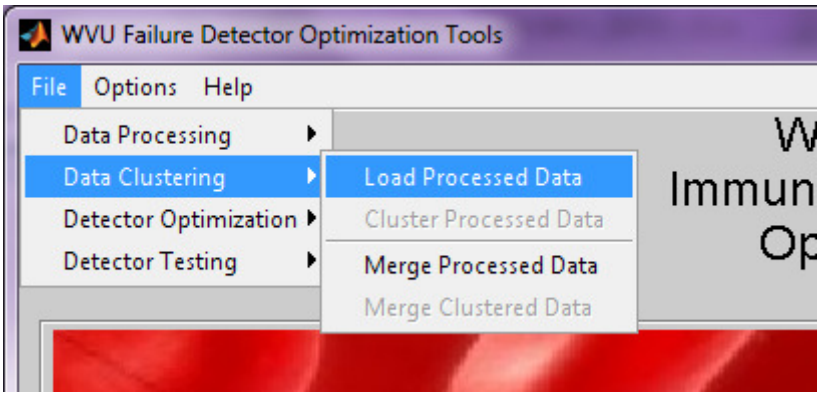

Figure B.68—Opening Load Processed Data Menu

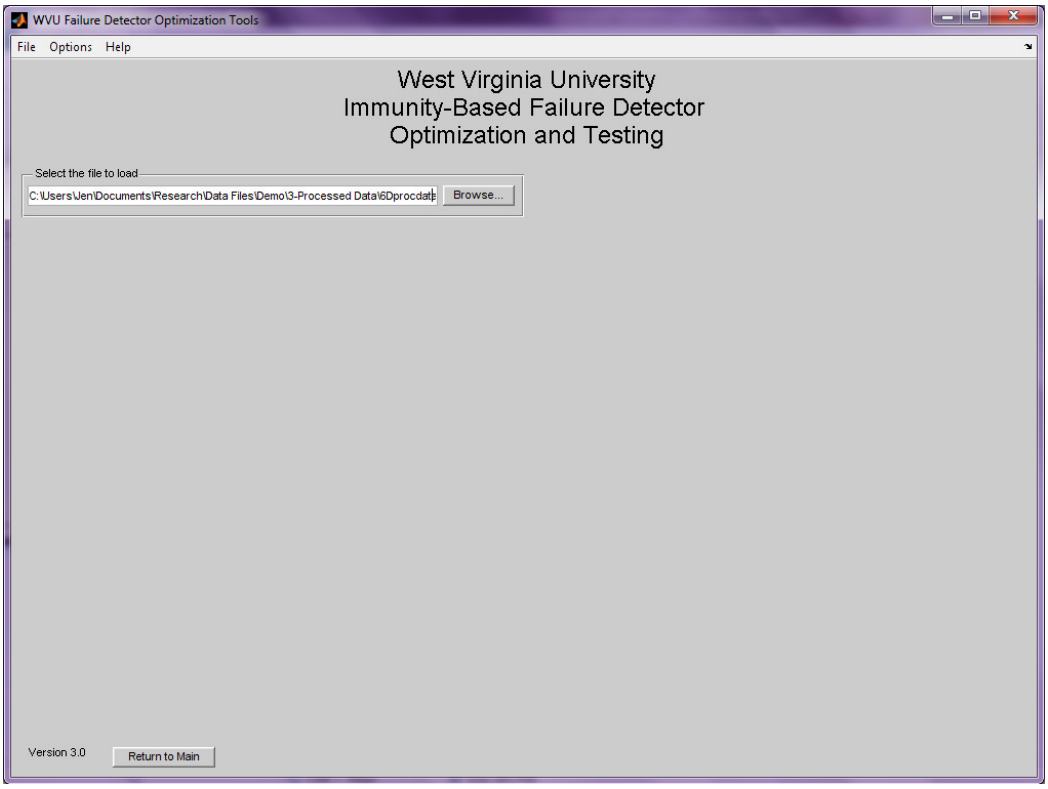

Figure B.69—Load Processed Data Menu

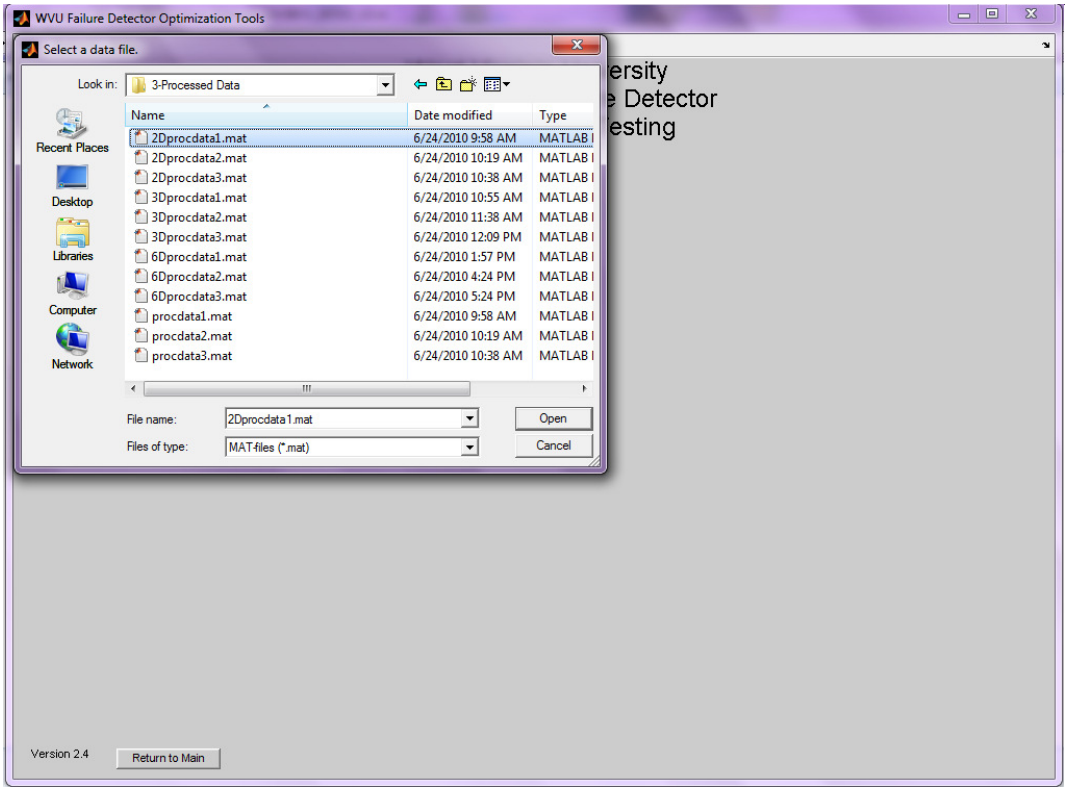

Figure B.70—Processed Data Browser Dialog

Once the processed data file has been selected, click on the 'File' menu, select 'Data Clustering' and then 'Cluster Processed Data'. This will load the menu seen in Figure B.71. This menu defaults to the M1 clustering method. To load the rectangle clustering menu select the 'Rectangles' radio button. This menu includes four parameters. These are the minimum cluster dimension, desired number of clusters, confidence percentage, and permitted error. The minimum cluster dimension is the smallest acceptable distance from the center to the edge in any dimension which may be assigned to a cluster. The desired number of clusters is the number of centers that will be generated by the k-means algorithm. The confidence percentage and permitted error are the Monte Carlo volume estimation parameters used to determine the accuracy desired when calculating the cover of the solution space and amount of overlapping present in the clustered set. Select these parameters and click the 'Cluster Data' button. To follow along with this guide, enter 0.002 as the minimum cluster dimension, 100 as the desired number of clusters, 98 as the confidence percentage, and 0.01 as the permitted error and click the 'Cluster Data' button. This will load the menu seen in Figure B.72.

Once the clustering has completed, a save dialog will open, as shown in Figure B.73. Navigate to the desired save location, enter the desired name for the clustered data file and click 'Save'. The file name chosen for this clustered data file is '2Dclust1_Rect.mat'. Then the clustering results will be displayed. If the data being clustered is 2-dimensional, the clusters will be plotted along with the self parameters, as in Figure B.74. Otherwise only the self parameters will appear, as shown in Figure B.75.

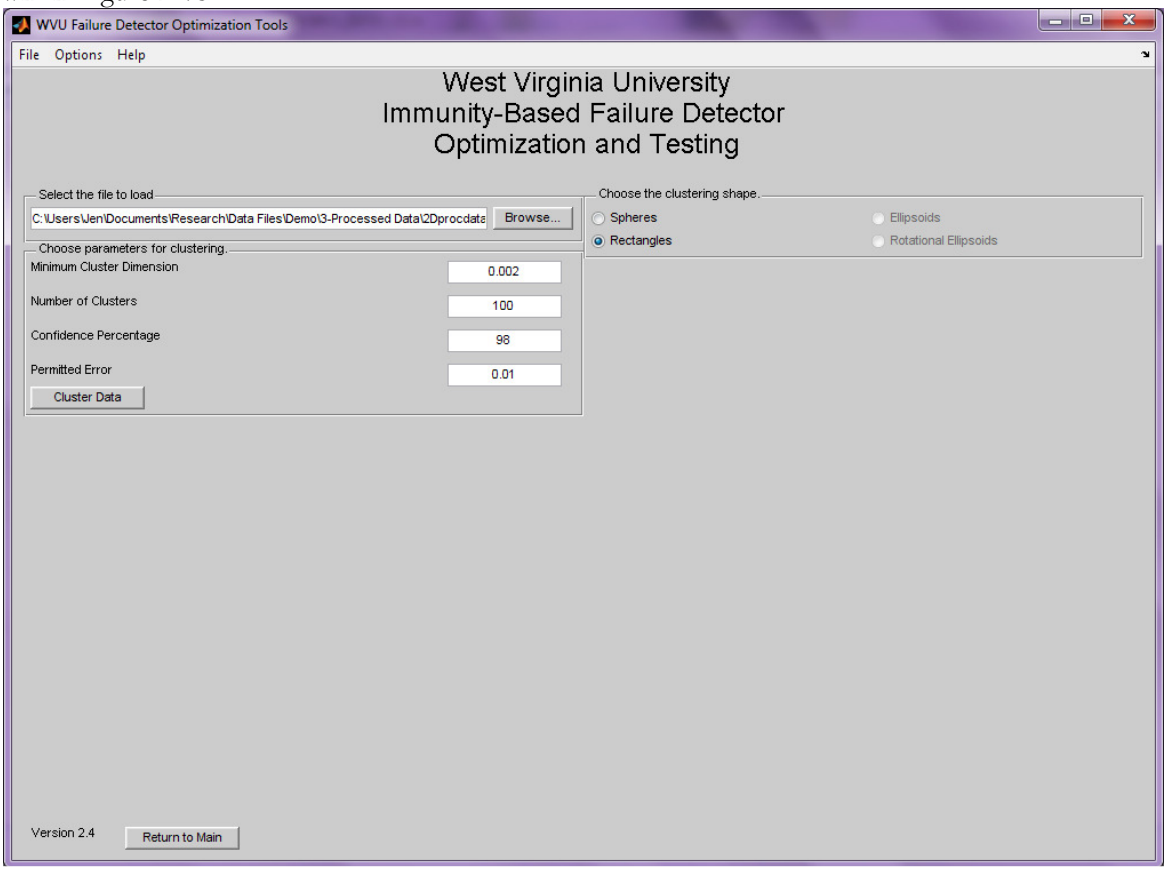

Figure B.71— Positive Detector Hyper-Rectangles Menu

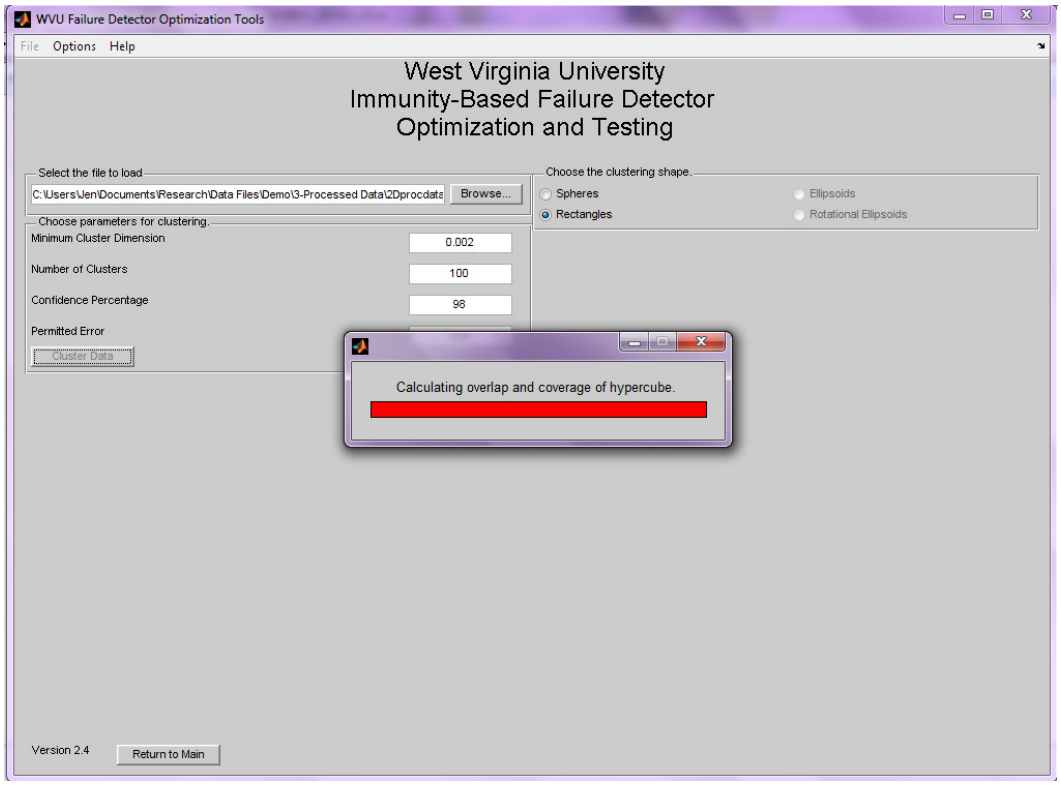

Figure B.72— Positive Detector Hyper-Rectangles in Progress

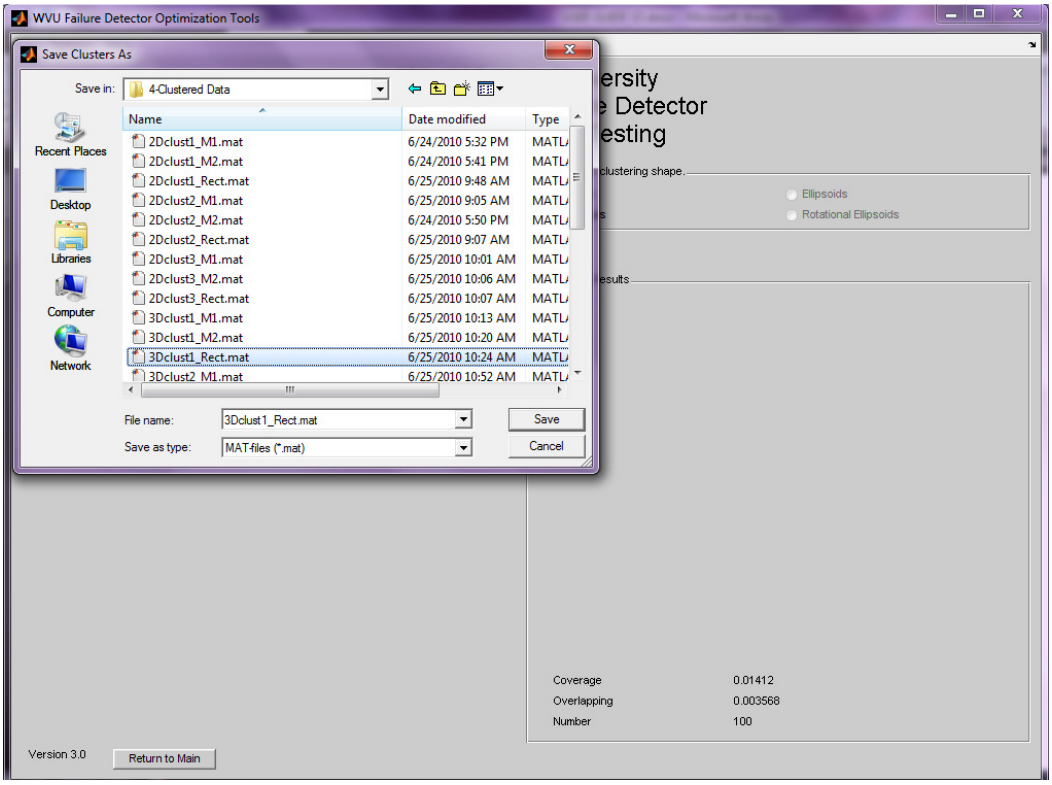

Figure B.73— Positive Detector Hyper-Rectangles Save Dialog

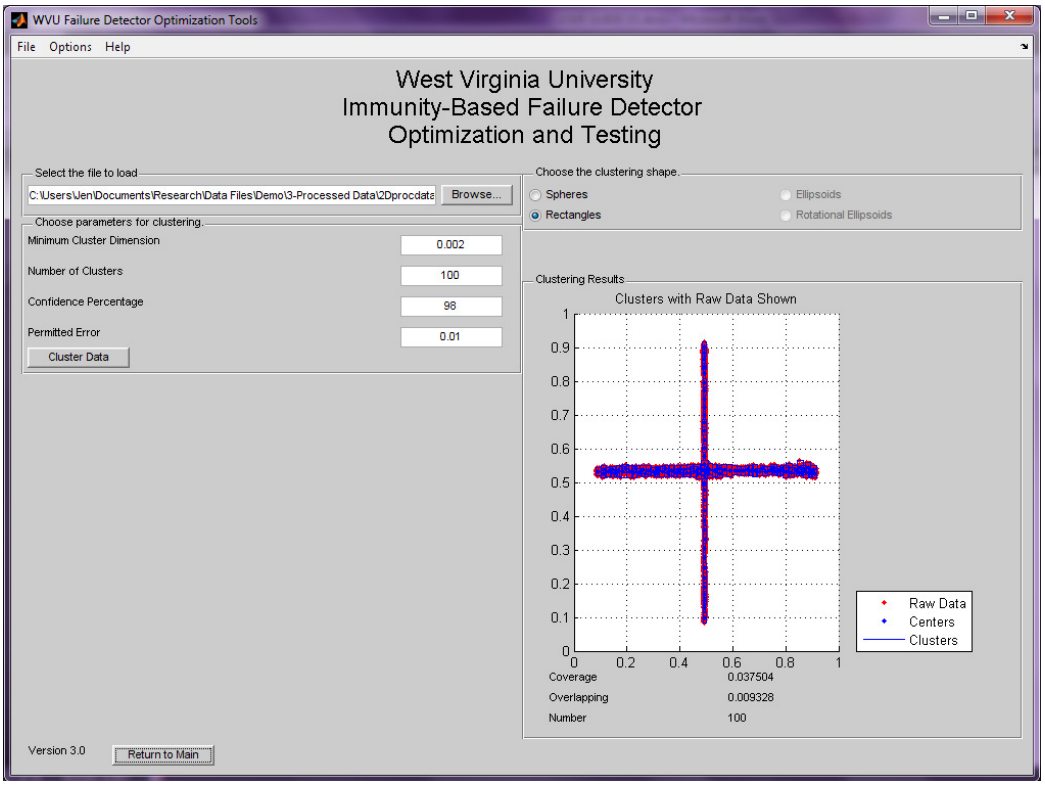

Figure B.74—Positive Detector Hyper-Rectangles 2-Dimensional Results

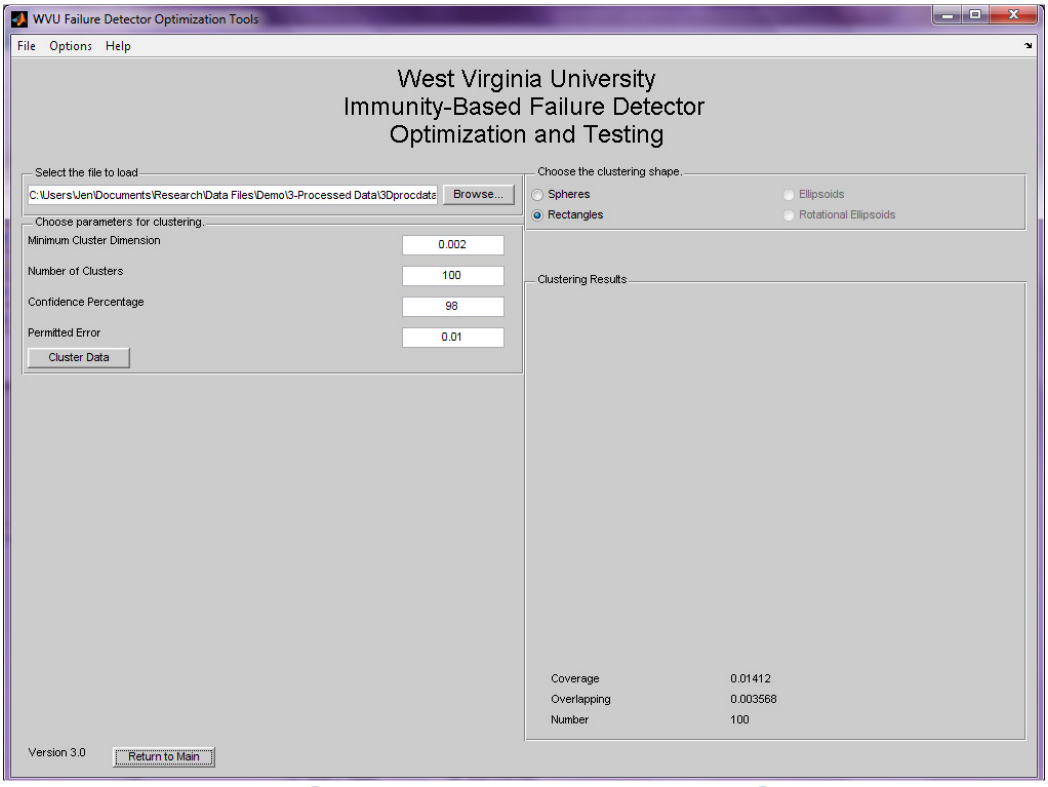

Figure B.75—Positive Detector Hyper-Rectangles Higher-Dimensional Results

Chapter 4—Creating Negative Selection Detectors and Optimization with Genetic Algorithm

Creating negative selection detectors requires a properly defined, processed, and clustered data set. Several methods and shapes are available for producing detectors within the categories of producing and optimizing detectors. These methods will be discussed in this chapter of the guide.

4.1 Creating a Single Detector Set

At times it is useful to generate only a single detector set, without optimization. This is useful for determining if the detector generation parameters being used are effective and if the identifiers contained in the data are capable of detecting the desired failures before optimization is performed. It is also possible to perform this task prior to optimization with the genetic algorithm, since the following algorithms are used to create the initial population for the genetic algorithm. When the evolutionary algorithm menu is loaded, a button labeled 'Check Detector Parameters' will appear in place of the 'Create Detectors' button in the menus below. In addition, for the check detectors functionality, the detector set will not be saved as it is used as an example only, and is not a part of the initial population of the genetic algorithm.

4.1.1 Creating Hyper-Sphere Detectors with NSA-RV

This method of generating hyper-sphere detectors generates centers at random within the solution space. If the center does not fall within another detector or within the self, the center is assigned a radius equal to the distance from the center to the nearest edge of the self. If this center is greater than a specified minimum threshold, the detector is accepted. This continues until too many random centers have been rejected for falling within the self, too many centers have been rejected for falling within existing detectors, or the specified number of detectors has been reached.

To create a set of detectors, begin by clicking on the 'File' menu, selecting 'Detector Optimization', then 'Negative Selection', then 'Load Clustered Data', as in Figure B.76. Click on the browse button and load a data file containing hyper-sphere clusters (See section 4.1.2 for hyper-rectangle clusters), as in Figure B.77. To follow along with this guide, select the file labeled '2Dclust1_M1.mat' within the '4-Clustered Data' folder in the 'Demo' directory. Click on the 'File' menu, select 'Negative Selection', then 'Create Detectors (Phase I only)', as seen in Figure B.78. This will load the menu seen in Figure B.79. This menu contains four parameters which must be chosen in order to perform detector generation. These are the minimum detector radius, maximum number of detectors, non-self coverage, and self coverage. The minimum detector radius is the smallest radius a center can be assigned and be accepted as a mature detector. The default for this value is 0.008. The maximum number of detectors is the largest desired number of detectors the set is permitted to contain. The default for this value is 100. The non-self coverage is a stopping criteria based on the number of random centers that have fallen within other detectors. This demonstrates coverage of the large non-self regions. This parameter should be nearly 1. The default for this value is 0.9999. The self coverage is a stopping criteria based on the number of random centers that have fallen within self clusters. This demonstrates coverage of the non-self regions near the self. This parameter should be nearly 1. The default for this value is 0.9999. To follow along with this guide, leave these parameters as the default values. Click on the 'Create Detectors' button. This will load a message box signifying the algorithm is running, as shown in Figure B.80. When the algorithm completes, a save dialog will appear, as in Figure B.81. Navigate to the desired save location, specify a name for the file, and click save. This detectors file is named '2Ddet1_spheres_M1.mat'. This will load the detector results menu shown in Figure B.82. If the

detectors contain greater than 2 dimensions, the results menu will appear as in Figure B.83. The detectors are now ready for implementation in the control scheme.

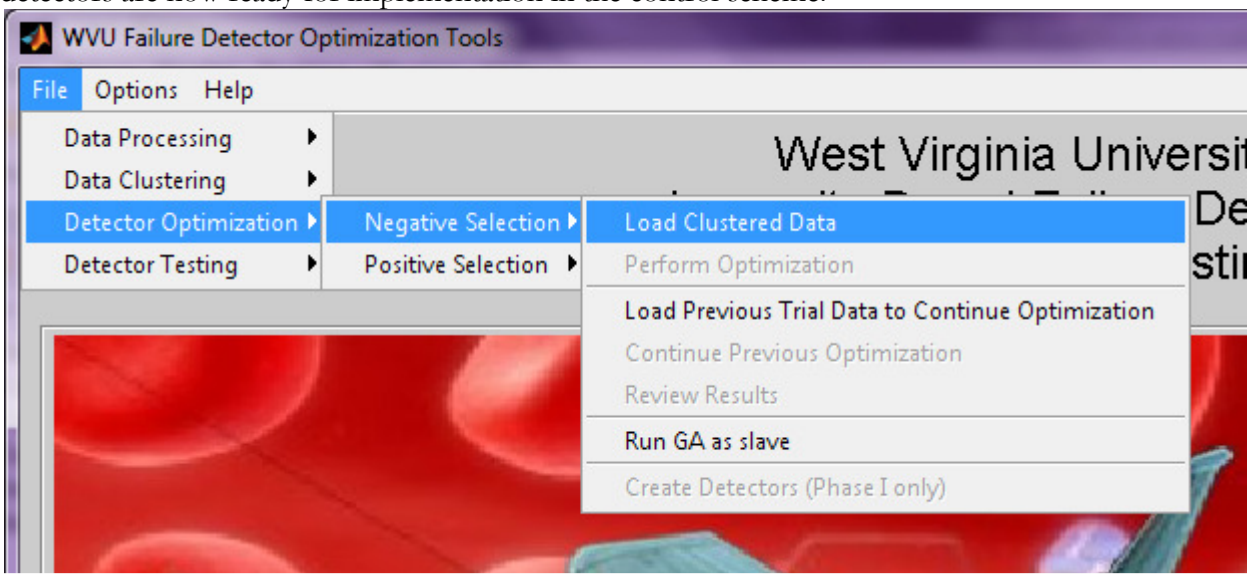

Figure B.76—Opening Load Clusters Menu

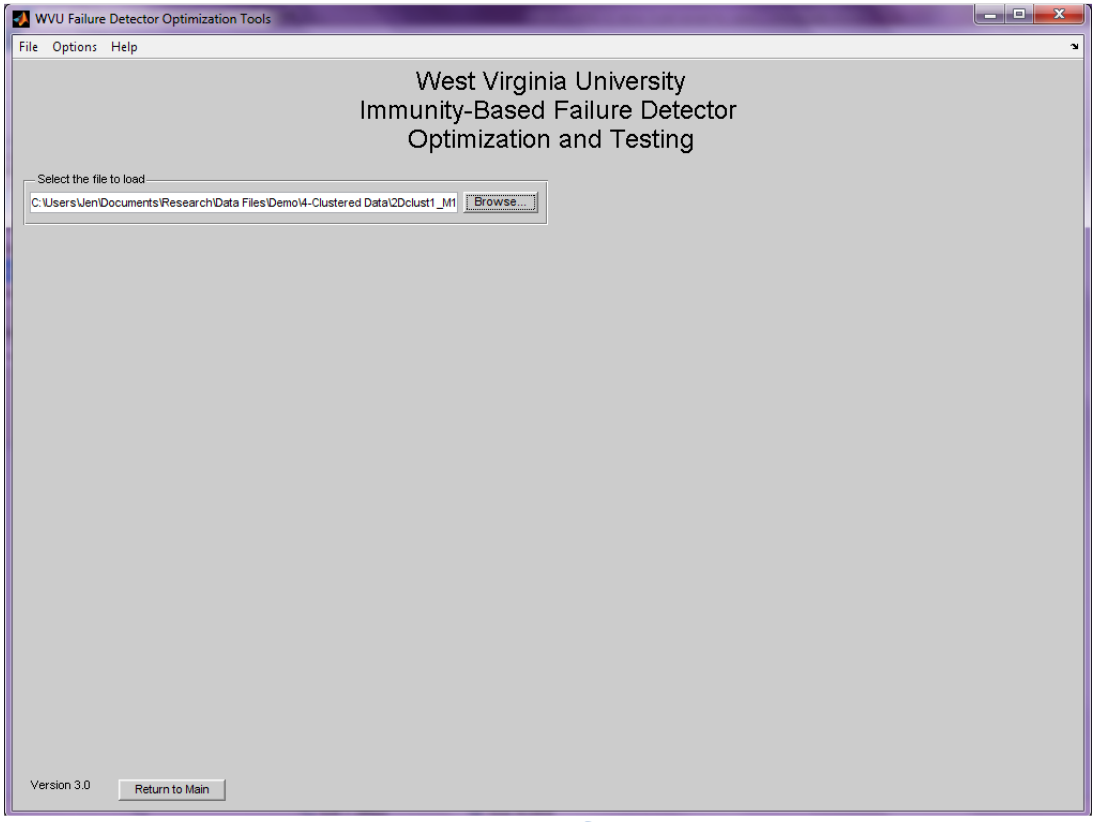

Figure B.77—Load Clusters Menu

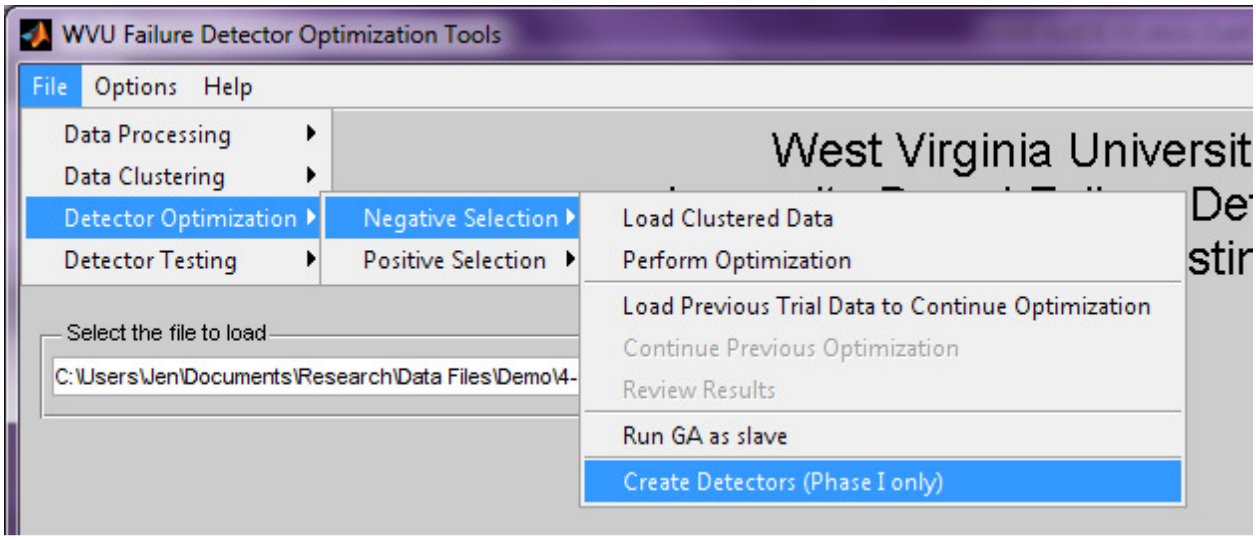

Figure B.78—Opening Create Detectors Menu

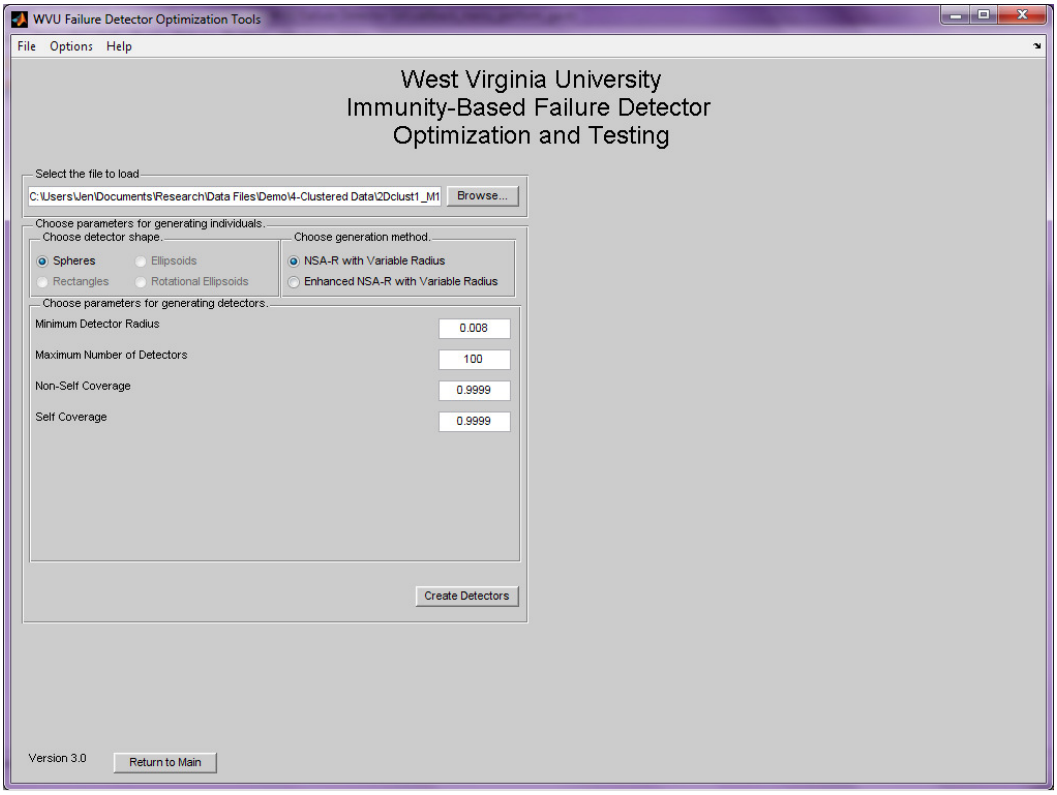

Figure B.79—Create Sphere Detectors Menu For Using NSA-RV

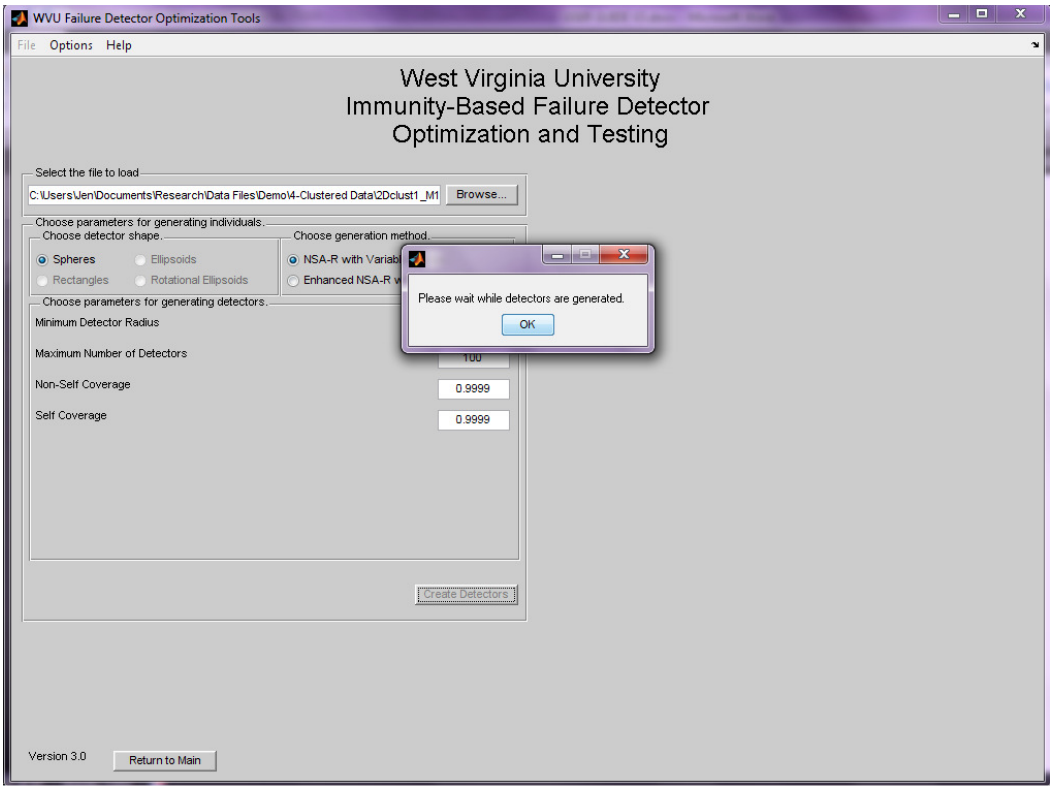

Figure B.80—Detector Creation in Progress

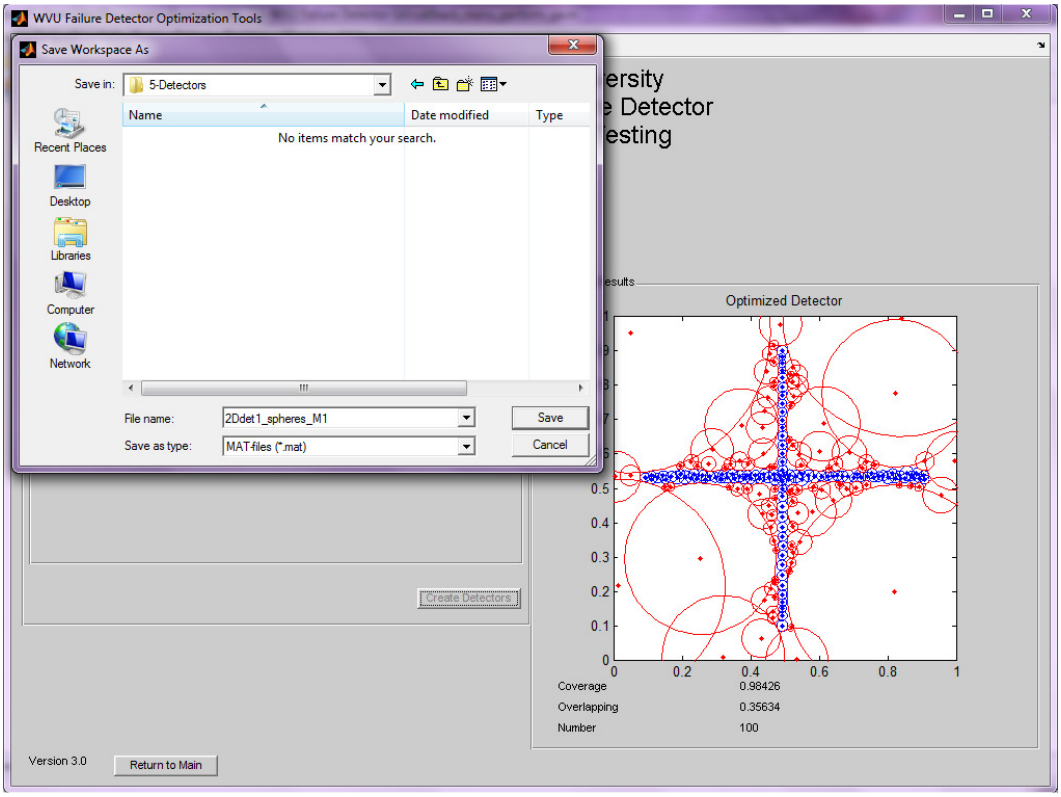

Figure B.81—Detector Creation Save Dialog

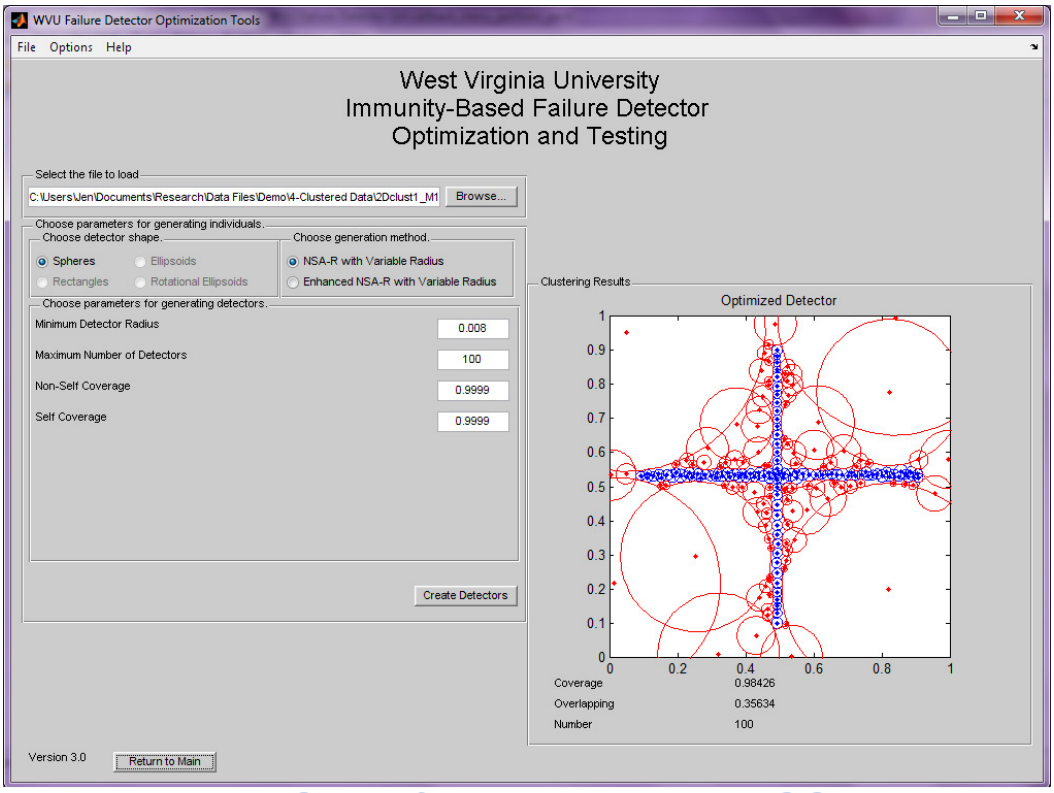

Figure B.82—Detector Creation Results Menu For 2-D Detectors

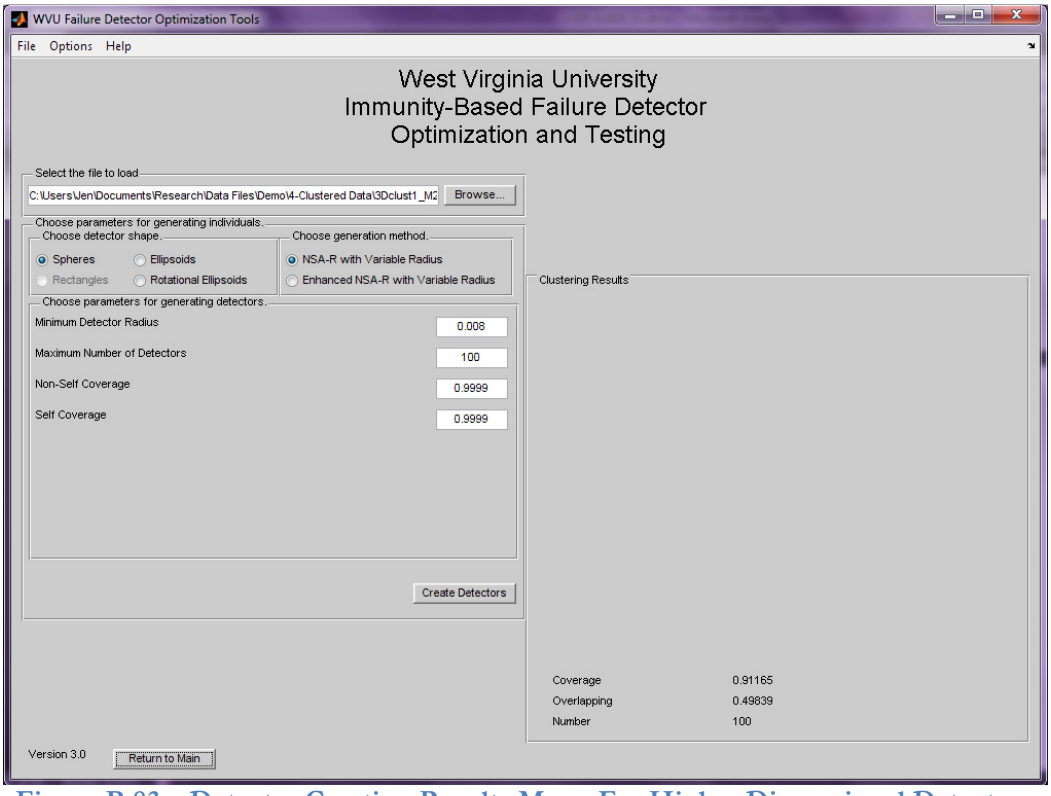

Figure B.83—Detector Creation Results Menu For Higher Dimensional Detectors

4.1.2 Creating Hyper-Rectangle Detectors with NSA-RV

This method of generating hyper-rectangle detectors generates centers at random within the solution space. If the center does not fall within another detector or within the self, the center is assigned a radius equal to the distance from the center to the nearest edge of the self. If this center is greater than a specified minimum threshold, the detector is accepted. This minimum detector radius decreases as the number of iterations of the detector generation algorithm increases. This continues until too many random centers have been rejected for falling within other detectors, or the specified number of detectors has been reached.

To create a set of detectors, begin by clicking on the 'File' menu, selecting 'Detector Optimization', then 'Negative Selection', then 'Load Clustered Data', as in Figure B.84. Click on the browse button and load a data file containing hyper-sphere clusters (See section 4.1.2 for hyper-rectangle clusters), as in Figure B.85. To follow along with this guide, select the file labeled '2Dclust1_Rect.mat' within the '4-Clustered Data' folder in the 'Demo' directory. Click on the 'File' menu, select 'Negative Selection', then 'Create Detectors (Phase I only)', as seen in Figure B.86. This will load the menu seen in Figure B.87. This menu contains four parameters which must be chosen in order to perform detector generation. These are the minimum detector dimension, maximum number of detectors, non-self coverage, and decay parameter. The minimum detector dimension is the smallest semi-side length a center can be assigned and be accepted as a mature detector. The default for this value is 0.002. The maximum number of detectors is the largest desired number of detectors the set is permitted to contain. The default for this value is 100. The non-self coverage is a stopping criteria based on the number of random centers that have fallen within other detectors. This demonstrates coverage of the large non-self regions. This parameter should be nearly 1. The default for this value is 0.9999. The decay parameter defines the rate at which the minimum detector dimension decreases and should increase as the desired number of detectors increases. The default value for this parameter is 140. To follow along with this guide, leave these parameters as the default values. Click on the 'Create Detectors' button. This will load a message box signifying the algorithm is running, as shown in Figure B.88. When the algorithm completes, a save dialog will appear, as in Figure B.89. Navigate to the desired save location, specify a name for the file, and click save. This detectors file is named '2Ddet1_rectangles.mat'. This will load the detector results menu shown in Figure B.90. If the detectors contain greater than 2 dimensions, the results menu will appear as in Figure B.91. The detectors are now ready for implementation in the control scheme.

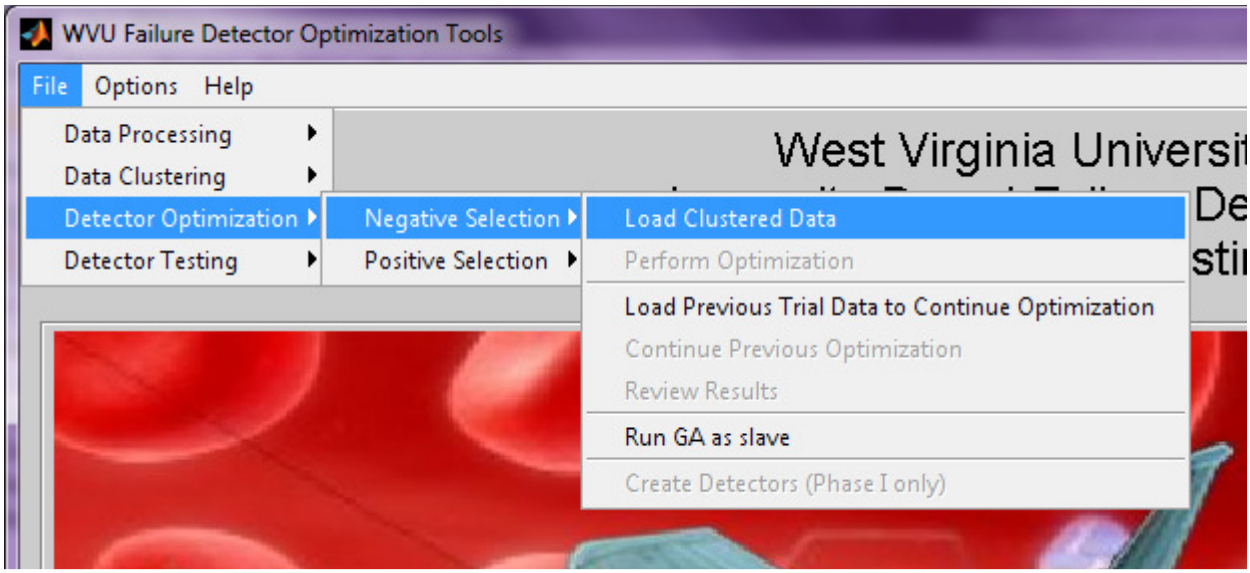

Figure B.84—Opening Load Clusters Menu

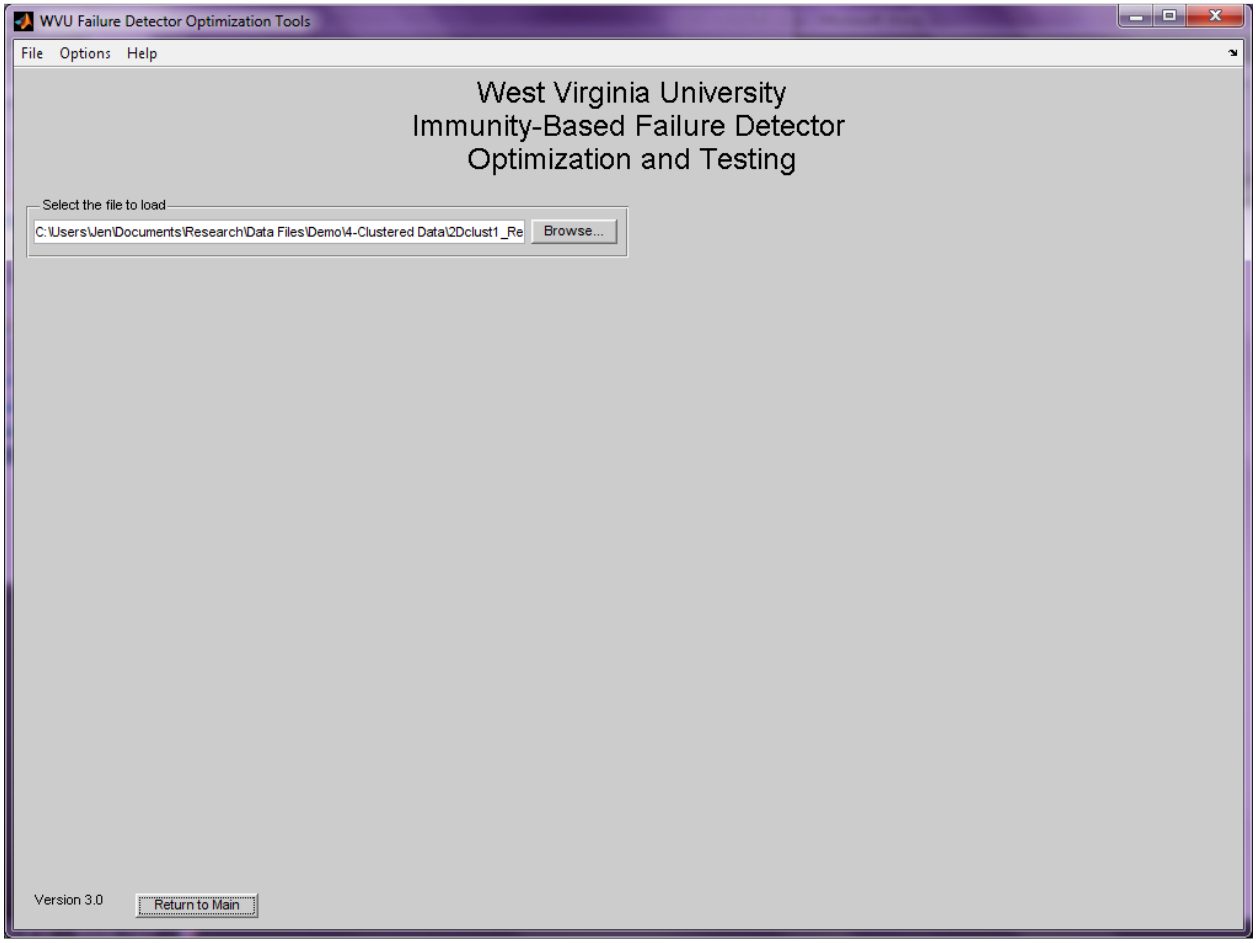

Figure B.85—Load Clusters Menu

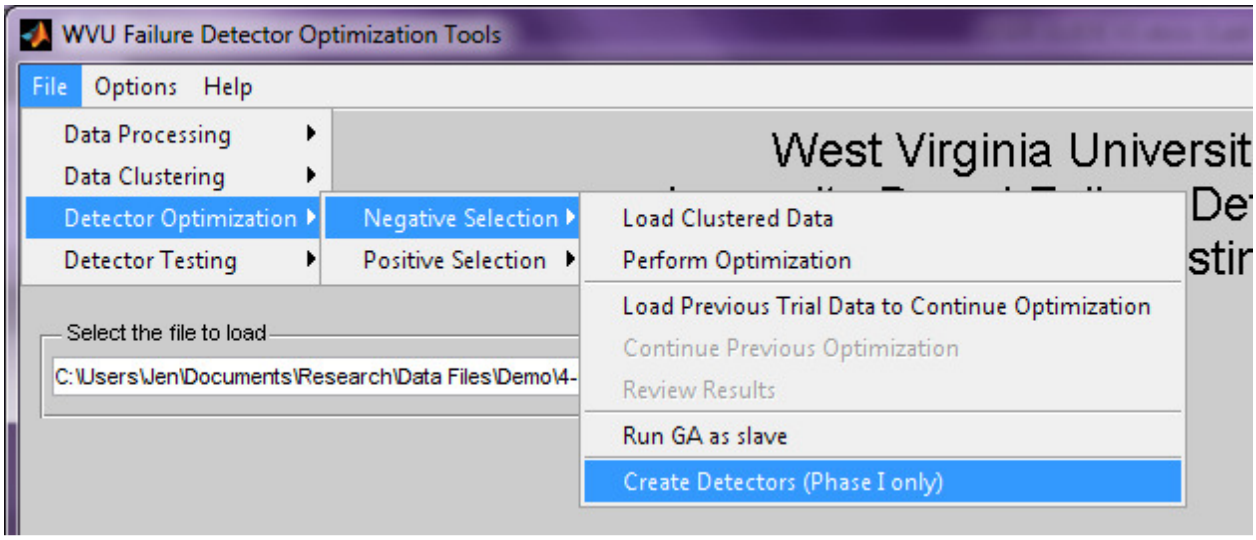

Figure B.86—Opening Create Detectors Menu

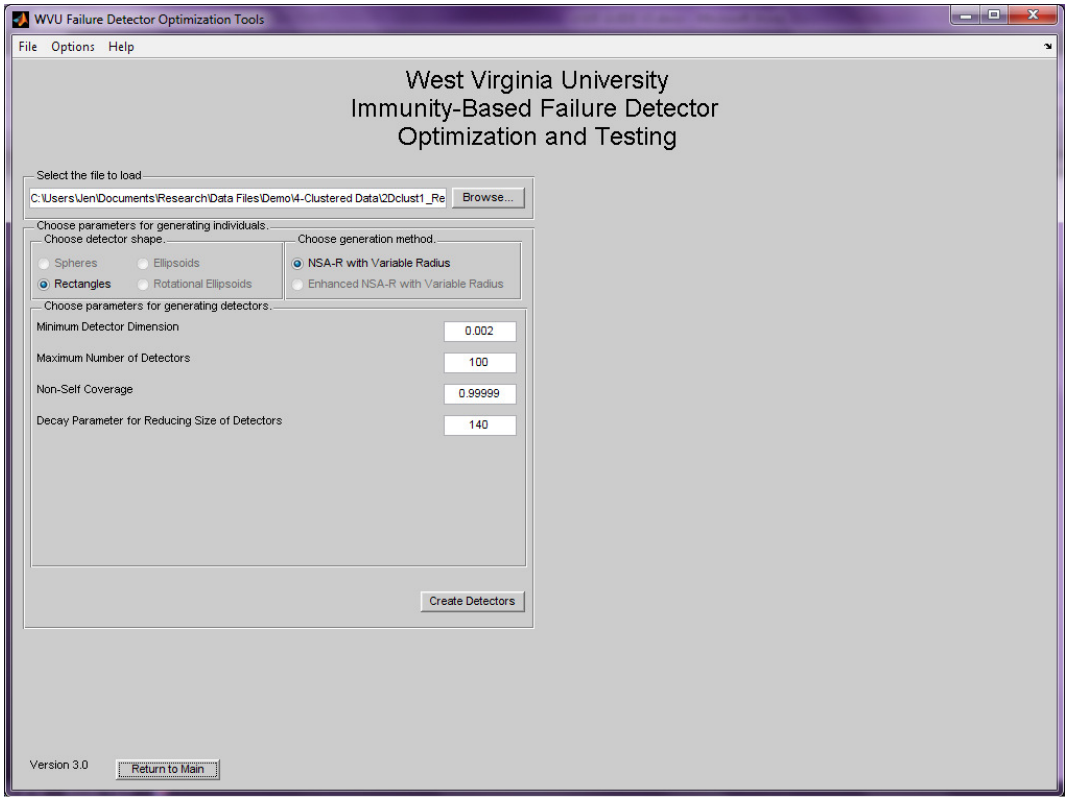

Figure B.87—Create Rectangle Detectors Menu

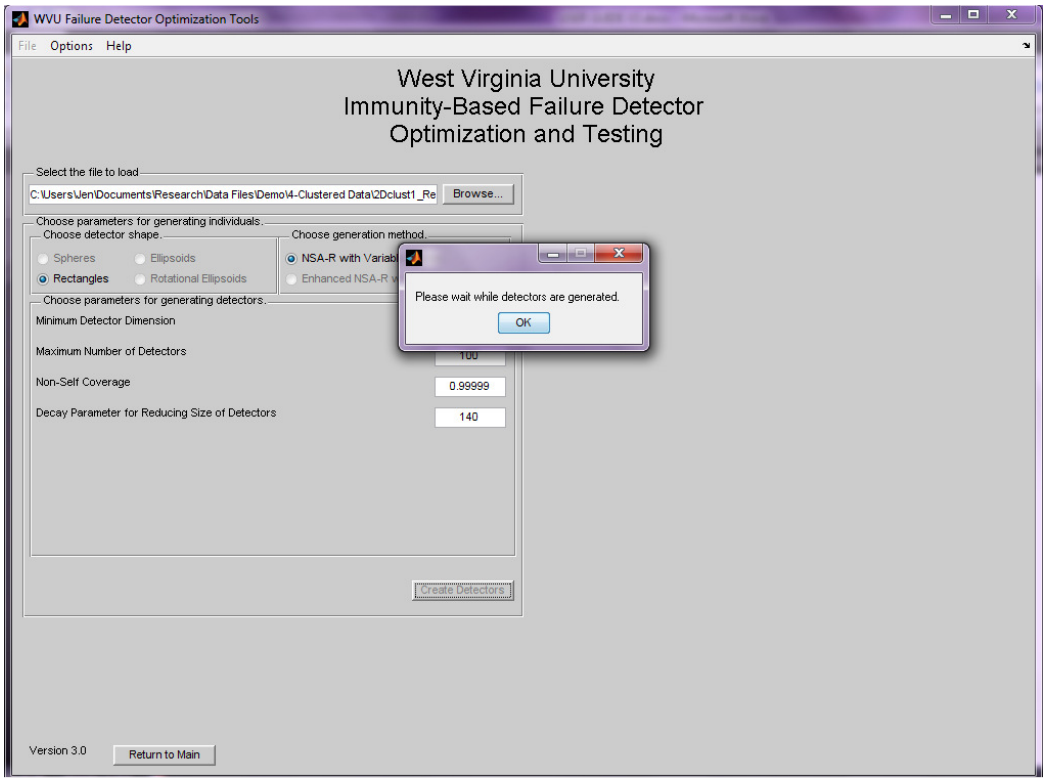

Figure B.88—Detector Creation in Progress

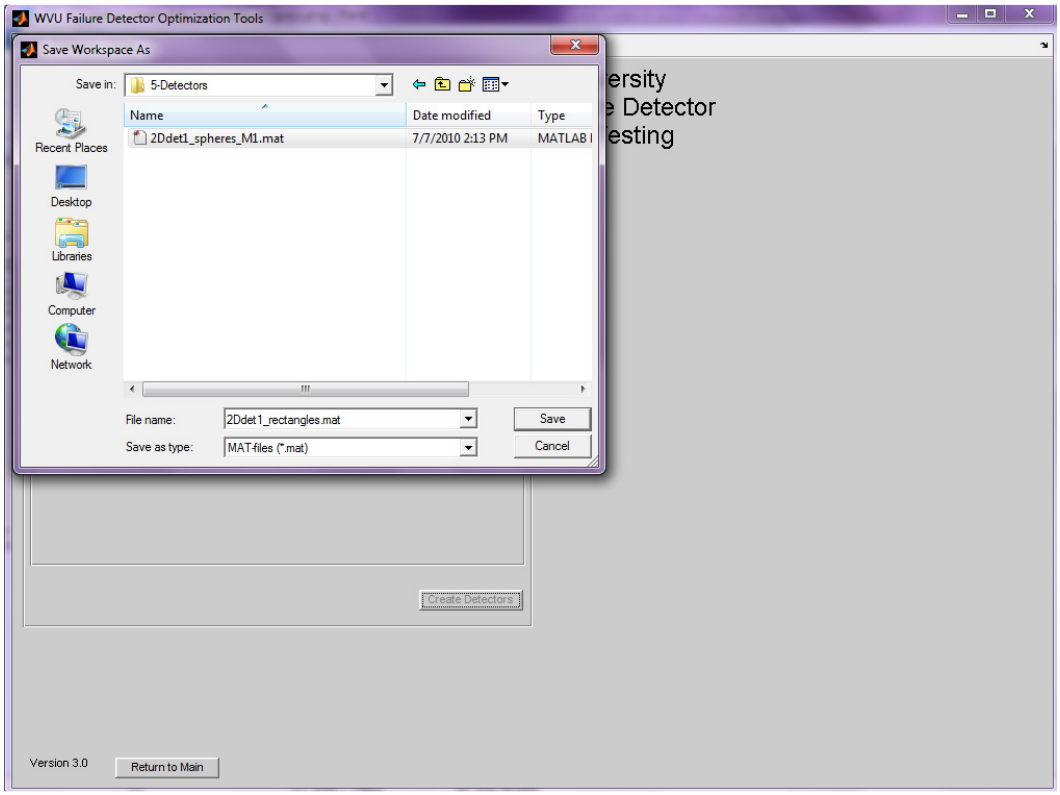

Figure B.89—Detector Creation Save Dialog

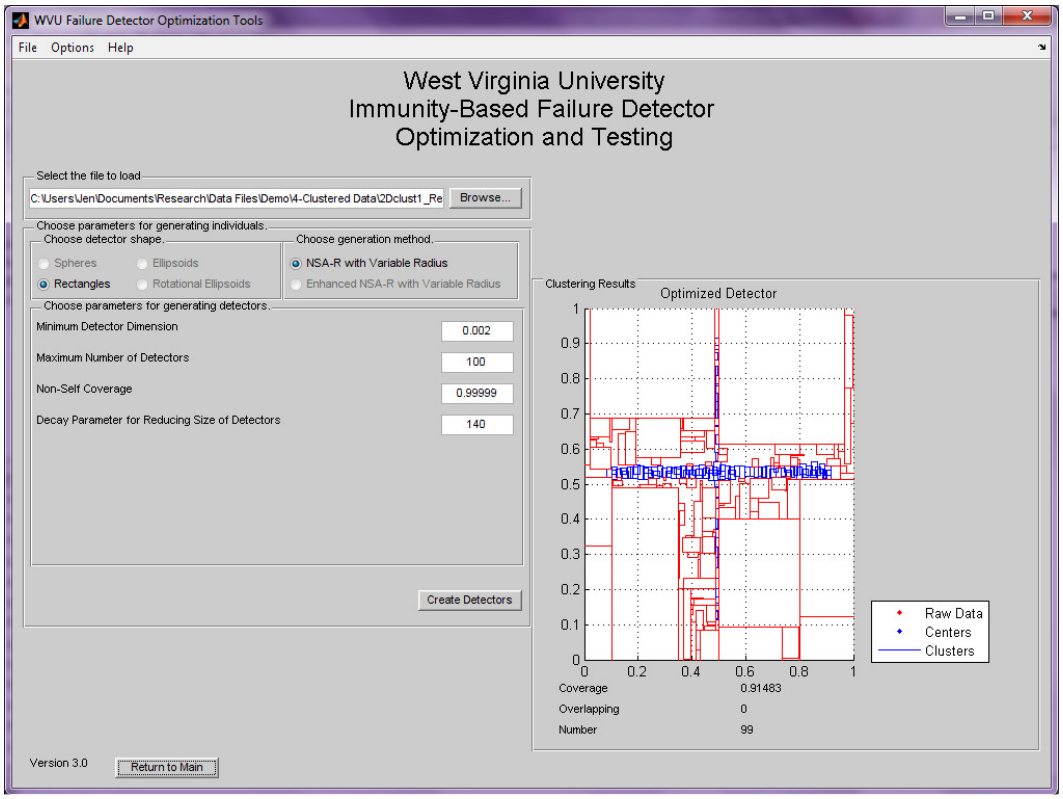

Figure B.90—Detector Creation Results Menu For 2-D Detectors

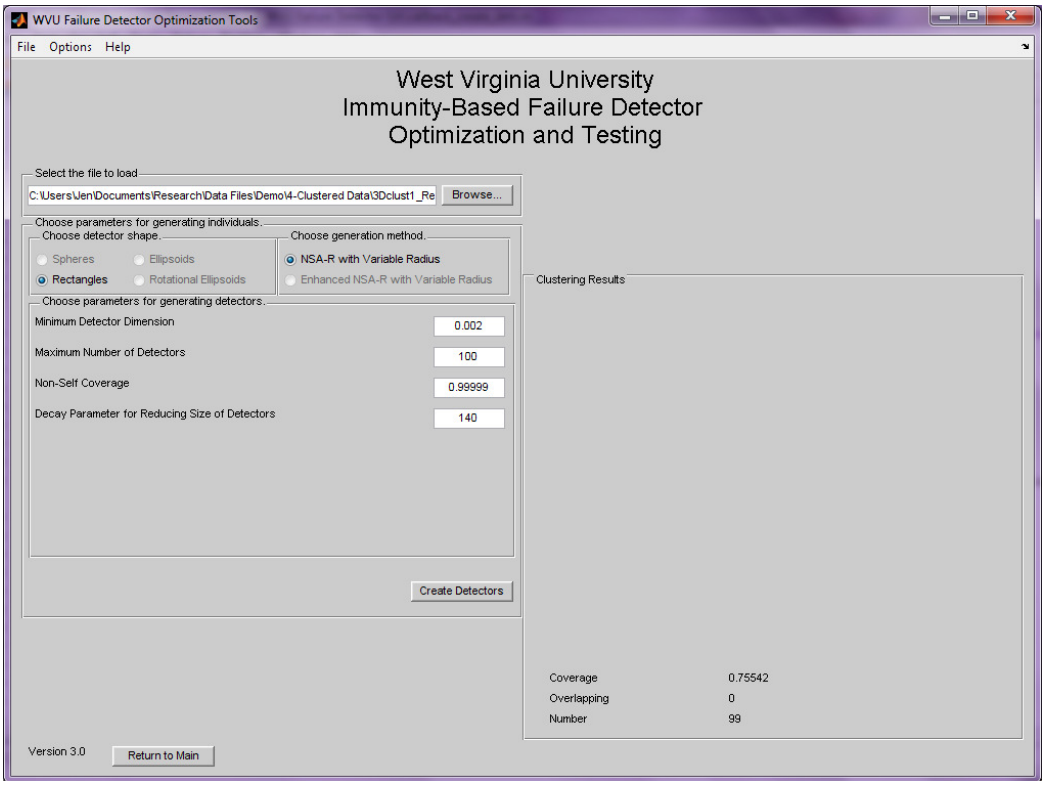

Figure B.91—Detector Creation Results Menu For Higher Dimensional Detectors

4.1.3 Creating Hyper-Sphere Detectors with Enhanced NSA-RV

This method of generating hyper-sphere detectors generates centers at random within the solution space. If the center does not fall within another detector or within the self, the center is assigned a radius equal to the distance from the center to the nearest edge of the self. If this center is greater than a specified minimum threshold, the detector is accepted. If the center is not acceptable, it is moved until it is acceptable or has been moved too many times and is rejected. If a detector is accepted as mature, a number of clones of the detector are created based on the overlapping present in the detector. This continues until the specified number of detectors has been reached.

To create a set of detectors, begin by clicking on the 'File' menu, selecting 'Detector Optimization', then 'Negative Selection', then 'Load Clustered Data', as in Figure B.92. Click on the browse button and load a data file containing hyper-sphere clusters (See section 4.1.2 for hyper-rectangle clusters), as in Figure B.93. To follow along with this guide, select the file labeled '2Dclust1_M1.mat' within the '4-Clustered Data' folder in the 'Demo' directory. Click on the 'File' menu, select 'Negative Selection', then 'Create Detectors (Phase I only)', as seen in Figure B.94. This will load the menu seen in Figure B.78. To load the Enhanced NSA-RV menu, click on the radio button labeled 'Enhanced NSA-RV with Variable Radius'. This loads the menu seen in Figure B.95 The minimum detector radius is the smallest radius a center can be assigned and be accepted as a mature detector. The default for this value is 0.008. The initial number of detectors is the number of intial random centers to generate. The default value for this parameter is 20. The maximum number of detectors is the largest desired number of detectors the set is permitted to contain. The default for this value is 100. The maximum number of iterations is the most iterations the algorithm can take to attempt to create the required number of detectors. The default value for this parameter is 100. The number of random detectors at each iteration is the number of new random centers to attempt in each iteration. The default value for this parameter is 1500. The number of detectors to move each iteration is the number of rejected centers that will be moved in an attempt to make them acceptable. The default value for this parameter is 10. The initial adaptation rate is the initial distance to attempt to move rejected centers. The default value for this parameter is 0.55. The decay parameter determines how this distance will decrease with increased iterations. The default value for this parameter is 15. The threshold for permitted overlapping is the allowable overlapping percentage for a detector to be acceptable. The default value for this parameter is 1.0, or complete overlapping. The number of point to consider for cloning is the number of nearest points used to determine where clones are located. The default value for this parameter is 2. The number of points to consider for moving is the number of nearest points used to determine which direction to move. The default value for this parameter is 2. The initial distance to located new clones is the distance from the detector its clones will be generated. The default value for this parameter is 1. These clones are moved if necessary to make them acceptable. The cloning decay parameter determines the decreased distance from the original detector clones will be located, as iterations increase. The default value for this parameter is 10. To follow along with this guide, leave these parameters as the default values. Click on the 'Create Detectors' button. This will load a message box signifying the algorithm is running, as shown in Figure B.96. When the algorithm completes, a save dialog will appear, as in Figure B.97. Navigate to the desired save location, specify a name for the file, and click save. This detectors file is named '2Ddet1_spheres_M1.mat'. This will load the detector results menu shown in Figure B.98. If the detectors contain greater than 2 dimensions, the results menu will appear as in Figure B.99. The detectors are now ready for implementation in the control scheme.

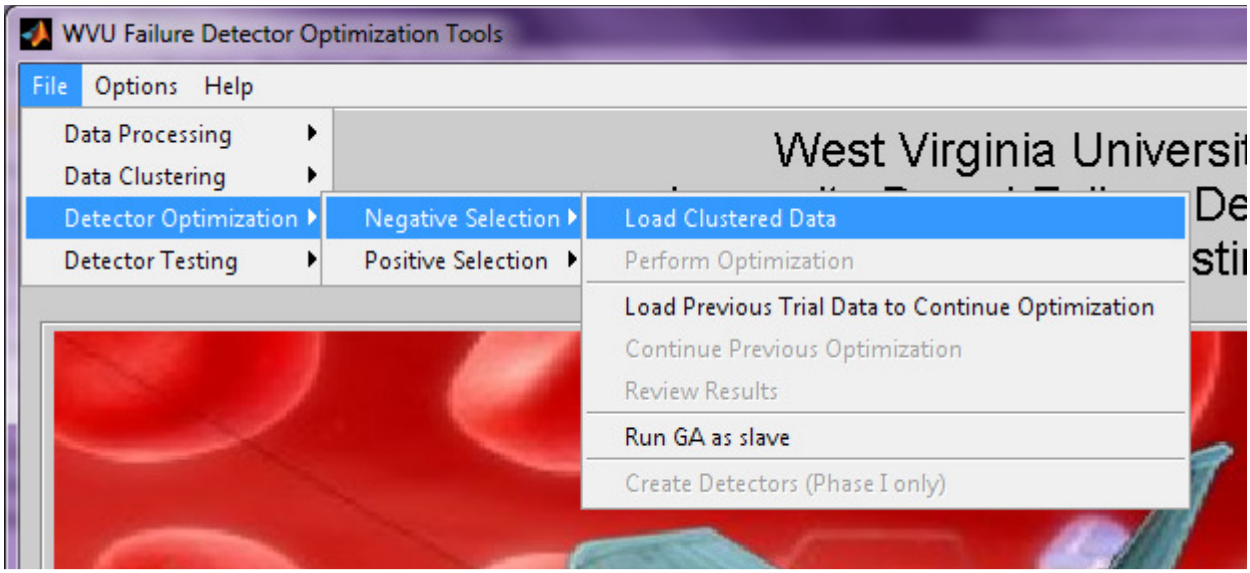

Figure B.92—Opening Load Clusters Menu

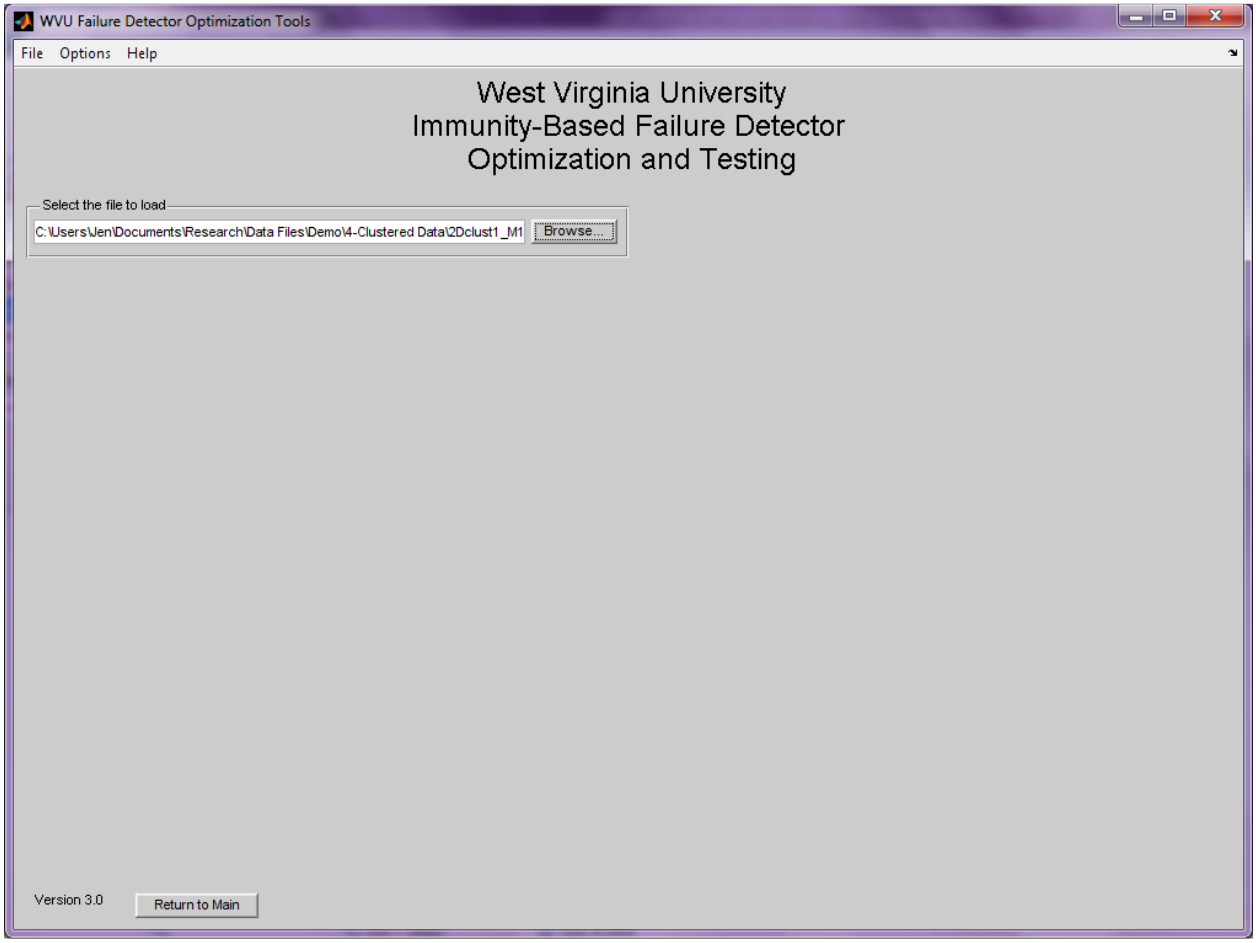

Figure B.93—Load Clusters Menu

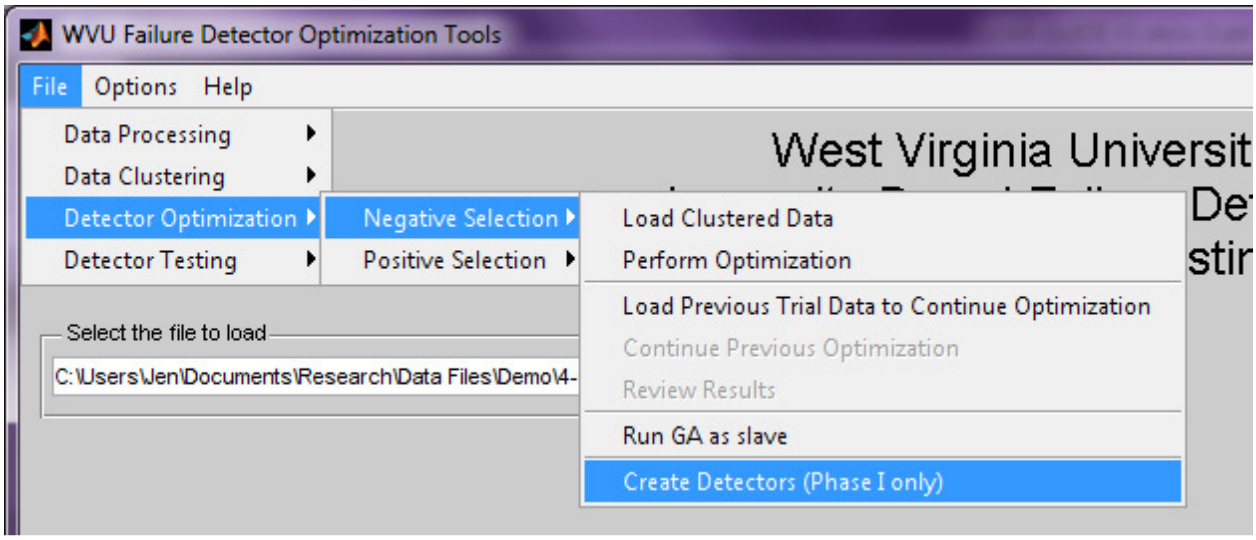

Figure B.94—Opening Create Detectors Menu

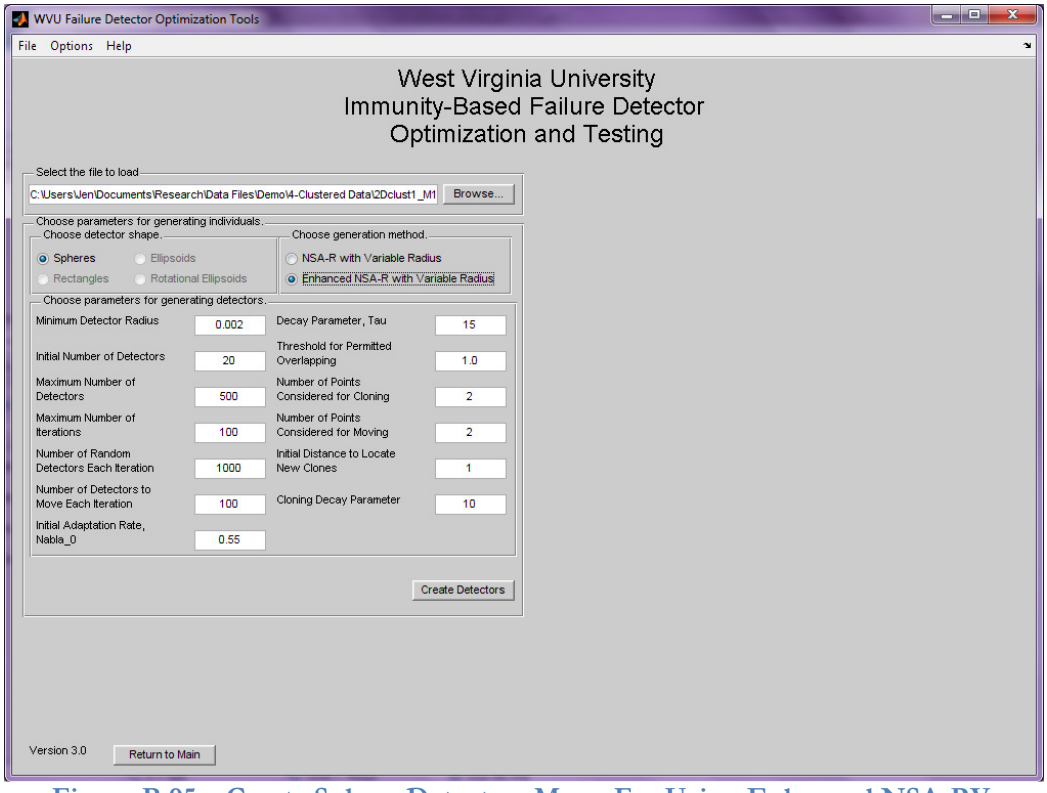

Figure B.95—Create Sphere Detectors Menu For Using Enhanced NSA-RV

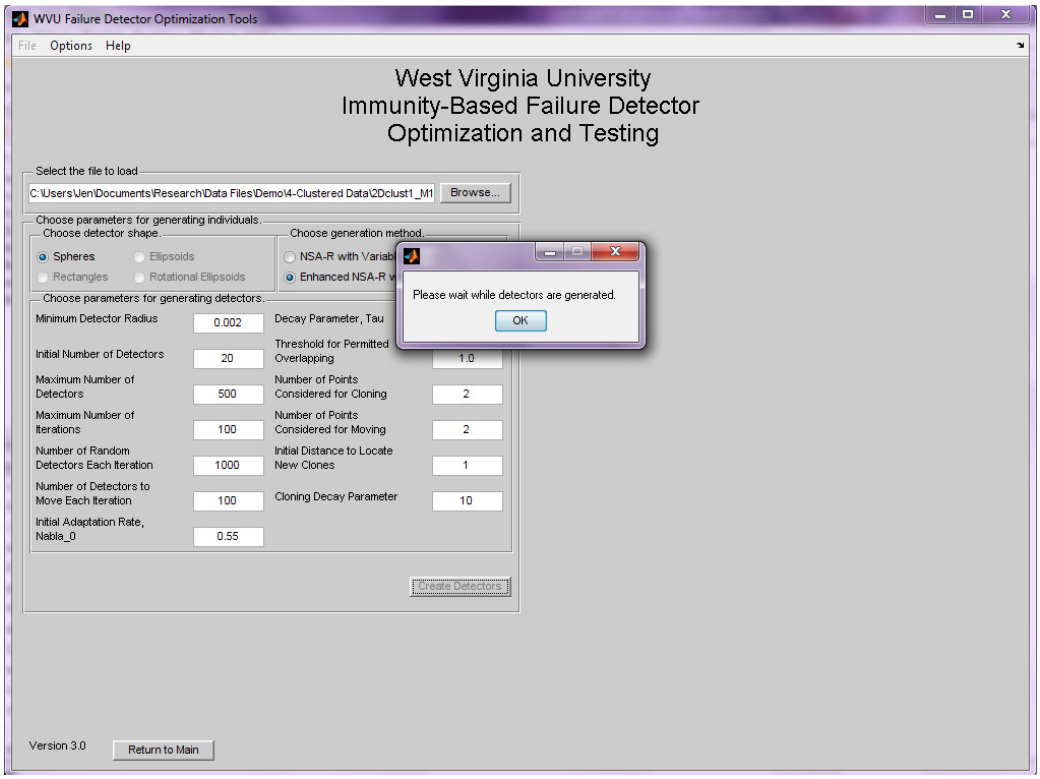

Figure B.96—Detector Creation in Progress

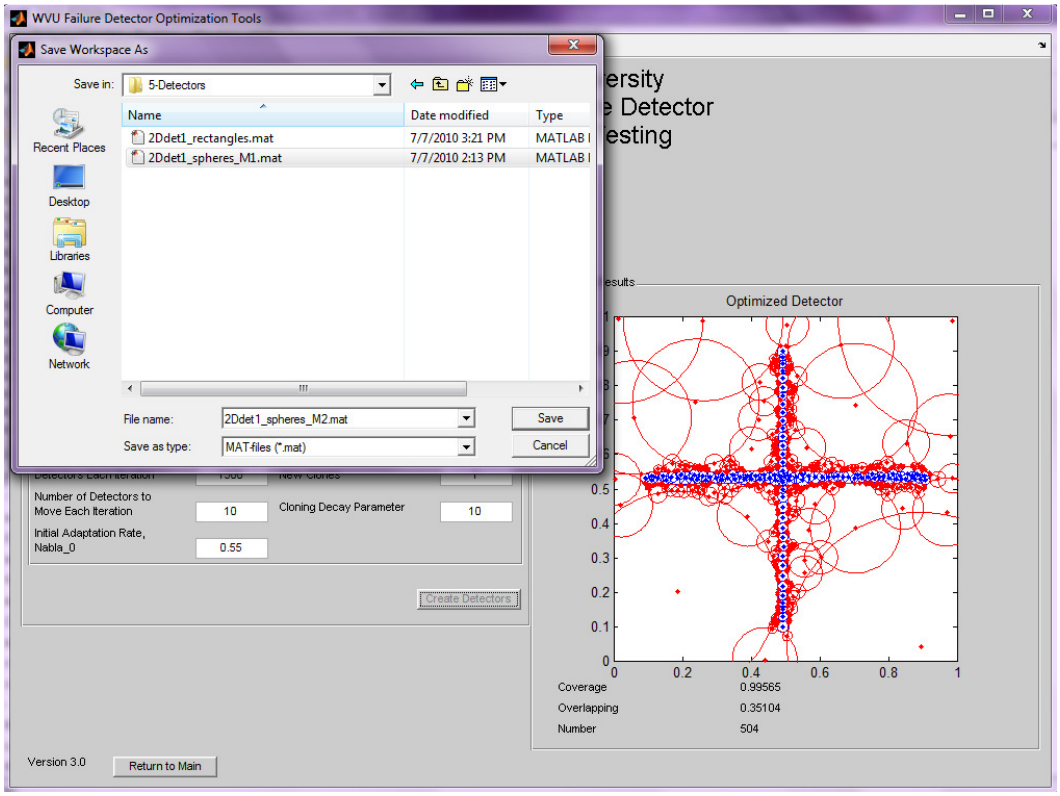

Figure B.97—Detector Creation Save Dialog

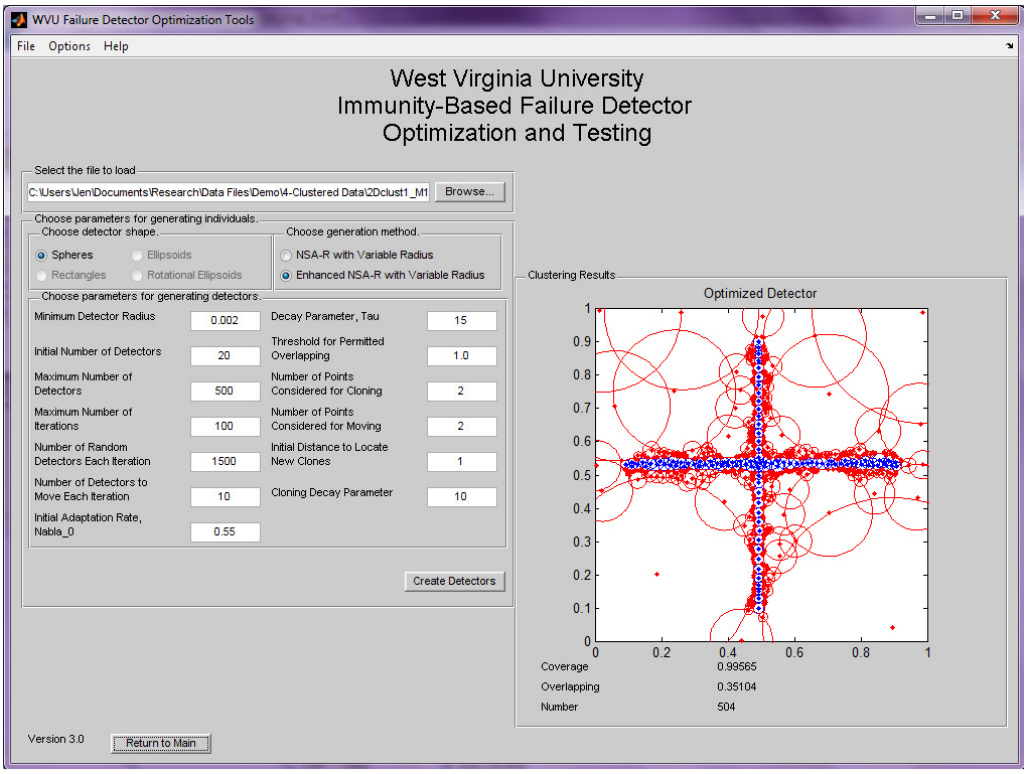

Figure B.98—Detector Creation Results Menu For 2-D Detectors

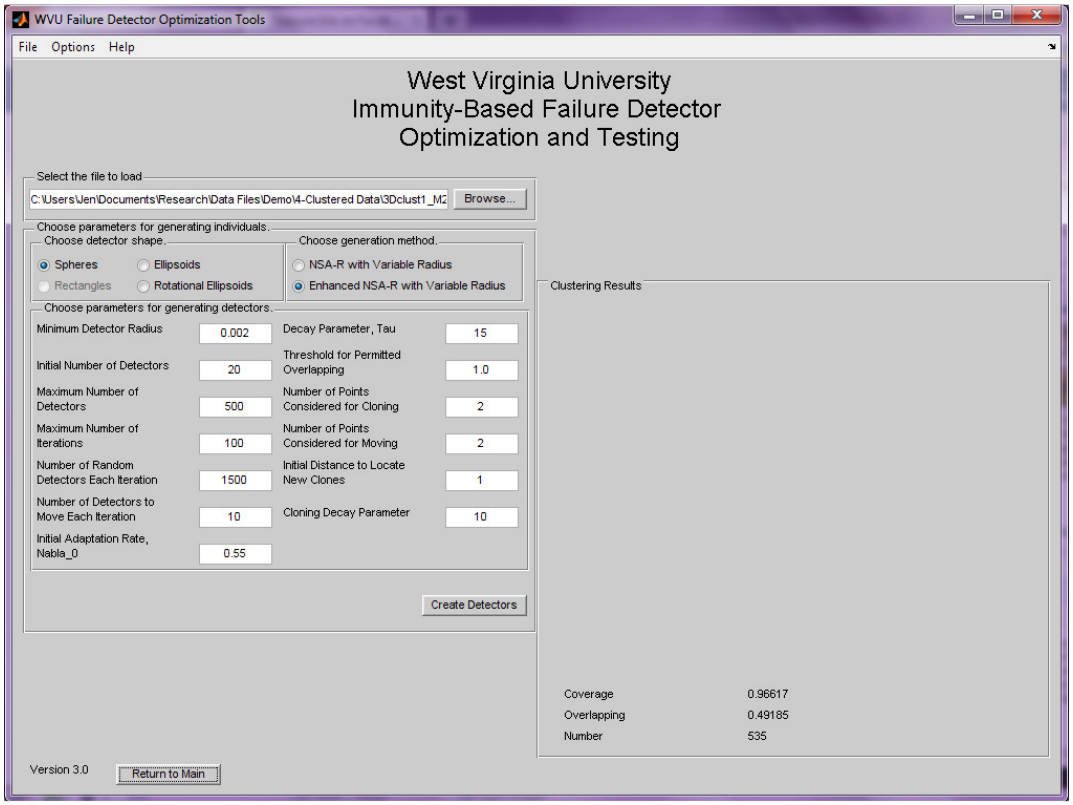

Figure B.99—Detector Creation Results Menu For Higher Dimensional Detectors

4.2 Detector Optimization

Detector optimization utilizes an evolutionary algorithm to optimize a set of detectors based on the coverage of the solution space, overlapping among detectors, and number of detector in a set. This can be an extremely time-consuming process, depending upon the shape used and the various parameters given. Time expectancy cannot be placed on this step, since the process is highly variable, and processing speed depends highly on the parameters used in the algorithm.

To optimize detectors using hyper-spheres, click on the 'File' menu, then select 'Detector Optimization', then 'Negative Selection' and 'Load Clustered Data', as seen in Figure B.100. This loads the menu seen in Figure B.101. Click on the 'Browse' and select the desired clusters file. The file chosen for this walkthrough is '2Dclust1_M1.mat'. The most important choice the user makes in the optimization phase is the choice of detector shape. If hyper-rectangle detectors are desired, a clusters file containing hyper-rectangle clusters must be chosen; otherwise, hyper-sphere clusters are needed.

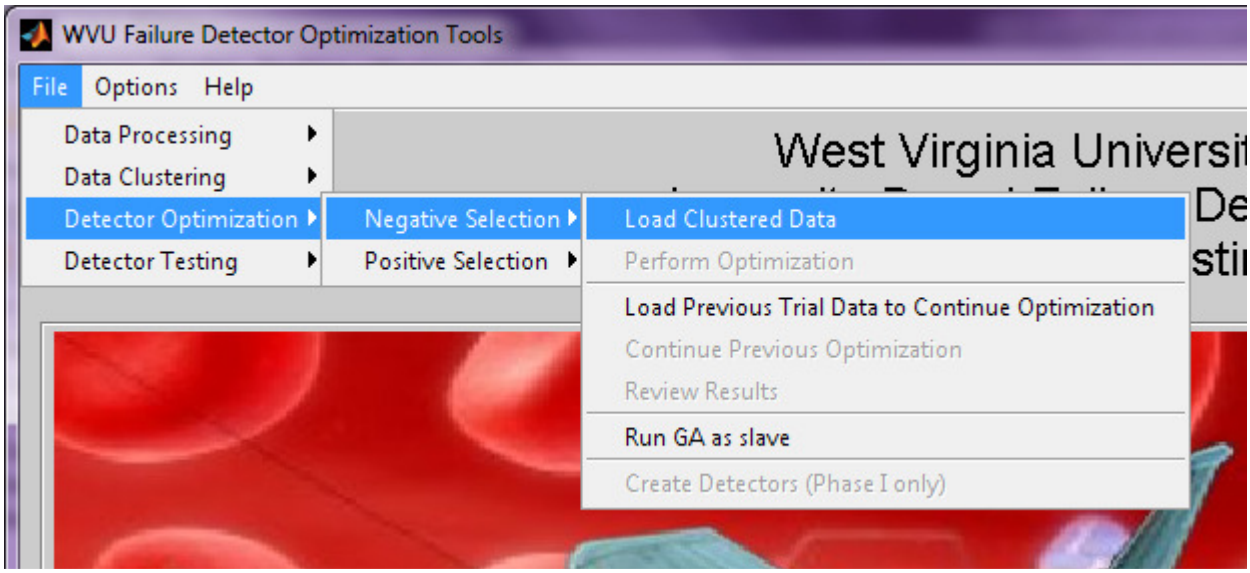

Figure B.100—Opening Load Clusters Menu

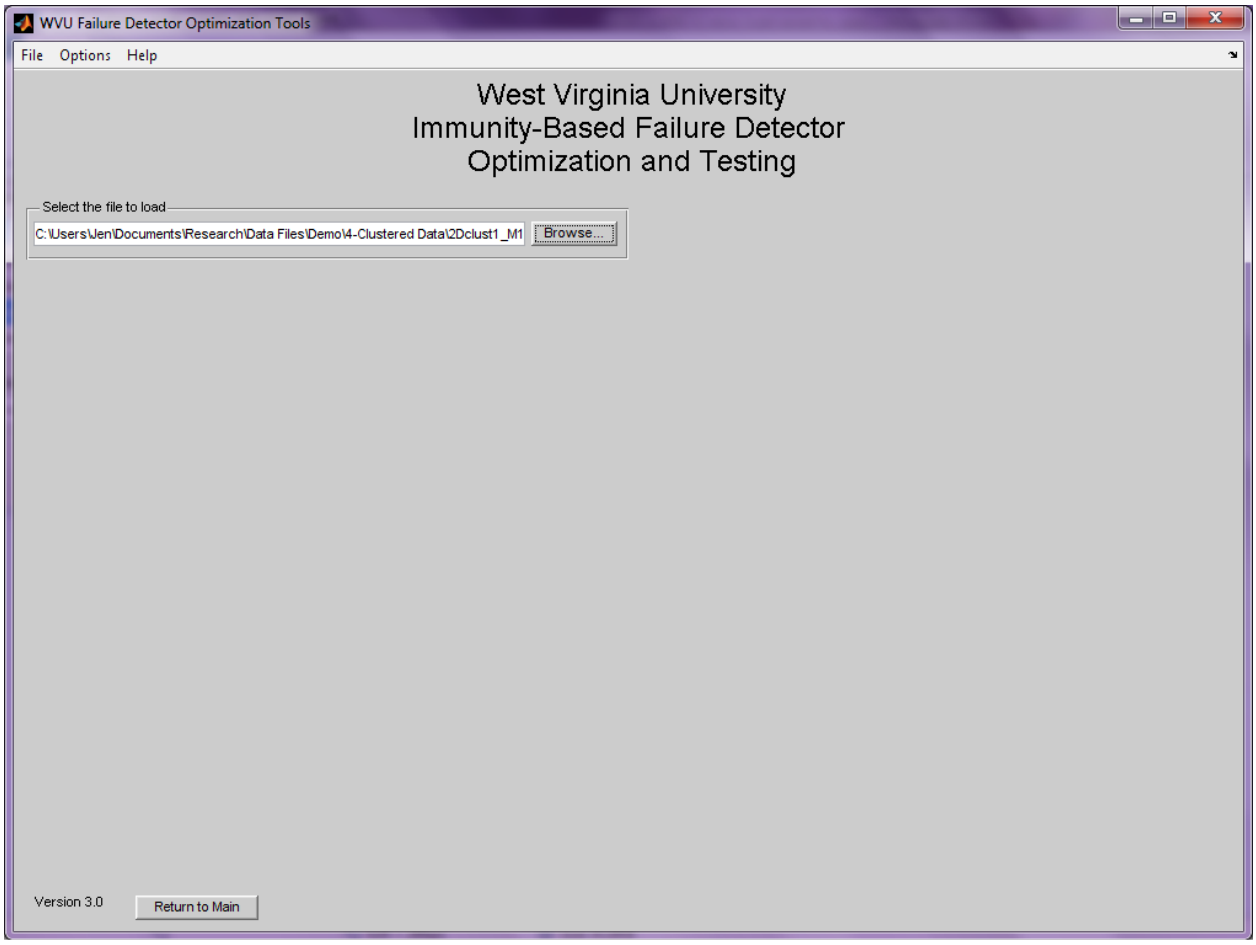

Figure B.101—Load Clusters Menu

With the properly clustered data file chosen, click on the 'File' menu, select 'Detector Optimization', then 'Negative Selection', then 'Perform Optimization', as in Figure B.102. This loads the menu seen in Figure B.103. The shape choices available will depend upon the shape of the clusters in the files chosen. If the clusters are composed of hyper-spheres, the radio button labeled 'Rectangles' will be disabled. If the clusters are composed of hyper-rectangles, only the radio button labeled 'Rectangles' will be enabled. Choose the shape desired. For this walkthrough, the shape chosen will be hyper-spheres. As options vary per shape, these will be discussed as they occur. For simplicity, all parameters will be left as the default value for this walkthrough.

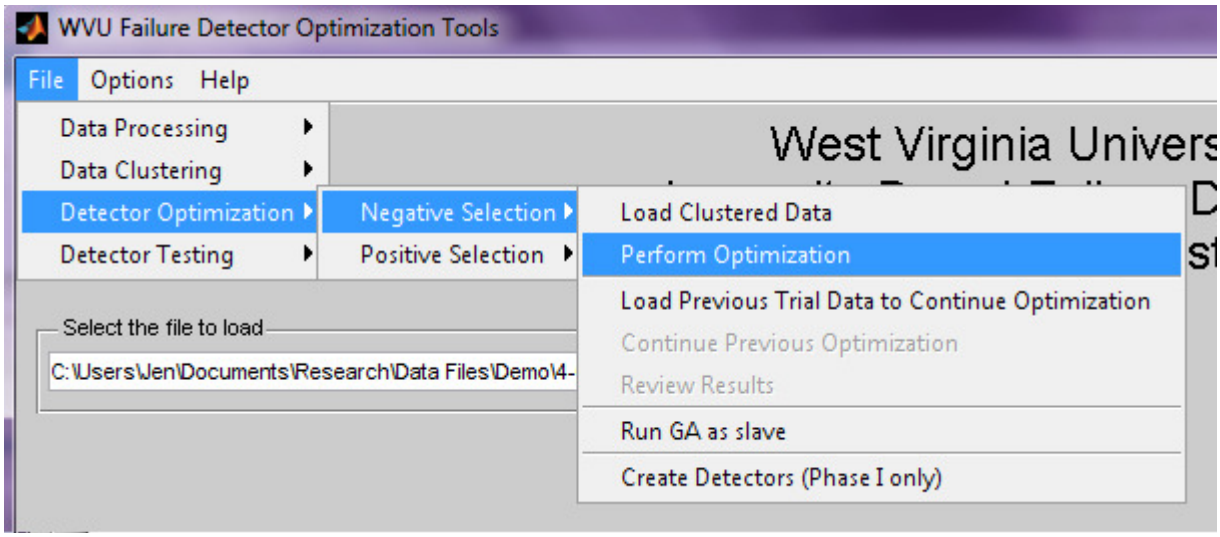

Figure B.102—Opening Detector Optimization Menu

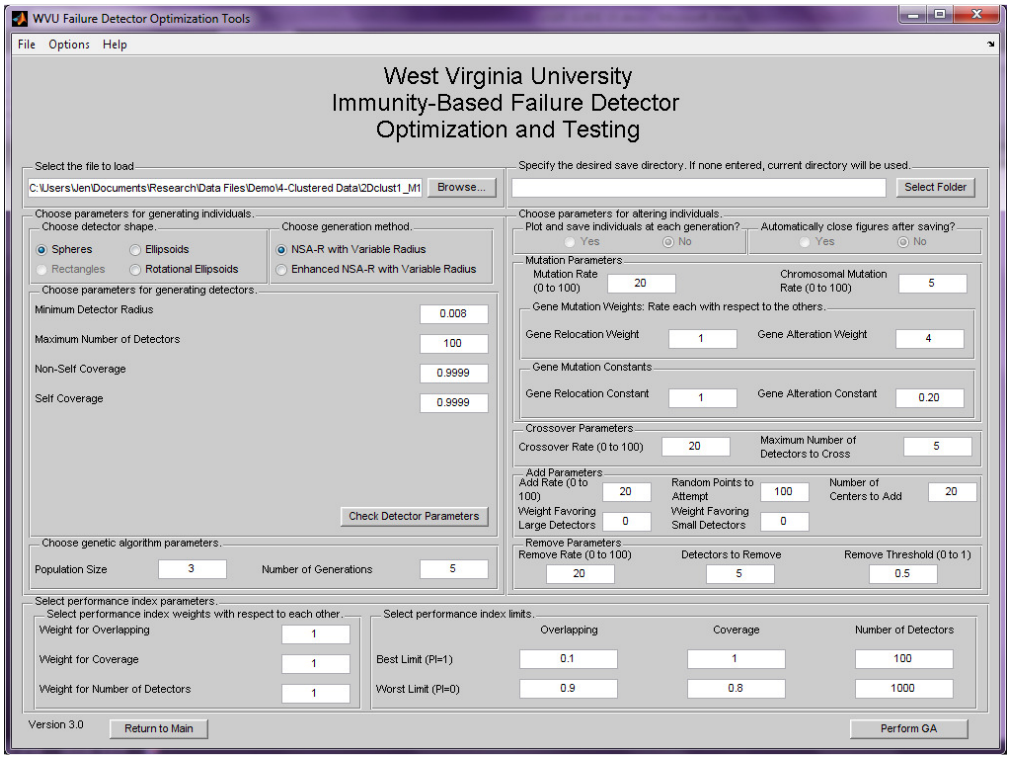

Figure B.103—Detector Optimization Menu

Based on the shape chosen, the detector generation options will vary. The section 4.1 discusses these options in detail, so they will not be covered here. Since the clusters are hyper-spheres, the NSA-RV detector generation method will be utilized.

The genetic algorithm parameters determine the number of individuals in the population, and the number of generations to perform. A higher number of individuals is desired for higher individual variability and better exploration of the solution space. The number of generations is the stopping criteria for the algorithm.

The plot and save parameters are only valid for 2-D detector sets. These options will allow the user the choice to plot each individual at each generation and whether and where to automatically save and close these figures.

The mutation parameters vary based upon the detector shape chosen. For hyper-spheres and hyper-rectangles, 6 parameters are needed. The mutation rate refers to the percentage of individuals that will undergo mutation in each generation. The chromosomal mutation rate refers to the percentage of detectors within each chosen individual that will be altered in each instance of the mutation operator. Gene relocation weight and gene alteration weight are weights that work the same as the performance index weights, to determine the likelihood of a particular type of mutation occurring. Gene relocation is the moving of the center of a detector. Gene alteration is the changing of a detector's radius. The gene relocation constant is the distance in multiples of the detector radius the center can be moved in one direction at a time. The gene alteration constant is the distance in multiples of the detector radius that the radius can be changed by at one time.

For the case of hyper-ellipsoids and hyper-rotational-ellipsoids, an additional mutation type exists, called gene rotation. This means that 8 parameters are needed. The mutation rate refers to the percentage of individuals that will undergo mutation in each generation. The chromosomal mutation rate refers to the percentage of detectors within each chosen individual that will be altered in each instance of the mutation operator. Gene relocation weight and gene alteration weight are weights that work the same as the performance index weights, to determine the likelihood of a particular type of mutation occurring. Gene relocation is the moving of the center of a detector. Gene alteration is the changing of a detector's radius. Gene rotation is the rotation of a detector about a certain axis. The gene relocation constant is the distance in multiples of the detector radius the center can be moved in one direction at a time. The gene alteration constant is the distance in multiples of the detector radius that the radius can be changed by at one time. The gene rotation constant is the maximum number of degrees a detector can be rotated at one time.

The crossover parameters consist of only 2 inputs. These are the crossover rate, or the probability an individual will undergo crossover, and number of detectors to cross, or the maximum number of detectors that can be traded between two individuals.

The detector addition parameters contain 5 parameters for hyper-spheres and hyper-rectangles. The add rate is the probability an individual will undergo addition of detectors in a generation. The number of random centers is the number of random centers that will be generated in an attempt to create detectors in previously uncovered areas. The number of detectors to add is the maximum number of detectors that can be added to an individual in one generation. As many detectors as possible will be added, up to this amount. The favor larger detector and favor smaller detector weights determine this probability that the algorithm will favor adding larger or smaller detectors for an individual. If both of these parameters are given as 0, the algorithm adds detectors randomly, paying no attention to the size of the detector. Using this option significantly increases the speed of calculation of the algorithm. An additional parameter is needed when hyper-ellipsoids or hyper-rotational-ellipsoids are used. This is the accuracy for calculating the radius of the new detectors. The radius assigned to the new detectors will be within this amount of touching the nearest object without overlapping.

The remove parameters contain 3 inputs. These are the remove rate, or the probability that an individual will undergo detector removal within a generation, the number of detectors to remove, and the removal threshold, which is the amount of overlapping a detector may have which is considered too low to remove.

The performance index parameters consist of 9 values. For the performance index, three weights must be entered to determine the weights of the three grading criteria. These should be chosen with respect to each other. This means that is all three have the same weight, they will be

equally weighted. However, for instance, if coverage has a weight of 2 and number and overlap have a weight of 1, the performance index will be composed 50% from the coverage rating, and 25% each from the number and overlap ratings. For each of these three criteria, grading limits must be set. A best value and worst value must be specified for each criterion.

After appropriately assigning all of these parameters based on the detector shape chosen, click the 'Perform GA' button. This loads a progress bar, as seen in **Figure B**.104. When the algorithm completes, a save dialog will be displayed, as in **Figure B**.105. Navigate to the desired save location, specify the file name for the optimization file and click save. The filename chosen for this optimization was '2Dopt1_spheres.mat'. The results of the trial will be displayed as in **Figure B**.106 below for 2 identifiers and as in **Figure B**.107 for more than 2 identifiers. The detectors are now ready for implementation into the control scheme.

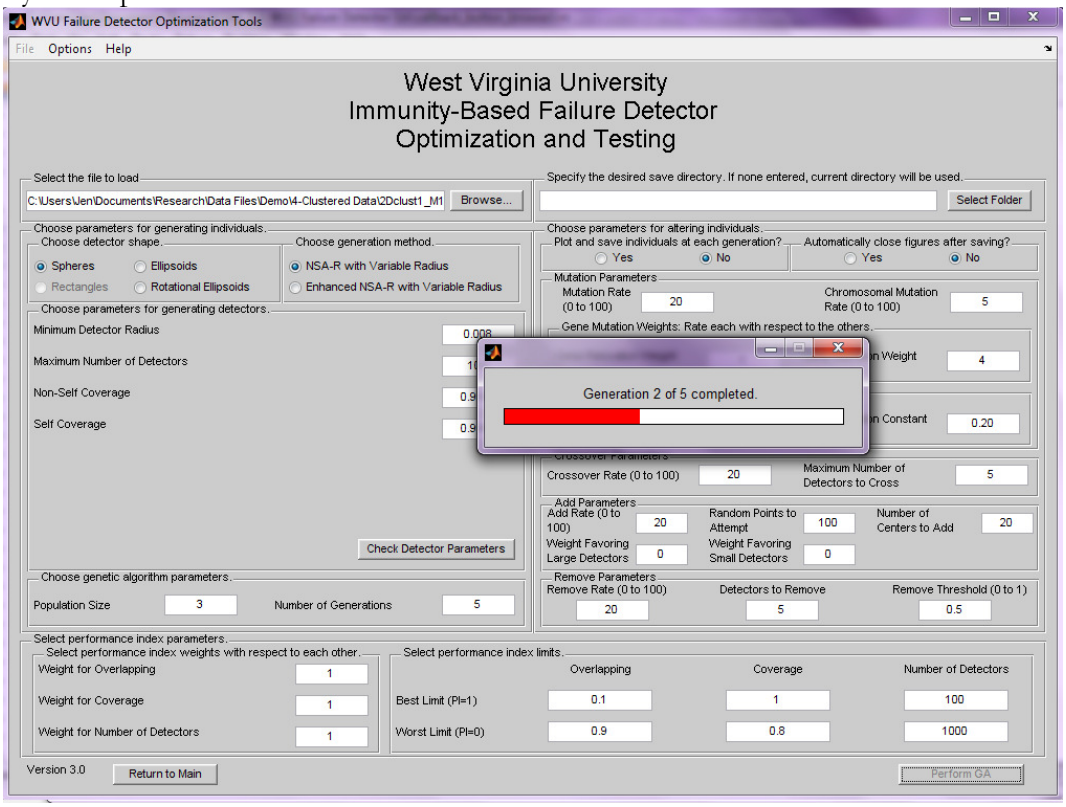

Figure B.104—Genetic Algorithm in Progress

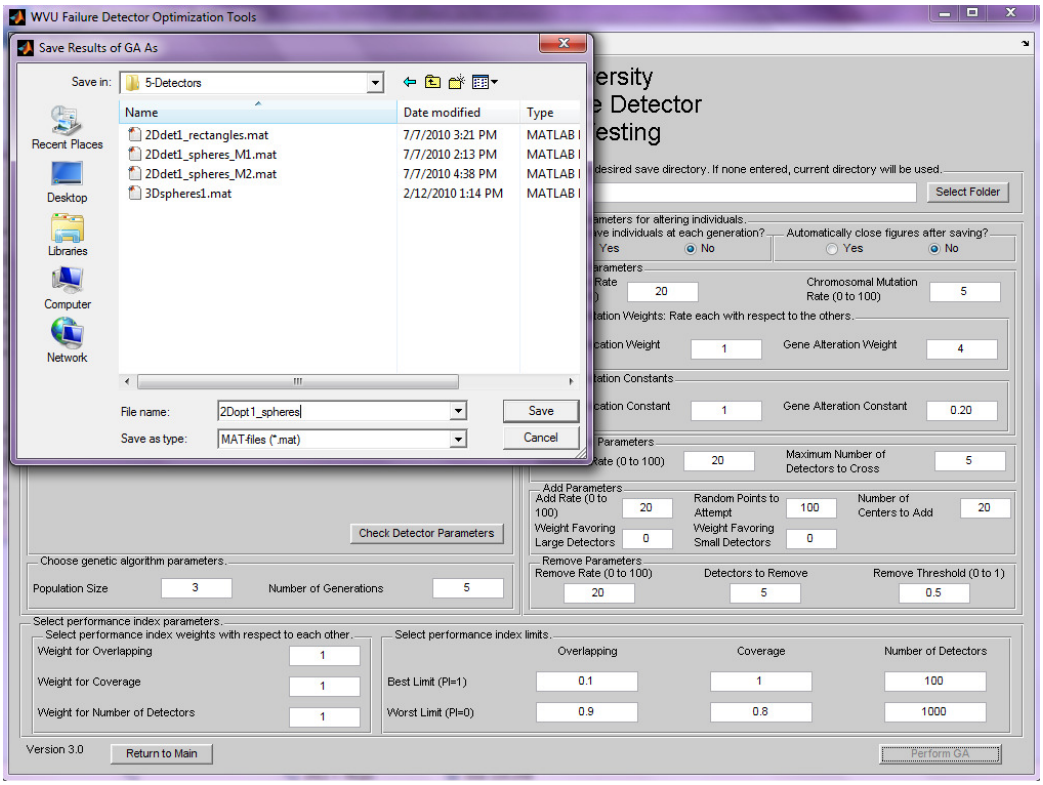

Figure B.105—Genetic Algorithm Save Dialog

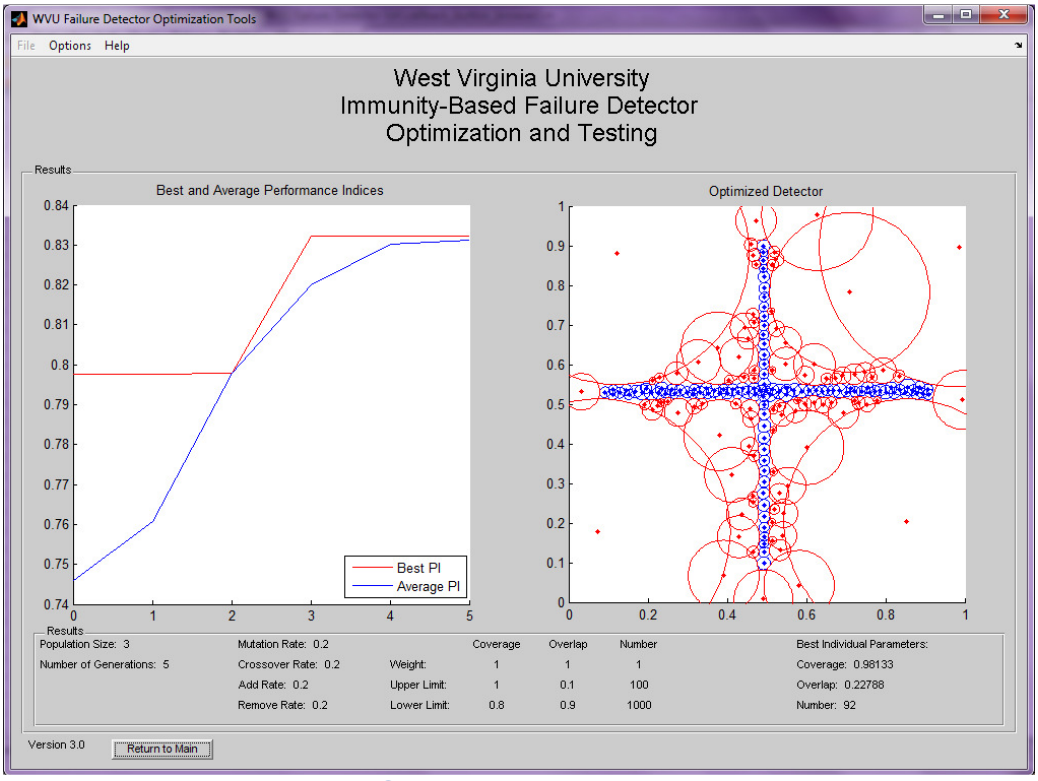

Figure B.106—Optimization Results for 2 Identifiers

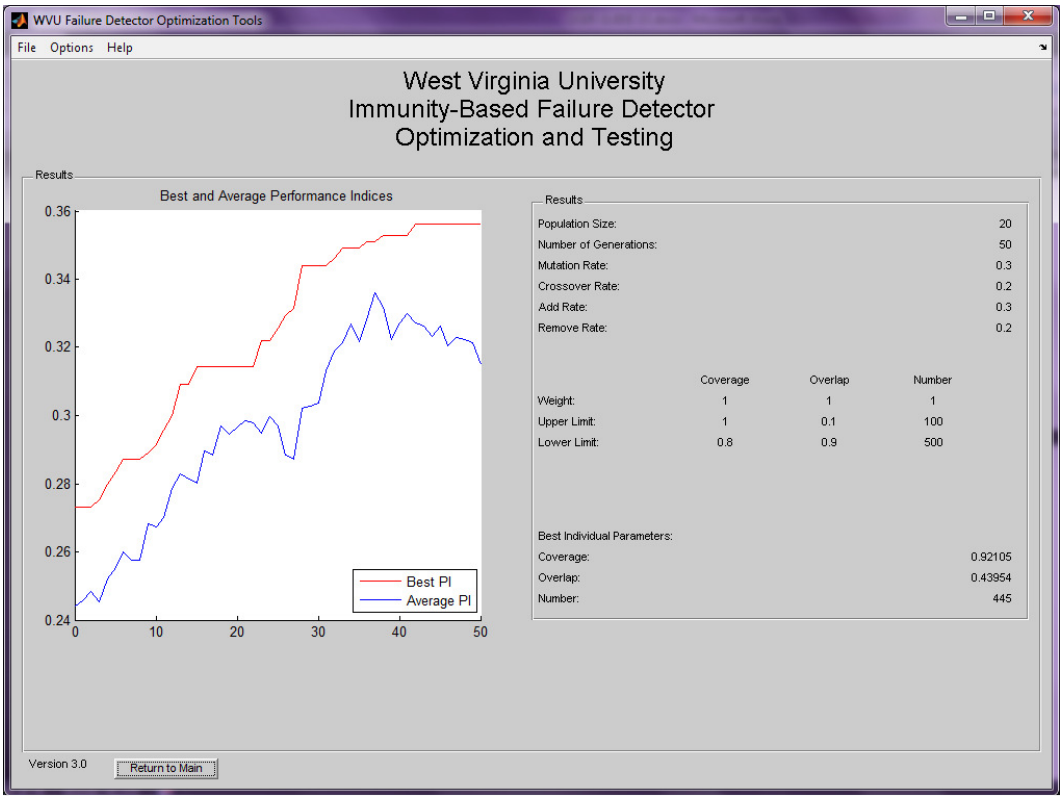

Figure B.107—Optimization Results for Greater Than 2 Identifiers

4.3 Continuing Optimization

At times it may be useful to run optimization for a particular detector set for more generations, to potentially arrive at a better solution. In order to do this, the user must have completed an initial trial using this utility. The results menu will be displayed differently depending upon whether the detectors contain 2 or more identifiers. The 2D trial used in the segment is labeled '2Dopt1_spheres.mat'. The 3-D trial used for this segment is labeled '3Dspheres1.mat'.

To continue optimizing a file, click on the 'File' menu, select 'Detector Optimization', then 'Negative Selection', then 'Load Previous Trial Data to Continue Optimization' as in Figure B.108. This loads the menu seen in Figure B.109. Click on the browse button and navigate to the desired trial data. Then click on the 'File' menu, select 'Detector Optimization', then 'Negative Selection', then 'Continue Previous Optimization', as in Figure B.110. This loads the menu seen in Figure B.111. The optimization parameters default to the values used in the trial loaded, with the exception of any optimization parameters which may not be altered. These parameters include the detector generation parameter, shape choice, and number of individuals in the population. These are not alterable, as they are set by the preceding trial, or are no longer needed. Change any parameters as desired and click the 'Continue GA' button. This will load the progress bar as seen in Figure B.112. When the trial completes, a save dialog will appear, as in Figure B.113. Navigate to the desired save location, enter the name for the trial, and click save. The save name for this file is '2Dopt1_spheres_continued.mat'. The results of the trial will be displayed as in Figure B.114 below for 2 identifiers and as in Figure B.115 for more than 2 identifiers. These detectors are ready for integration into the control scheme.

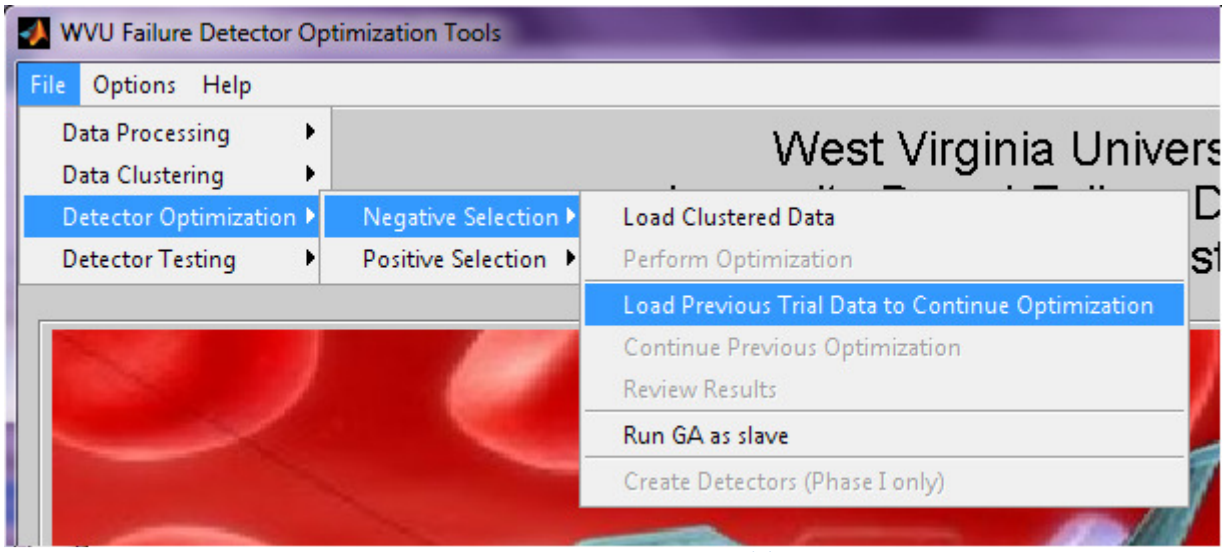

Figure B.108—Opening Load Previous Trial Data Menu

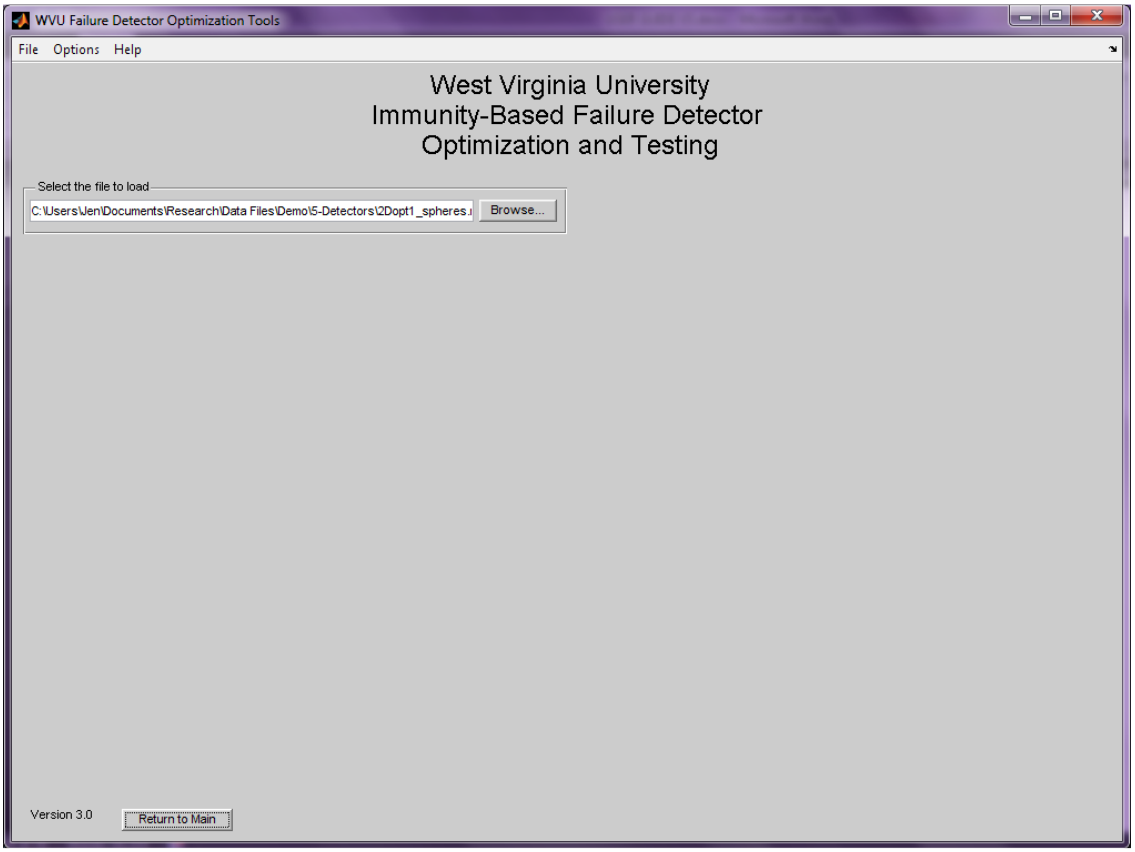

Figure B.109—Previous Trial Data Menu

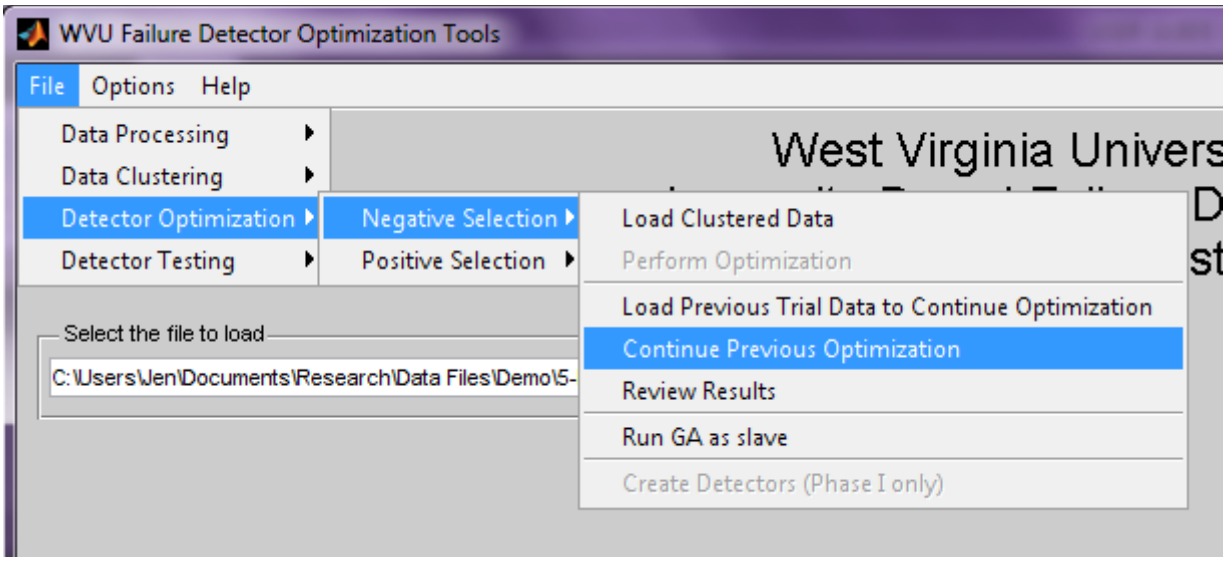

Figure B.110—Opening Continue Optimization Menu

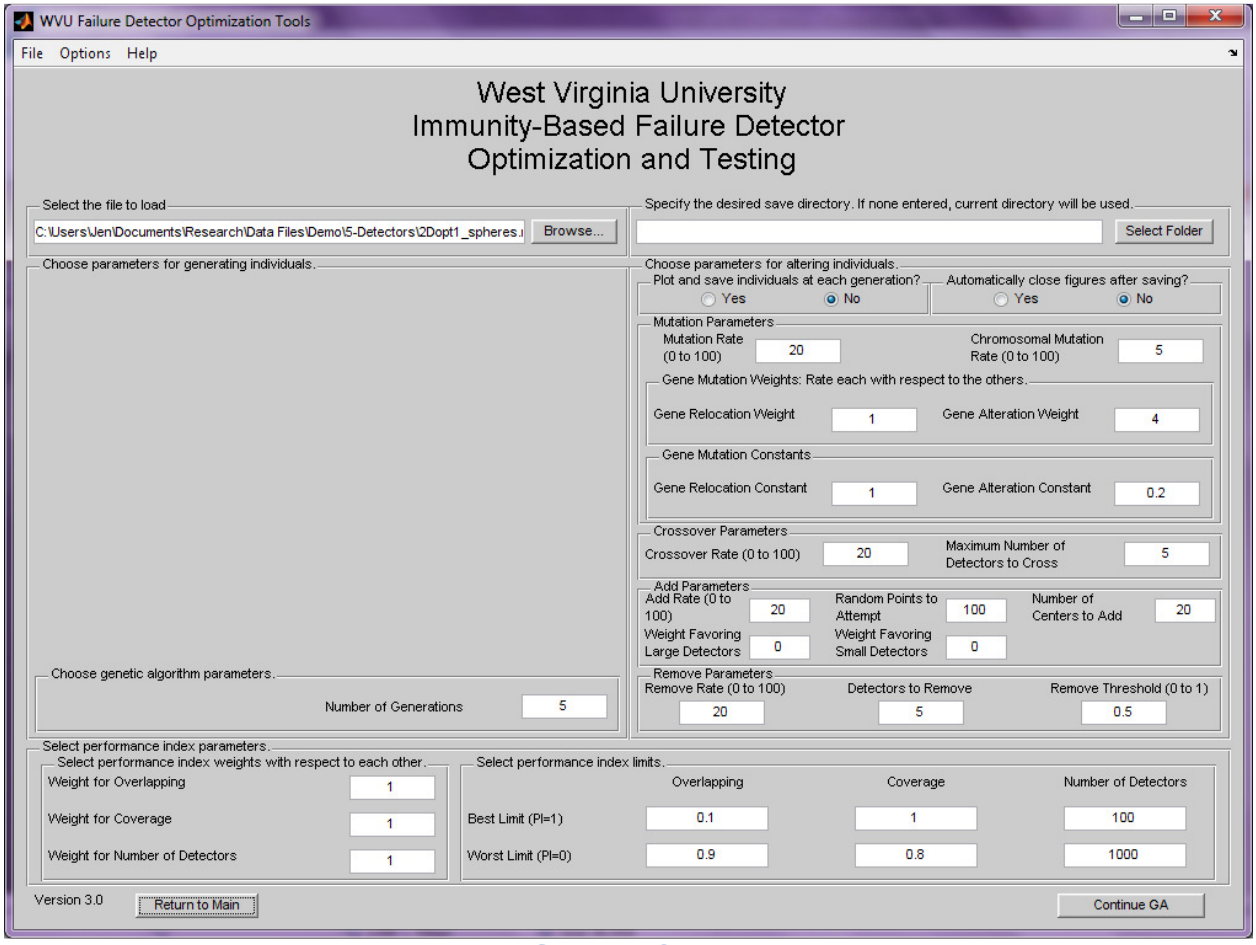

Figure B.111—Continue Optimization Menu

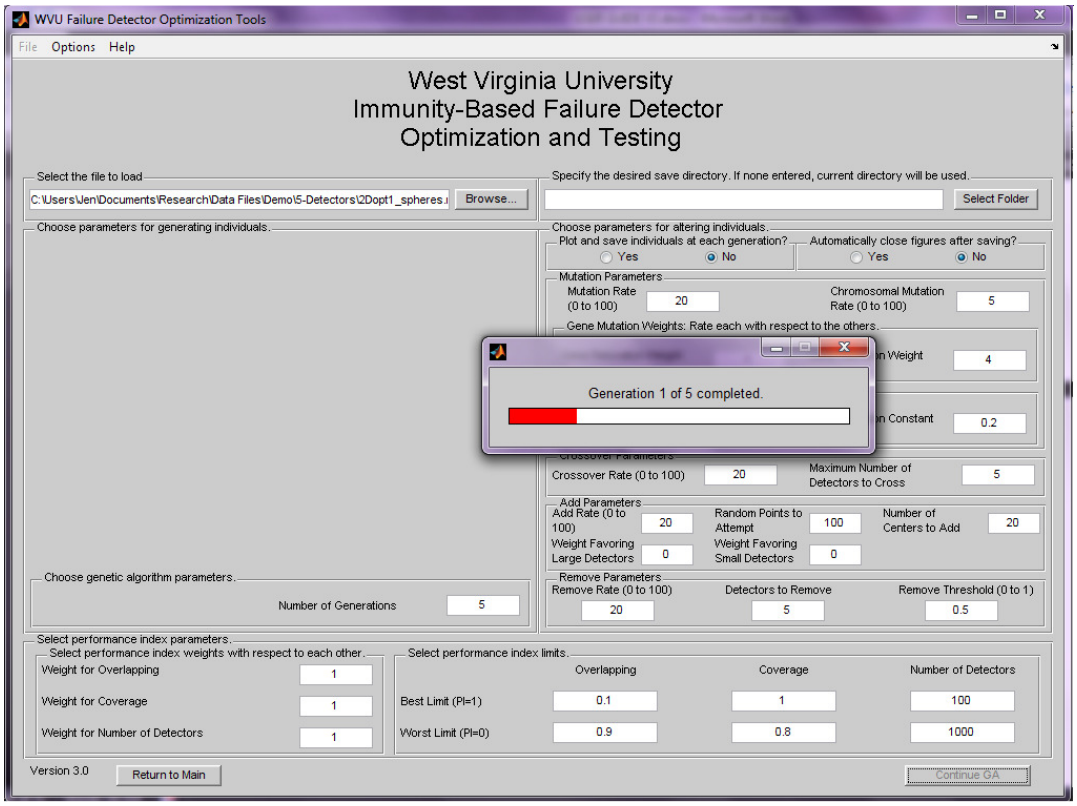

Figure B.112— Continue Optimization in Progress

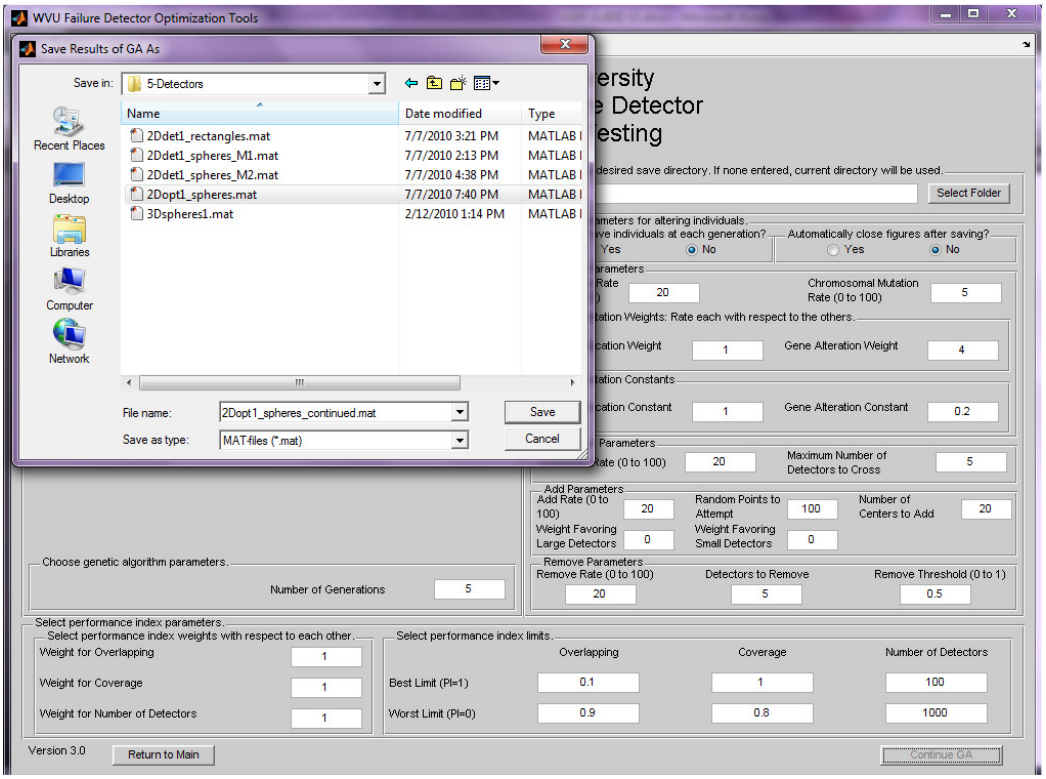

Figure B.113—Genetic Algorithm Save Dialog

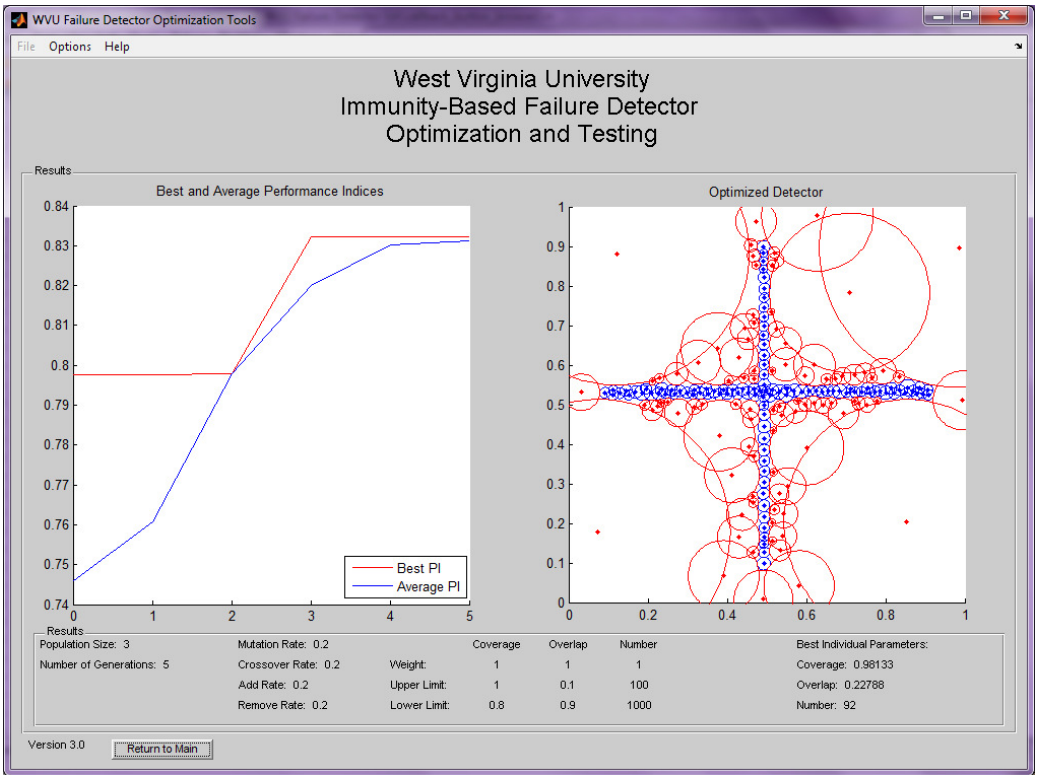

Figure B.114—Optimization Results for 2 Identifiers

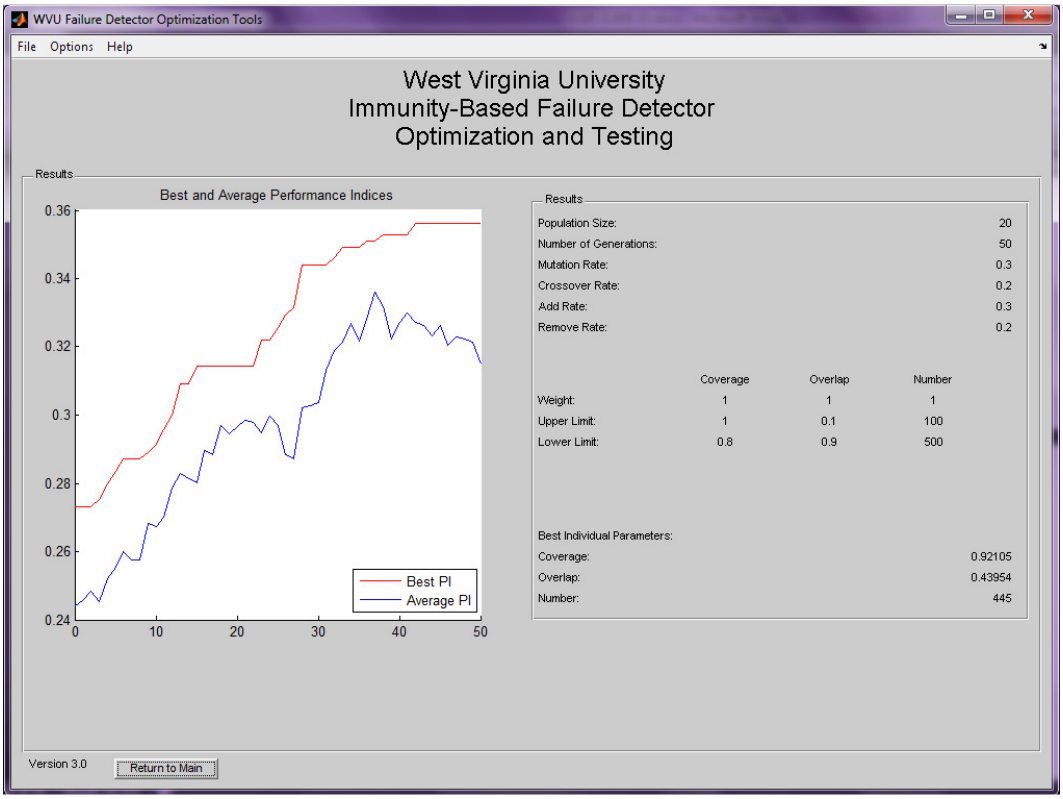

Figure B.115—Optimization Results for Greater Than 2 Identifiers

4.4 *Displaying Results*

At times it may be useful to display the optimization results contained within a file. The results menu will be displayed differently depending upon whether the detectors contain 2 or more identifiers. The 2D trial used in the segment is labeled '2Dopt1_spheres.mat'. The 3-D trial used for this segment is labeled '3Dspheres1.mat'.

To display optimization results, click on the 'File' menu, select 'Detector Optimization', then 'Negative Selection', then 'Load Previous Trial Data to Continue Optimization', as in Figure B.116. This loads the menu seen in Figure B.117. Click on the browse button and navigate to the desired trial data. Then click on the 'File' menu, select 'Detector Optimization', then 'Negative Selection', then 'Review Results'., as in Figure B.118. This loads the menu seen in Figure B.119 if the detectors contain 2 identifiers, or the menu seen in Figure B.120 if the detectors contain more than 2 identifiers.

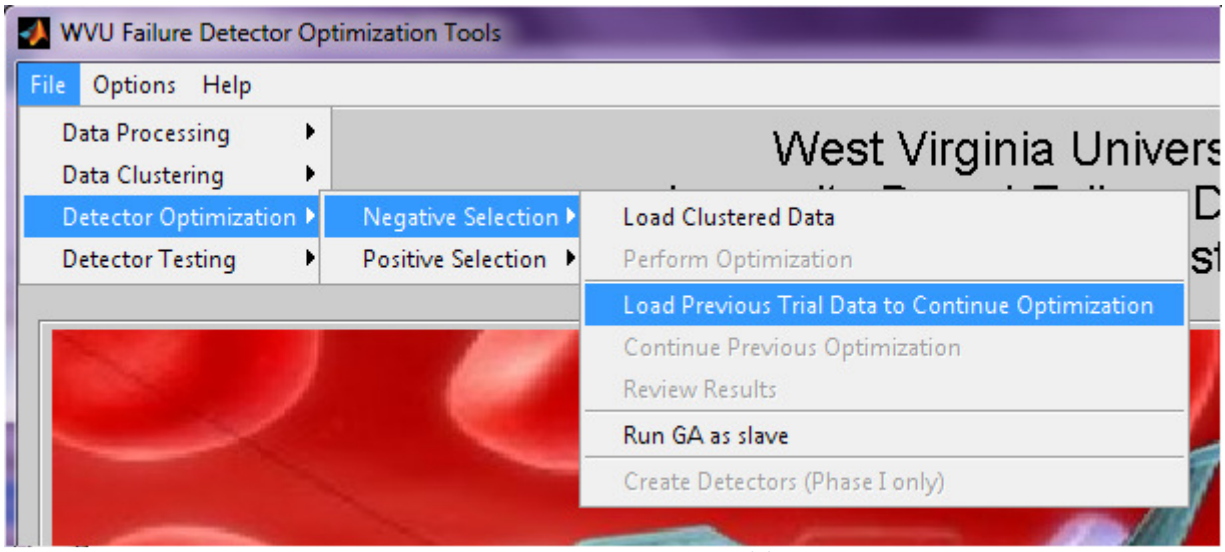

Figure B.116—Opening Load Previous Trial Data Menu

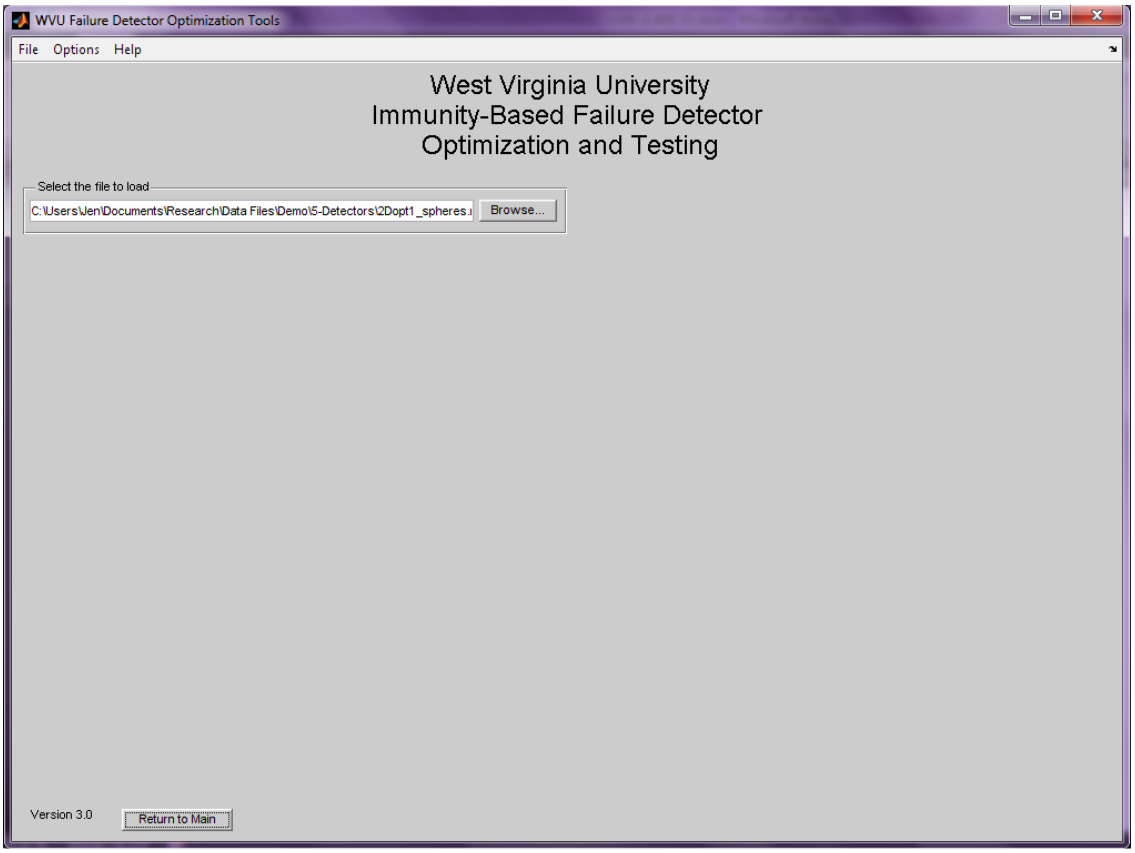

Figure B.117—Load Previous Trial Data Menu

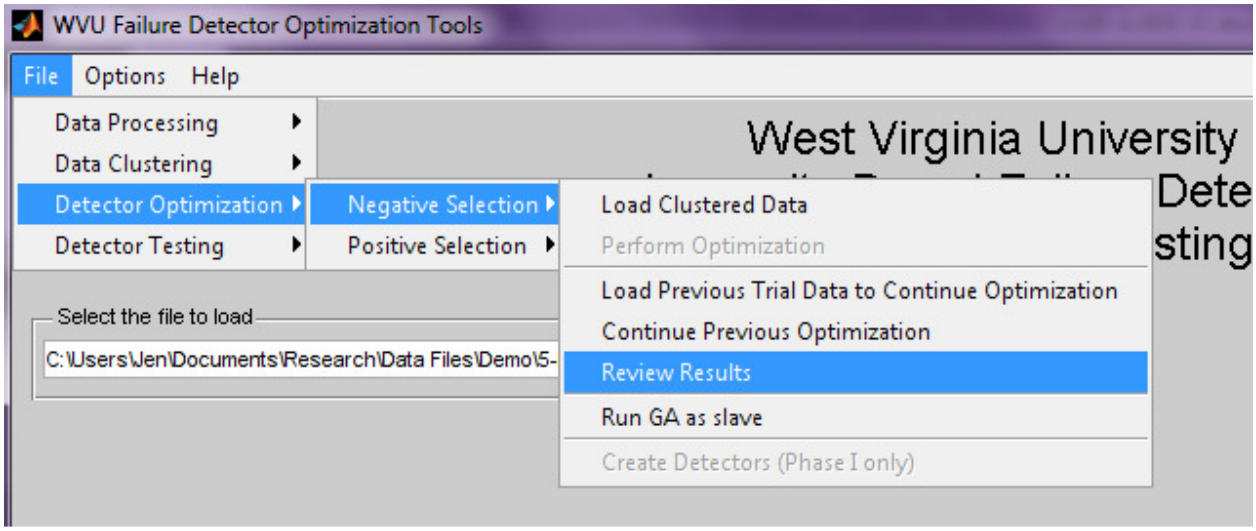

Figure B.118—Opening Review Results Menu

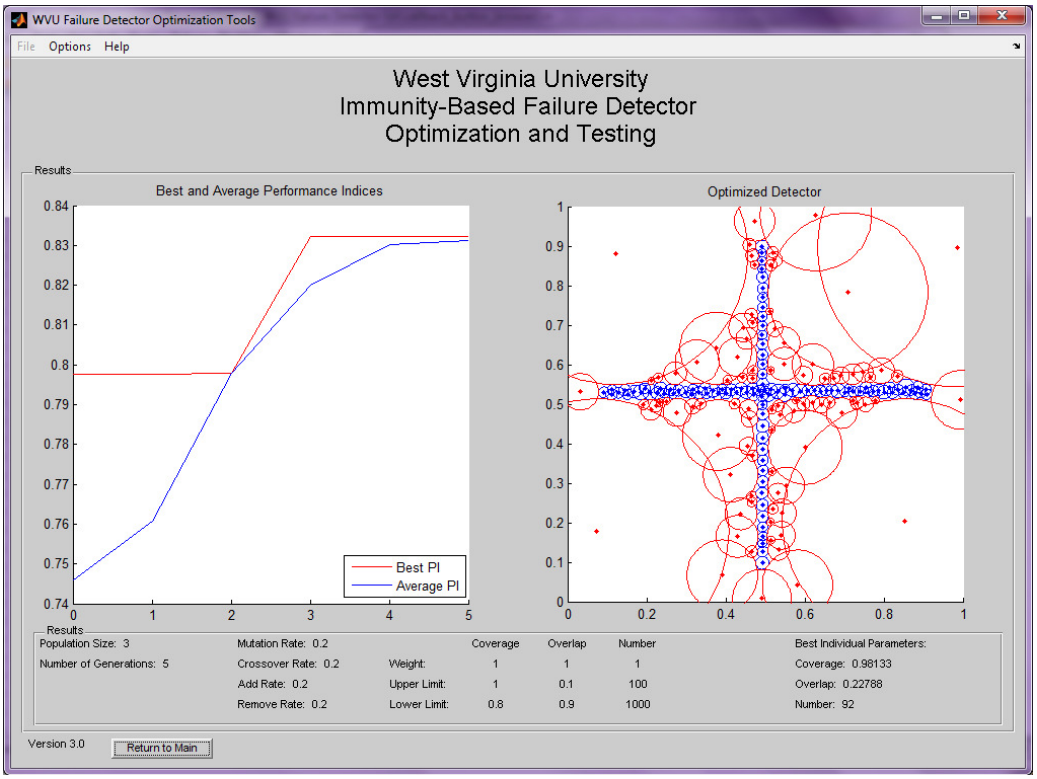

Figure B.119—Optimization Results for 2 Identifiers

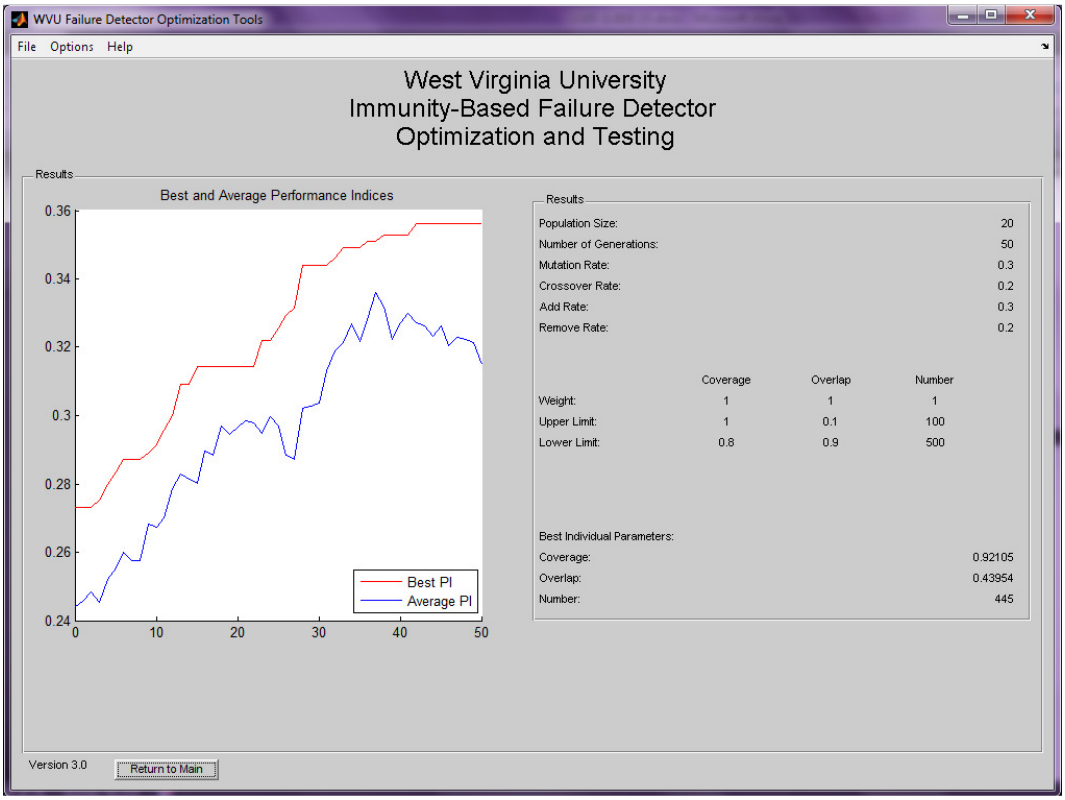

Figure B.120—Optimization Results for Greater Than 2 Identifiers

Chapter 5—Running Detection

5.1 Negative Selection Detectors

Checking detection is the finishing step for the creation of a detector set. Before implementation in an actual control scheme, it is important to know the detection performance of a detector set for the various failures it may need to detect. Detector sets may be saved using the variable name 'antibodies' or 'optdetector' depending on the method used to generate them. Both are acceptable for the detection testing section. Flight data files must be saved with the variable name 'dataN'. Detection testing may be performed for any of the detector shapes producible through the use of this utility.

An example of each detector shape has been included in the 'Demo' directory in the folder labeled '7-Detection Data'. In addition, one failure file and one normal file have been added, which are compatible with detector sets. The default program values are valid for these data. Sample rate is 50Hz, activation window is 50 samples, time of failure occurrence is 40 seconds, and size of the point radius is 0.008.

To perform detection, begin by clicking on the 'File' menu, select 'Detector Testing', then 'Run Detection', as in Figure B.121. This will load the file seen in Figure B.122. Begin by loading the file containing the detectors into the menu using the topmost 'Browse' button. Beneath this button in the same panel are radio buttons for choosing the shape of the detector contained in the file, and the type of detectors. The shape will be chosen automatically if the detectors were generated using this utility. If this is not the case, it is the responsibility of the user to select the correct detector shape. The use must also input the detector type. The default is negative selection detectors, which may be selected using the radio buttons if necessary. In the middle panel is contained another 'Browse" button. Click this and select the data file the detectors are being tested against. This file may contain either normal or abnormal data. The user must correctly report this using the radio buttons included in the panel in order to avoid detection errors. The bottom panel contains a number of parameters the user is responsible for, in order for the detection results to be valid. These are the sampling rate at which the data was collected; the activation window desired, usually 1 second of data at the current sampling rate; the time in seconds the failure occurred, 0 if the data is normal; and the size of the point radius, used for determining the volume of a data point., usually chosen by the distance between consecutive normal data points. With these parameters chosen, click the 'Perform Detection' button. A message box will load to signify that the detection is running. When the detection is complete the results will be displayed as in Figure B.123. If the data contained in the detection file is normal data, the detection results will appear as in Figure B.124. Note that the detection rate will be displayed as "NaN", meaning not a number, since there is no definition of detection if there exists no failure.

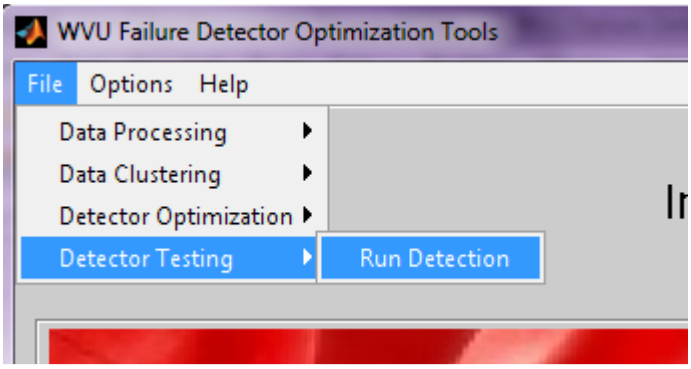

Figure B.121—Opening Detection Menu

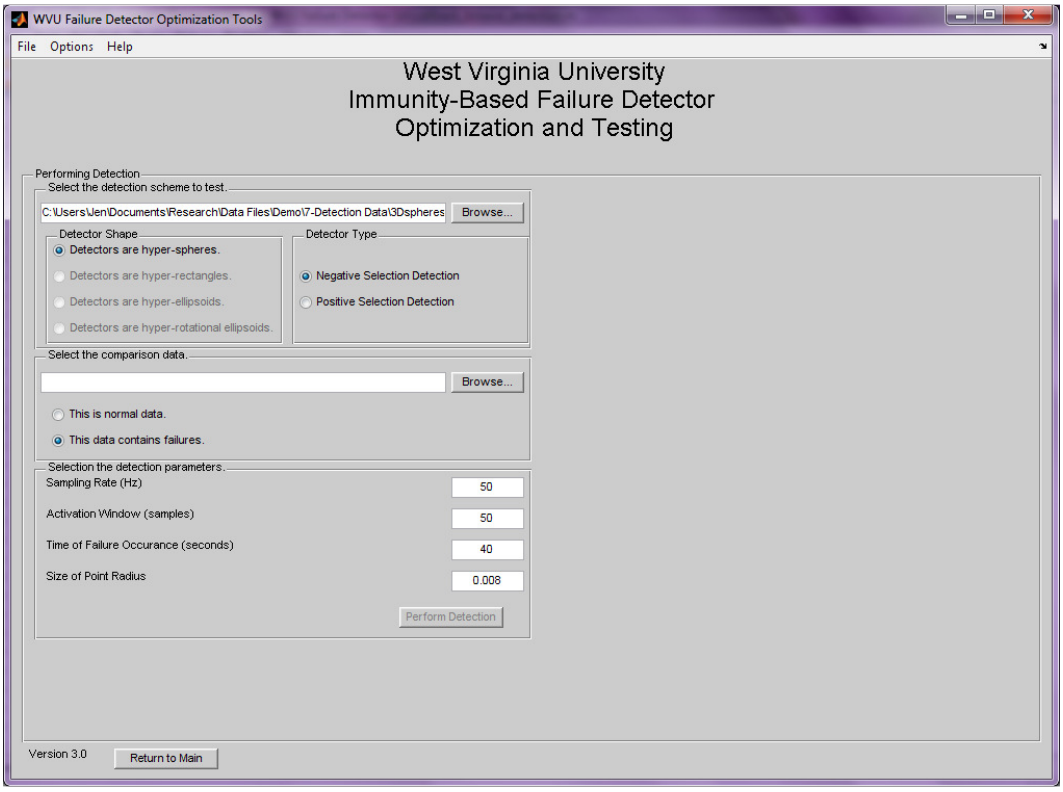

Figure B.122—Detection Menu

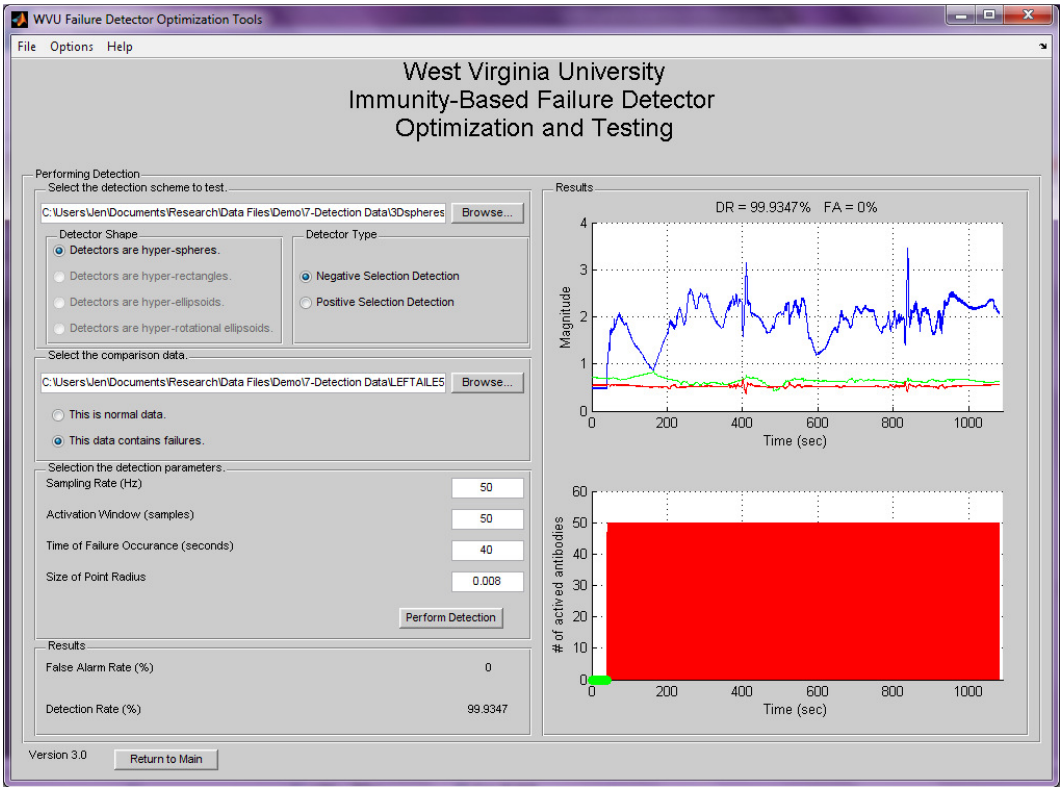

Figure B.123—Detection Results for Failure Data

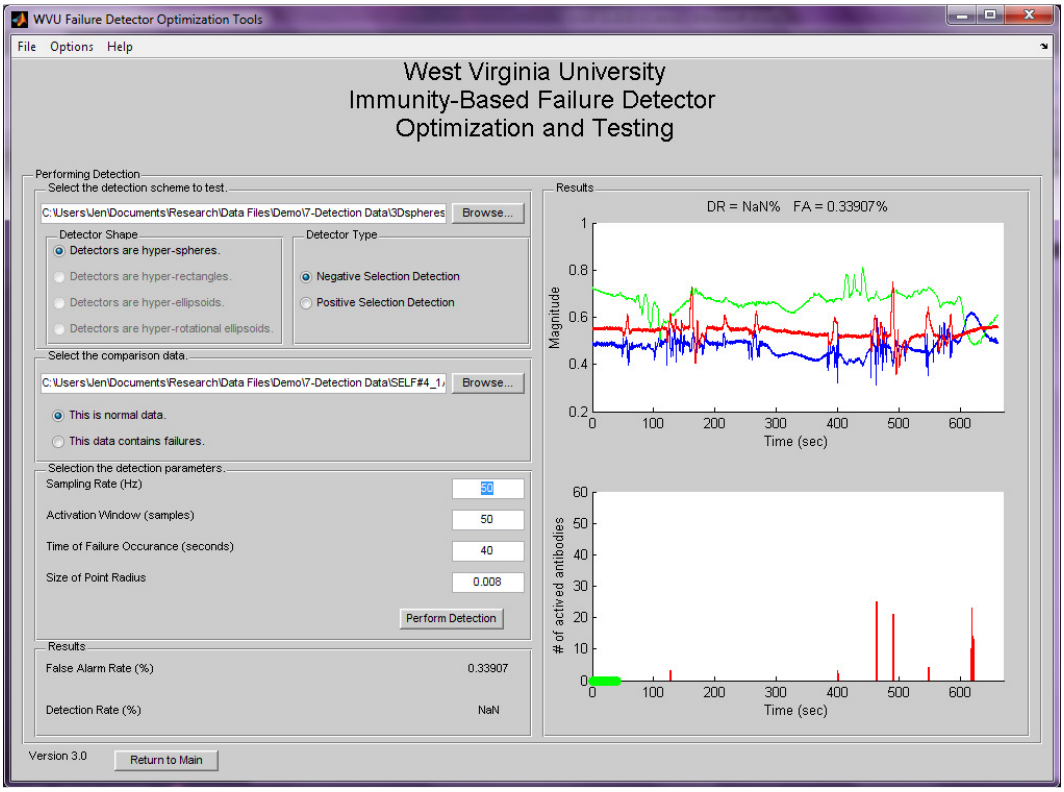

Figure B.124—Detection Results for Normal Data

5.2 Positive Selection Detectors

Checking detection is the finishing step for the creation of a detector set. Before implementation in an actual control scheme, it is important to know the detection performance of a detector set for the various failures it may need to detect. Note that positive selection detectors are only capable of determining than a failure has occurred, not identifying the type of failure. This is why negative selection is favored. Detector sets may be saved using the variable name 'antibodies' or 'optdetector' depending on the method used to generate them. Both are acceptable for the detection testing section. Flight data files must be saved with the variable name 'dataN'. Detection testing may be performed for any of the detector shapes producible through the use of this utility.

An example of positive selection hyper-spheres has been included in the 'Demo' directory in the folder labeled '7-Detection Data'. In addition, one failure file and one normal file have been added, which are compatible with this detector set. The default program values are valid for these data. Sample rate is 50Hz, activation window is 50 samples, time of failure occurrence is 40 seconds, and size of the point radius is 0.008.

To perform detection, begin by clicking on the 'File' menu, select 'Detector Testing', then 'Run Detection', as in Figure B.125. This will load the file seen in Figure B.126. Begin by loading the file containing the detectors into the menu using the topmost 'Browse' button. Beneath this button in the same panel are radio buttons for choosing the shape of the detector contained in the file, and the type of detectors. The shape will be chosen automatically if the detectors were generated using this utility. If this is not the case, it is the responsibility of the user to select the correct detector shape. The use must also input the detector type. The default is negative selection detectors, which may be selected using the radio buttons if necessary. For positive selection

detectors, be sure to select the radio labeled 'Positive Selection Detection'. In the middle panel is contained another 'Browse" button. Click this and select the data file the detectors are being tested against. This file may contain either normal or abnormal data. The user must correctly report this using the radio buttons included in the panel in order to avoid detection errors. The bottom panel contains a number of parameters the user is responsible for, in order for the detection results to be valid. These are the sampling rate at which the data was collected; the activation window desired, usually 1 second of data at the current sampling rate; the time in seconds the failure occurred, 0 if the data is normal; and the size of the point radius, used for determining the volume of a data point., usually chosen by the distance between consecutive normal data points. With these parameters chosen, click the 'Perform Detection' button. A message box will load to signify that the detection is running. When the detection is complete the results will be displayed as in Figure B.127. If the data contained in the detection file is normal data, the detection results will appear as in Figure B.128.

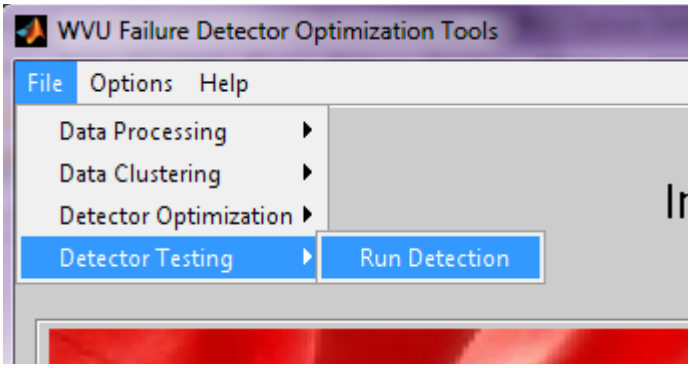

Figure B.125—Opening Detection Menu

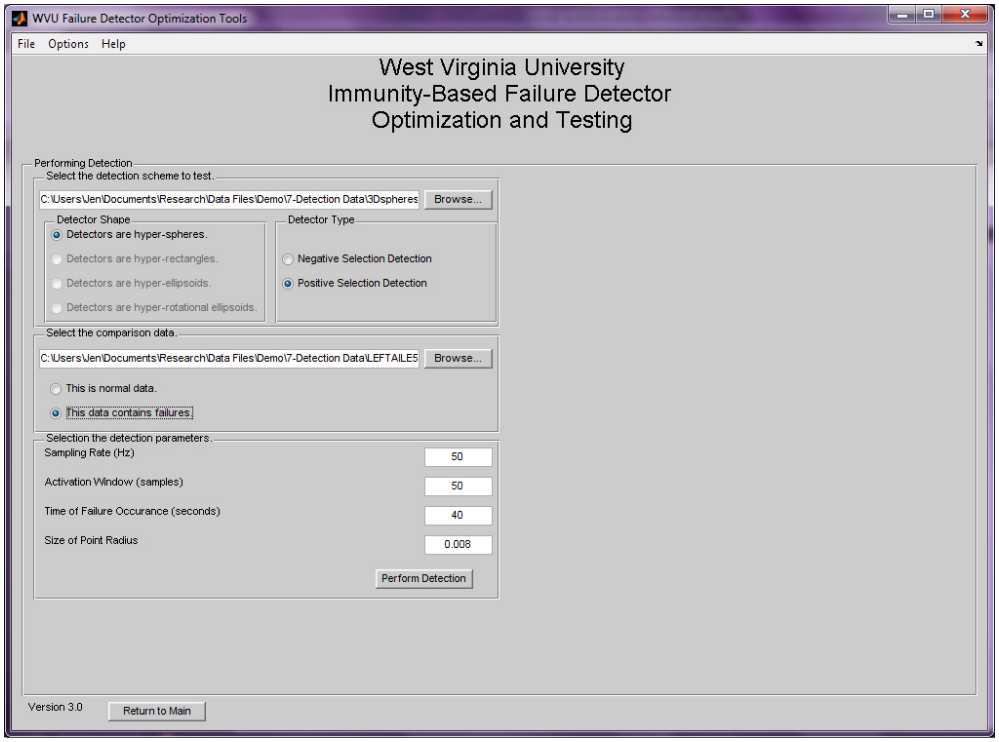

Figure B.126—Detection Menu

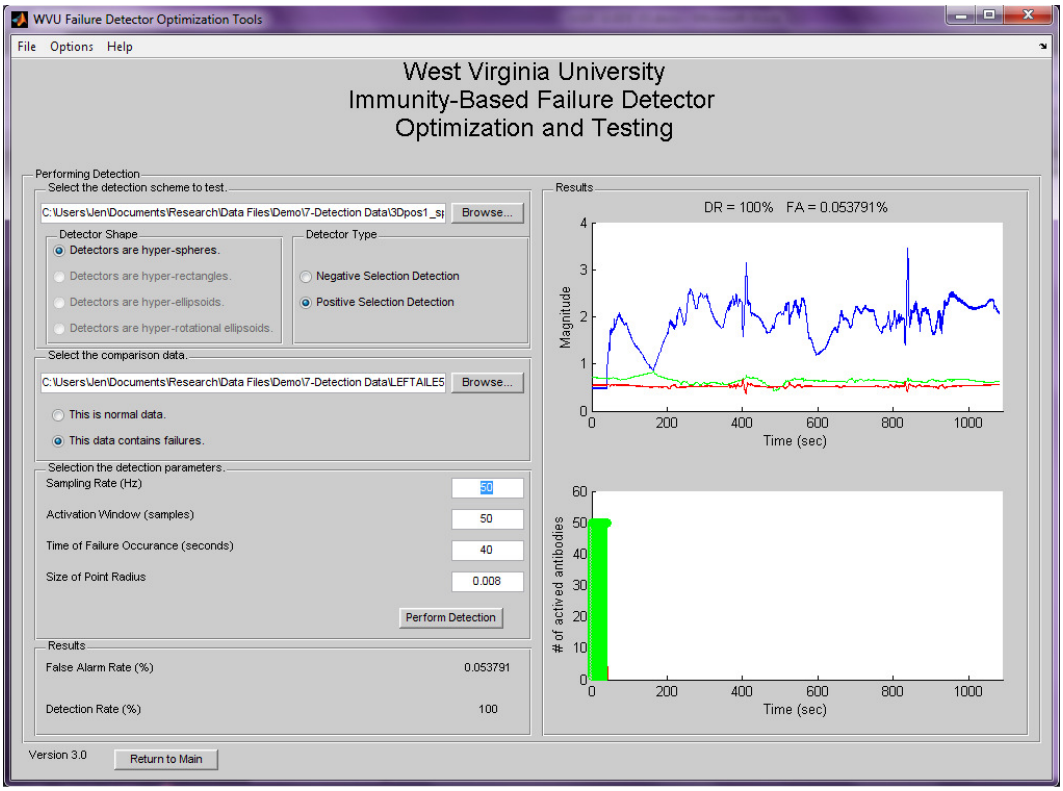

Figure B.127—Detection Results for Failure Data

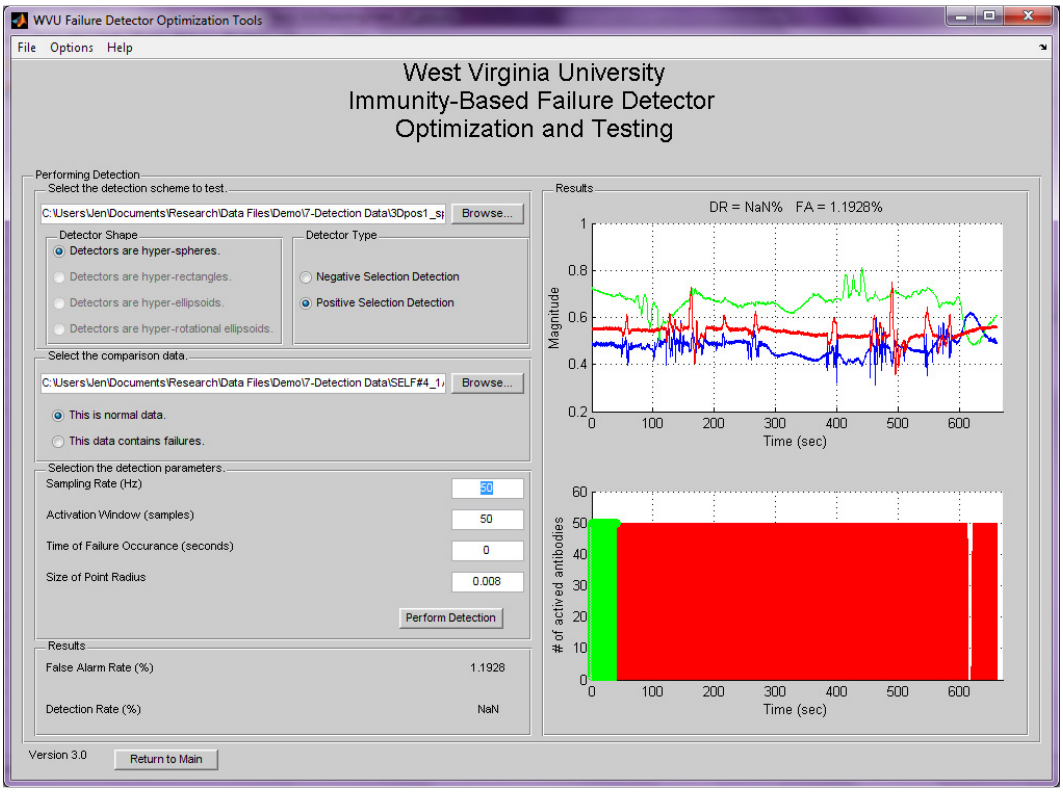

Figure B.128—Detection Results for Normal Data

Chapter 6—Merging Data Files

Many instances may occur when it becomes useful to merge data files into one. Merging data can occur at many times throughout the detector creation process. These methods have been included for added user convenience, and are not necessary to the detector creation process.

6.1 Merging Raw Data

Merging raw data files is useful for generating a self to cover many areas of the flight envelope from data files that may each cover only a few. In order to merge raw data files, each of the files must contain the same number of columns of data. Be aware that the program can only check that each file contains the correct number of columns of data. It is the responsibility of the user to ensure that these columns of data contain the same identifiers, all of the same identifiers, and in the same order. This method is not recommended for processed data, but if this method is used with processed data, the data must also have been normalized using the same limits for each file. The variable name for processed data must also be changed from 'selfdata' to 'sensors' and then back to 'selfdata' before clustering, making this much less convenient than using the method for merging processed data.

This function is simple. Two data files exist, which are assumed compatible for this guide. Note that if the two files chosen do not have the same number of columns of data, an error message will be received, as in Figure B.129 below. In no case will the program allow the user to attempt to merge two raw data files with incompatible dimensions. The data array in each file must also have the variable name 'sensors'. Once the two files have been chosen, the data from each is loaded, and the two data arrays are concatenated by adding the data points in the second file below the data points in the first file. This new, larger array is then saved to the variable name 'sensors'. This file is then ready for processing.

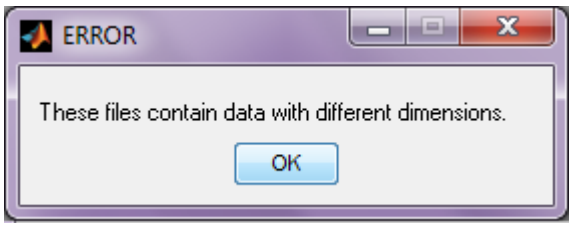

Figure B.129—Incompatible Raw Files Error Message

To merge raw data, click on the 'File' menu, select 'Data Processing', then select 'Merge Raw Data', as in Figure B.130 below. This loads the menu shown in Figure B.131 below. It contains two file loading panels with two 'Browse' buttons. Click on each of these 'Browse' buttons and navigate to the desired files. This is shown in Figure B.132. The files chosen for this walkthrough are 'selfdata1-2D.mat' and 'selfdata2-2D.mat', within the folder labeled '2-Truncated Raw Data' in the 'Demo' directory. Click on the 'Merge Raw Data' button. This process is very up quick, thus there is no need for a progress bar. When the function, finished, a save dialog will appear, as in Figure B.133. Navigate to the desired save directory, name the file, and click save. The save name used for this file is 'selfdata12-2D.mat'.

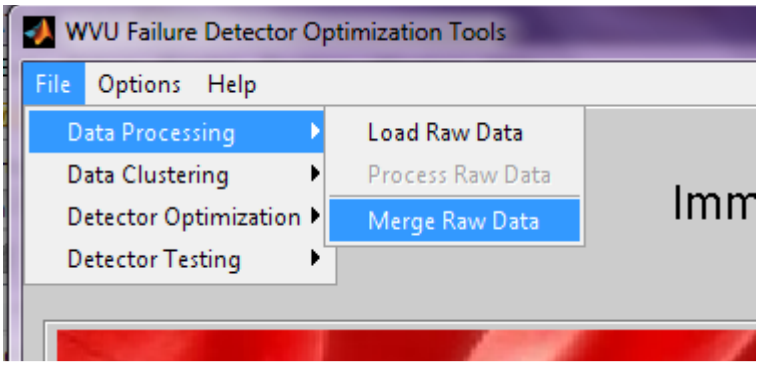

Figure B.130—Opening Merge Raw Data Menu

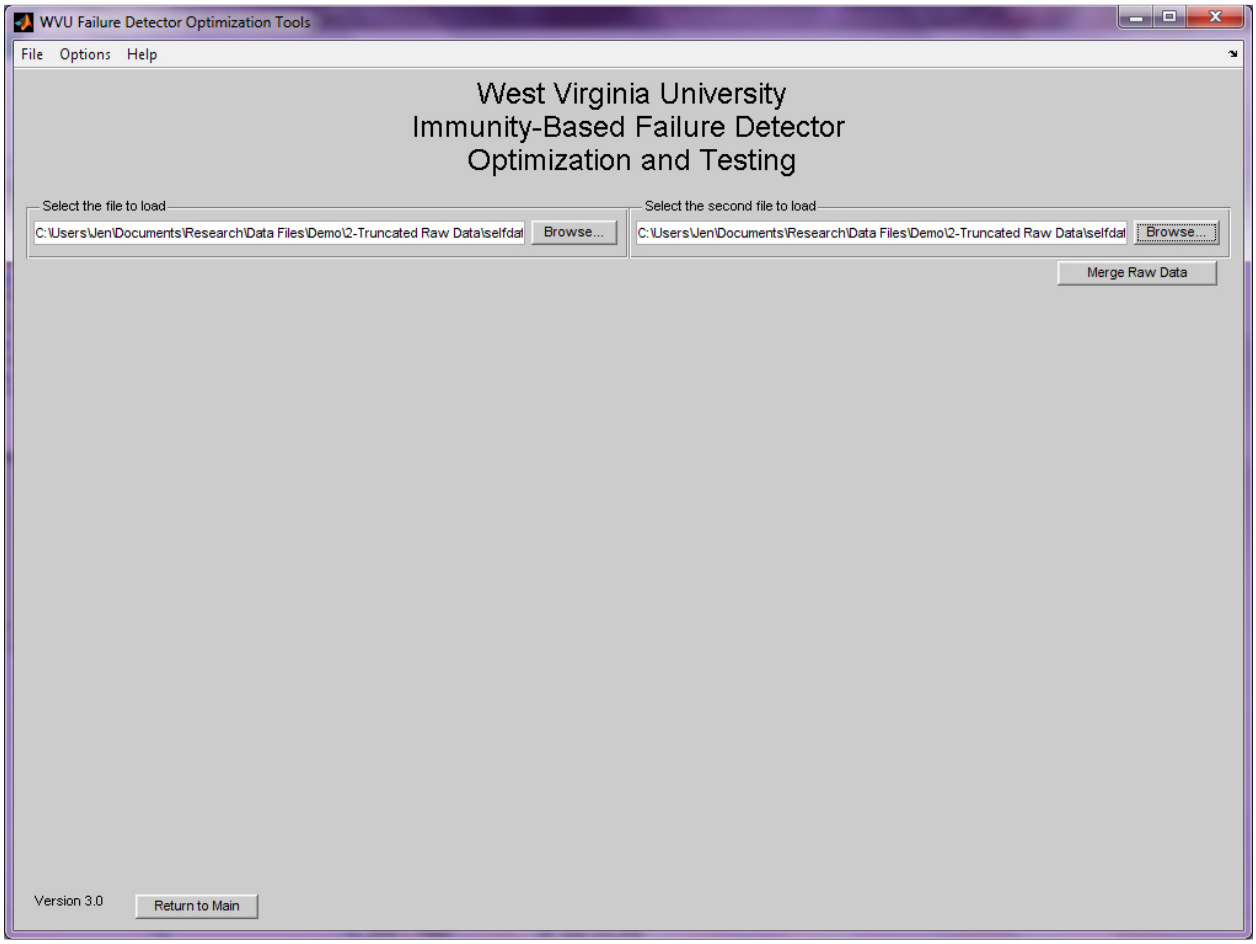

Figure B.131—Merge Raw Data Menu

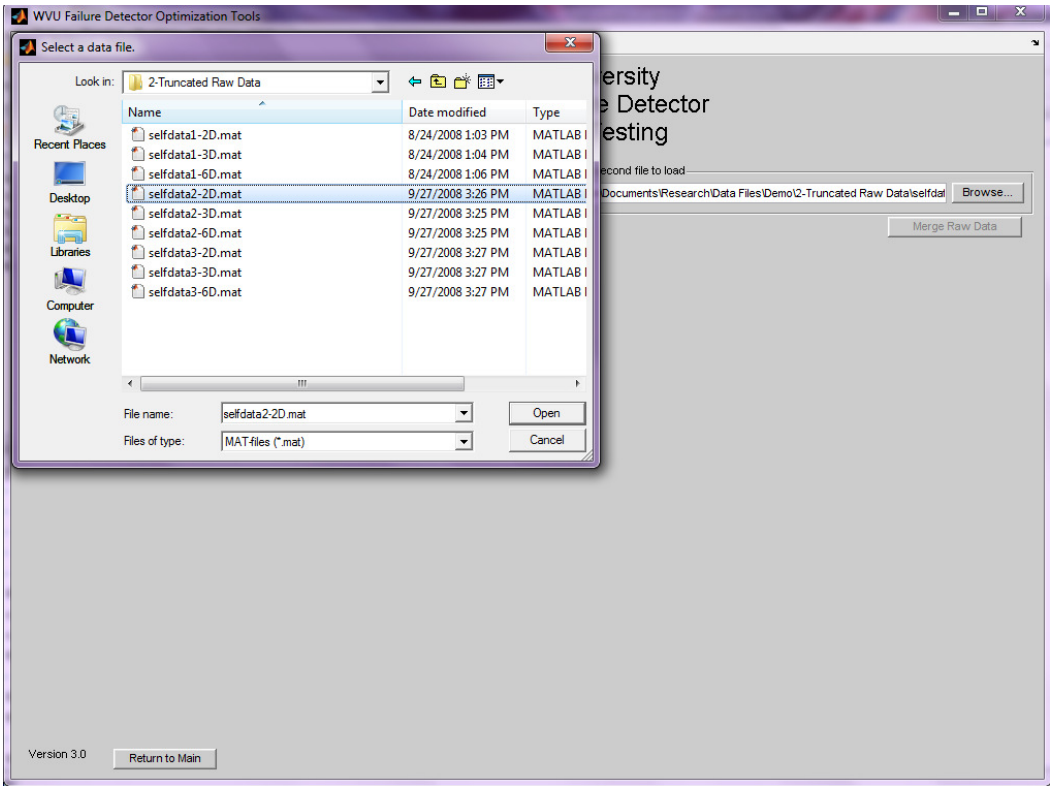

Figure B.132—Merge Raw Data Browse

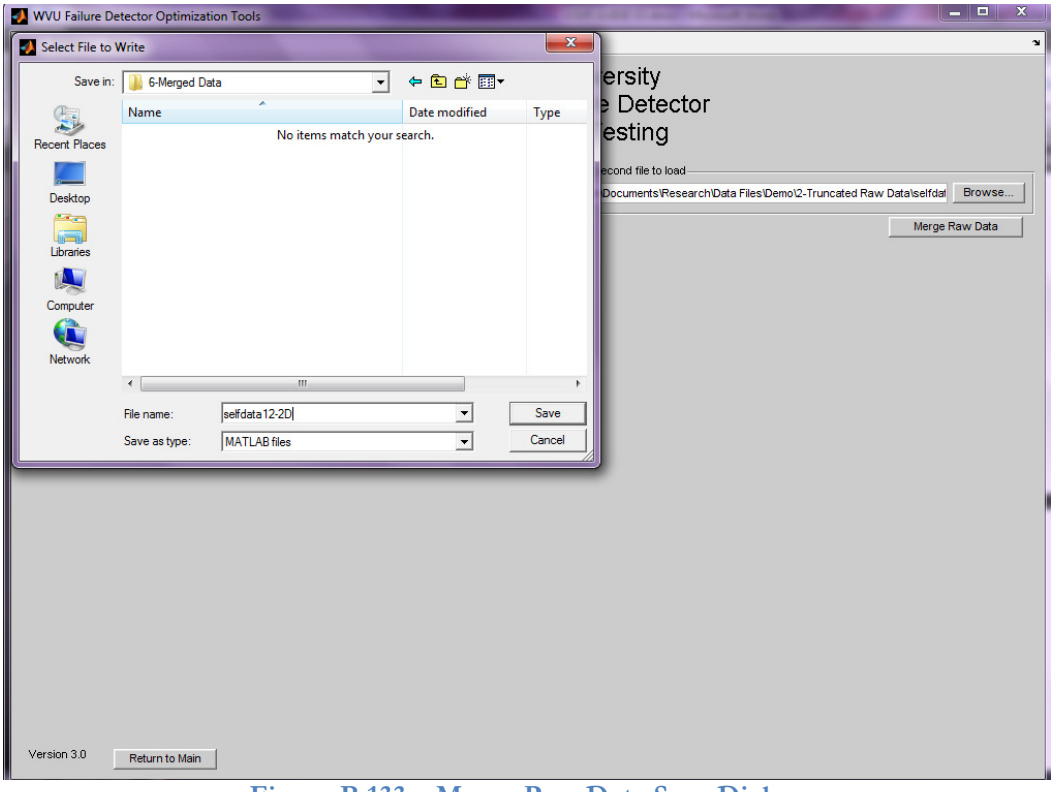

Figure B.133—Merge Raw Data Save Dialog

6.2 Merging Processed Data

Merging processed data files is useful for generating a self to cover many areas of the flight envelope from data files that may each cover only a few. In order to merge processed data files, each of the files must contain the same number of columns of data. Be aware that the program can only check that each file contains the correct number of columns of data. It is the responsibility of the user to ensure that these columns of data contain the same identifiers, all of the same identifiers, and in the same order. For this method, the processed data files do not necessarily need to be normalized to the same limits.

This function loads both the data arrays and normalization limits from each data file. In no case will the program allow the user to attempt to merge two processed data files with incompatible dimensions. The variables should be 'normmaximums', 'normminimums', and 'selfdata'. These will already be correct if the processing was performed using the IFDOT Utility. For each of the data files, the normalized data is converted back to its original values using the normalization limits stored with the file. The restored raw data is then concatenated by placing the data points from the second data array at the end of the first data array. The normalization limits of the two files are then compared and new limits are chosen. Minimums are compared and the smaller minimum from each column in the two files is chosen as the new 'normminimums'. Maximums are compared and the larger maximum from each column is chosen as the new 'normmaximums'. These new limits are then used to renormalize the data and any duplicate points are removed from the merged set. This data is saved to 'selfdata', and is then ready for clustering.

To merge processed data, click on the 'File' menu, select 'Data Clustering', then select 'Merge Processed Data', as in Figure B.134 below. This loads the menu shown in Figure B.135 below. It contains two file loading panels with two 'Browse' buttons. Click on each of these 'Browse' buttons and navigate to the desired files. This is shown in Figure B.136. The files chosen for this walkthrough are '2Dprocdata1.mat' and '2Dprocdata2.mat', within the folder labeled '3-Processed Raw Data' in the 'Demo' directory. Click on the 'Merge Processed Data' button. This process will open a progress bar as in Figure B.137. When the function, finished, a save dialog will appear, as in Figure B.138. Navigate to the desired save directory, name the file, and click save. The save name used for this file is 'procdata12-2D.mat'.

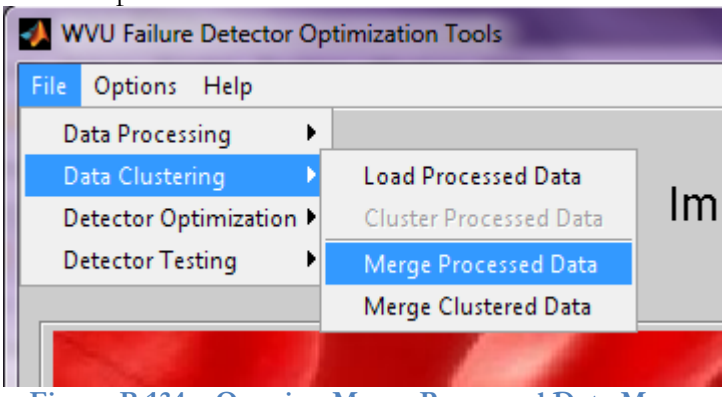

Figure B.134—Opening Merge Processed Data Menu

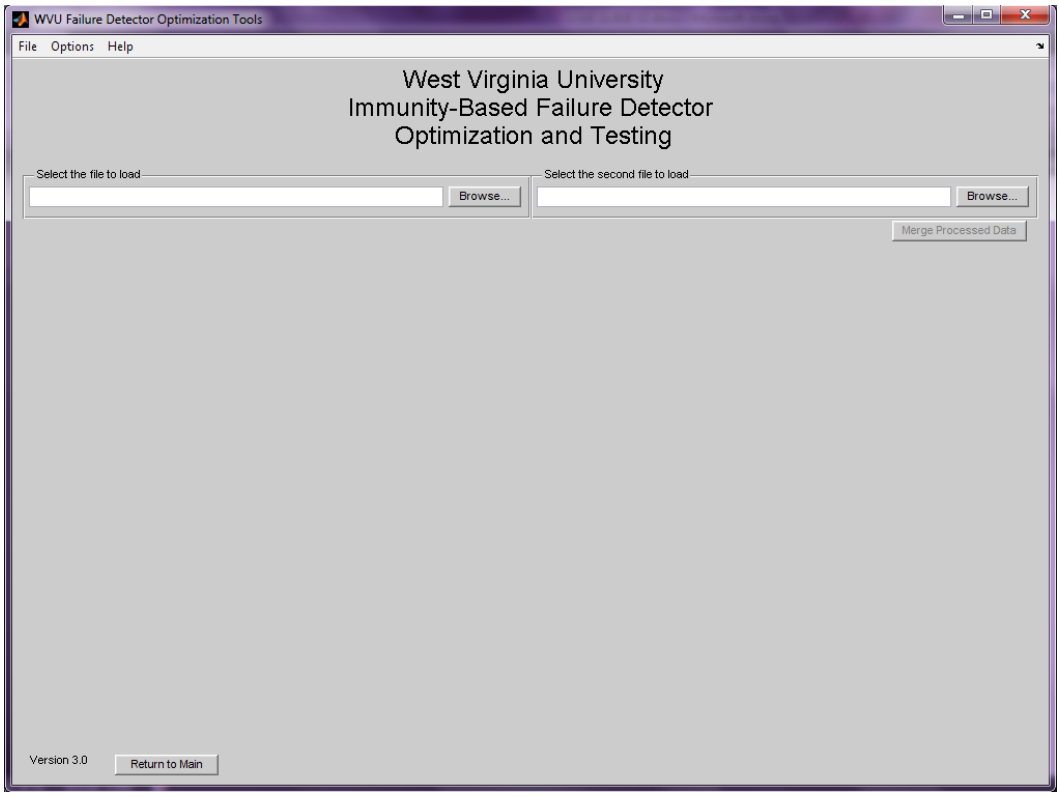

Figure B.135—Merge Processed Data Menu

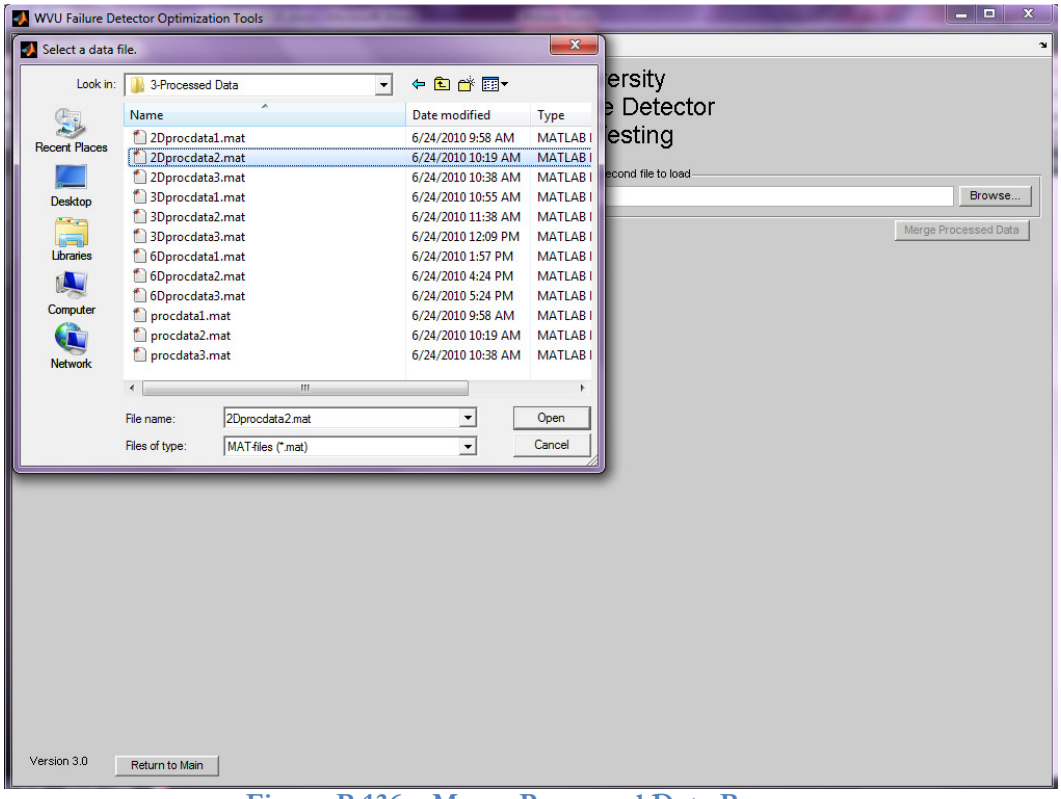

Figure B.136—Merge Processed Data Browse

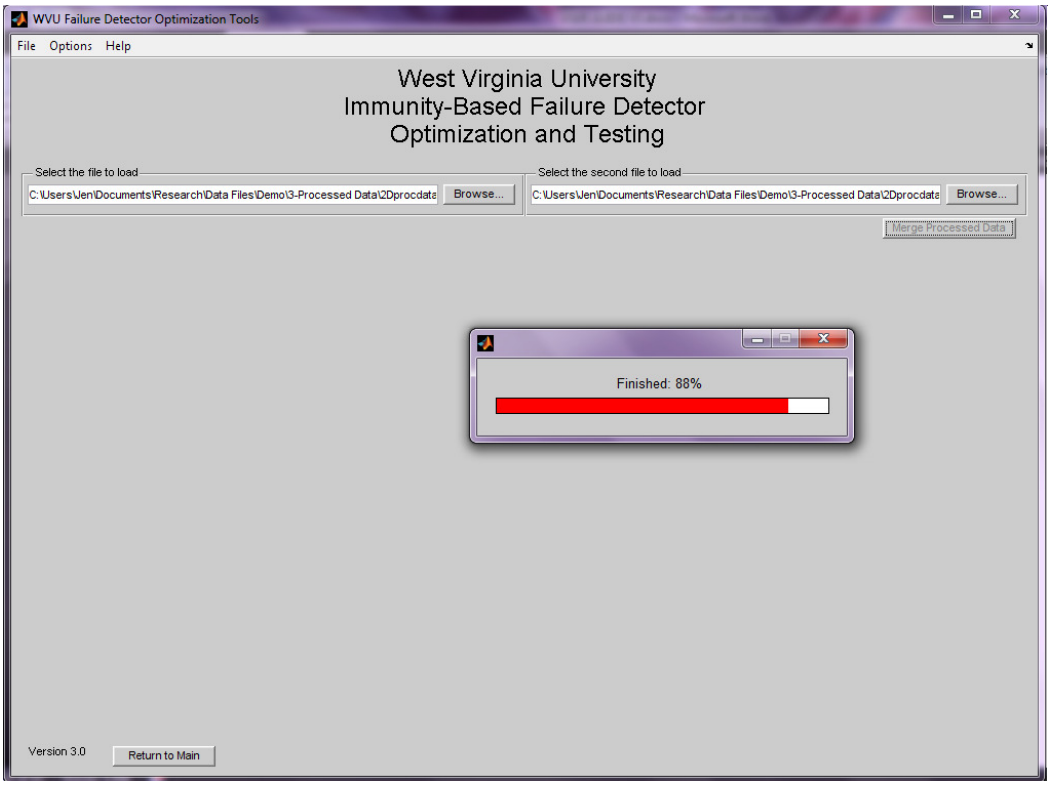

Figure B.137—Merge Processed Data In Progress

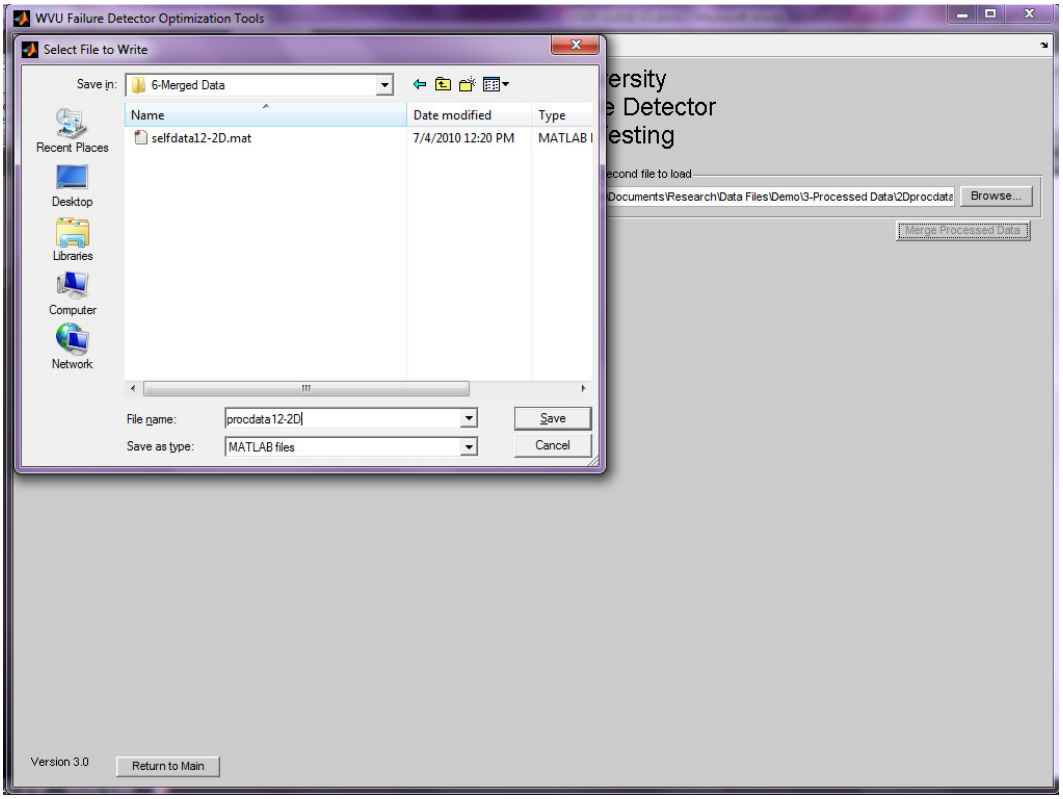

Figure B.138—Merge Processed Data Save Dialog

6.3 Merging Clustered Data

Merging clusters is useful for creating a self definition for the full flight envelop made from several sets of clusters which were created independently and cover different areas of the flight envelop. As with other merging functions in this program, it is the responsibility of the user to ensure that the data contained in the clustered set are compatible with each other. In the case of clusters, each clustered set needs to have been clustered from normal condition data that was normalized to the same limits, even if the normal condition data was taken over varying parts of the flight envelope.

The program cannot ensure that the clusters encompass the appropriate data; however, the program will reject any attempt to merge clusters that do not contain the same number of dimensions, or are not composed of the same shape. For instance, a set of hyper-sphere clusters cannot be merged with a set of hyper-rectangle clusters, even if they contain the same number of identifiers.

This function not only merges the two sets of clusters, but eliminates any redundant clusters from the new set. To merge clustered data, click on the 'File' menu, select 'Data Clustering', then select 'Merge Clustered Data', as in Figure B.139 below. This loads the menu shown in Figure B.140 below. It contains two file loading panels with two 'Browse' buttons. Click on each of these 'Browse' buttons and navigate to the desired files. This is shown in Figure B.141. The files chosen for this walkthrough are '2Dclust1_M1.mat' and '2Dclust2_M2.mat', within the folder labeled '3-Clustered Data' in the 'Demo' directory. Click on the 'Merge Clustered Data' button. This process will open a progress bar as in Figure B.142. When the function, finished, a save dialog will appear, as in Figure B.143**.** Navigate to the desired save directory, name the file, and click save. The save name used for this file is '2Dclust12.mat'.

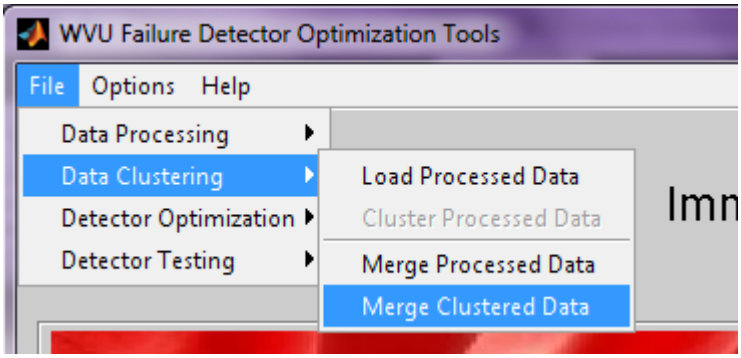

Figure B.139—Opening Merge Clustered Data Menu

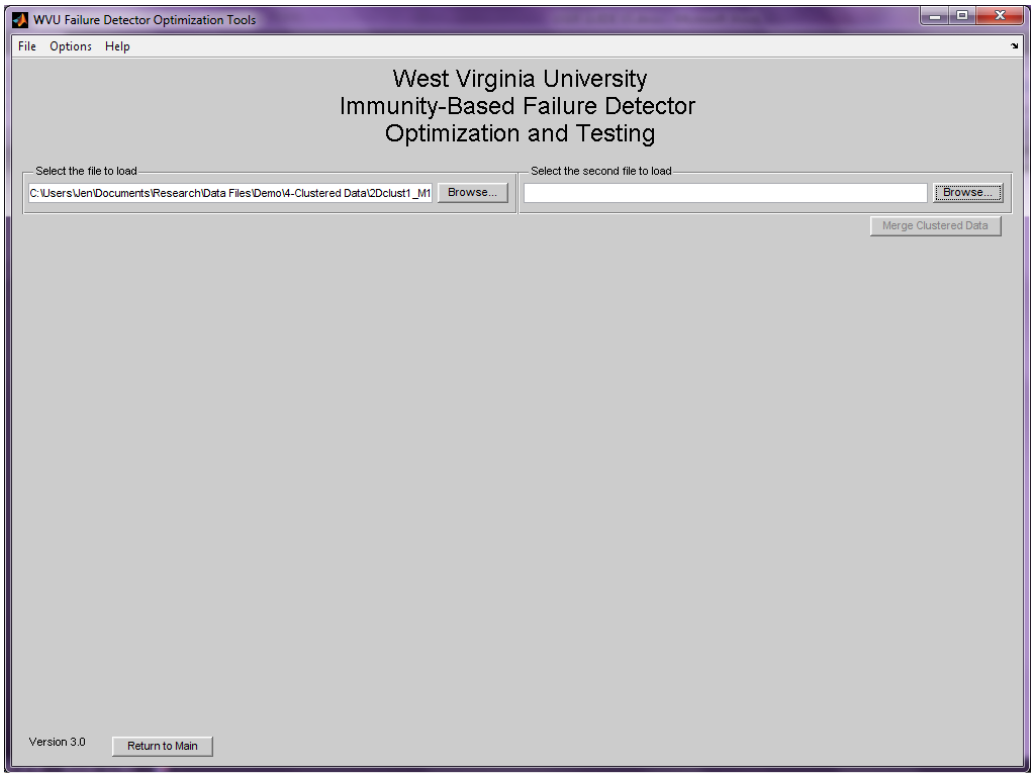

Figure B.140—Merge Clusters Menu

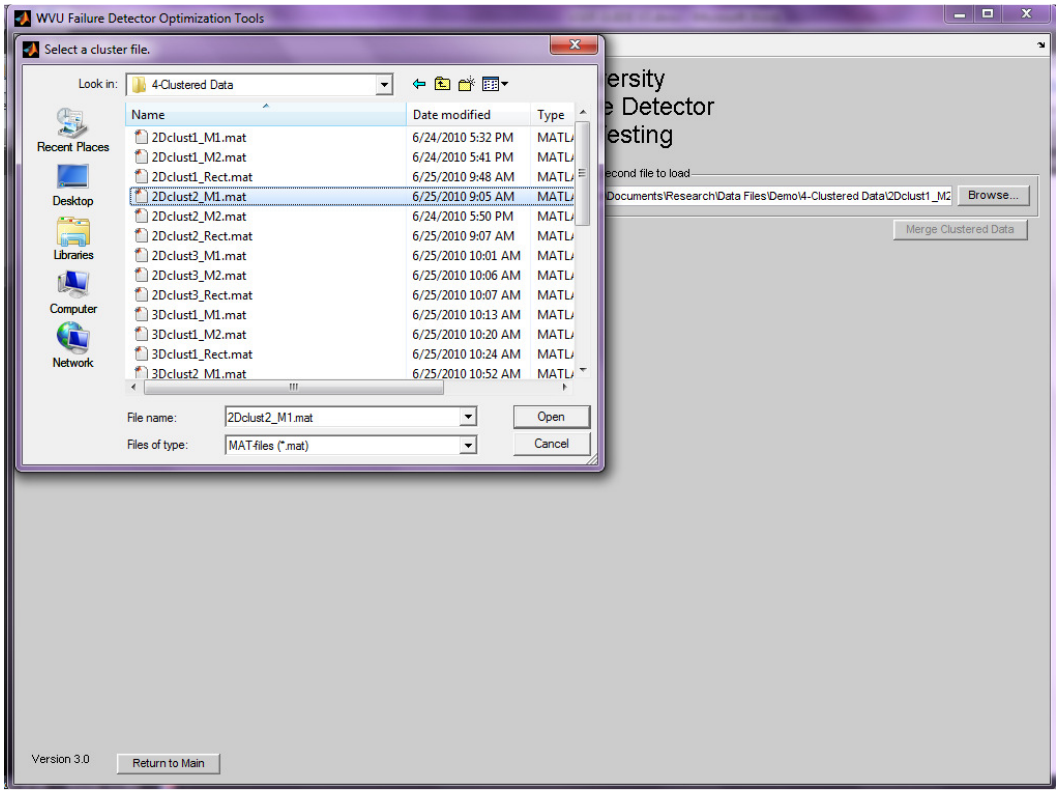

Figure B.141—Merge Clusters Browse

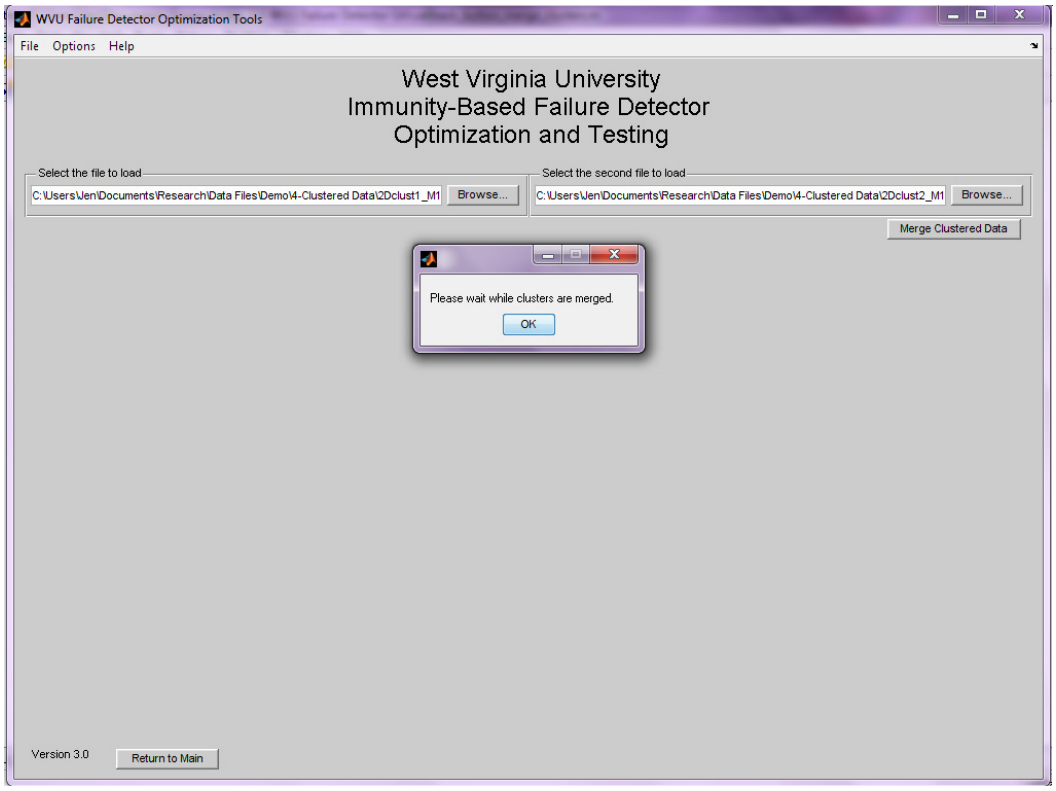

Figure B.142—Merge Clusters In Progress

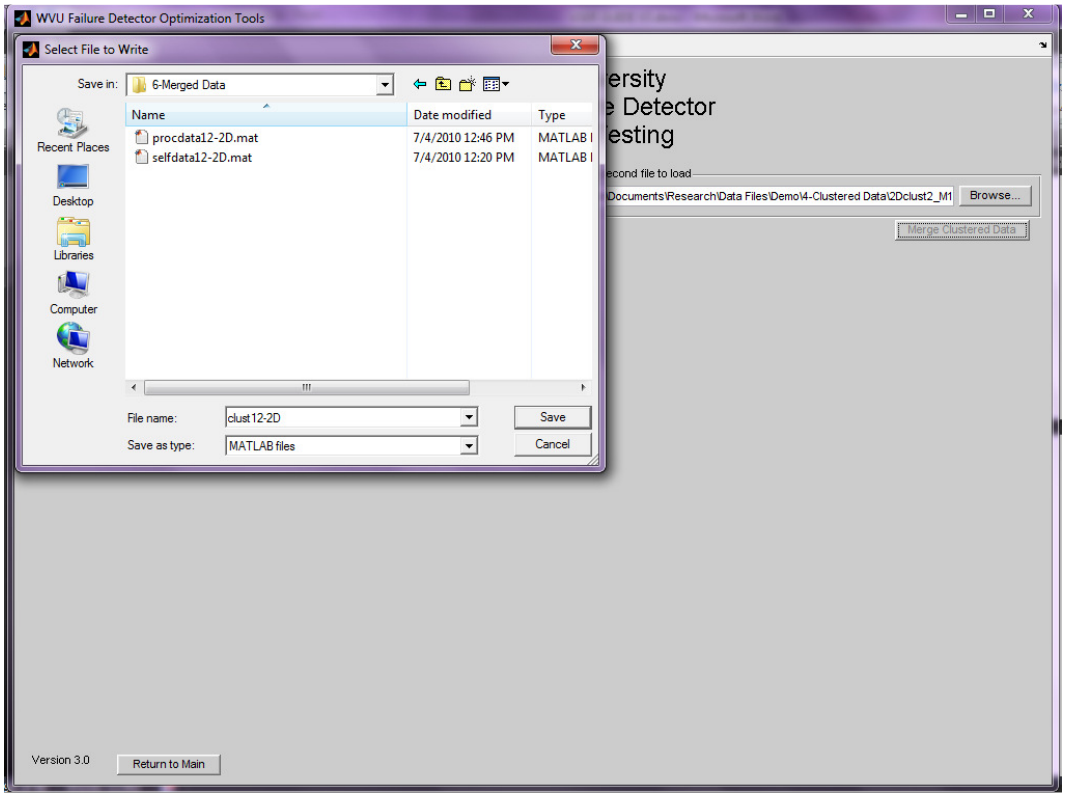

Figure B.143—Merge Clusters Save Dialog

6.4 Merging Positive Selection Detectors

Merging positive detectors is useful for creating a self definition for the full flight envelop made from several sets of positive selection detectors which were created independently and cover different areas of the flight envelop. As with other merging functions in this program, it is the responsibility of the user to ensure that the data contained in the detector sets are compatible with each other. In the case of positive selection detectors, each detector set needs to have been clustered from normal condition data that was normalized to the same limits, even if the normal condition data was taken over varying parts of the flight envelope.

The program cannot ensure that the positive detectors encompass the appropriate data; however, the program will reject any attempt to merge positive detectors that do not contain the same number of dimensions, or are not composed of the same shape. For instance, a set of hyper-sphere detectors cannot be merged with a set of hyper-rectangle detectors, even if they contain the same number of identifiers.

This function not only merges the two sets of positive selection detectors, but eliminates any redundant detectors from the new set. To merge positive detectors, click on the 'File' menu, select 'Data Clustering', then select 'Merge Clustered Data', as in Figure B.144 below. This loads the menu shown in Figure B.145 below. It contains two file loading panels with two 'Browse' buttons. Click on each of these 'Browse' buttons and navigate to the desired files. This is shown in Figure B.146. The files chosen for this walkthrough are '2Dpos1_M1.mat' and '2Dpos2_M2.mat', within the folder labeled '3-Clustered Data' in the 'Demo' directory. Click on the 'Merge Clustered Data' button. This process will open a progress bar as in Figure B.147. When the function, finished, a save dialog will appear, as in Figure B.148. Navigate to the desired save directory, name the file, and click save. The save name used for this file is 'pos12-2D.mat'.

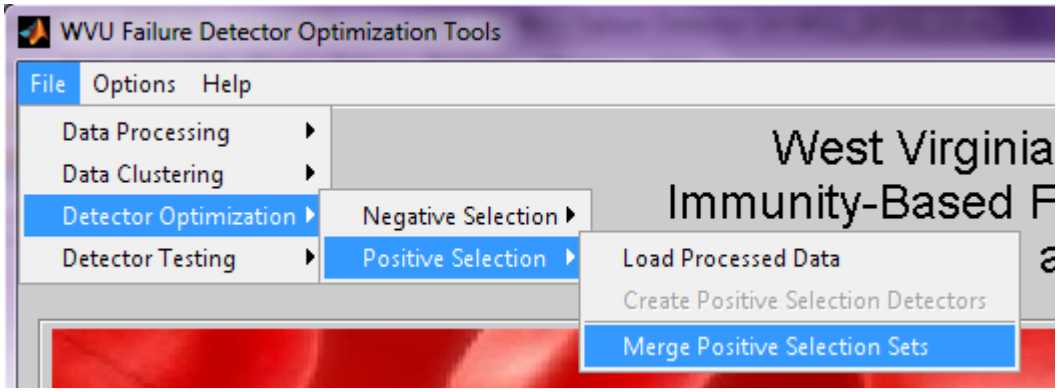

Figure B.144—Opening Merge Positive Detectors Menu

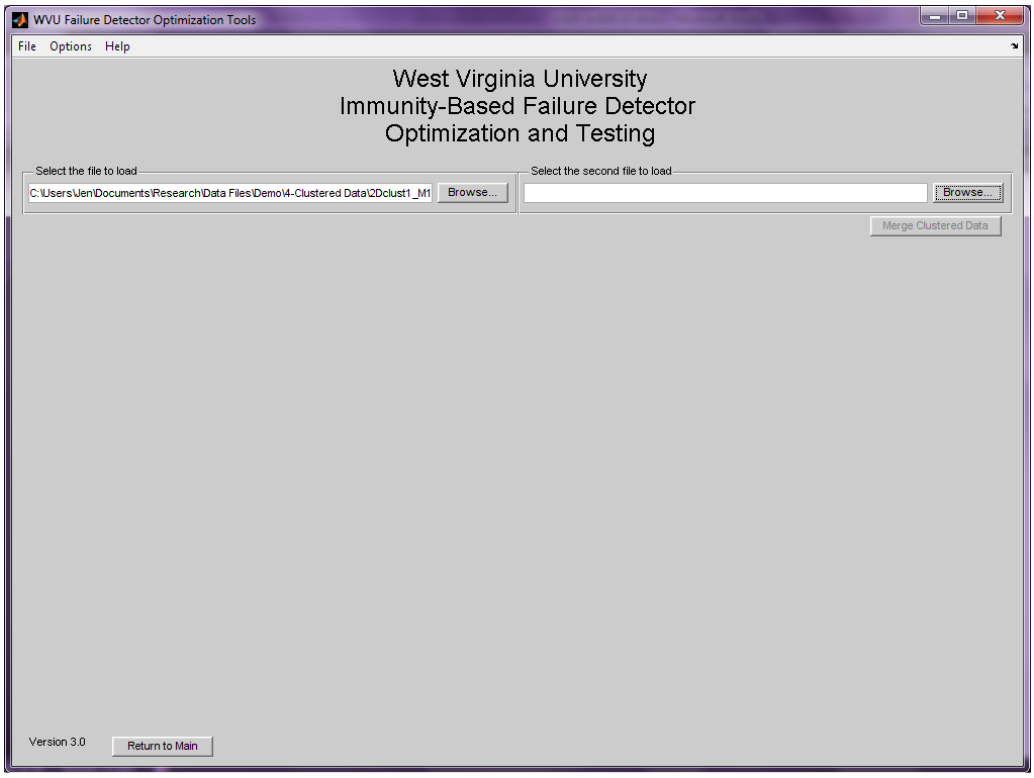

Figure B.145—Merge Clusters Menu

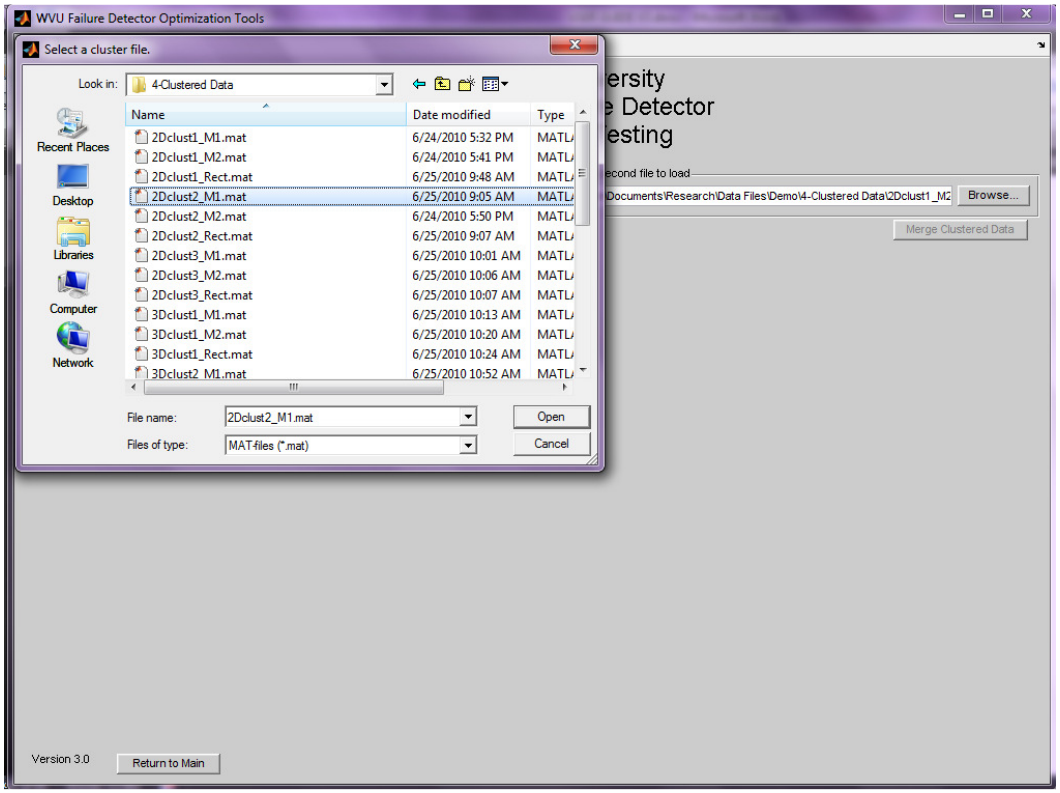

Figure B.146—Merge Clusters Browse

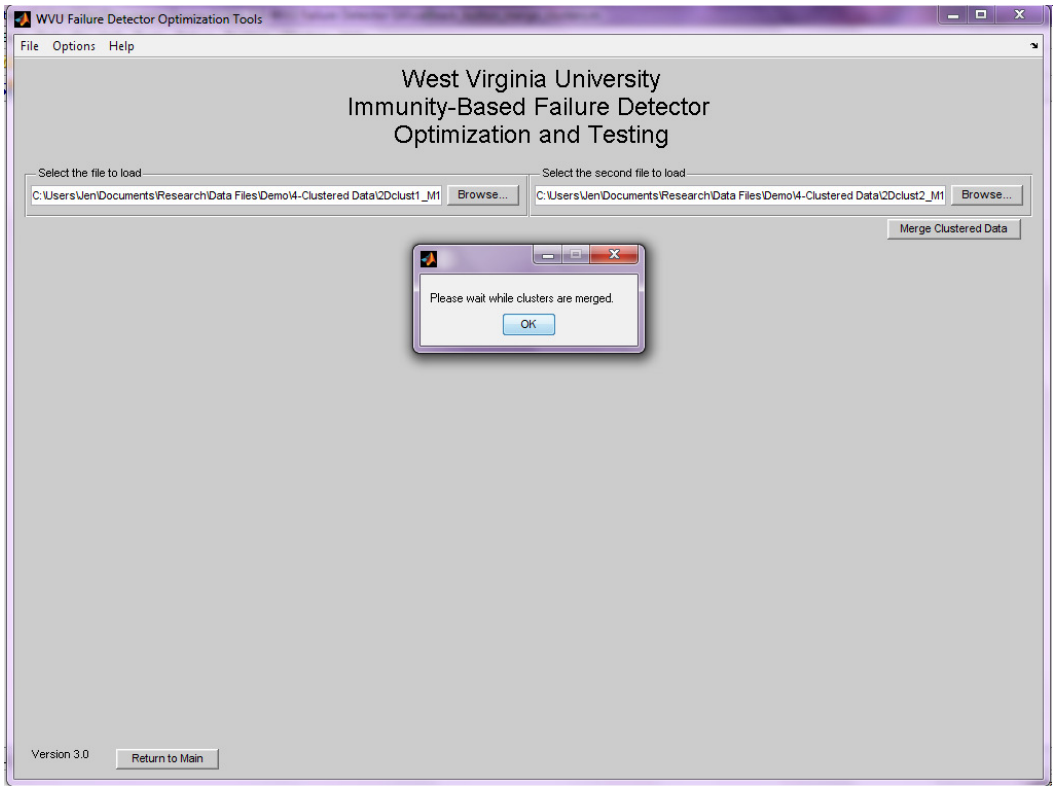

Figure B.147—Merge Clusters in Progress

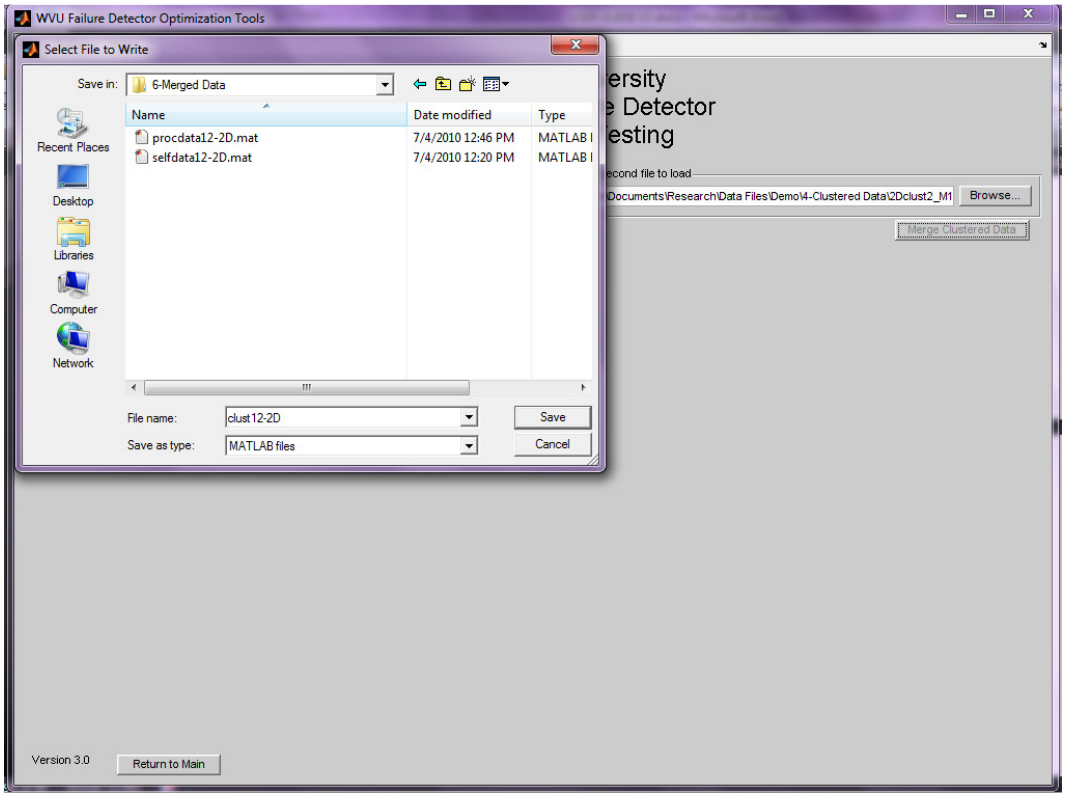

Figure B.148—Merge Clusters Save Dialog