Clustering Analysis of Nuclear Proliferation Resistance Measures

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Zachary Kyle Jankovsky B.S.

Graduate Program in Nuclear Engineering

The Ohio State University

2014

Master's Examination Committee:

Dr. Richard Denning, Advisor

Dr. Tunc Aldemir

Abstract

The development of Generation IV nuclear fuel cycles has led to a renewed interest in nuclear proliferation studies for fast-spectrum reactors and recycling systems. One such system, conceptualized specifically for the purpose of proliferation studies, is the Example Sodium-cooled Fast Reactor (ESFR). The ESFR is an actinide burner, capable of using Light Water Reactor (LWR) Spent Fuel (SF) as an input. A reprocessing plant exists on-site, which separates plutonium and uranium from other actinides and fission products, and forms new ESFR fuel assemblies. This production of relatively-pure plutonium presents a proliferation concern.

PRCALC was developed at Brookhaven National Laboratory to represent material diversion from an ESFR system as a Markov model, with absorbing states of detection, technical failure, and success from the point of view of the would-be proliferator. Scenarios are created by varying diversion targets, diversion rates, and safeguards conditions on various elements of the fuel cycle.

Since there are relatively large uncertainties on the scenario constituents, this thesis focuses on a methodology to create and analyze a large number of PRCALC scenarios which may result from a sensitivity study. These scenarios are clustered using the mean-shift and *k*-means algorithms, in addition to an adaptive-bandwidth variation of mean-shift. Clustering is performed to allow analysis of important inputs without

examining each of potentially thousands of scenarios by hand. A simple case study was performed as an example of how an analyst may use the methodology in a real-world application.

A tool is developed and described to mechanize much of the scenario creation, clustering, and analysis processes. This tool is called Ohio State University Proliferation Resistance (OSUPR) and is written in MATLAB as a companion tool to PRCALC. A sample set of scenarios is run from start to finish using OSUPR to show its use to an analyst. Some trends can be seen in the sample set that would be of interest to an analyst.

Dedicated to my family: here, gone, and not here yet

and to

the music makers, the dreamers of dreams, the movers and shakers

Acknowledgements

I am indebted to:

- Drs. Richard Denning and Tunc Aldemir for their guidance not only in regards to this research, but through the entire course of my graduate studies
- Drs. Lap-Yan Cheng and Meng Yue of Brookhaven National Laboratory for their cooperation on this project
- The Department of Energy Office of Nuclear Energy's Nuclear Energy University Programs for the funding that made this work possible
- My fellow students for their support and challenges
- Dr. John Engdahl for my first introduction to nuclear topics
- My brother Pete for setting the bar and pushing me to try to raise it
- My parents for always being there
- My wonderful and patient wife Jenni for staying with me despite incessant rambling about trivia, nuclear energy, and goats

Vita

September 1988 .....Born, Lima, Ohio

2012.....B.S. Mechanical Engineering, Bradley University

2012 to present .....Graduate Research Associate, Nuclear Engineering Program, The Ohio State University

Publications

Jankovsky, Zachary, Zamalieva, Daniya, Yilmaz, Alper, Denning, Richard, and Aldemir, Tunc. "A Clustering Analysis of Probabilistic Proliferation Resistance Measures in an Example Nuclear Fuel System." ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis. (2013)

Jankovsky, Zachary, Zamalieva, Daniya, Denning, Richard, Yilmaz, Alper, and Aldemir, Tunc. "A Comparison of Various Clustering Schemes for Proliferation Resistance Measures." Transactions - American Nuclear Society, 109, Part 1. (2013): 261

Fields of Study

Major Field: Nuclear Engineering

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

In this chapter, a brief history of nuclear fuel reprocessing and an overview of its proliferation implications are given (1.1), followed by a description of the specific system considered (1.2). Having introduced the problem, the goals of this thesis are stated (1.3). Finally, the organization of this thesis is established (1.4).

1.1 Nuclear Reprocessing & Proliferation

The terrible prospect of a nuclear war has been evident to the world since the atomic bombings of Hiroshima and Nagasaki in August, 1945. With deaths on the order of a quarter million people from only 2 weapons, the power of a fission weapon was seen quite clearly in human terms. The Cold War arms race put the world even more on edge as the United States of America (USA) and the Union of Soviet Socialist Republics (USSR) each amassed tens of thousands of much more powerful thermonuclear weapons. By 1968, three more nations had joined the "Nuclear Club" and a treaty was crafted to address the risk that the spread of nuclear weapons posed to all people.

Since the Treaty on the Non-Proliferation of Nuclear Weapons (NPT) went into force in 1970, proliferation concerns have been explicitly considered in decision-making for the development of new technologies and for the construction of nuclear facilities worldwide. Nuclear fuel reprocessing is of particular concern with regards to proliferation. Typical light water reactor (LWR) fresh fuel is ~5% U-235 and ~95% U-

238. By the end of its life in the reactor, the proportion of U-235 is substantially less, some of the U-238 having been transformed into various actinides, and lighter fission products from the fission of both U-235 and plutonium isotopes having been created. Reprocessing typically separates fission products from heavier elements such as uranium and plutonium, and then separates uranium from plutonium. The separated plutonium can be used as fuel in the form of a mixed oxide (uranium/plutonium oxide) and recycled back into the reactor to obtain more energy. The plutonium could, however, be potentially diverted for illicit use.

US Presidents Ford and Carter both cited proliferation concerns in suspending proposals for commercial fuel reprocessing in the USA. Quoting Ford's policy statement, "No single nation, not even the United States, can realistically hope - by itself - to control effectively the spread of reprocessing technology and the resulting availability of plutonium." However, it was felt that the U.S. should be a role model for all countries in foregoing the potential economic and resource benefits of plutonium recycle in order to minimize the likelihood of proliferating nuclear weapon technology and capability. Both presidents agreed that the benefits and proliferation risks of reprocessing ought to be re-considered from time to time.

Because France and Japan have limited fossil resources, they have committed to significant nuclear energy programs, which include the capability for plutonium recycling. Inexpensive uranium resources are limited worldwide. Many of the Generation IV nuclear plants that are being designed for the future would require some form of reprocessing to enable breeding to extend the capability of our uranium and thorium resources. In addition, there has been significant attention paid recently to the potential

benefits of recycling actinides from LWR fuel in order to reduce the long term impact on geologic waste repositories.

1.2 System Under Consideration

The hypothetical fuel system analyzed in this work mates a reprocessing and fabrication plant to a population of sodium-cooled fast spectrum reactors used for both power production and actinide burning. The concept is named the Example Sodium-Cooled Fast Reactor (ESFR) [1]. For the analysis of proliferation resistance, it is assumed that the facility would be operated within a country that is a non-nuclear weapons state (NNWS), which is a signatory to the NPT under International Atomic Energy Agency (IAEA) safeguards. It is assumed that the operator of the facility wishes to covertly divert material from some part (or multiple parts) of the facility with the end goal of obtaining one or more significant quantities (SQs) of fissile material for eventual use in a nuclear explosive device.

An important quantity in safeguards is the significant quantity (SQ). An SQ is defined [2] as: "the approximate amount of nuclear material for which the possibility of manufacturing a nuclear explosive device cannot be excluded." An SQ equivalent is the amount of material of non-weapons-usable form that could be converted and manufactured into an SQ of weapons-usable material. An SQ of plutonium is 8 kg of Pu containing less than 20% Pu-238. An SQ of uranium is 25 kg of U-235 with enrichment over 20% U-235.

Each step in the reprocessing plant contains different nuclear materials flowing at different rates, each with different applicable safeguards. For example, spent fuel pellets

from a LWR that are being chopped in preparation for reprocessing are quite a different material compared to freshly separated plutonium stream prior to the fabrication of fuel for the actinide burner reactor. Each different material requires different physical and chemical processing towards their end use. In the analysis of the system from a proliferation viewpoint, the parts of the fuel system that are targeted are varied in each scenario, as are the material diversion rates from each target. The safeguard conditions are also varied in some scenarios, to represent an actor circumventing the IAEA safeguards. For example, the actor may coerce inspectors into returning a false negative result in an inspection report or may convincingly replace suspicious surveillance footage with benign footage.

1.3 Thesis Goal

In order to gauge the sensitivity of various proliferation measures to inputs, many thousands of scenarios may be created. The aim of this work is to develop a methodology to analyze a large number of diversion scenarios in order to identify the most attractive strategies for the actor. Clustering is used to group scenarios by their results, because manual inspection would be infeasible for large data sets. This knowledge may then be applied to allocating safeguarding resources effectively. The cluster or clusters with properties most favorable to the actor are compared to the data at large to identify the choices of targets, rates or safeguard circumventions that lead to the largest changes in outcomes. An existing tool called PRCALC (see 3.1) is used to estimate proliferation resistance measures to be used in the clustering analysis. The new tool created will use

PRCALC as a module and will be referred to in this work as Ohio State University Proliferation Resistance (OSUPR).

1.4 Thesis Organization

This thesis is divided into 6 chapters. Chapter 2, Background, describes the NPT as well as IAEA safeguards more fully. Previous work in the areas of proliferation resistance and clustering is examined with regards to its relevance to this thesis. Chapter 3 first describes PRCALC in terms of its development, capabilities, operation, and previous work published using it as an analysis tool. The process and rationale of scenario creation is then explained, as well as the running of PRCALC. Processing done to prepare data for clustering is also included in this chapter. Chapter 4 describes the various clustering algorithms to be used.

Results from PRCALC and clustering are discussed in Chapter 5. The results of different clustering approaches are compared to each other and analyzed to determine the inputs with large effects on cluster assignments. The development and operation of the OSUPR tool is also described in this chapter with screenshots, and a walk-through explanation of its use are included. Chapter 6 presents the conclusions that can be drawn from the work, as well as opportunities for future research.

Chapter 2: Background

This chapter describes the terms, membership, and events associated with the NPT and nuclear proliferation in general (2.1). The importance and basic structure of IAEA safeguards is given (2.2), and previous work in the areas of proliferation resistance and clustering is examined (2.3).

2.1 Non-Proliferation Treaty

The NPT entered into effect in 1970 [3] amidst a nuclear arms race between the USA and the USSR, whose nuclear arsenals by that time numbered in the tens of thousands of warheads. At attempt was made to balance concerns with the spread of nuclear weapons with the rights of all nations to enjoy the benefits of peaceful nuclear technology. The world's nations were divided into 2 categories: nuclear weapon states (NWS), who had detonated a nuclear explosive device by January 1, 1967, and non-nuclear weapon states (NNWS). NWS recognized by the treaty are: China, France, the USSR (now Russia), the United Kingdom, and the USA.

By ratifying the treaty:

- NWS promise not to assist or encourage a NNWS in any attempt to acquire nuclear weapons
- NNWS promise not to attempt to acquire nuclear weapons

- NNWS consent to IAEA safeguards on all nuclear material under their control to ensure compliance with the treaty
- NWS agreed not to transfer fissionable material or related equipment to NNWS unless under safeguards
- NWS agree to cooperate with NNWS in sharing information and technology that will assist the NNWS in peaceful use of nuclear energy
- All nations agree to pursue total nuclear disarmament
- A nation may withdraw from the treaty, having given the United Nations 3 months' notice, if continued membership "threatens the supreme interests of the nation"

A small number of nations are not currently parties to the NPT: India, Pakistan, Israel, North Korea, and South Sudan. South Sudan is a newly-formed nation that is not generally suspected to have nuclear weapon ambitions. North Korea was a party to the NPT from 1985 to 2003. India, Pakistan and Israel have never been members. India and Pakistan have both conducted nuclear tests [4], as has North Korea [5]. Some facilities in Israel are safeguarded [6].

Common ways a nation can be found out of compliance with the NPT are failing to declare a nuclear facility or interfering with safeguards. For example, at various times throughout its membership North Korea denied access to IAEA inspectors and was found to be non-compliant [7]. The IAEA found Iran to be non-compliant in 2005 [8] and 2006 [9] for its failure to declare certain nuclear facilities and other actions that caused an "absence of confidence that Iran's nuclear programme is exclusively for peaceful

purposes". These non-compliances are worrying to the international community, as they may be taken as signs of nefarious actions being committed by the state involved.

2.2 IAEA Safeguards

Safeguards that are put in place as part of a nation's membership in the NPT are aimed at declared facilities. Different safeguards approaches may include material accountancy, surveillance, and containment [10]. Accountancy is composed of inspections and measurements of materials, either as a matter of routine or in response to special requests. Often, a comprehensive account of materials is made once per year and smaller-scale inspections are made more often. Surveillance is generally in place continuously, and data from cameras and other monitoring equipment is reviewed either continually or periodically. Containment measures, which include tamper-evident seals on sensitive process and safeguards equipment, are typically checked at some regular interval.

A member nation may also be encouraged to enact an Additional Protocol (AP) to ensure compliance. AP safeguards are in place to rule out the existence of undeclared nuclear facilities. As a result, AP agreements typically allow inspectors to travel freely throughout the nation and within any facility that may house nuclear material or activities. Inspectors can visit most sites with 24 hours of advance notice and some with only two hours of notice, making it much more difficult for the host nation to effectively conceal a nuclear weapons program.

2.3 Previous Work

To place this thesis within the literature, a short review is made of existing nuclear proliferation resistance assessment tools and applications of clustering. This thesis is believed to be the first to use clustering analysis in the area of nuclear nonproliferation.

2.3.1 Nuclear Non-Proliferation

There is a significant body of work relating to nuclear non-proliferation, both on a fuel cycle level and on a per-component basis. Analysis is often focused on categorizing materials and preventing diversion of attractive materials [11]. The development of Generation IV technology in particular has led to a number of proliferation resistance models for fast-spectrum reactor systems with recycling [12]. One such model is PRCALC, which is described in Chapter 3 and is used throughout this thesis. PRCALC models material diversion, transportation, and transformation into a weapons-usable form. PRCALC represents diversion from the fuel cycle as a Markov model, in which the absorbing states are detection, technical failure and success.

Another recent tool is the Proliferation Resistance Analysis and Evaluation Tool for Observed Risk, or PRAETOR, developed at Texas A&M University [13]. PRAETOR is built on the multi-attribute utility (MAU) methodology, often used in business decision analysis to identify optimal choices when numerous attributes of alternatives are involved in the optimization. PRAETOR models material diversion, transportation, transformation, and weaponization. Each of 63 attributes has been given a weight by expert elicitation.

The tool calculates and outputs a single measure of proliferation resistance for a given scenario.

2.3.2 Clustering

Cluster analysis has been applied to various problems of pattern recognition, growing rapidly in popularity in the 1970s [14]. The mean-shift algorithm in particular [15], which identifies cluster centroids as points of maximum density (see Chapter 4), may be used to automatically track an object between frames with good accuracy. *K*-means clustering is often used as an initial sorting step for numerical data due to its fast running speed, but for in-depth analysis *k*-means often must be combined with other techniques [16].

With the dramatic increase in data processing ability over the past decades, data sets have grown in size, creating more demand for data reduction techniques, including cluster analysis. Within nuclear engineering, there have been a few examples of the use of clustering to classify scenarios in probabilistic risk assessment. One such example applies the technique to a steam generator tube rupture event in order to group similar accident progressions [17]. Sixty scenarios were clustered using the fuzzy c-means algorithm into 4 "classes" of events, one of which was not anticipated.

The dissertation of Diego Mandelli [18] applies clustering more broadly to dynamic probabilistic risk assessment and gives a number of example nuclear engineering analyses including aircraft crash, station blackout, and pump seal leakage. Various clustering algorithms were compared, and mean-shift was chosen for the analyses. This thesis uses the MATLAB mean-shift implementation that is described in

the Mandelli dissertation and was previously applied directly to Level 2 and Level 3 Probabilistic Risk Assessment in the dissertation of Douglas Osborn [19].

Chapter 3: PRCALC and Its Adaptation for Mechanistic Scenario Generation

This chapter describes PRCALC and how it is used. The target fuel system and general method of analysis are first explained, and an overview is given of previous work accomplished using PRCALC (3.1). The creation of PRCALC scenarios for this thesis is then fully described (3.2), as is the method of running PRCALC (3.3). Some data processing was required after PRCALC and before clustering, and is explained in the final section (3.4).

3.1 PRCALC Development, Operation & Previous Studies

PRCALC is a software tool developed at Brookhaven National Laboratory [20] to estimate measures related to proliferation resistance. Although PRCALC contains data for a number of potential fuel cycles, its main focus is on the hypothetical ESFR fuel system, with the main locations shown in Figure 1. LWR spent fuel (SF) and transuranic (TRU) elements are used as fuel sources for this fast-neutron spectrum, actinide-burner reactor. Reprocessing of LWR spent fuel and ESFR spent fuel and recycling of molten salt are accomplished on-site. The goal of this system is to close the nuclear fuel cycle, using much more of the fertile material present in LWR fuel than a once-through LWR reactor, as well as to reduce the amount of long half-life waste that would ultimately be sent to a geologic repository.

Figure 2 shows part of a Markov chain of the fuel system with regards to material diversion. A Markov chain consists of multiple states that flow into each other with specific transition probabilities. The sum of transition probabilities from an initial state to others (including itself) is one. An absorbing state is one that transitions only to itself, and thus acts as a sink. In a Markov process transitions between states depend only on current conditions and not on the prior history. A more thorough explanation is given by A. A. Markov [21].

An attempt may be made from any declared stage of the fuel system to divert material to a clandestine transportation stage. This may be thwarted by IAEA safeguards or by internal barriers such as the material being too thermally or radioactively "hot" to be transported. From there the various different material types are converted to a weapons-usable form in a clandestine chemical plant. At this point detection is no longer a concern, but technical failure (defined later in this section) is a significant risk. This is because the chemical conversion is done without internationally-sanctioned support, and so the actor may be lacking in equipment or expertise.

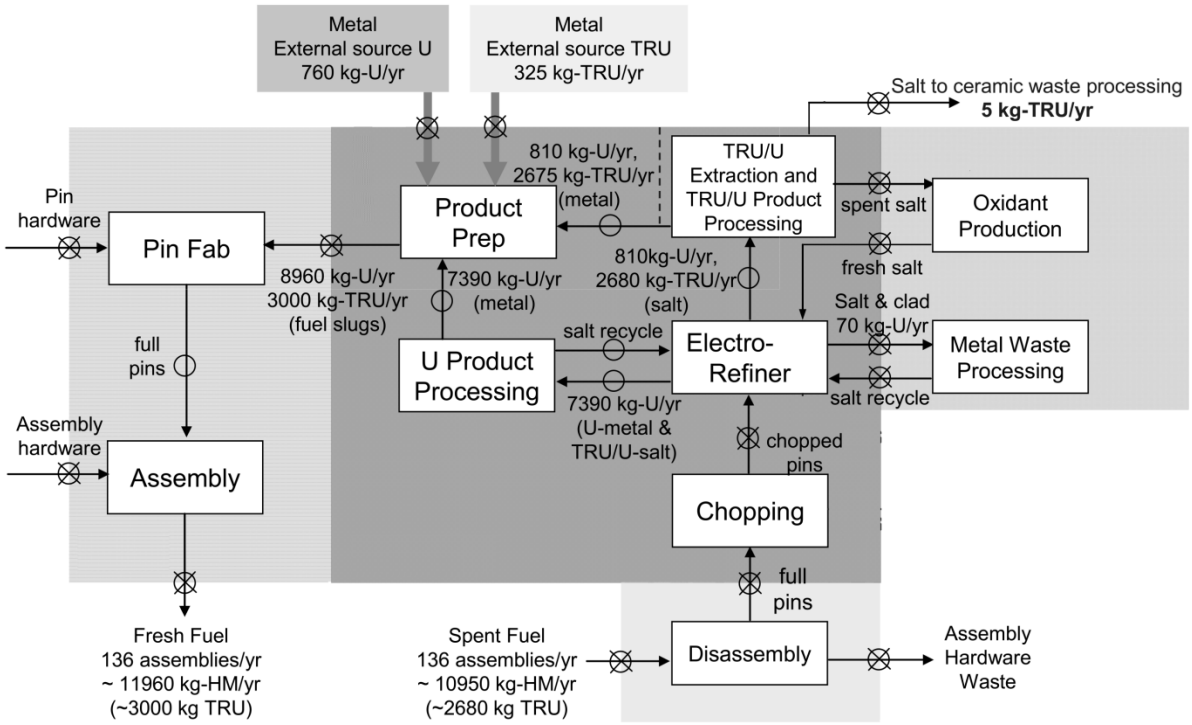


Figure 1. ESFR Fuel System Schematic [22]

PRCALC outputs the following measures:

- Probability of Detection (DP), the likelihood that a diversion attempt will be detected, which will invariably lead to the attempt being stopped
- Probability of Failure (PF), the likelihood that a diversion and clandestine conversion attempt will be thwarted by difficulties in obtaining material or converting it into a weapons-usable form
- Probability of Success (PS), the likelihood that a diversion attempt will result in the actor obtaining an SQ equivalent and converting it into a weapons-usable form
- Proliferation Time (PT), the time required in weeks to obtain an SQ equivalent of material and convert it to a weapons-usable form

- Material Type (MT), the mass-averaged attractiveness of material being diverted

PRCALC was developed to address proliferation resistance and physical protection (PR&PP) in support of Generation IV nuclear technology. The decision was made to use a Markov model, in which the analysis moves between states according to calculated transition rates. Because failure, detection and success are absorbing states of the Markov model, their probabilities sum to 1 for any given scenario. This also indicates that they are correlated, which will be investigated in a later section.

The transition rate to each detection state depends on the safeguards in place and the diversion rate from the stage in question. If there is no diversion, there is no chance of detection. Likewise, the transition rate to each failure state depends on presumed internal barriers in the declared stages and technical difficulties in the clandestine stages. The diversion rate corresponds to the fraction of the material in any given stage that yields a similar detection transition rate and is represented by a lowercase sigma. Therefore, a diversion rate of 2σ from some stage indicates that the material diversion rate is such that the detection transition rate for that stage is twice that of a stage with diversion rate of 1σ.

The key to PRCALC is the calculation of characteristic times and associated rates of interactions. Each safeguard has a characteristic time to detect an anomaly, and a time to confirm that the anomaly was caused by diversion and is not a false alarm. The detection rate is given by Equation 1

$$r_i = \sum_{j=1}^{n} \frac{1}{T_{D(i,j)}} \tag{1}$$

where r_i is the total detection rate for stage i, n is the number of safeguard approaches for stage i, and $T_{D(i,j)}$ is the total detection time for safeguard j on stage i. This greatly simplifies the Markov model, as opposed to including a detection state and transition rate for each safeguard approach at each stage of the fuel cycle. The total failure rate for each stage is handled in a similar fashion, being a combination of multiple failure mechanisms.

The safeguard approaches included in PRCALC are listed in Table 1. Safeguards are applied to each stage element as appropriate. The Category is given by the numeral, and the individual approach within the Category by the letter. The categories are:

1. Audit of various nuclear material accounting records or reports
2. Material verification such as physical inventory verification (PIV) of all nuclear material in a nuclear energy system
3. Surveillance and monitoring
4. Containment

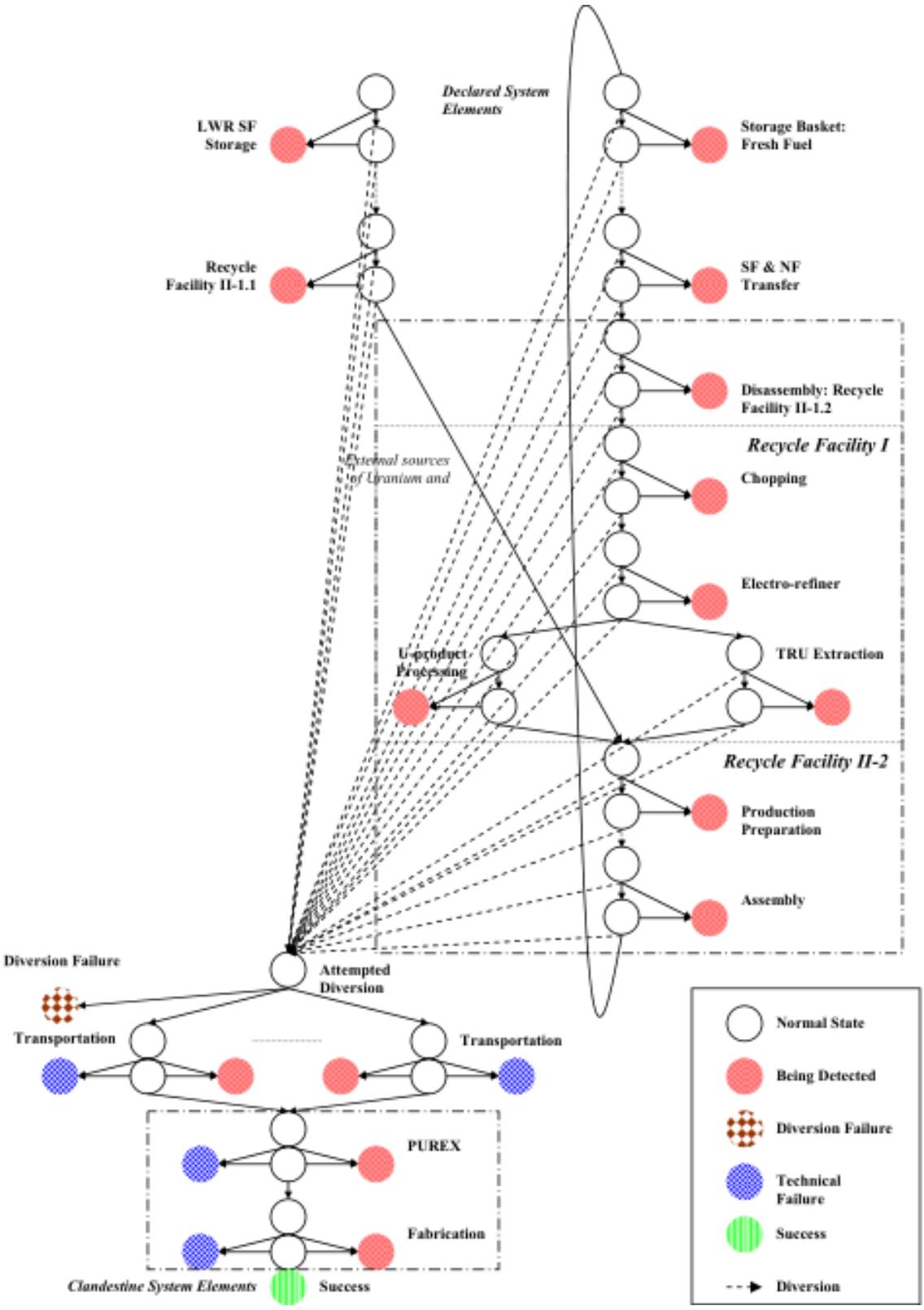


Figure 2. ESFR Fuel System Markov Model [23]

1A	Comparison of records of power history and shutdown dates at International Atomic Energy Agency (IAEA) inspection
1B	comparison of inspector and system element book inventory listing (BIL) at IAEA inspection
1C	Comparison of fresh fuel shipment and receipt inventory change reports (ICRs) when the fresh fuel is shipped
1D	Comparison of outstanding state transfer documents (STDs) with subsequent receipt and shipment ICRs from reactor when the transfer occurs from reactor
1E	Comparison of (SF) shipment ICRs and STDs with crane movement records (CMR) when SF is moved
1F	Comparison of SF shipment and receipt ICRs when SF is shipped
1G	Comparison of state system for accountancy and control reports
1H	Comparison of records of power history with reported nuclear material production and loss at IAEA inspection
2A	Comparison of system BIL with item count and o/r mass
2B	Visual inspection of fresh fuel/SF or other types of material
2C	Comparison of fuel element identification number with system element BIL
2D	Comparison of number of assemblies exhibiting Cerenkov glow with number of assemblies reported recently irradiated using different devices
2E	Comparison of measured value of fresh fuel enrichment with BIL value
2F	Comparison of total assembly ^{235}U content or plutonium with value as measured by active neutron method and plutonium assay using Pu/Cm ratio technique via destructive analysis (DA)
2G	Comparison of measured burnup with burnup value on system element BIL
2H	Gamma spectroscopy of spent fuel via nondestructive analysis (NDA) that indicates whether the fuel has been irradiated
3A	Comparison of surveillance record in SF pool area and other system elements with the CMR for casks of SF or fresh fuel
3B	Review of surveillance in reactor area for undeclared shutdown and head removal or unusual activity during refueling
3C	Review of surveillance record for shipment or receipt of fuel during refueling
3D	Nearly real-time accounting and monitoring system, including using neutron counting to determine the amount of transferred material to be protected
3E	Assay of Pu via Pu/Cm ratio and NDA
4A	Analysis of seal on reactor vessel
4B	Analysis of seal of spent-/fresh-fuel shipping cask
4C	Analysis of seal of safeguards equipment

Table 1. PRCALC Safeguard Approaches [23]

A clarification must be made in regards to the terms “stage” and “stage element”. In the operation of PRCALC each stage element belongs to a stage and there are often multiple elements to each stage. A mapping of stages and stage elements is given as Table 2. Safeguard approaches and diversion targets are selected in PRCALC on a stage

element level. When discussing the Markov model in general, the term stage is used more generally.

Stage I	LWR SF Storage
	Storage Basket: Fresh Fuel
Stage II	LWR Transfer Port
	ESFR Reactor
Stage III	LWR Transfer
	Storage Basket: Spent Fuel
Stage IV	ESFR Transfer Port
Stage V	ESFR Transfer
Stage VI	Staging/Washing Area
Stage VII	Staging/Washing Transfer Port
Stage VIII	Staging/Washing Transfer
Stage IX	SF & NF Storage Cell
Stage X	SF & NF Transfer Port
Stage XI	SF & NF Transfer
Stage XII	LWR SF Disassembly
	ESFR SF Disassembly
Stage XIII	Chopping
Stage XIV	Electro-refiner
Stage XV	U-product Processing
	TRU Extraction
Stage XVI	Product Preparation
Stage XVII	Pin Fabrication
Stage XVIII	Assembly
Stage XIX	Clandestine Transportation
Stage XX	Clandestine Chemical Conversion
Stage XXI	Clandestine Chemical Separation

Table 2. ESFR Fuel System Stages & Elements

Recent work with PRCALC focuses on an actor who switches strategies once the covert diversion attempt has reached some high probability of detection but before it can be terminated [24]. This is possible due to the time required, once diversion is detected, for the international community to take action to stop it. A representative Markov model

is shown as Figure 3. Once diversion reaches a threshold Probability of Detection at any System Element i, the actor enters a Breakout scenario and begins to divert material as quickly as possible from any other System Element j. The same technical failure mechanisms are present for this new mode of diversion, but detection is not considered during Breakout. Instead, it is replaced by Being Terminated, a state that represents the international community taking action to end all diversion attempts. As modeled, switching strategies can only increase the actor's chance of success, and so the analysis focuses on reducing its impact.

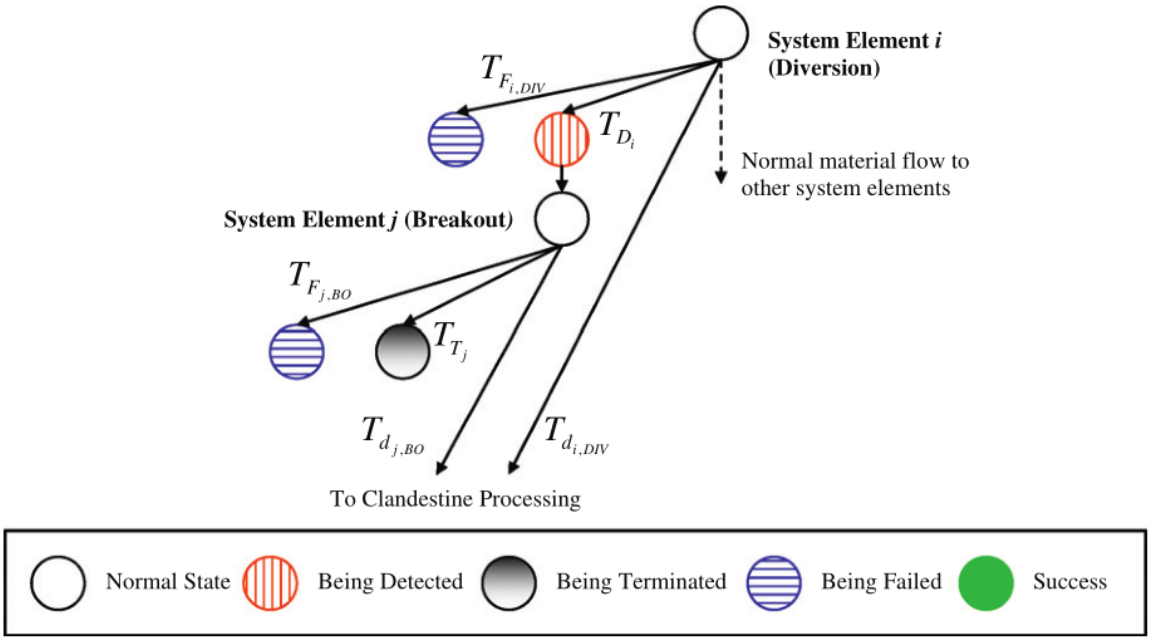


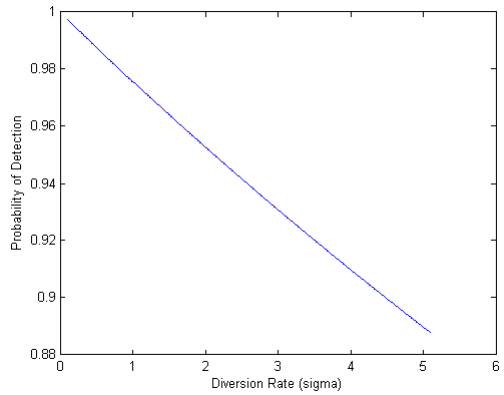
Figure 3. Strategy Switching Diversion Scenario [24]

3.2 PRCALC Scenario Creation

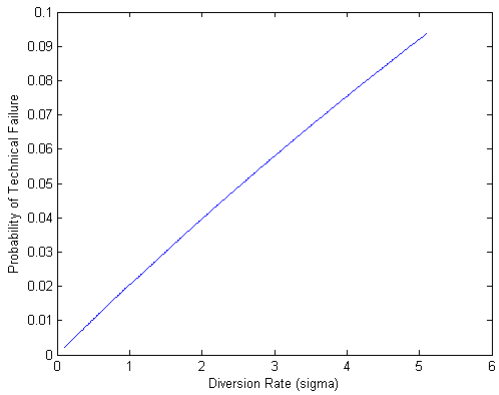
As originally written, the PRCALC program allows the user to create one scenario at a time. This is most useful when the user has a specific set of conditions in

mind to be used for analysis. When seeking to cover a large amount of the parameter space, however, a method is desired to create a large number of scenarios quickly. Two such methods were explored in this work. All scenarios were based on a default scenario file that is included with PRCALC. The structure of the scenario file was examined and documented, and input parameters were directly manipulated using MATLAB rather than the PRCALC interface.

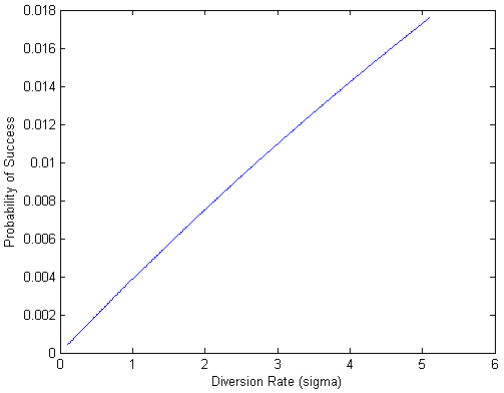
The first method was to vary an input parameter by increments. A target and set of safeguards were chosen, and the diversion rate was varied by some small increment to gauge the sensitivity of the outputs to the diversion rate. A study was done for 500 scenarios using the Electro-refiner as the target with default safeguards. The diversion rate was varied from 0.11σ to 5.1σ in steps of 0.01σ. The results are shown in Fig. 4. It can be seen that DP, PF, and PS (Figure 4 (a, b, and c)) vary roughly linearly with diversion rate. PT and diversion rate have an inverse relationship (Figure 4 (d)), and a doubling of the diversion rate will nearly halve the total time required for material gathering and processing.



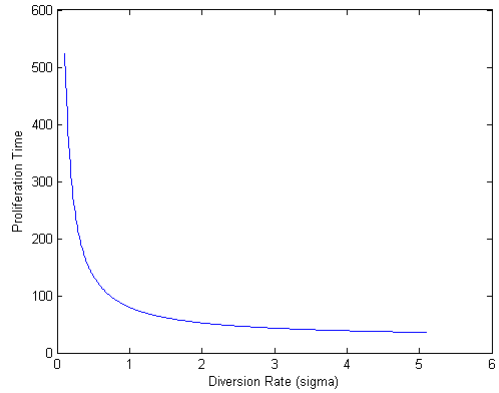
(a) DP vs Diversion Rate



(b) PF vs Diversion Rate



(c) PS vs Diversion Rate



(d) PT vs Diversion Rate

Figure 4. PRCALC Outputs vs Diversion Rate

To create larger and more complex data sets a method was used which varied input parameters through every combination of the chosen targets, diversion rates, and safeguard conditions. A list of potential targets was created based on interest and their perceived attractiveness. A list of diversion rates was also created, often including 0σ to represent no diversion attempt from a particular stage. Finally, a list of safeguard conditions was created, usually including the default set of safeguards for each stage as built into the fuel system model in PRCALC. Safeguard condition modifications often

involved the removal of entire categories, such as surveillance and monitoring or physical inventory verification. A matrix was created for each set, its length being the number of scenarios in the set as calculated by Equation 2

$$N_{scen} = r^{N_{stage}} * N_{SG} \quad (2)$$

where N_{scen} is the total number of scenarios, N_{stage} is the number of target stages, r is the number of possible diversion rates, and N_{SG} is the number of safeguard conditions.

The first large set created used 7 target stage elements, 4 possible diversion rates, and 4 safeguard conditions. The targets are listed in Table 3. The diversion rates were 0σ, 1σ, 2σ, and 4σ. The 4 safeguard conditions were the default safeguards in place in the PRCALC model, all Category 2 safeguards removed on all stage elements, all Category 3 safeguards removed, and all Category 4 safeguards removed. This yielded 65,536 scenarios. Because one of the possible diversion rates was 0σ, 4 scenarios were created where the diversion rate for every target was 0σ. These were eliminated from the set, for a final size of 65,532 scenarios. This was referred to as the Deep Set, because many diversion rates were considered.

Target Number	Stage Element Name
1	TRU Extraction
2	Electro-refiner
3	Pin Fabrication
4	Storage Basket: Fresh Fuel
5	ESFR SF Disassembly
6	Chopping
7	LWR SF Storage

Table 3. Deep Set Target Stage Elements

After analysis of the results of the Deep Set, it was seen that DP, PS, and PF were highly correlated (see 3.4.1). To ensure that this was not an artifact of the targets selected, a set was created with many more targets and fewer diversion rate options. The target stage elements are shown in Table 4. The diversion rate options were 0σ and 2σ, and the safeguard conditions were the same as in the Deep Set. This yielded 262,144 scenarios, 4 of which had a diversion rate of 0σ for every target. The final size was 262,140 scenarios, and this was referred to as the Wide set.

Target Number	Stage Element Name
1	LWR Spent Fuel Storage
2	Storage Basket: Fresh Fuel
3	LWR Transfer Port
4	LWR Transfer
5	Storage Basket: Spent Fuel
6	Staging/Washing Area
7	SF & NF Storage Cell
8	SF & NF Transfer
9	LWR SF Disassembly
10	ESFR SF Disassembly
11	Chopping
12	Electro-refiner
13	TRU Extraction
14	Product Preparation
15	Pin Fabrication
16	Assembly

Table 4. Wide Set Target Stage Elements

3.3 Running PRCALC Scenarios

Once scenarios were created, they had to be fed into the PRCALC engine. PRCALC includes an option to run a batch of input files. The names and paths of scenario files were added to a PRCALC-compatible batch file, which could be run with

as much effort as a single scenario in the PRCALC interface. By default PRCALC displays a chart of relevant outputs for each scenario after it is finished (see Figure 5), as well as a progress bar for each scenario as it is running (see Figure 6). A progress bar for each scenario was not useful for a batch, as it did not indicate the progress of the entire batch. The output charts were also not applicable to a large batch, as windows were created quickly and were not labeled by scenario. These PRCALC features were disabled for this work.

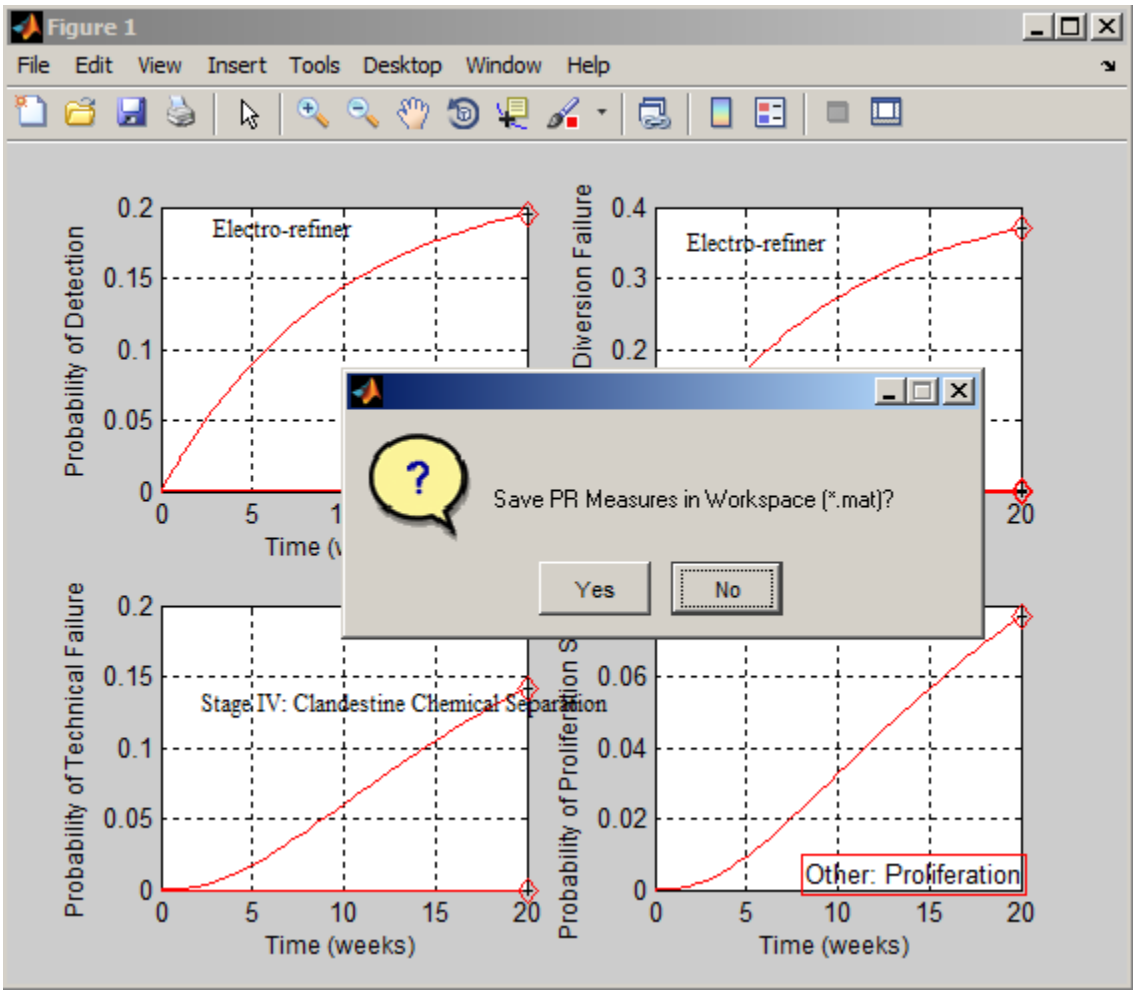


Figure 5. PRCALC Output Chart

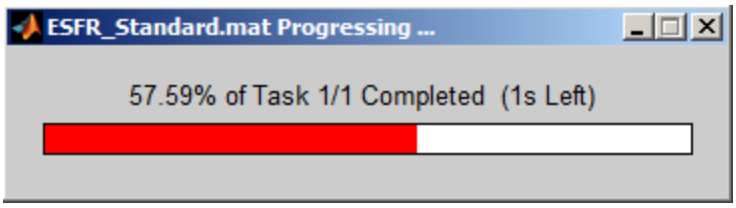


Figure 6. PRCALC Scenario Progress Bar

PRCALC runs on one computing thread at a time. When it was written circa 2005, multi-core personal computers were not common in most office environments. Today, the potential for parallel processing cannot be ignored. Rather than change the structure of PRCALC, it was decided to run multiple instances of PRCALC. When batching scenarios for parallel processing, they may be divided into a number of batches equal to the number of processing threads to be used as shown in Figure 7. Multiple instances of MATLAB and PRCALC are then opened, and the batches are loaded sequentially into each instance. The operating system automatically allocates each instance of PRCALC to a processing thread for efficient use of the processor.

To determine the performance gains from using multiple instances of PRCALC, a set of 6,558 scenarios was created with a variety of rates, stages, and safeguard conditions. Scenarios and information files were first created using a method that does not allow for the use of multiple cores. Time was recorded from when PRCALC started running until it had finished and the results had been compiled into a single output file. All experiments were performed on the same HP xw6400 workstation, which was restarted in between experiments. The results are shown in Figure 8. A trend line is given in black to represent a perfect inverse power law ($y=x^{-1}$) relationship. It can be seen that with a doubling of cores, there is nearly a halving of the time required. This was

expected, because each scenario is run independently in PRCALC. It is likely the task of compiling the results of multiple cores that causes the time to not exactly halve with each doubling of cores. In all, running PRCALC with 8 cores required 15% of the wall clock time when compared to 1 core.

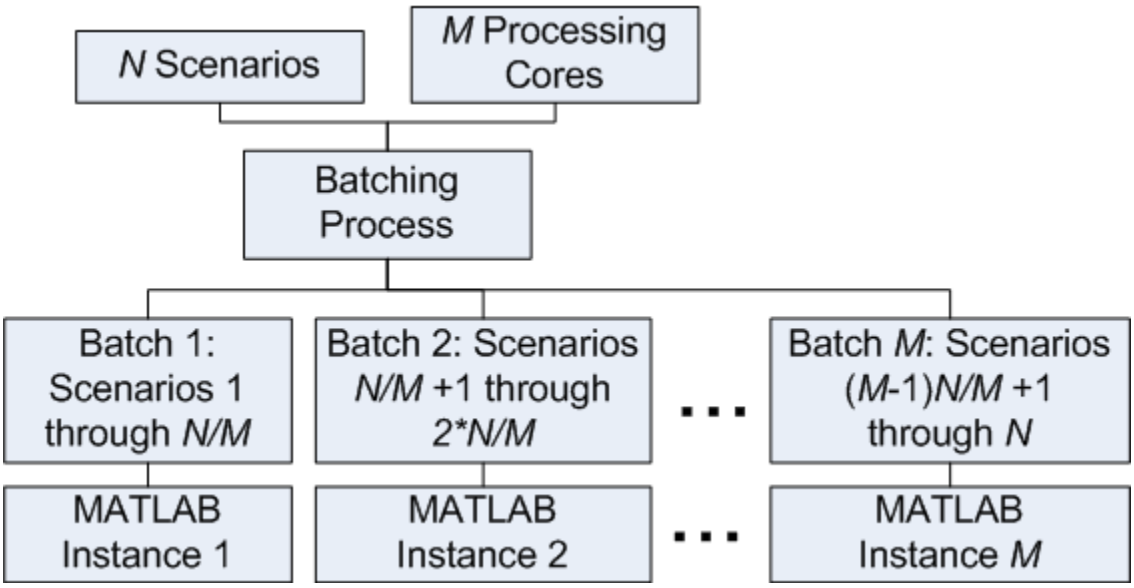


Figure 7. PRCALC Batching for Parallel Operation

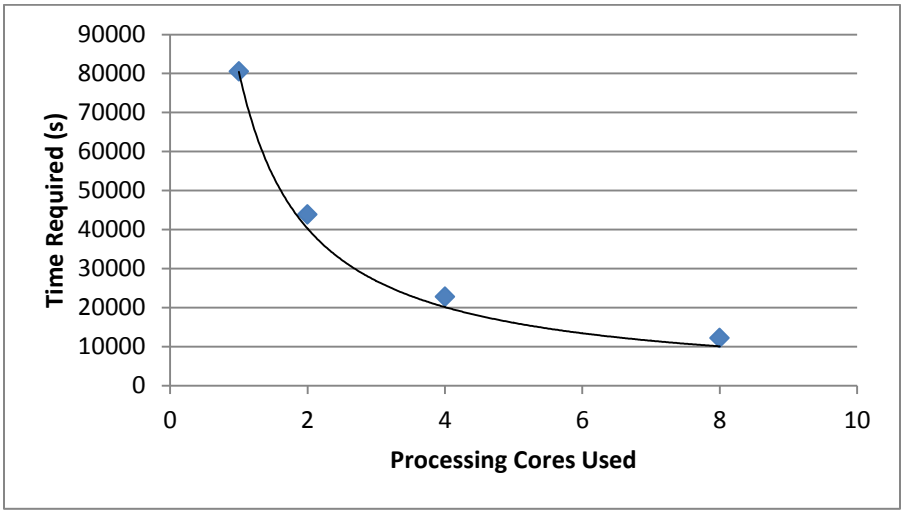


Figure 8. Parallel PRCALC Processing - Time Required vs Cores Used

3.4 Pre-Clustering Processing of PRCALC Outputs

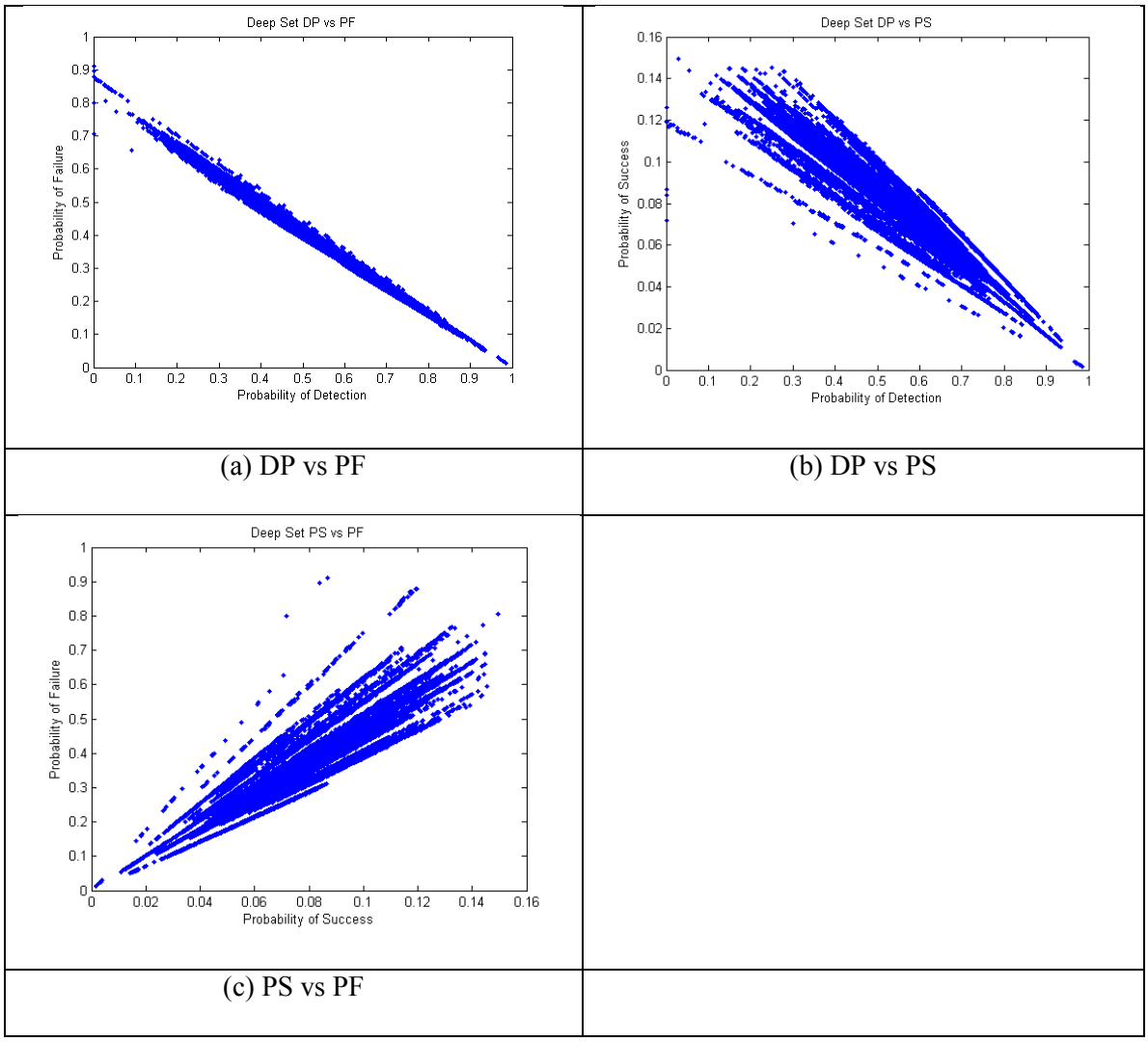
3.4.1 Correlated Variables

Because DP, PF and PS are the absorbing states of the Markov chain, they sum to 1 for each scenario. This indicates some correlation between them, which was investigated in preparation for clustering. The Deep Set of scenarios was chosen for this analysis. First, the covariance of each combination of the 5 PRCALC outputs was found using Equation 3. The correlation coefficient was then calculated using Equation 4

$$C(x,y) = \frac{\sum_{i=}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n} \quad (x, y = \ ,\cdots,5) \tag{3}$$

$$R(x,y) = \frac{C(x,y)}{\sqrt{C(x,x)C(y,y)}} \quad (x, y = \ ,\cdots,5) \tag{4}$$

where x and y denote PRCALC output variables, $\bar{x}$ and $\bar{y}$ are their corresponding arithmetic means, and n is the number of scenarios. The correlation coefficients are shown in Table 5. A positive value indicates that the variables rise together, and a negative value indicates that a rise in one variable corresponds to a fall in the other. A value near 0 indicates no linear correlation, and a value of 1 indicates perfect linear correlation. It can be seen that DP and PF have a very strong negative correlation (Figure 9 (a)), and that DP and PS are also negatively correlated to a lesser degree (Figure 9 (b)). PS and PF are positively correlated to nearly the same degree as DP and PS (Figure 9 (c)).



(a) DP vs PF

(b) DP vs PS

(c) PS vs PF

Figure 9. Deep Set Correlated PRCALC Output Variables

	DP (x=1)	PF (x=2)	MT (x=3)	PT (x=4)	PS (x=5)
DP (y=1)	1				
PF (y=2)	-0.9972	1			
MT (y=3)	0.0648	-0.0862	1		
PT (y=4)	0.3934	-0.3811	-0.0002	1	
PS (y=5)	-0.9368	0.9087	0.0416	-0.4179	1

Table 5. Deep Set Correlation Coefficients of PRCALC Outputs

3.4.2 Normalization of Variables

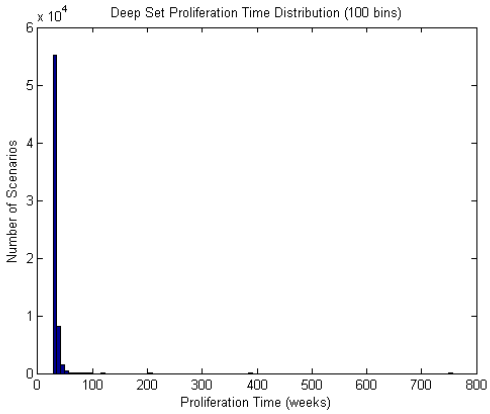
While the DP, PF, PS, and MT outputs of PRCALC vary from 0 to 1, PT varies from approximately 20 to 800 weeks. Used directly, this has the potential to skew the clustering analysis by giving extra weight to PT. This is corrected by normalizing each variable in a given set to vary from 0 to 1. The method for this process is given as

$$X_{(i,norm)} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{5}$$

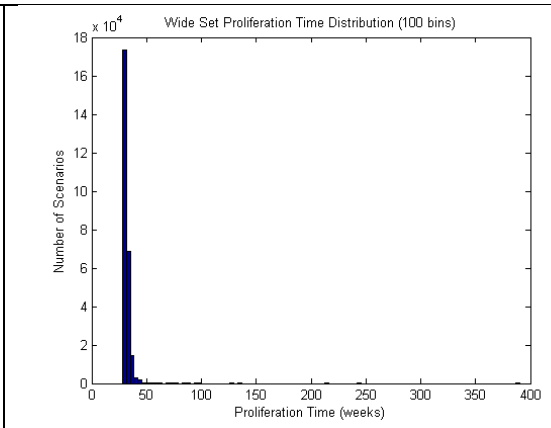
where $X_{(i,norm)}$ is the normalized value of point i of variable X, X_i is the raw value of point i of variable X, and X_{min} and X_{max} are the minimum and maximum values of variable X within the set, respectively. By this method, each variable used in clustering is assured the same weight for any clustering algorithm.

3.4.3 Very High Proliferation Times

For both the Deep and Wide Sets, the vast majority of scenarios fell between 20 and 40 weeks for PT. A few scenarios, however, took as long as 756 weeks. Figure 10 (a) shows the distribution for the Deep Set, with most bins being empty between the main group of scenarios and a high of 756 weeks. Figure 10 (b) similarly shows the Wide set, with a maximum PT of 391 weeks. These outlying scenarios complicated the clustering process as they would clearly be their own clusters, but they also caused most other scenarios to have very similar normalized values for PT.



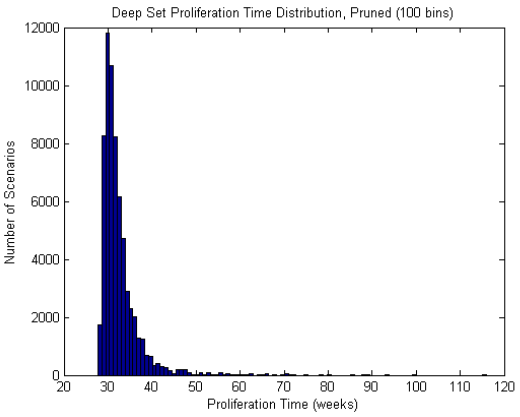
(a) Deep Set



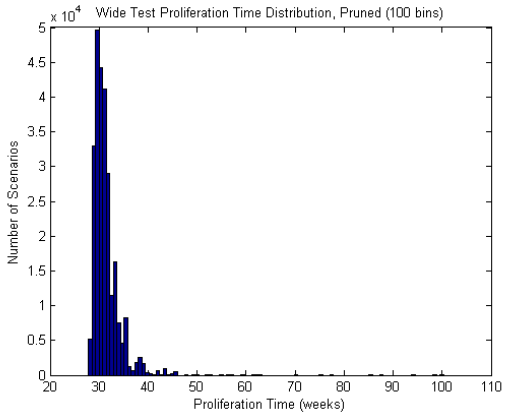
(b) Wide Set

Figure 10. PT Distribution, Original

It was also seen that, in both sets, those scenarios with very high values of PT also had high values of DP and low values of PS, making them unattractive to a proliferator. It was decided to prune scenarios after the first half-year (26 week) gap in PT. This criterion excluded 12 scenarios above 115 weeks in the Deep set, and 40 scenarios above 100 weeks in the Wide set. It can be seen in Figure 11 (a) that the pruned distribution for the Deep Set shows much more detail of the main mass of scenarios than the original in Figure 10 (a). The same can be seen in comparing Figure 11 (b) to Figure 10 (b) for the Wide Set.



(a) Pruned Deep Set



(b) Pruned Wide Set

Figure 11. PT Distribution, Pruned

Chapter 4: Clustering

This chapter describes the clustering algorithms used to sort scenarios by PRCALC output variables. Mean-shift and *k*-means are fully described, as is a variant based on mean-shift that is referred to as adaptive mean-shift (4.1). The time required to run each algorithm is compared, as are differences in scenario cluster assignments (4.2).

4.1 Clustering Algorithms

When a large set of scenarios is considered, it is often preferable to reduce the number to be analyzed. Clustering is often used to sort data into groups with similar characteristics, allowing for detailed analysis of the most interesting clusters of scenarios. In this case, various outputs of PRCALC are used as clustering variables in order to identify options taken by a proliferator that would lead to similar results, particularly clusters that would be most attractive to the proliferator. Three clustering algorithms are considered: mean-shift (4.1.1), adaptive mean-shift (4.1.2), and *k*-means (4.1.3). The output of each clustering method is a mapping of scenarios to clusters and the calculated centroid of the PRCALC output variables for each cluster.

4.1.1 Mean-Shift

The mean-shift algorithm [25] divides the data space into neighborhoods and finds the point of highest data density within each neighborhood as shown:

$$m\ x_s\ = \frac{\sum_i^N K\ x_i - x_s\ * x_i}{\sum_i^N K\ x_i - x_s} \tag{6}$$

where N is the number of points in the neighborhood, x_s is the initial guess at the neighborhood mean, x_i is data point i which represents the scenario within the neighborhood, and $K(x)$ is a kernel that weighs the distance between x_s and x_i. For this work, $K(x)$ is a Gaussian kernel as defined by Equation 7. Mean-shift clustering typically uses a Gaussian kernel unless a more data-appropriate kernel is evident. With each iteration x_s is updated until $m(x_s)$ is below a certain cutoff value. In this work, the cutoff value was chosen as 5×10^{-4}. This was chosen through experimentation as a compromise between solution stability and computation time required. The variable $m(x_s)$ represents the mean shift, or the distance (according to the kernel) from the mean to a local density maximum.

$$K\ x_i - x_s\ = e^{-\frac{|x_i - x_s\ |^2}{h^2}} \tag{7}$$

In Equation 7, h is the radius of inclusion for each neighborhood, and is referred to as the bandwidth. The bandwidth is a required input parameter for mean-shift. At the extremes a very large bandwidth leads to all scenarios being included in a single cluster, and a very small bandwidth leads to each scenario being its own cluster.

4.1.2 Adaptive Mean-Shift

The term adaptive mean-shift is used to refer to a variation on the mean-shift algorithm in which the bandwidth is varied across the data [26]. That is, different-sized neighborhoods are created in response to the local density of the data. This is often

accomplished using a pilot run of a fraction of the total data points to determine where densities are relatively high or low.

Fast adaptive mean-shift (FAMS) defines the bandwidth on a point-by-point basis using

$$h_i = ||x_i - x_{i,k}|| \tag{8}$$

where x_i is the data point under consideration, $x_{i,k}$ is the k^{th} nearest neighbor to x_i, and k is determined by a pilot run of the algorithm. The pilot run is repeated with differing values of k in order to find a value that returns a low average point-to-center distance within each cluster and a high center-to-center distance between clusters.

4.1.3 K-means

K-means is one of the most commonly-used clustering algorithms [27] due to its simplicity and fast computation. It is named for its parameter, *k*, which defines the number of clusters to be returned. The algorithm can be described in three steps:

1. Assign *k* cluster centroids randomly within the data space
2. Assign each point to the cluster represented by the closest centroid
3. Calculate new centroids based on the points within the cluster

Steps 2 and 3 are iterated until cluster assignments no longer change. The centroid of a cluster is defined as the average values of its members in each dimension. The Euclidean distance between a data point and the cluster centroids is used to determine the closest cluster. The algorithm is used as built into MATLAB and is repeated 20 times, giving 20 sets of initial cluster centroids. The run with the lowest sum of distances from points to their cluster centroids is used.

4.2 Clustering Algorithm Comparison

4.2.1 Clustering Speed

The time required to cluster a set of scenarios is often of concern. In general, *k*-means is a very fast algorithm. Mean-shift is very slow, and adaptive mean-shift is considerably faster than traditional mean-shift. A comparison was performed [28] for the Deep Set without pre-pruning high proliferation times (see Section 3.2), for a total of 65,532 scenarios. The clustering algorithms were run consecutively with the same computer as in Section 3.3, with the computer restarted in between tests. For a bandwidth of 0.5, mean-shift returned 4 clusters and required 14 hours. Adaptive mean-shift returned 8 clusters and required just under an hour. *K*-means was run with a k-value of 4 and required one second.

4.2.2 Cluster Assignments

Mean-shift and *k*-means naturally group any particular set of data into differently-shaped clusters. The most appropriate grouping may depend on the given data set. Figure 12 shows the results of clustering for the Deep Set using mean-shift with a bandwidth of 0.5. Four clusters were returned, and two of them were composed of easily-identifiable outliers: the scenarios with very high Proliferation Times. These are indicated with red arrows. The rest of the data points were divided roughly in half. Figure 13 shows the results of clustering using *k*-means with a *k* value of 4.

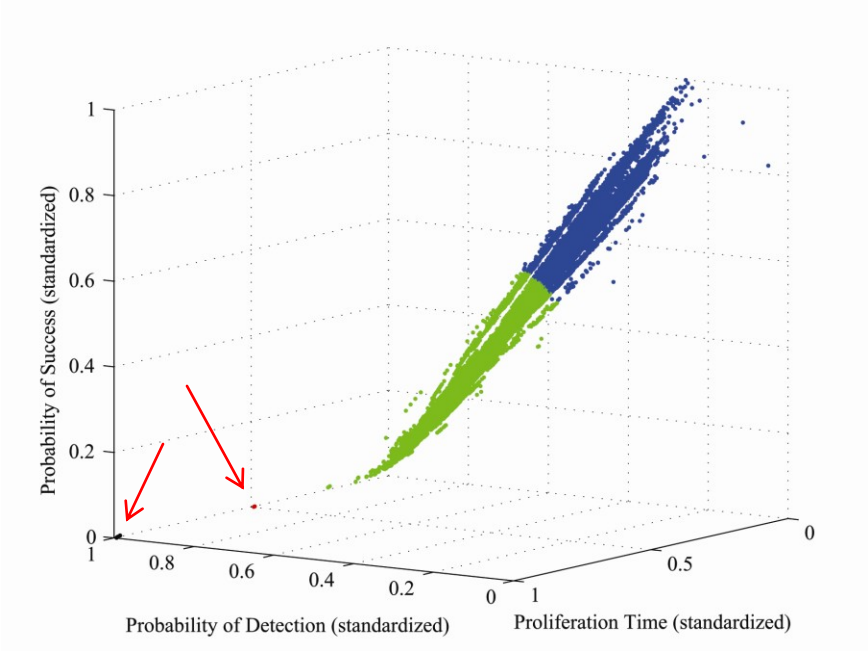


Figure 12. Deep Set, Mean-Shift, Bandwidth 0.5

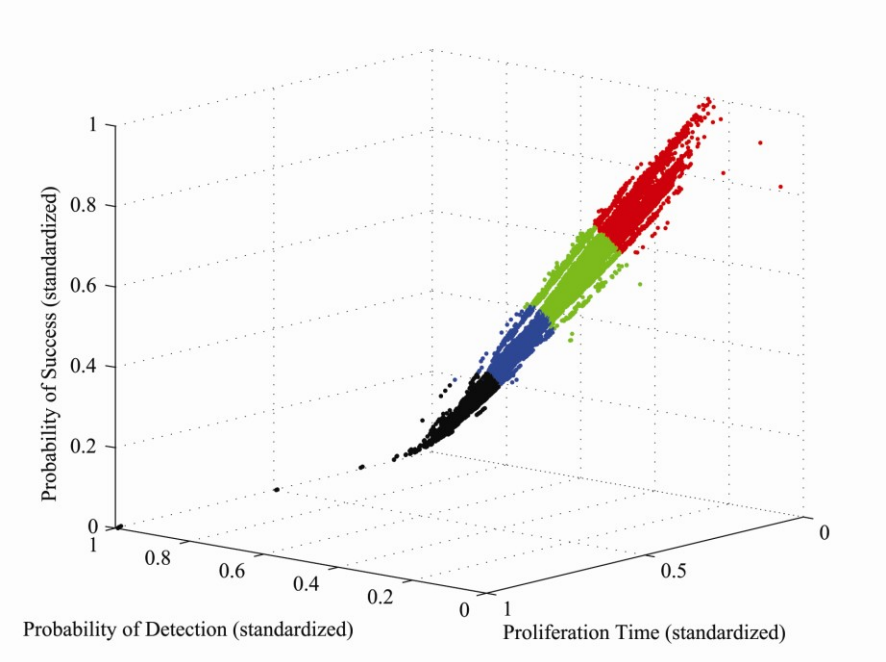


Figure 13. Deep Set, *K*-means, *k*=4

While there are still 4 clusters, the treatment of the outliers is different with *k*-means. Because points are assigned to the closest cluster centroid, the algorithm tends toward returning *k* clusters of equal size in the variable space. This may be troublesome for certain shapes of data because scenarios that are fundamentally different may be clustered together. Ultimately, the analyst must consider the results and decide whether a particular clustering method makes sense. In this case, the outliers with high Proliferation Time are obscured within a cluster that has an average Proliferation Time near that of the other clusters.

Chapter 5: Results & Analysis

This chapter presents the PRCALC outputs for the Deep and Wide data sets (5.1). The results of clustering the PRCALC outputs with the three clustering methods as described in Chapter 4 are also given (5.2). A cluster of interest is analyzed as an example of the work that this methodology allows (5.3). The development and operation of the OSUPR tool is described, and a small set of scenarios is run from creation to individual cluster analysis as an example (5.4).

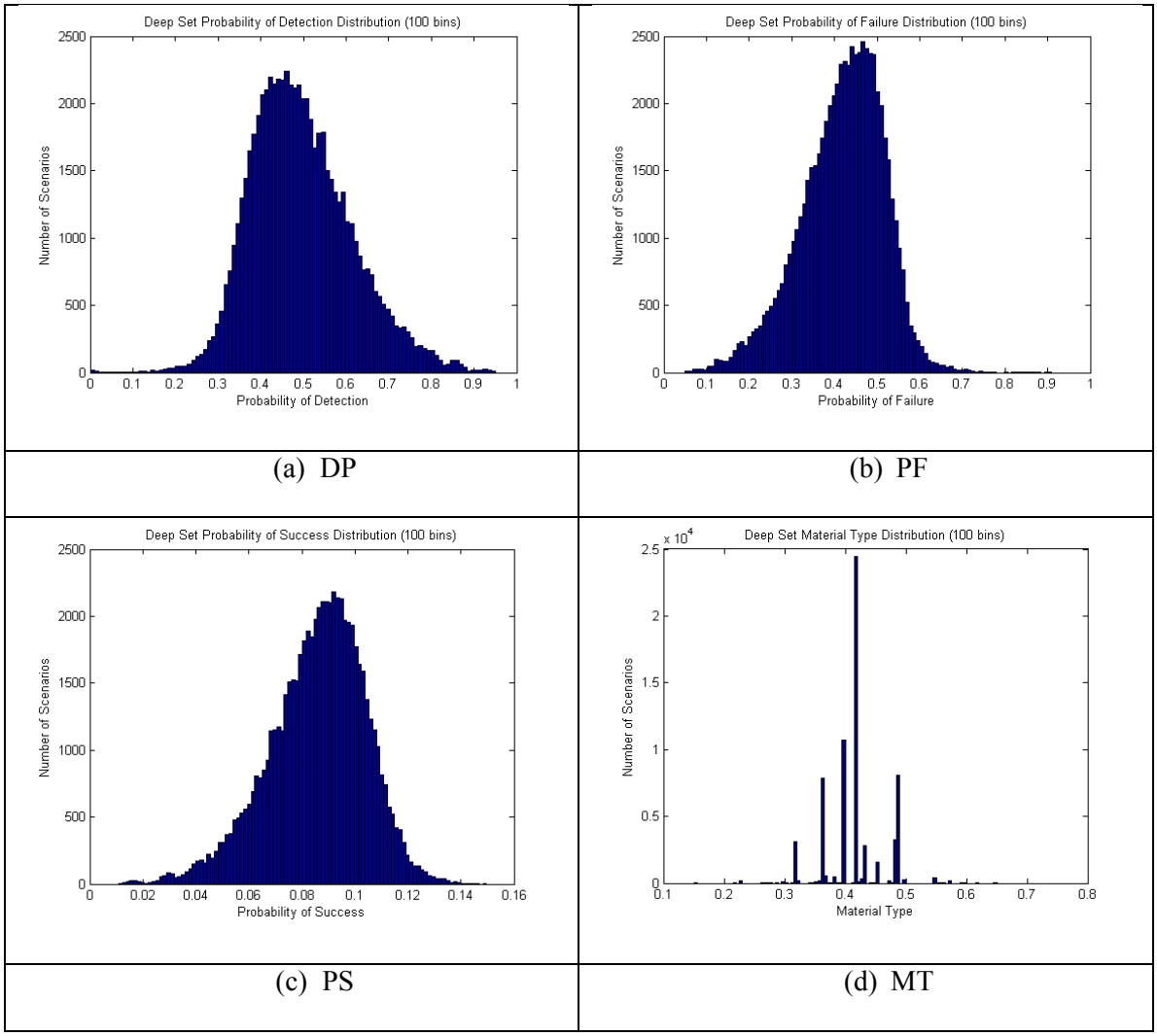
5.1 PRCALC Outputs

5.1.1 Deep Set PRCALC Outputs

The results of the first large set of scenarios, designated the Deep Set and created as described in Section 3.2, are shown in this section. As described in Section 3.4.3, some scenarios had very high values of PT and were set aside as distinct groups before the clustering analysis began. The correlation coefficients and plots of correlated PRCALC output variables are shown for this set in Section 3.4.1.

Figure 14 (a-d) shows the distributions of 4 of the 5 PRCALC output variables for the Deep Set. The distribution of PT is shown in Figure 11 (a). It can be seen from Figure 14 (a) and (b) that DP and PF generally cover the entire 0 to 1 interval, with averages of 0.49 and 0.42, respectively. Normalization does not greatly change the value of these variables. PS, shown in Figure 14 (c), has an average value of only 0.09 and occupies a

smaller range, from approximately 0 to 0.16. Normalization has the effect of stretching the PS distribution, giving it equal weight to other variables in the clustering algorithm. As Figure 14 (d) shows, MT is more concentrated on a few values, with an average value of 0.42 and 37% of scenarios with a value of exactly 0.42.



(a) DP (b) PF

(c) PS (d) MT

Figure 14. Deep Set Distribution of PRCALC Output Variables

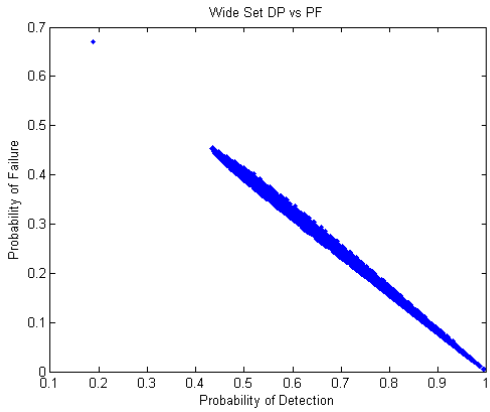
5.1.2 Wide Set PRCALC Outputs

The results of the second large set of scenarios, designated the Wide Set and created as described in Section 3.2, are shown in this section. As described in Section 3.4.3, some scenarios had very high values of PT and were set aside as distinct groups before the clustering analysis began. The set began with 262,140 scenarios, and was left with 262,100 after scenarios with very high values of PT were set aside.

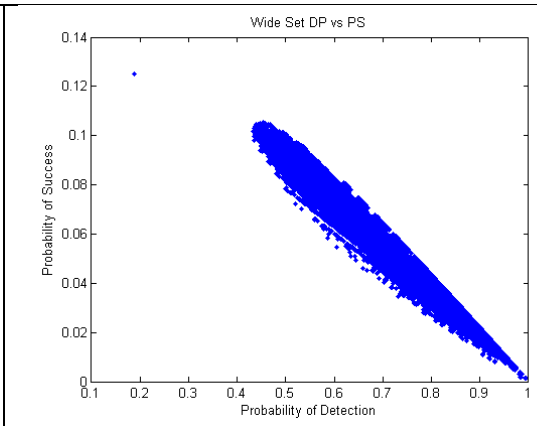
The correlation coefficients and plots of correlated PRCALC output variables were determined as described in Section 3.4.1 and are shown for this set in Table 6 and Figure 15. Similarly to the Deep Set, DP and PF are very strongly negatively correlated (Figure 15 (a)). DP and PS are also negatively correlated to a slightly less degree (Figure 15 (b)) than DP and PF. PS and PF are correlated ((Figure 15 (c)) to approximately the same degree as DP and PS. For this set, there is not a strong correlation between MT and PT with each other or any other PRCALC output.

	DP (x=1)	PF (x=2)	MT (x=3)	PT (x=4)	PS (x=5)
DP (y=1)	1				
PF (y=2)	-0.9992	1			
MT (y=3)	-0.119	0.1278	1		
PT (y=4)	0.4163	-0.4119	-0.0951	1	
PS (y=5)	-0.9872	0.9802	0.078	-0.4128	1

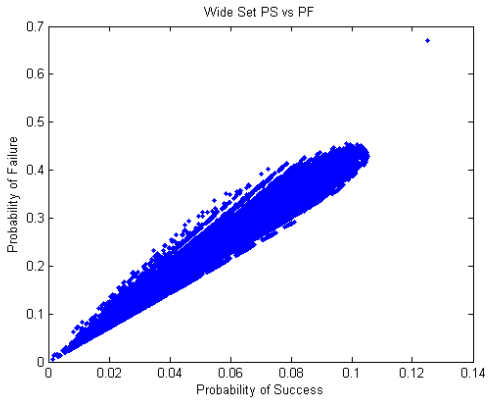
Table 6. Wide Set Correlation Coefficients of PRCALC Outputs



(a) DP vs PF



(b) DP vs PS



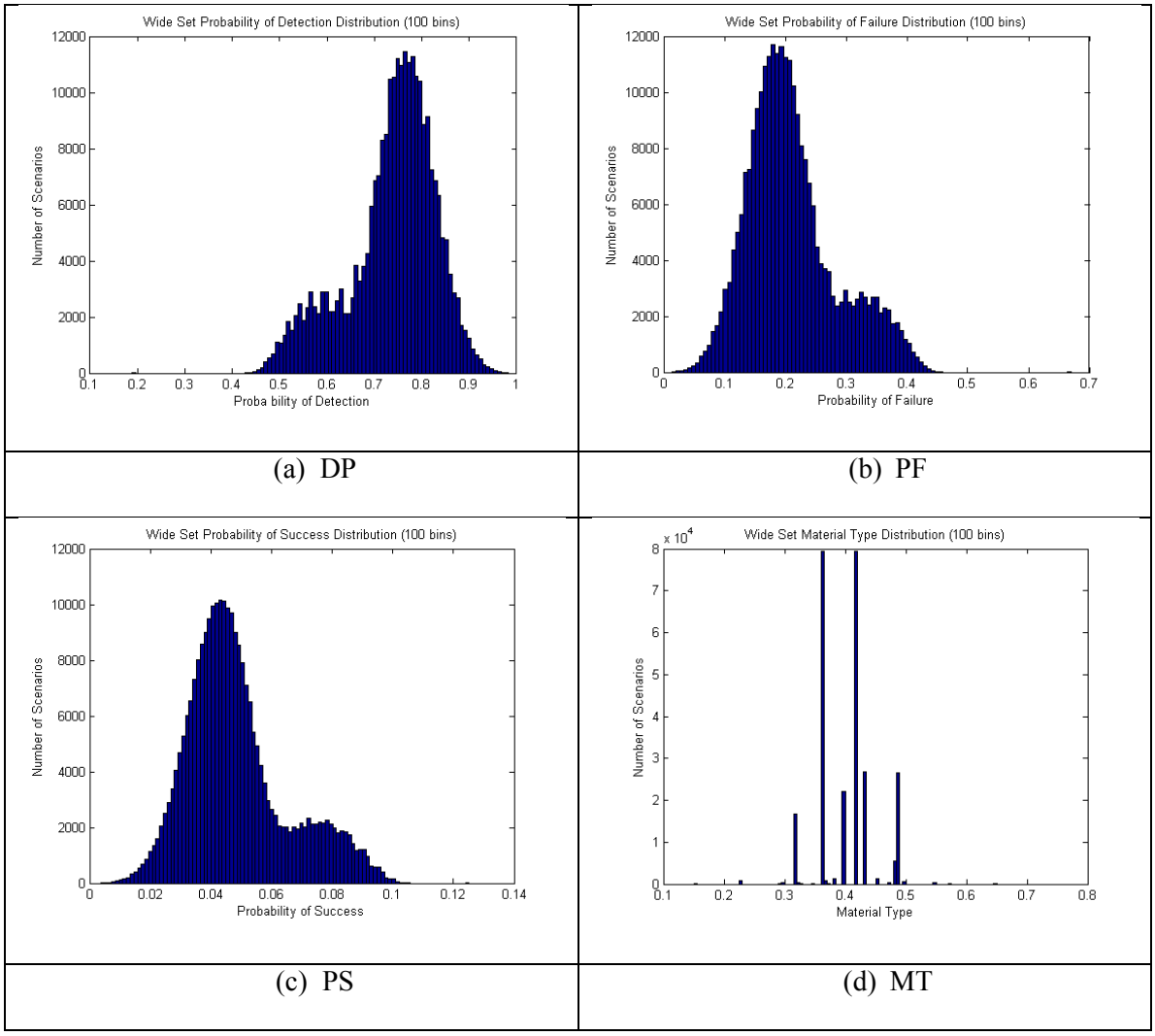
(c) PS vs PF

Figure 15. Wide Set Correlated PRCALC Output Variables

Figure 16 shows the distributions of 4 of the 5 PRCALC output variables for this set. The distribution of PT is shown in Figure 11 (b). It can be seen from Figure 16 (a) that DP generally varies from 0.4 to 1, with an average of 0.74. PF, shown in Figure 16 (b), generally varies from 0 to 0.5, with an average value of 0.21. PS, Figure 16 (c), generally varies from 0 to 0.11, with an average value of 0.05. Normalization affects DP and PF significantly more than in the Deep set, as they are concentrated in smaller ranges

of values. MT (Figure 16 (d)) is again mostly concentrated on a few values. 30% of scenarios have a value of 0.36, and 30% have a value of 0.42. The average value is 0.40.

DP, PF, and PS showed an interesting bimodal distribution for the Wide Set that was not present in the Deep Set. This separation may be due to the binary nature of the diversion rates used in the Wide set. Overall, outputs of the Wide set had smaller ranges than the Deep set.



(a) DP (b) PF

(c) PS (d) MT

Figure 16. Wide Set Distribution of PRCALC Output Variables

5.2 Clustering Outputs

The PRCALC outputs to be clustered were varied in order to gauge their effects on cluster assignment. The most common configuration was to use PS, DP, and PT, normalized to vary from 0 to 1 and thus given equal weight. On these three dimensions and for the data sets in question, a mean-shift bandwidth on the order of 0.1 to 0.5 returned a manageable number of clusters: on the order of 2 to 20.

5.2.1 Influence of Bandwidth

For the *k*-means algorithm, the number of clusters is selected a priori. For adaptive mean-shift, the bandwidth may differ for each point. The number of clusters returned by mean-shift for a given bandwidth cannot be calculated a priori. For a very small bandwidth, the number of clusters approaches the number of scenarios. For a large bandwidth, the number of clusters approaches 1. This is expected: the variable space may be visualized as a cube with edges of length 1, and the cluster neighborhood by a sphere of a radius equal to the bandwidth. For a bandwidth of 1 the entire set could be enveloped by a single cluster, or a few. For a bandwidth of 0.05, thousands of spheres may be required to cover the variable space.

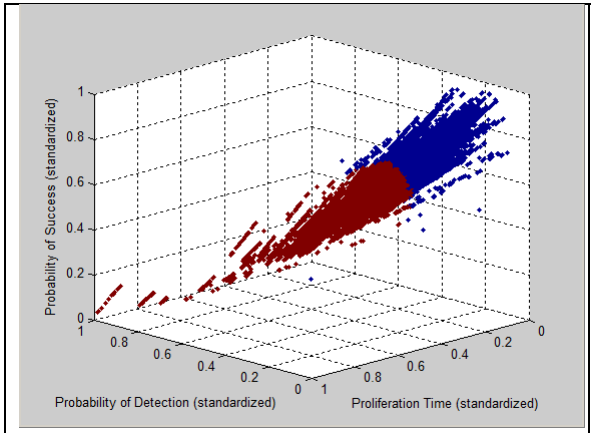
A smaller mean-shift bandwidth also requires more computation time. More clusters are created at the start and their centers must be tracked until they are declared empty, reach a local maximum, or collide with another cluster. There is some room to reduce the wall-clock time required to cluster by parallelizing the calculations, but the output of each core must be synchronized for each iteration. This possibility is explored more fully in Reference 18.

5.2.2 Comparison of Clustering Methods

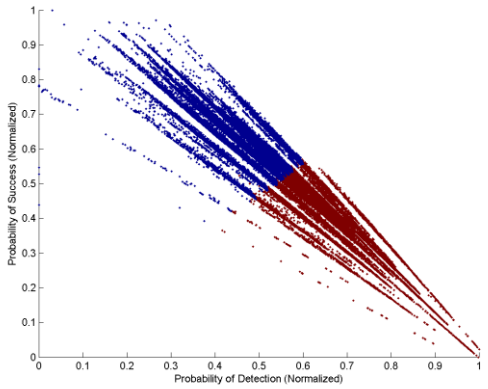
Mean-shift and *k*-means return clusters of different characteristic shapes, as described in Section 4.2.2. The plots compared in this section have had scenarios of very high PT removed in accordance with Section 3.4.3. Results are compared between the two mean-shift methods and *k*-means using the same number of clusters. First, in Figure 17, mean-shift with bandwidth 1 is compared with *k*-means with a *k* of 2 for the Deep Set.

Figure 17 (a) and (d) show the clustering results in a 3-D form, and it is immediately clear that *k*-means divided the set into roughly equal-sized clusters, while mean-shift grouped almost all scenarios into a single cluster. Figure 17 (b) and (e) show DP vs PS, and Figure 17 (c) and (f) compare the Proliferation Time vs Probability of Success for the two methods. These results represent a very large bandwidth for mean-shift. The small cluster for mean-shift has high PT, low DP, and middle-range PS, which may be of interest for a patient proliferator.

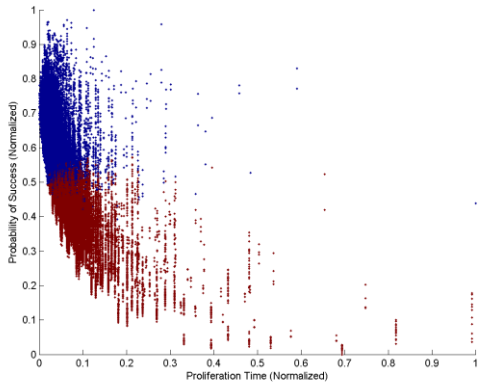
The next step smaller bandwidth chosen for mean-shift was 0.5. The results are shown in Figure 18 (a-c), and may be compared to Figure 17 (d-f). The same four points that comprised the smaller cluster under bandwidth 1 are distinct under bandwidth 0.5. These are indicated by a black arrow in Fig. 17 (b) and Fig. 18 (a). In addition, there is a third cluster of points with high PT, high DP, and low PS, indicated by a red arrow in Fig. 18 (a). This cluster does not appear to be appealing to a proliferator, but is an example of how the set of scenarios is further divided with a smaller bandwidth.



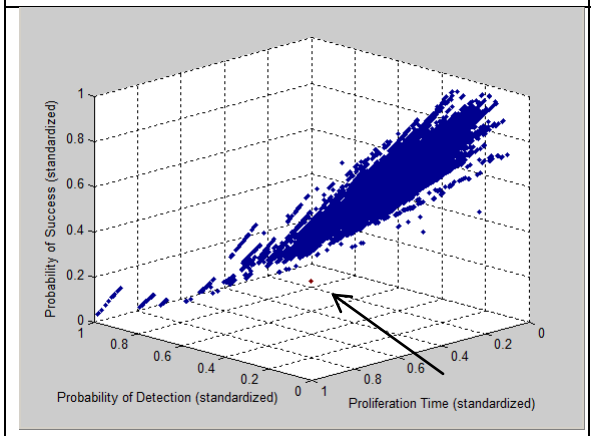
(a) Deep Set, *K*-means, 3D



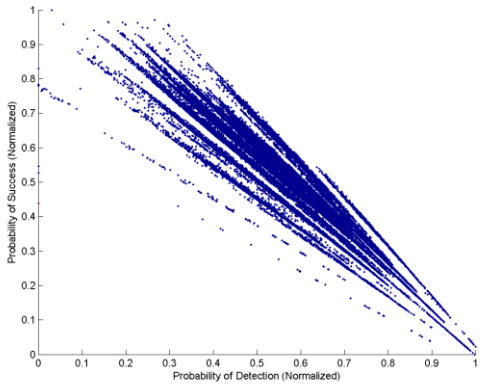
(b) Deep Set, *K*-means, DP vs PS



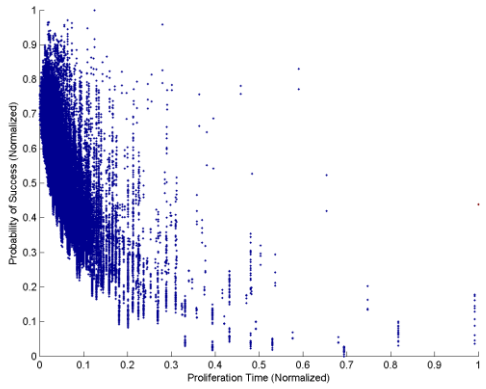
(c) Deep Set, *K*-means, PT vs PS



(d) Deep Set, Mean-Shift, 3D

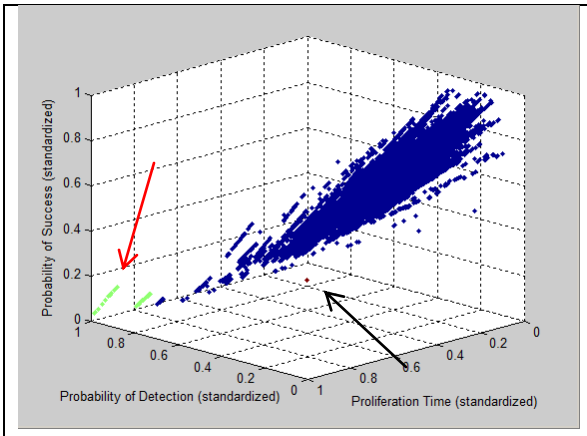


(e) Deep Set, Mean-Shift, DP vs PS

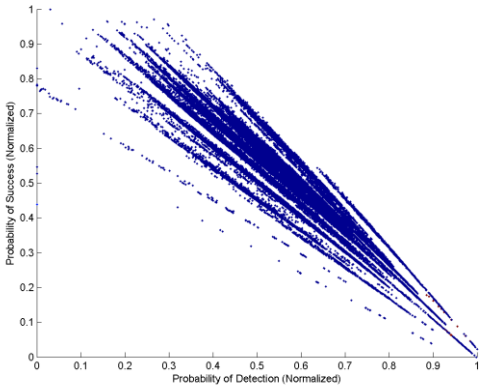


(f) Deep Set, Mean-Shift, PT vs PS

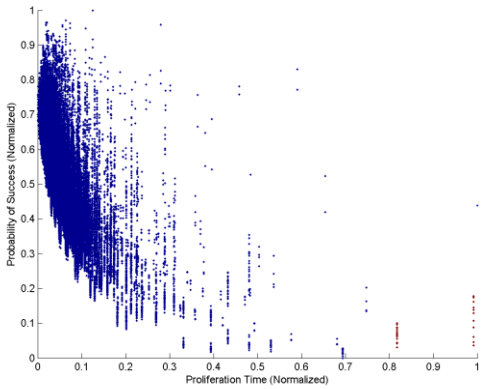
Figure 17. Deep Set, Comparison of *K*-means with *k*=2 to Mean-Shift with Bandwidth 1



(a) Deep Set – Mean-Shift, 3D



(b) Deep Set – Mean-Shift, DP vs PS



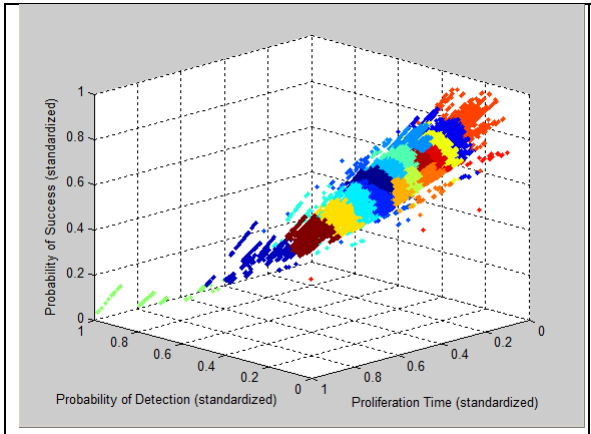
(c) Deep Set, Mean-Shift, PT vs PS

Figure 18. Deep Set PRCALC Outputs, Mean-Shift with Bandwidth 0.5

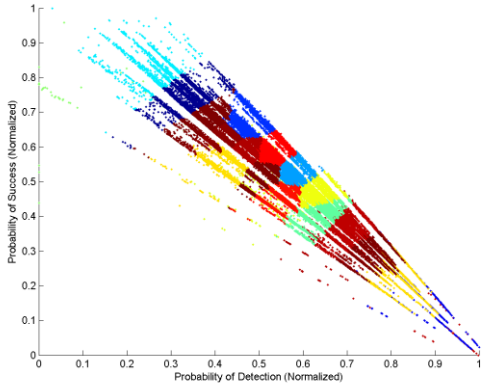
Next, in Figure 19, the results of adaptive mean-shift were compared to *k*-means with a *k* of 21. Both methods returned 21 clusters. While the clusters may easily be counted for *k*-means, most of the clusters under adaptive mean-shift are in the area indicated by a black circle in Fig. 19 (d). Figure 19 (a) and (d) again show 3-D plots of the clustered data. *K*-means has once again divided the set into roughly equal-sized clusters. Adaptive mean-shift has assigned most scenarios to one of two large clusters,

with each group of outliers occupying their own clusters. DP vs PS are compared in Figure 19 (b) and (e), and PT vs PS is compared in Figure 19 (c) and (f). The large dark blue cluster for adaptive mean-shift may be of interest, as it has a low average PT, low DP, and high PS, indicating relatively favorable conditions for a proliferator. This cluster is further examined in Section 5.3. The number of clusters returned is easily handled, and may be helpful without being overwhelming to an analyst.

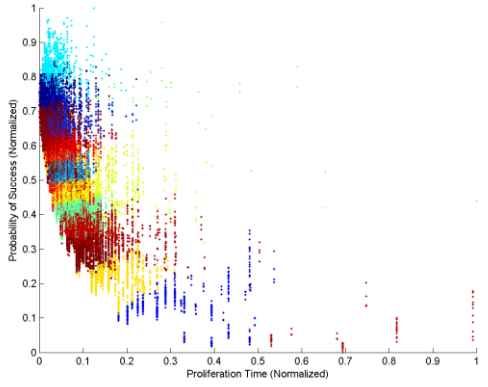
Figure 20 shows the results of a mean-shift bandwidth of 0.01, which is a small bandwidth for this data set. Indeed, 7,964 clusters were returned. For comparison, the results of *k*-means with a *k* of 7,964 are also shown. Figure 20 (a) and (d) compare 3-D plots of the clustered data set, (b) and (e) compare DP vs PS, and (c) and (f) compare PT vs PS. It is clear that 0.01 is a small mean-shift bandwidth for this data set. While clusters can generally be tracked across the data for mean-shift, clusters are nearly unidentifiable with *k*-means (Fig. 20 (a)). It is very difficult to visually or numerically pick out clusters according to attractiveness to a proliferator, and so this bandwidth may be inappropriately small for analysis.



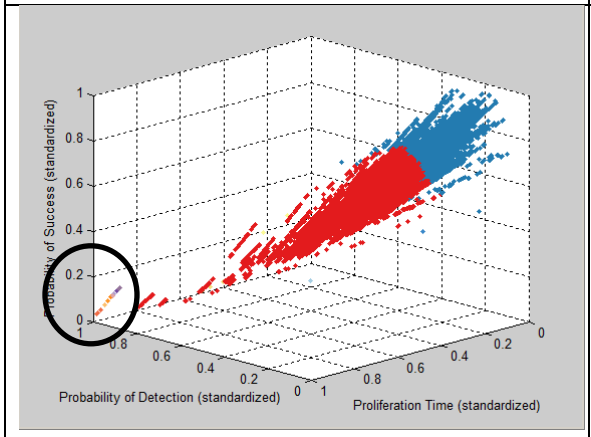
(a) Deep Set, *K*-means, 3D



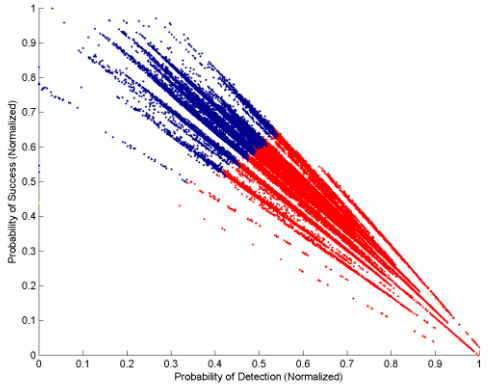
(b) Deep Set, *K*-means, DP vs PS



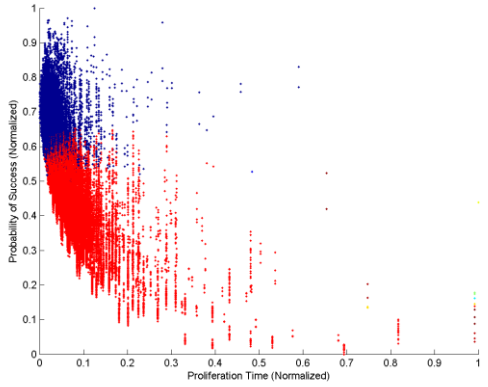
(c) Deep Set, *K*-means, PT vs PS



(d) Deep Set, FAMS, 3D

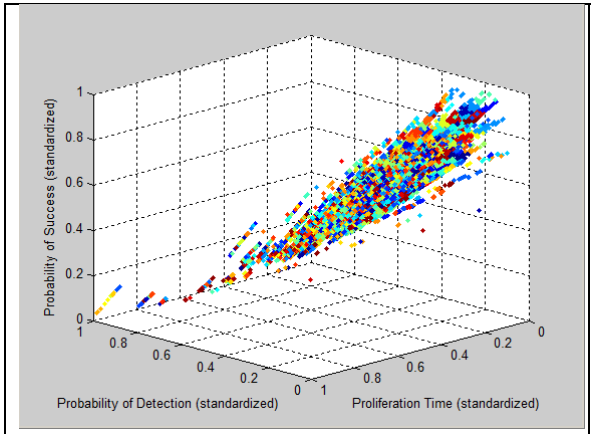


(e) Deep Set, FAMS, DP vs PS

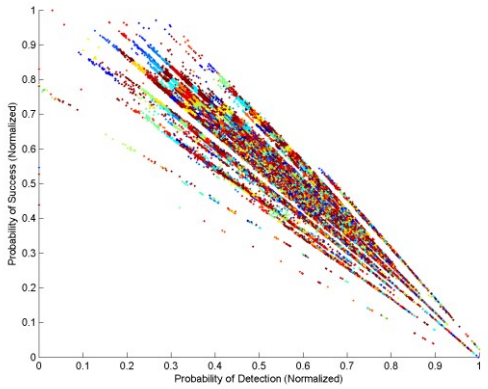


(f) Deep Set, FAMS, PT vs PS

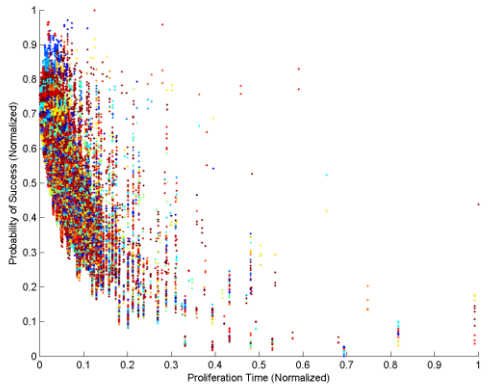
Figure 19. Deep Set, Comparison of *K*-means with *k*=21 to Fast Adaptive Mean-Shift (FAMS)



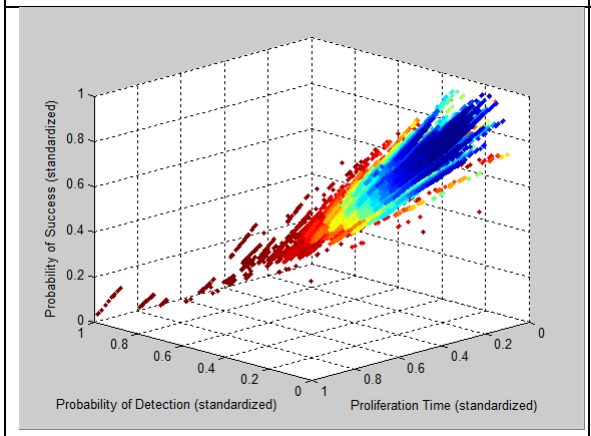
(a) Deep Set, *K*-means, 3D



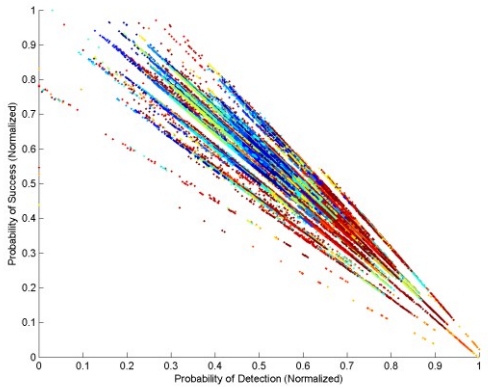
(b) Deep Set, *K*-means, DP vs PS



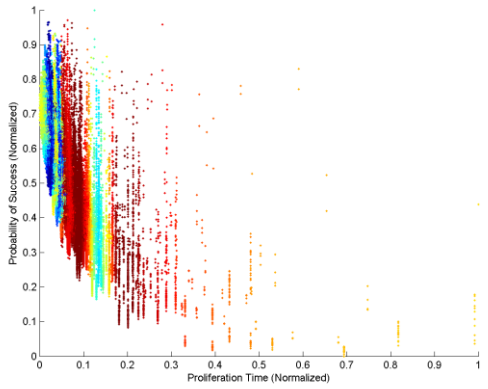
(c) Deep Set, *K*-means, PT vs PS



(d) Deep Set, Mean-Shift, 3D



(e) Deep Set, Mean-Shift, DP vs PS



(f) Deep Set, Mean-Shift, PT vs PS

Figure 20. Deep Set, Comparison of *K*-means with *k*=7964 to Mean-Shift with Bandwidth 0.01

5.3 Analysis of Individual Clusters

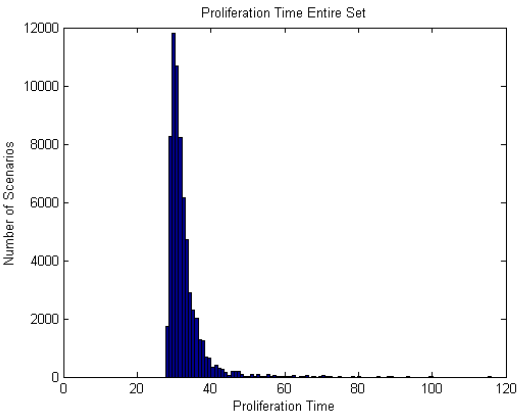
To arrive at the objective of the thesis, the analysis of important factors in the success of proliferation attempts, individual clusters were examined for trends. Each cluster of interest was first identified by its outcomes. The inputs that led to inclusion in such a cluster were then probed for clues to sensitive safeguard conditions and targets.

5.3.1 Deep Set, Fast Adaptive Mean-Shift, Cluster 3

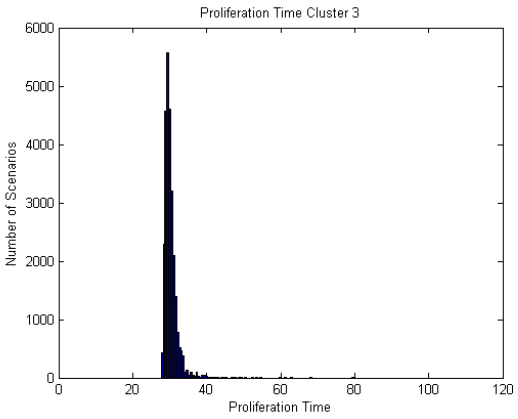
The output of the FAMS analysis resulted in 21 clusters but most of the scenarios were grouped into just two large clusters, the blue cluster and the red cluster shown in Figure 19 (d-f). The other clusters all fell within the region of the circle in Figure 19 (d) involving low success probability, high detection probability and long acquisition times, which would be avoided by the proliferator. The most attractive outcomes for the proliferator are associated with the dark blue cluster (Cluster 3). Thus, the focus of the analysis is on the underlying characteristics, targets for proliferation and safeguard characteristics of the Cluster 3 scenarios that led to these outcomes. To a large extent the assessment is based on a comparison between Cluster 3 events and the data set as a whole. In Figure 21, the principal outcomes of the FAMS Cluster 3 events are compared with the entire data set. Comparing Figure 21 (a) and (b), these scenarios roughly follow the distribution of the larger set with regards to Proliferation Time. Thus, the attractiveness of Cluster 3 does not appear to be associated with particularly short proliferation times. As can be seen in Figure 21 (c) and (d), the FAMS Cluster 3 contains mostly scenarios on the lower part of the Probability of Detection distribution. Similarly, in Figure 21 (e) and (f), it is seen that the distribution of the Probability of Success

represents the upper portion of the distribution of the overall population of scenarios. Thus, the common characteristics of the FAMS Cluster 3 scenarios that set it apart are scenarios leading to high Probability of Success and low Detection Probability.

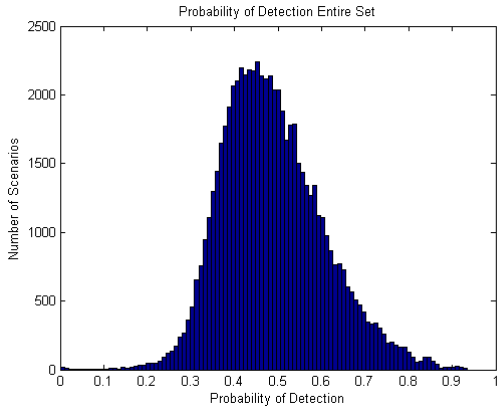
The specific inputs (safeguards conditions and diversion rates) that led to the results in Figure 21 are shown in Figure 22. The safeguard Conditions refer to the removal of safeguard Categories. In Condition 0, the default safeguards are in place. In Conditions 2, 3, and 4, all safeguards of Categories 2, 3, or 4 respectively are removed. Because of the manner in which the scenarios are generated, overall there are the same numbers of sequences with default safeguards (Condition 0 in Figure 22 (a)) and with safeguards categories 2, 3 and 4 removed, as indicated in Figure 22 (a) by the equal heights of the four bars. Similarly, Figure 22 (c), (e) and (g) have equal sized bars associated with proliferation from TRU extraction, LWRSF storage and pin fabrication as target areas for proliferation. Figure 22 (b) shows that a smaller proportion of scenarios within the cluster had all safeguards intact, when compared with the entire set. It was expected that any removal of safeguards would lead to more favorable conditions for the proliferator. A large number of scenarios with safeguard Category 2 removed were included in the cluster. Category 2 safeguards are related to yearly inventory inspections, thus the results indicate that they are of great importance.



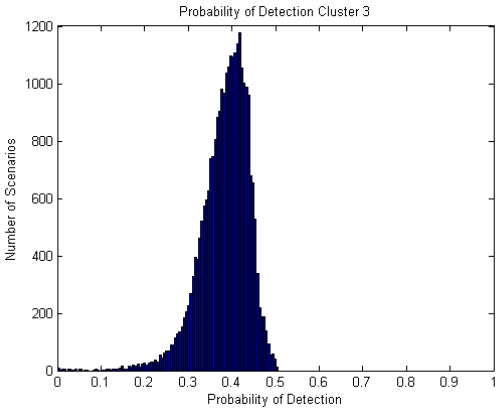
(a) PT – Entire Set



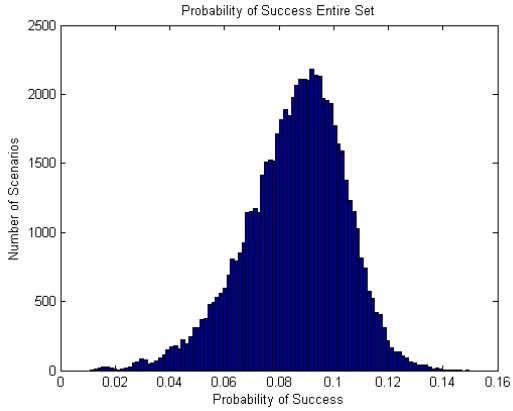
(b) PT – FAMS Cluster 3



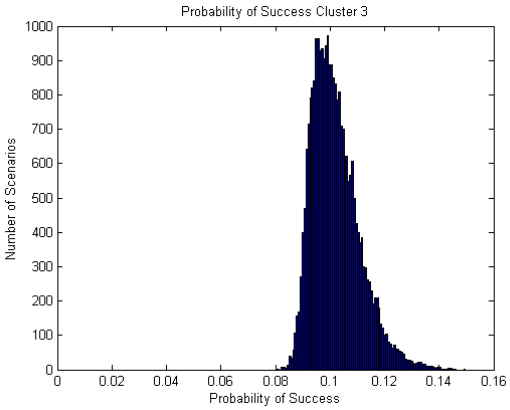
(c) DP – Entire Set



(d) DP – FAMS Cluster 3

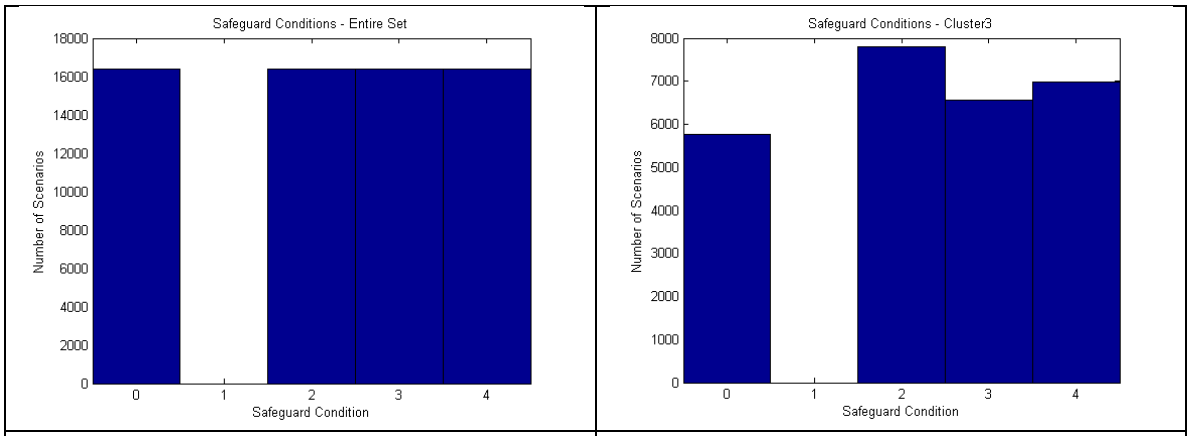


(e) PS – Entire Set

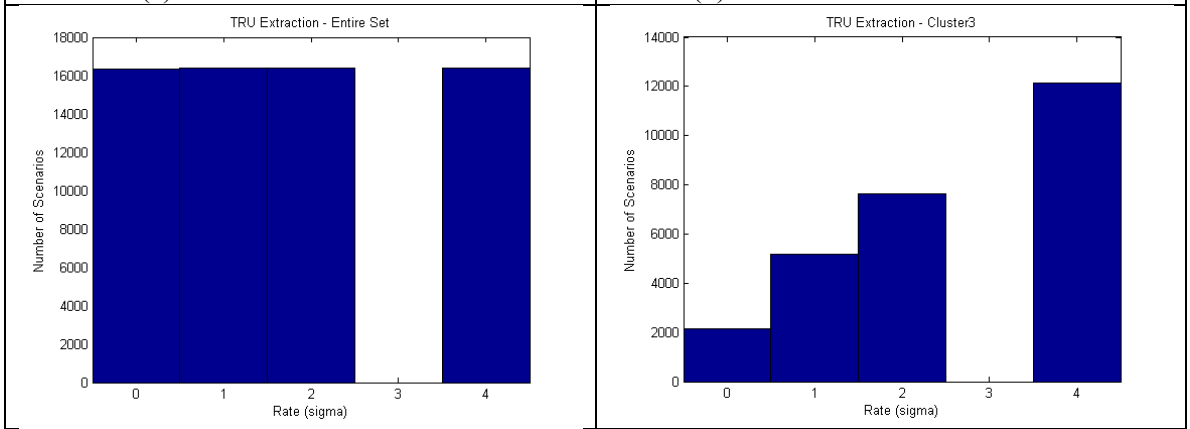


(f) PS – FAMS Cluster 3

Figure 21. Histograms for PRCALC Outputs, Deep Set, Fast Adaptive Mean-Shift, Cluster 3



(a) SG Conditions – Entire Set | (b) SG Conditions – FAMS Cluster 3

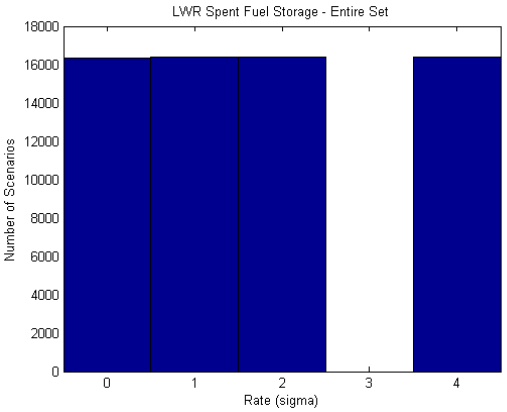


(c) TRU Extraction – Entire Set | (d) TRU Extraction – FAMS Cluster 3

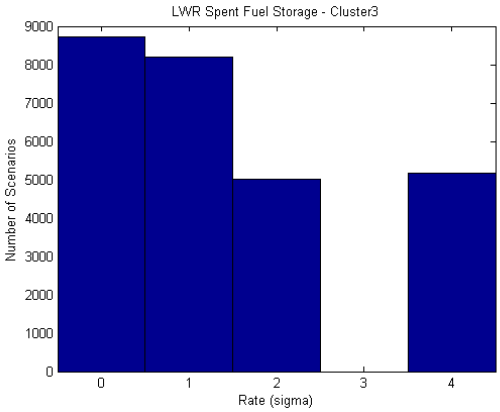
Figure 22. Histograms for PRCALC Inputs, Deep Set, Fast Adaptive Mean-Shift, Cluster 3

(continued)

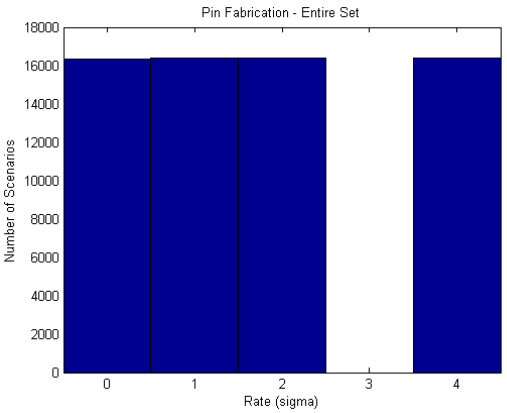
(Figure 22 continued)



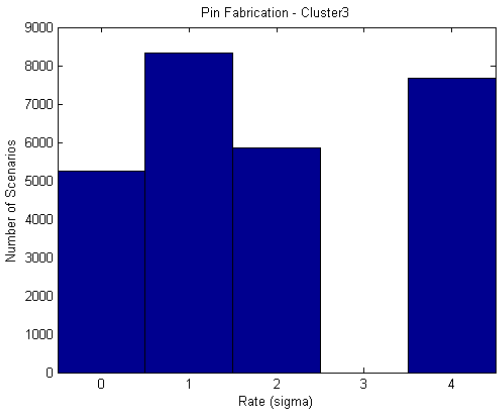
(e) LWRSF Storage – Entire Set



(f) LWRSF Storage – FAMS Cluster 3



(g) Pin Fabrication – Entire Set



(h) Pin Fabrication – FAMS Cluster 3

Figure 22 (d) indicates that scenarios with high rates of material diversion from the TRU area of the facility are found in Cluster 3. This behavior is also characteristic of proliferation from the Electro-refiner. High rates of diversion from these targets lead to better outcomes for the proliferator. This knowledge could be used to modify processes or to apply additional safeguards selectively. Figure 22 (f) shows a target, LWR Spent Fuel Storage, where the opposite trend is true: where a higher diversion rate leads to exclusion from Cluster 3. This indicates that LWR Spent Fuel Storage is not the most

attractive target for a proliferator. Figure 22 (h) is representative of targets where a clear trend is not seen between diversion rate and inclusion in the cluster of interest. This is true of Pin Fabrication, Storage Basket: Fresh Fuel, Chopping, and ESFR SF Disassembly.

5.4 OSUPR

OSUPR was developed as a tool for scenario clustering for PRCALC diversion scenarios. It is an easy-to-use assembly of the MATLAB scripts and functions that were created for the analysis described in this thesis.

5.4.1 Software Development

OSUPR was designed to be a drop-in addition to PRCALC. PRCALC is completely functional on an installation of MATLAB without any additional toolboxes. OSUPR is similarly self-contained. PRCALC is platform-agnostic, and works equivalently under the Windows and Unix versions of MATLAB. OSUPR maintains this to some degree. In its current state, the multi-threading scheme is Windows-specific. PRCALC batch files are not, by their nature, portable between computers. An absolute path is required for loading of scenarios in the PRCALC batch mode. This can be a limitation if a user desires to create a set of scenarios on a personal computer and then run it on a more powerful computer.

Figure 23 shows the flow of information in OSUPR. User Inputs are given to the Scenario Creation Interface (see Section 5.4.2.1) and the Clustering & Analysis Interface (see Section 5.4.2.2). Each interface processes user inputs in some way and creates output

files in a common folder. The Scenario Creation Interface creates scenarios, batches them, and runs them through PRCALC. User choices as well as PRCALC outputs are saved in a master file, as described in Section 5.4.2.1. The Clustering & Analysis Interface allows the user to choose parameters for clustering PRCALC outputs and then examine individual clusters more deeply. This interface also updates the master file with clustering choices and outputs.

5.4.2 OSUPR Operation

OSUPR operation consists of two MATLAB interfaces: scenario creation, in which scenarios are created and run through PRCALC (5.4.2.1), and clustering and analysis, in which data sets may be clustered and examined for trends (5.4.2.2). The code behind the scenario creation interface is given in Appendix A, and the code for the clustering and analysis interface is given in Appendix B.

5.4.2.1 OSUPR Scenario Creation Interface

The OSUPR scenario creation interface is shown as Figure 24. The steps that the operator follows are listed in Table 7. Diversion rates in terms of σ, as described in Section 3.1, are entered in Step 2. In Step 3, Safeguard Conditions are chosen using the categories described in Section 3.1. In Step 4, any set of targets is chosen as shown in Table 2. Step 5 approximates the number of scenarios in the set using Equation 2 and displays the result in a pop-up window. This gives the user an idea of the size of the data set and the relative time required to run PRCALC and the clustering algorithms.

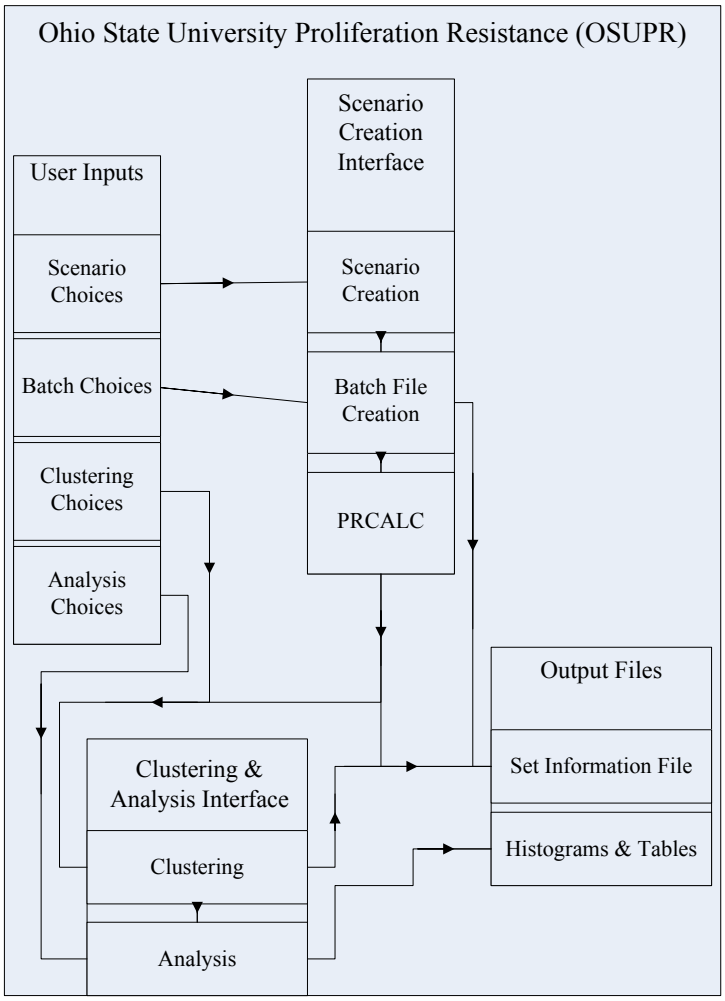


Figure 23. OSUPR Schematic

Step #	Description
1	Series Name
2	Enter Diversion Rates
3	Select Safeguard Conditions
4	Select Stage Elements
5	Calculate Number of Scenarios
6	Choose Folder
7	Create Input & Info Files
8	Get Number of Cores
9	Enter Number of Batches
10	Create Batch Files
11	Run PRCALC

Table 7. OSUPR Scenario Creation Steps

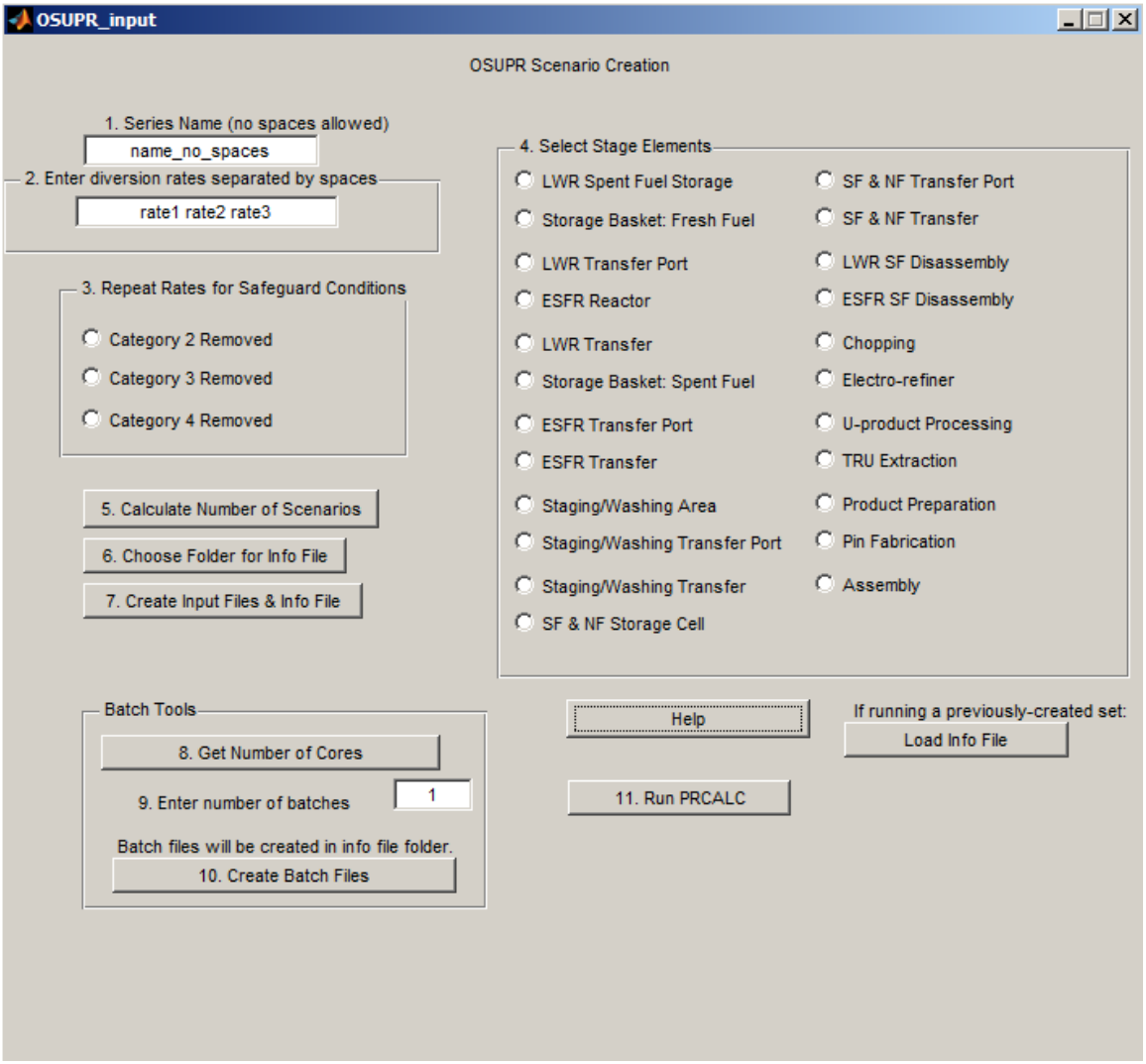


Figure 24. OSUPR Scenario Creation Interface

To contain all of the information associated with the set, a single Information File is created. This file's contents are listed in Table 8. At the point of scenario creation in Step 7, only the Names and Rates fields are populated. The Information File may be inspected in MATLAB, where it is assembled as a structure array. A structure array may contain various different types of data, including other structure arrays, numerical matrices, cells and strings. Step 8 polls the system for the number of cores available, and displays the result in a pop-up window. In Step 9 the user enters the number of batches to

create, which often equals the number of cores available. The benefit of running on multiple cores is described in Section 3.3. Step 10 creates batch files and updates the Information File with batch assignments and file locations.

Field	Description
Names	The name of each scenario file
Rates	Diversion rates associated with each scenario
BatchAssign	Assignment of each scenario to a batch
Batches	Batch file names and locations
ConfirmFile	The name of a small file used to confirm that a thread is finished
PRResults	PRCALC outputs, both raw and normalized
KmeansInfo	Cluster assignment and statistics for *k*-means
MeanShiftInfo	Cluster assignment and statistics for mean-shift
FAMSInfo	Cluster assignment and statistics for adaptive mean-shift

Table 8. Information File Contents

PRCALC is finally run as Step 11. Steps 1-10 in Figure 24 may be skipped if a previously-created set is selected using the "Load Info File" button. This may be convenient if a set is created on a workstation during the day and saved to run later, or if many sets are created at once. Each batch is run as a separate instance of PRCALC, and thus MATLAB. Once each processing node is finished, the results are re-combined into a single set and saved to the Information File.

As a demonstration, Figure 25 shows an example set with possible diversion rates of 0, 1, and 2σ. To simulate an attack on the reprocessing facility itself, the targets are the Electro-refiner, U-product Processing, TRU Extraction, and Product Preparation. The set is repeated a total of three times: once with default safeguards, once with Category 2 safeguards removed, and once with Category 4 safeguards removed. According to Equation 2, this indicates 243 scenarios (see Fig. 26). OSUPR automatically removes

scenarios with all 0 diversion rates, i.e. those in which no diversion is occurring, but that is not done until Step 7. Once a folder is selected, scenarios are created in Step 7. Depending on the number of scenarios, this may take from a few seconds (~250 scenarios) to a few hours (~250,000 scenarios). A progress bar is displayed as seen in Figure 25 for the scenario file creation process, and a pop-up window informs the user when file creation is finished. A batch file creation progress bar is displayed, but this step generally only takes seconds for even a very large data set.

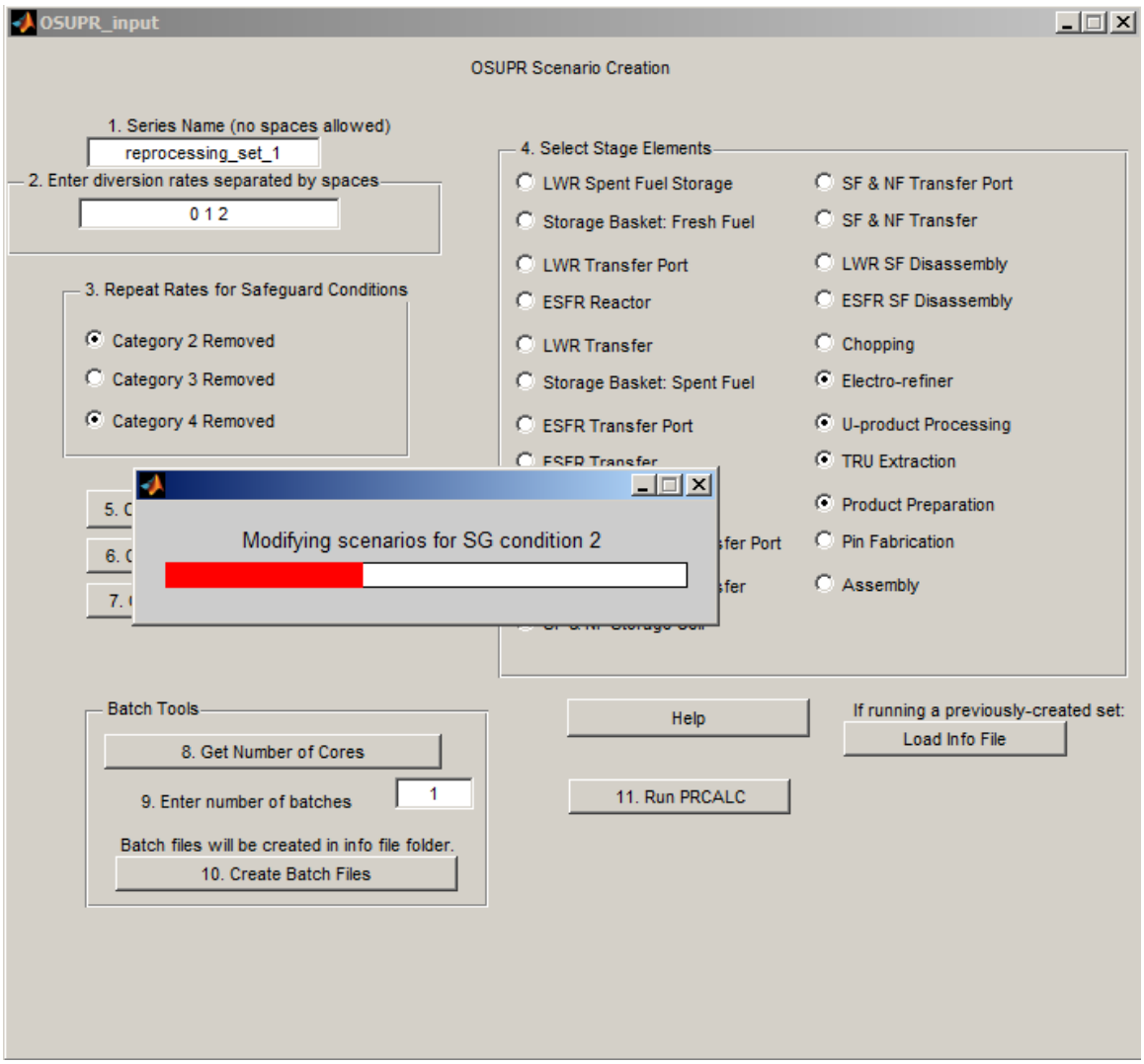


Figure 25. OSUPR Scenario Creation Progress Indicator

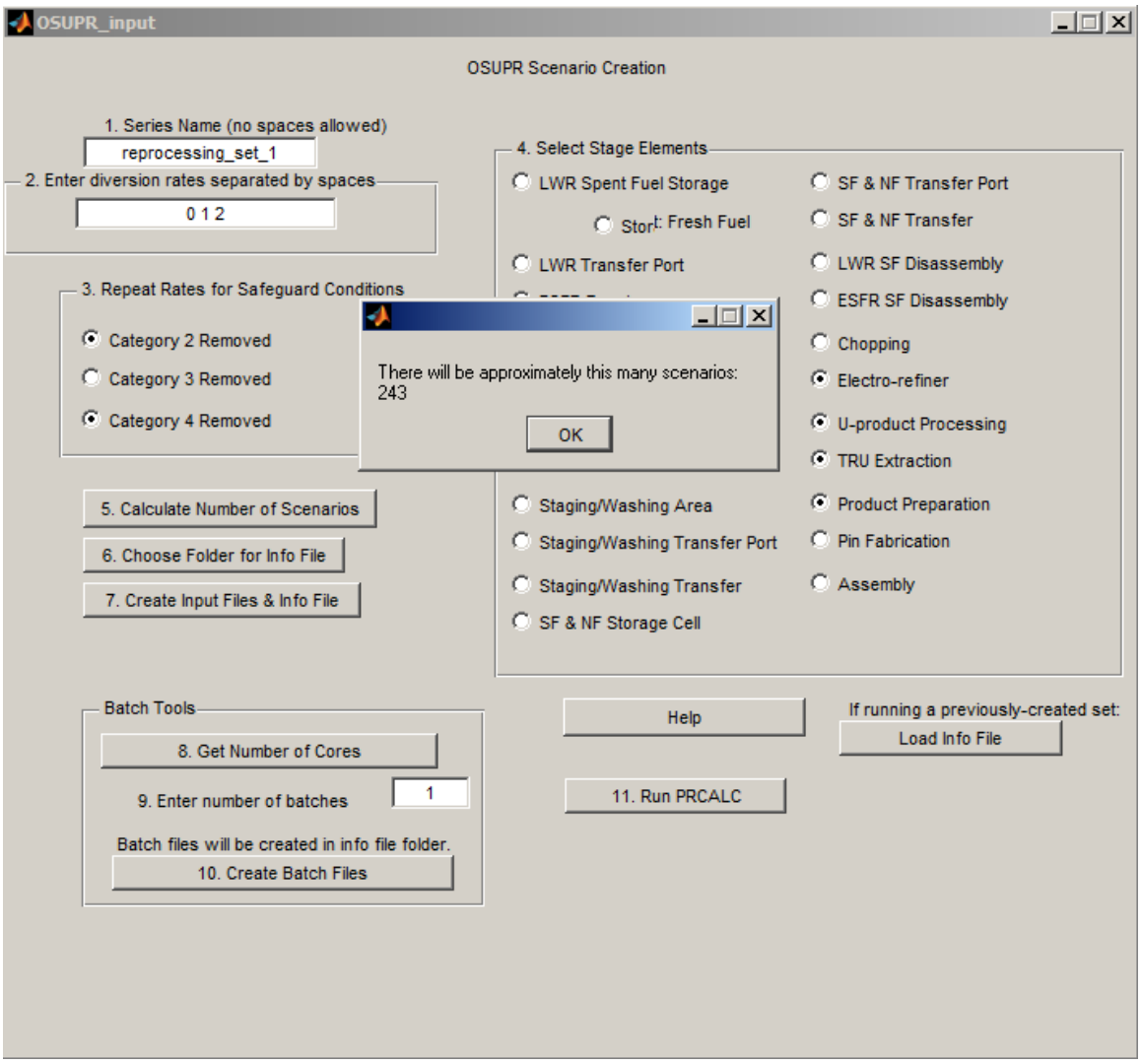


Figure 26. OSUPR Scenario Creation Interface - Number of Scenarios Calculation

This sample set comprises 240 scenarios and requires 300 seconds to run in 4 parallel batches on a laptop computer with an Intel Core i3-3217U processor. In large part due to the size of the scenario files, the data created at this point requires 5.7 MB of disk space. The Information File, which fully represents the set for clustering and analysis, is only 12 KB. This reduction may be especially helpful in large sets, where the scenario files may require gigabytes of space. At this point the Information File can be manually inspected using MATLAB, or loaded into the Clustering & Analysis Interface.

5.4.2.2 OSUPR Clustering & Analysis Interface

The OSUPR Clustering & Analysis Interface is illustrated in Figure 27, with the steps presented as Table 9. The user must first load an Information File for a data set (Step 1). This is the same Information File that is created using the Scenario Creation Interface and described in Table 8. During this step, the PRCALC outputs are normalized according to Section 3.4.2 and Equation 5. The user then enters a parameter for mean-shift or *k*-means (Step 2) and runs the chosen algorithm (Step 3). A progress bar is not available for clustering because the time required for convergence cannot easily be calculated a priori. A pop-up window does inform the user when a particular method is finished. Clustering methods can only be used one at a time, but all 3 can be used on the same set of data.

Step #	Description
1	Load Information File
2 a. b.	Enter Clustering Parameter Mean-Shift *k*-means
3 a. b. c.	Run Clustering Algorithm Mean-Shift *k*-means Adaptive mean-shift
4	Choose Algorithm for Analysis
5	View Cluster Centroids
6	Choose Cluster of Interest
7	Analyze Cluster of Interest

Table 9. OSUPR Clustering & Analysis Steps

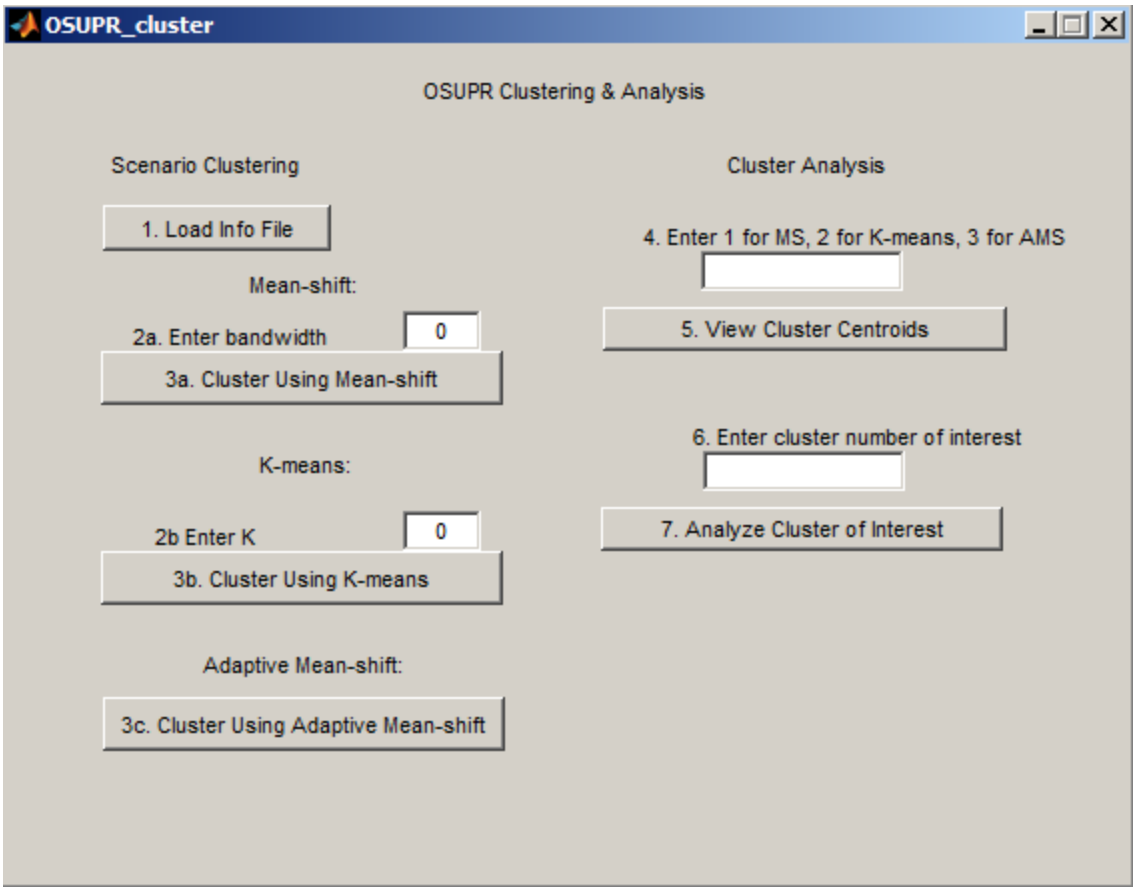


Figure 27. OSUPR Clustering & Analysis Interface

A set of clustering parameters is chosen for the sample set from Section 5.4.2.1 and entered into the interface as shown in Figure 28. For a bandwidth of 0.5, mean-shift returns 6 clusters. A value of $k = 6$ is chosen for k-means, which will produce the same number of clusters. The adaptive mean-shift algorithm returns 8 clusters, which might be expected to yield comparable results to mean-shift.

In Step 4 of Figure 27, the user chooses a clustering method to examine more closely. In Fig. 28, Mean-Shift has been chosen for the sample set, as indicated by the red arrow. Step 5 displays the cluster centroids for that method. Figure 29 shows the values for the sample set for the Mean-Shift method. This allows the user to choose a cluster of

interest. For instance, the user may focus on a low Probability of Detection and pick Cluster 1 (indicated by a red arrow), which has the lowest average value of DP. That cluster also has the highest average PS, indicating that the scenarios in it may be preferred by a proliferator.

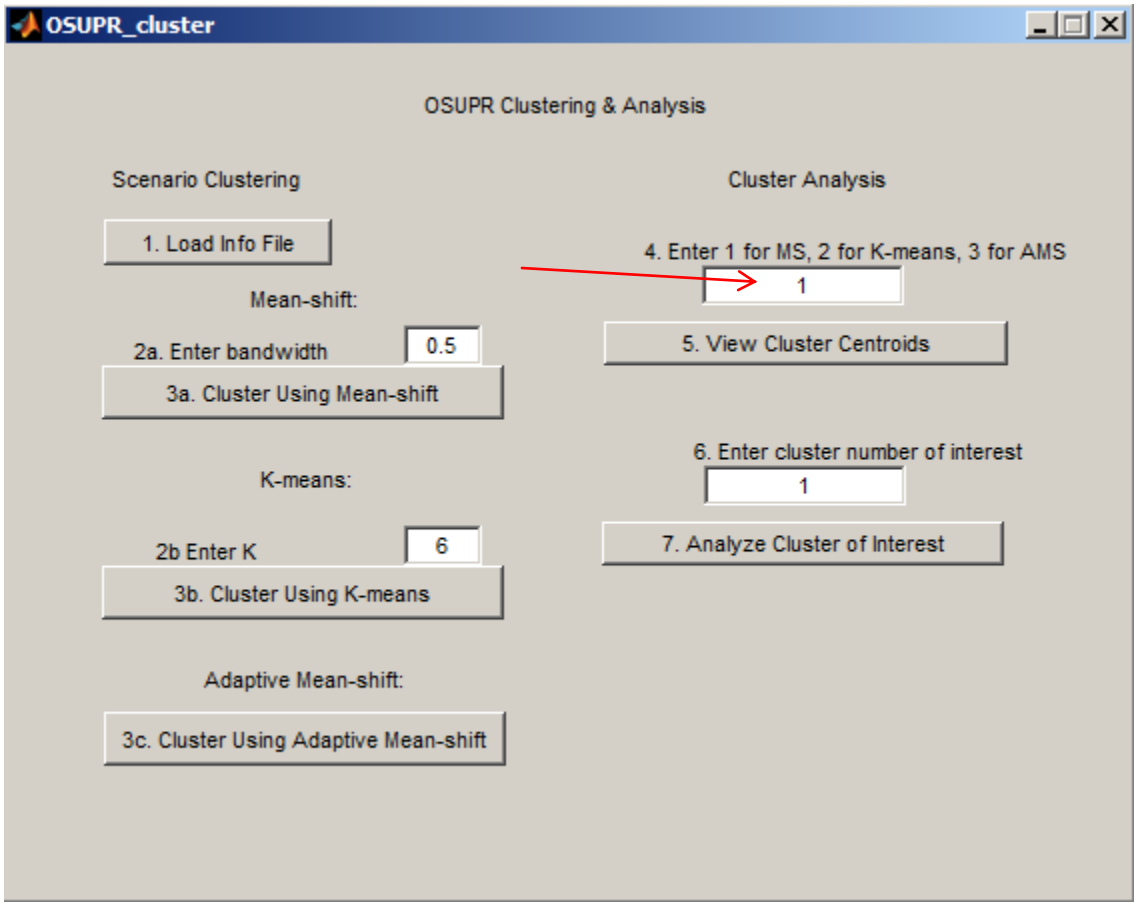


Figure 28. Clustering & Analysis Parameters for Sample Set

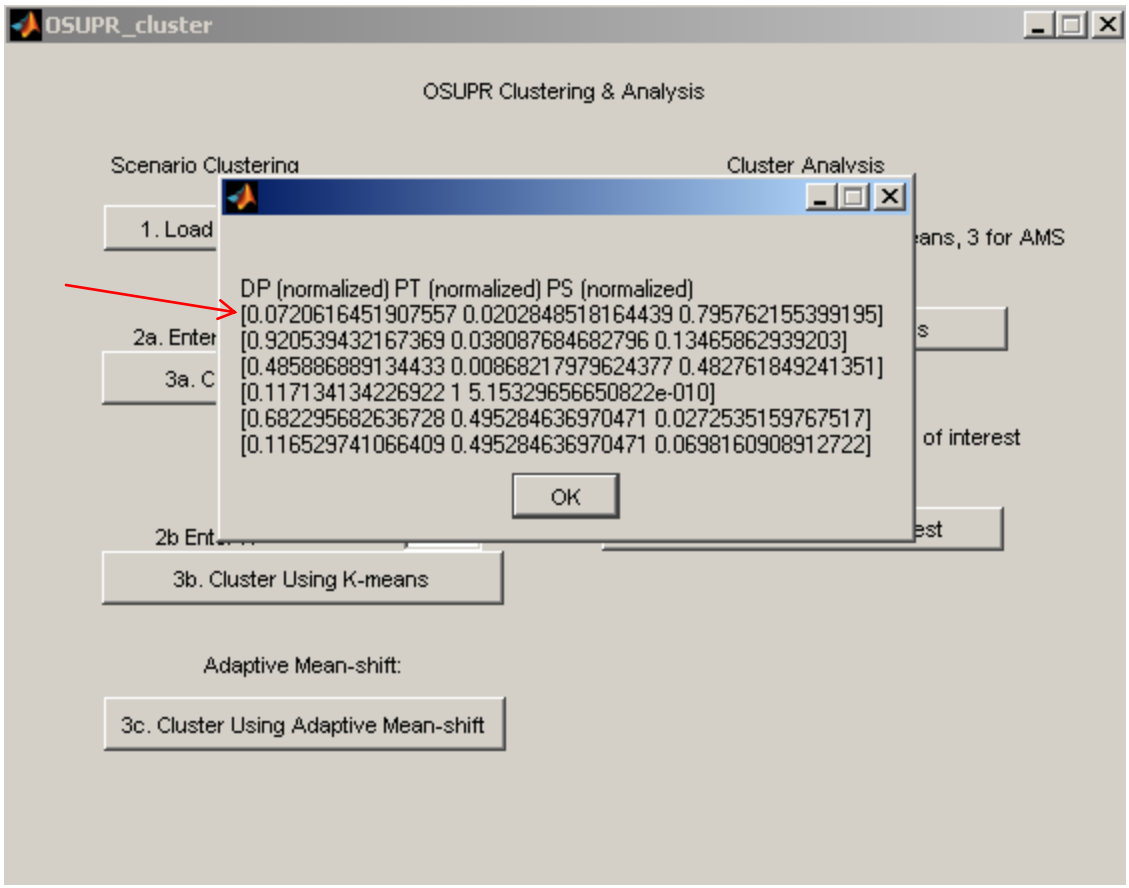
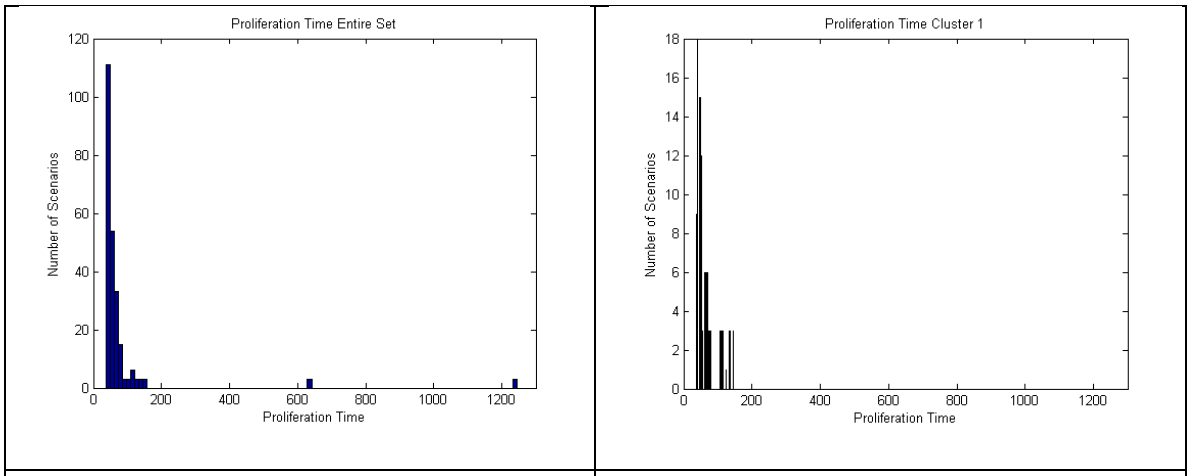


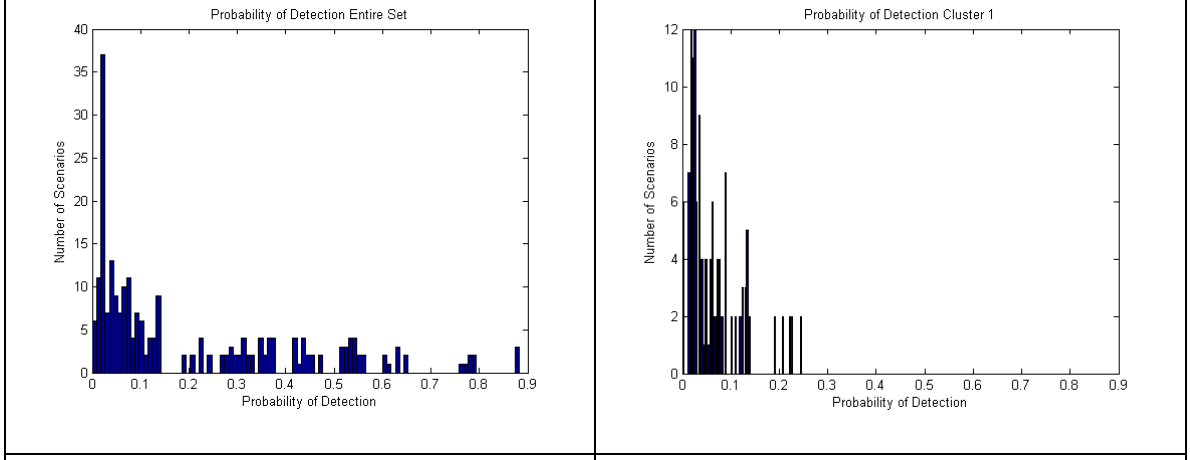
Figure 29. OSUPR Cluster Centroid Display

With Step 7 in Figure 27, OSUPR creates a number of figures that can be used to compare the cluster of interest to the set at large. The first of these, as seen in Figure 30, relate to the outputs of PRCALC. It can be seen in Figure 30 (b) that the values of PT in the cluster of interest are all less than 150 weeks, and so are fairly typical of the entire set (Fig. 30 (a)). The cluster also appears to contain the scenarios with the lowest values of DP (Figure 30 (c) and (d)), and the highest values of PS (Figure 30 (e) and (f)). The next set of histograms, as seen in Figure 31, examine the input parameter decisions (diversion rates and safeguard conditions) that led to inclusion in this cluster.



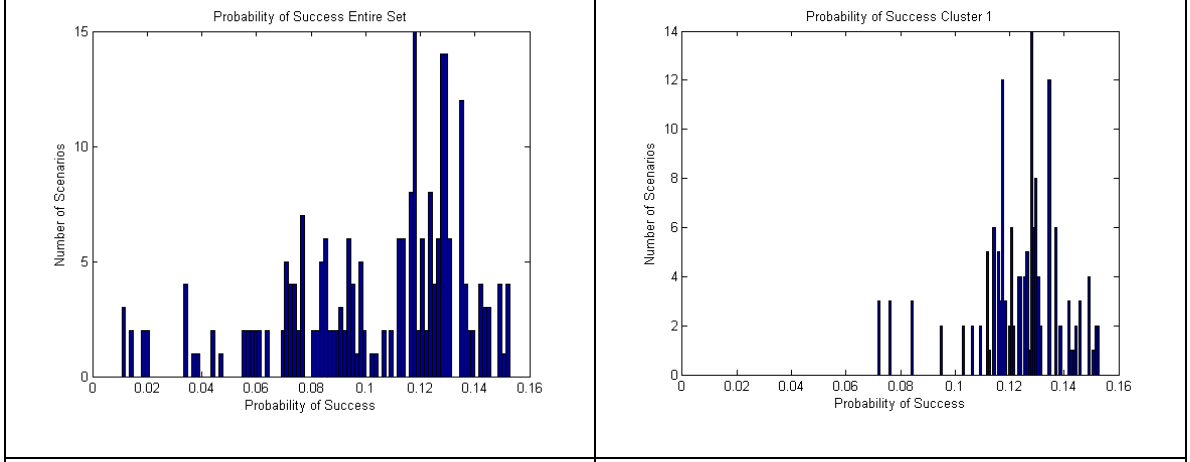
(a) PT – Entire Set

(b) PT – Cluster 1



(c) DP – Entire Set

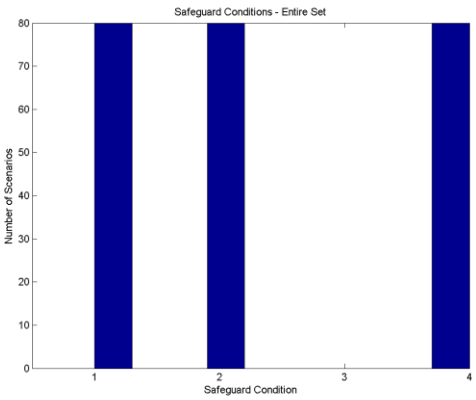
(d) DP – Cluster 1



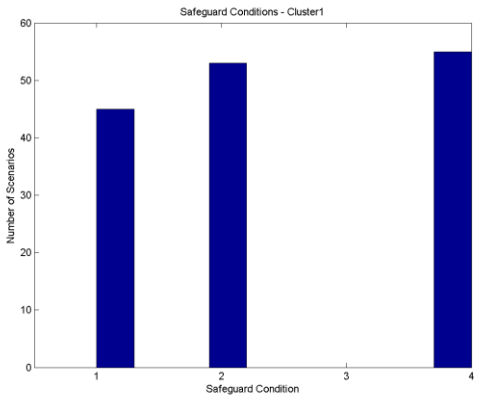
(e) PS – Entire Set

(f) PS – Cluster 1

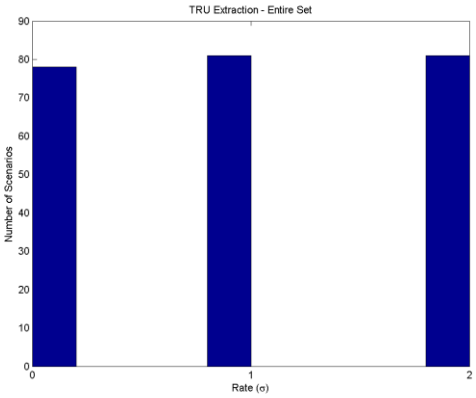
Figure 30. OSUPR Histograms for PRCALC Outputs, Sample Set



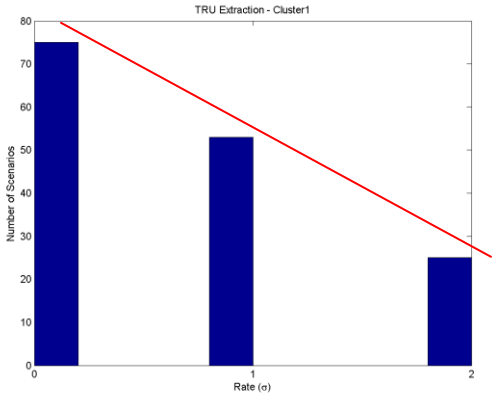
(a) SG Conditions – Entire Set



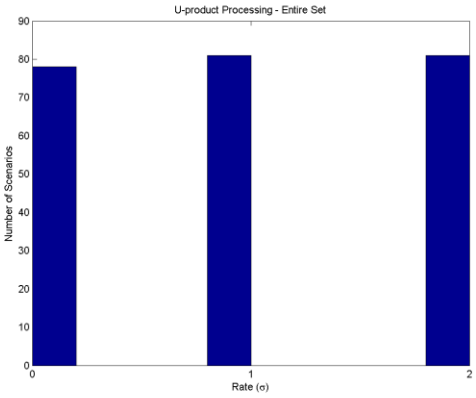
(b) SG Conditions – Cluster 1



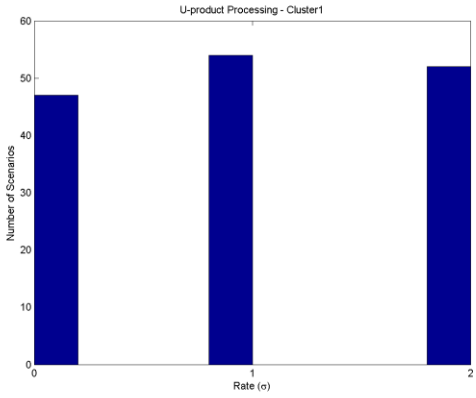
(c) TRU Extraction – Entire Set



(d) TRU Extraction – Cluster 1



(e) U-product Processing – Entire Set



(f) U-product Processing – Cluster 1

Figure 31. OSUPR Histograms for PRCALC Inputs, Sample Set

Frames (a) and (b) of Figure 31 show the safeguards conditions for the entire set (a) and for the cluster of interest (b). The default set of safeguards is represented by Condition 1, the set with Category 2 removed is Condition 2, and the set with Category 4 removed is Condition 4 (see Table 1). Due to the method of scenario creation, there are equal numbers of each condition in the entire set. It can be seen that the cluster of interest (Fig. 31 (b)) has a greater proportion of scenarios with safeguard Categories 2 or 4 removed, versus all safeguards being intact. This indicates that the presence of these categories of safeguards is important to preventing diversion of material.

Similarly to safeguard conditions, in the entire set there are an equal number of scenarios created for each possible diversion rate for each target. This can be seen in Fig. 31 (c) and (e). Figure 31 (c) and (d) represent a target (TRU Extraction) where increased diversion under these experimental conditions leads to scenarios not being included in the cluster of interest. The trend is indicated with a red line across Fig. 31 (d). This indicates that, with all else held equal, increased diversion from TRU Extraction leads to less desirable outcomes for the proliferator. Figure 31 (e) and (f) show a target (U-product Processing) where there is not a clear upward or downward trend, which was the case for 3 of 4 targets in this sample set. In this case, the diversion rate does not seem to affect the outcomes for the proliferator.

Chapter 6: Conclusions & Future Work

This chapter presents the conclusions that may be drawn from the work described in this thesis (6.1). The limitations of the methodology are discussed (6.2), as well as the potential for future research in the area (6.3).

6.1 Conclusions

A methodology was developed to create, batch, and run PRCALC scenarios, and then cluster them for further analysis with regards to sensitive inputs for proliferation resistance. A tool was developed to mechanistically implement the methodology. The tool includes some time-saving features, such as multi-core PRCALC operation and an adaptive-bandwidth variant of mean-shift that requires significantly less time to run. The tool is easy to understand, and a drop-in companion for PRCALC.

Sample analyses of clusters of interest have been provided. In the Deep set of scenarios, it was seen that PIV safeguards were the most critical to affecting the outcome of a proliferation attempt. With PIV safeguards removed, the Electro-refiner and TRU Extraction fuel cycle elements were found to be the most vulnerable elements tested. It was expected that these elements would be particularly sensitive due to the separations that are performed in each element. Because the fuel system under consideration is hypothetical, the purpose of the analyses in this thesis was to demonstrate the

methodology. The process of discovering sensitive inputs, including certain sets of safeguards and target elements of the fuel system, has been described.

6.2 Limitations

The methodology described in this thesis may be applied to other proliferation resistance tools, or different areas of study in which a large number of scenarios must be sorted. The specific software tool developed, however, is narrowly tailored to PRCALC and its data formats. PRCALC models a number of proliferation resistance and physical protection situations including diversion from many different fuel systems, sabotage, and an invasion by force.

6.3 Future Work

The development of a generic PRCALC clustering module could be beneficial. The ability to apply the methodology to fuel systems that currently exist such as the PUREX process or a light water reactor may lead to helpful insights into more efficient allocation of safeguarding resources. The application of a parallelized version of mean-shift within the tool could greatly accelerate the clustering analysis, as mean-shift is currently the slowest algorithm in use by far. A platform- and computer-independent method of multi-core PRCALC operation may lead to the ability to use high performance computing clusters, which are typically under Unix-based operating systems.

References

[1] Bari, R., Therios, P. I., & Whitlock, J. J. "Proliferation Resistance and Physical Protection Evaluation Methodology Development and Applications." Proceedings of the Generation IV International Forum Symposium, Paris, France. 2009.

[2] IAEA. IAEA Safeguards Glossary. 2001 Edition. Vienna: 2002.

[3] International Atomic Energy Agency. Notification of entry into force, Treaty on the Non-Proliferation of Nuclear Weapons. 1970.

[4] United Nations Security Council. S/RES/1172 (1998), Resolution 1172. 1998.

[5] United Nations Security Council. S/RES/2094 (2013), Resolution 2094. 2013.

[6] International Atomic Energy Agency. GOV/2010/49-GC(54)/14, Israeli Nuclear Capabilities. 2010.

[7] International Atomic Energy Agency. GOV/2012/36-GC(56)/11, Application of Safeguards in the Democratic People's Republic of Korea. 2012.

[8] International Atomic Energy Agency. GOV/2005/77, Implementation of the NPT Safeguards Agreement in the Islamic Republic of Iran. 2005.

[9] International Atomic Energy Agency. IAEA, GOV/2006/14, Implementation of the NPT Safeguards Agreement in the Islamic Republic of Iran. 2006.

[10] International Atomic Energy Agency. The Safeguards System of the International Atomic Energy Agency.

[11] Bathke, Charles, Ebbinghaus, Bartley, Collins, Brian, Sleaford, Brad, et al. "The Attractiveness of Materials in Advanced Nuclear Fuel Cycles for Various Proliferation and Theft Scenarios." Nuclear Technology Vol. 179. (2012): 5-30.

[12] Bari, Robert, Jeremy Whitlock, Ike Therios, and Per Peterson. "Proliferation Resistance and Physical Protection Working Group: Methodology and Applications." 2012 Winter Meeting. 2012.

[13] Metcalf, Richard. New Tool for Proliferation Resistance Evaluation Applied to Uranium and Thorium Fueled Fast Reactor Fuel Cycles. MS Thesis. Texas A&M University, 2009.

[14] Dubes, R., & Jain, A. “Clustering Techniques: The User’s Dilemma.” Pattern Recognition, Vol. 8. (1976): 247-260.

[15] Comaniciu, D., Ramesh, V., & Meer, P. “Real-time tracking of non-rigid objects using mean shift.” Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2. (2000): 142-149.

[16] Hamerly, G., & Elkan, C. “Learning the K in K-Means.” Neural Information Processing Systems. 2003.

[17] Mercurio, Davide, Luca Podofillini, Enrico Zio, and Vinh Dang. "Identification and Classification of Dynamic Event Tree Scenarios via Possibilistic Clustering: Application to a Steam Generator Tube Rupture Event." Accident Analysis and Prevention. 41. (2009): 1180-1191.

[18] Mandelli, Diego. Scenario Clustering and Dynamic Probabilistic Risk Assessment. Diss. The Ohio State University, 2011.

[19] Osborn, Douglas. Seamless Level 2/Level 3 Probabilistic Risk Assessment Using Dynamic Event Tree Analysis. Diss. The Ohio State University, 2013.

[20] Yue, Meng, Cheng, Lap-Yan, and Bari, Robert. “Quantitative Assessment of Probabilistic Measures for Proliferation Resistance.” Transactions - American Nuclear Society, 93, (2005): 333-335.

[21] Markov, A. A. “Extension of the limit theorems of probability theory to a sum of variables connected in a chain.” Translated by Petelin, S. Printed in Dynamic Probabilistic Systems, Vol. 1. (1971): 552-576.

[22] Yue, Meng, Cheng, Lap-Yan, and Bari, Robert. “Markov Model Application to Proliferation Risk Reduction of an Advanced Nuclear System.” Proceedings of the INMM 49th Annual Meeting. 2008.

[23] Yue, Meng, Cheng, Lap-Yan, and Bari, Robert. “PRCALC – A Software Tool for Proliferation Resistance and Physical Protection Assessment: Algorithm, Modeling, and User’s Guide.” Brookhaven National Laboratory. 2008.

[24] Yue, Meng, Cheng, Lap-Yan, and Bari, Robert. "Modeling and Evaluating Proliferation Resistance of Nuclear Energy Systems for Strategy Switching Proliferation." Annals of Nuclear Energy. 54. (2012): 11-26.

[25] Fukunaga, Keinosuke, and Hostetler, Larry. "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition." IEEE Transactions on Information Theory. 21. (1975): 32-40.

[26] Georgescu, Bogdan, Ilan Shimshoni, and Peter Meer. "Mean Shift Based Clustering in High Dimensions: A Texture Classification Example." Proceedings of the Ninth IEEE International Conference on Computer Vision. (2003): 456-463.

[27] MacQueen, J. "Some Methods for Classification and Analysis of Multivariate Observations." Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1. (1967): 281-297.

[28] Jankovsky, Zachary, Zamalieva, Daniya, Denning, Richard, Yilmaz, Alper, and Aldemir, Tunc. "A Comparison of Various Clustering Schemes for Proliferation Resistance Measures." Transactions - American Nuclear Society, 109, Part 1. (2013): 261.

Appendix A: OSUPR Scenario Creation Interface Code

The MATLAB code behind the OSUPR scenario creation interface is given in this appendix. It was created using the GUIDE tool in MATLAB, and is seen in Fig. 22:

```
function varargout = OSUPR_input(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @OSUPR_input_OpeningFcn, ...
   'gui_OutputFcn',  @OSUPR_input_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

% Written by Zachary Jankovsky, 2012-2014.
% The Ohio State University

function OSUPR_input_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

global Data;
global chosenstages;
global Stages;
global Folder;
global Elements;
global Rates;
global Name;
global PRMode;
global SG;
global N;
global S;
global Info;
global InfoFolder;
```

```
global Batches;
global infofilename;
global OSUPRGUI_dir
OSUPRGUI_dir=pwd;
addpath(OSUPRGUI_dir)
load('ESFR_Standard.mat')
%Creates strings of stage names for identification of stages in scenarios.
%This is necessary because a given stage (ie Chopping) will not necessarily
%occupy the same index between all scenarios.
Stages=cell(23,1);
Stages(1,1)=cellstr('LWR Spent Fuel Storage');
Stages(2,1)=cellstr('Storage Basket: Fresh Fuel');
Stages(3,1)=cellstr('LWR Transfer Port');
Stages(4,1)=cellstr('ESFR Reactor');
Stages(5,1)=cellstr('LWR Transfer');
Stages(6,1)=cellstr('Storage Basket: Spent Fuel');
Stages(7,1)=cellstr('ESFR Transfer Port');
Stages(8,1)=cellstr('ESFR Transfer');
Stages(9,1)=cellstr('Staging/Washing Area');
Stages(10,1)=cellstr('Staging/Washing Transfer Port');
Stages(11,1)=cellstr('Staging/Washing Transfer');
Stages(12,1)=cellstr('SF & NF Storage Cell');
Stages(13,1)=cellstr('SF & NF Transfer Port');
Stages(14,1)=cellstr('SF & NF Transfer');
Stages(15,1)=cellstr('LWR SF Disassembly');
Stages(16,1)=cellstr('ESFR SF Disassembly');
Stages(17,1)=cellstr('Chopping');
Stages(18,1)=cellstr('Electro-refiner');
Stages(19,1)=cellstr('U-product Processing');
Stages(20,1)=cellstr('TRU Extraction');
Stages(21,1)=cellstr('Product Preparation');
Stages(22,1)=cellstr('Pin Fabrication');
Stages(23,1)=cellstr('Assembly');
PRMode=1;
SG=[1 0 0 0];
Elements=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
N=0;
S=0;
return;

function varargout = OSUPR_input_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% Toggle diversion from 1.
function radiobutton3_Callback(hObject, eventdata, handles)
global Elements;
Elements(1)=0;
if get(hObject,'Value')
   Elements(1)=1;
else
   Elements(1)=0;
end
return

% Toggle diversion from 2.
```

```
function radiobutton4_Callback(hObject, eventdata, handles)
global Elements;
Elements(2)=0;
if get(hObject,'Value')
   Elements(2)=1;
else
   Elements(2)=0;
end
return

% Toggle diversion from 3.
function radiobutton5_Callback(hObject, eventdata, handles)
global Elements;
Elements(3)=0;
if get(hObject,'Value')
   Elements(3)=1;
else
   Elements(3)=0;
end
return

% Toggle diversion from 4.
function radiobutton6_Callback(hObject, eventdata, handles)
global Elements;
Elements(4)=0;
if get(hObject,'Value')
   Elements(4)=1;
else
   Elements(4)=0;
end
return

% Toggle diversion from 5.
function radiobutton7_Callback(hObject, eventdata, handles)
global Elements;
Elements(5)=0;
if get(hObject,'Value')
   Elements(5)=1;
else
   Elements(5)=0;
end
return

% Toggle diversion from 6.
function radiobutton8_Callback(hObject, eventdata, handles)
global Elements;
Elements(6)=0;
if get(hObject,'Value')
   Elements(6)=1;
else
   Elements(6)=0;
end
return

% Toggle diversion from 7.
```

```
function radiobutton9_Callback(hObject, eventdata, handles)
global Elements;
Elements(7)=0;
if get(hObject,'Value')
   Elements(7)=1;
else
   Elements(7)=0;
end
return

% Toggle diversion from 8.
function radiobutton10_Callback(hObject, eventdata, handles)
global Elements;
Elements(8)=0;
if get(hObject,'Value')
   Elements(8)=1;
else
   Elements(8)=0;
end
return

% Toggle diversion from 9.
function radiobutton11_Callback(hObject, eventdata, handles)
global Elements;
Elements(9)=0;
if get(hObject,'Value')
   Elements(9)=1;
else
   Elements(9)=0;
end
return

% Toggle diversion from 10.
function radiobutton12_Callback(hObject, eventdata, handles)
global Elements;
Elements(10)=0;
if get(hObject,'Value')
   Elements(10)=1;
else
   Elements(10)=0;
end
return

% Toggle diversion from 11.
function radiobutton13_Callback(hObject, eventdata, handles)
global Elements;
Elements(11)=0;
if get(hObject,'Value')
   Elements(11)=1;
else
   Elements(11)=0;
end
return

% Toggle diversion from 12.
```

```
function radiobutton14_Callback(hObject, eventdata, handles)
global Elements;
Elements(12)=0;
if get(hObject,'Value')
   Elements(12)=1;
else
   Elements(12)=0;
end
return

% Toggle diversion from 13.
function radiobutton15_Callback(hObject, eventdata, handles)
global Elements;
Elements(13)=0;
if get(hObject,'Value')
   Elements(13)=1;
else
   Elements(13)=0;
end
return

% Toggle diversion from 14.
function radiobutton16_Callback(hObject, eventdata, handles)
global Elements;
Elements(14)=0;
if get(hObject,'Value')
   Elements(14)=1;
else
   Elements(14)=0;
end
return

% Toggle diversion from 15.
function radiobutton17_Callback(hObject, eventdata, handles)
global Elements;
Elements(15)=0;
if get(hObject,'Value')
   Elements(15)=1;
else
   Elements(15)=0;
end
return

% Toggle diversion from 16.
function radiobutton18_Callback(hObject, eventdata, handles)
global Elements;
Elements(16)=0;
if get(hObject,'Value')
   Elements(16)=1;
else
   Elements(16)=0;
end
return

% Toggle diversion from 17.
```

```
function radiobutton19_Callback(hObject, eventdata, handles)
global Elements;
Elements(17)=0;
if get(hObject,'Value')
   Elements(17)=1;
else
   Elements(17)=0;
end
return

% Toggle diversion from 18.
function radiobutton20_Callback(hObject, eventdata, handles)
global Elements;
Elements(18)=0;
if get(hObject,'Value')
   Elements(18)=1;
else
   Elements(18)=0;
end
return

% Toggle diversion from 19.
function radiobutton21_Callback(hObject, eventdata, handles)
global Elements;
Elements(19)=0;
if get(hObject,'Value')
   Elements(19)=1;
else
   Elements(19)=0;
end
return

% Toggle diversion from 20.
function radiobutton22_Callback(hObject, eventdata, handles)
global Elements;
Elements(20)=0;
if get(hObject,'Value')
   Elements(20)=1;
else
   Elements(20)=0;
end
return

% Toggle diversion from 21.
function radiobutton23_Callback(hObject, eventdata, handles)
global Elements;
Elements(21)=0;
if get(hObject,'Value')
   Elements(21)=1;
else
   Elements(21)=0;
end
return

% Toggle diversion from 22.
```

```
function radiobutton24_Callback(hObject, eventdata, handles)
global Elements;
Elements(22)=0;
if get(hObject,'Value')
   Elements(22)=1;
else
   Elements(22)=0;
end
return

% Toggle diversion from 23.
function radiobutton25_Callback(hObject, eventdata, handles)
global Elements;
Elements(23)=0;
if get(hObject,'Value')
   Elements(23)=1;
else
   Elements(23)=0;
end
return

% Set diversion rates.
function edit1_Callback(hObject, eventdata, handles)
global Rates;
Rates=str2num(get(hObject,'String'));
return

% Set diversion rates.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% Set series name.
function edit2_Callback(hObject, eventdata, handles)
global Name;
Name=get(hObject,'String');
return

% Set series name.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% Category 2 SG removal toggle.
function radiobutton26_Callback(hObject, eventdata, handles)
global SG;
if get(hObject,'Value')
   SG(2)=1;
else
   SG(2)=0;
end
return
```

```
% Category 3 SG removal toggle.
function radiobutton27_Callback(hObject, eventdata, handles)
global SG;
if get(hObject,'Value')
   SG(3)=1;
else
   SG(3)=0;
end
return

% Category 4 SG removal toggle.
function radiobutton28_Callback(hObject, eventdata, handles)
global SG;
if get(hObject,'Value')
   SG(4)=1;
else
   SG(4)=0;
end
return

% Estimate number of scenarios.
function pushbutton1_Callback(hObject, eventdata, handles)
global Elements;
global Rates;
global SG;
global S;
global N;
S=0;
S=sum(SG);
E=0;
N=0;
E=sum(Elements);
N=size(npermutek(Rates,E),1);
msgbox(cat(2, {'There will be approximately this many scenarios: '},num2str(S*N)))
return

% Choose a base case to use for the input files.
%function pushbutton4_Callback(hObject, eventdata, handles)
%global Data;
%uiopen('Select model input file.mat');
%return

% Choose a folder for the input files.
%function pushbutton2_Callback(hObject, eventdata, handles)
%global Folder;
%Folder=uigetdir(pwd,'Folder to save series to?');
%return

% Choose a folder for the info file.
function pushbutton7_Callback(hObject, eventdata, handles)
global InfoFolder;
global Folder;
global Name;
InfoFolder=strcat(uigetdir(pwd,'Folder to save info file to?'),'\');
```

```
mkdir(InfoFolder,strcat(Name,'_files'))
Folder=strcat(InfoFolder,Name,'_files');
return

% Create input files.
function pushbutton3_Callback(hObject, eventdata, handles)
global PRMode;
global Data;
global Folder;
global Name;
global Elements;
global Rates;
global SG;
global Stages;
if PRMode==1
   chosenstages=cell(1,23);
   a=1;
   for i=1:23
      if Elements(i)==1
         chosenstages(a)=Stages(i);
      end
      a=a+1;
   end
   %Delete stages that are not chosen for diversion
   chosenstages(cellfun(@isempty,chosenstages))=[];
   Matrix=npermutek(Rates,size(chosenstages,2));
   Matrix=Matrix';
   %Handle a scenario with all zero rates
   if sum(Matrix(:,1))==0
      Matrix(:,1)=[];
   end
   h = waitbar(0,'Creating first set of scenarios');
   for i=1:length(Matrix)
      waitbar(i/length(Matrix),h)
      for n=1:size(chosenstages,2)
         Data.AutoData.ParameterSequence(n,3)=chosenstages(n);
         Data.AutoData.ParameterSequence(n,4)=cellstr(num2str(Matrix(n,i)));
      end

      %Writes other AutoData information to each line in ParameterSequence.
      %Default time is 50 weeks.
      for n=1:size(chosenstages,2)
         Data.AutoData.ParameterSequence(n,1)=cellstr('1');
         Data.AutoData.ParameterSequence(n,2)=cellstr('DR');
         Data.AutoData.ParameterSequence(n,5)=cellstr('50');
         Data.AutoData.ParameterSequence(n,6)=cellstr(' ');
      end
      %Eliminates lines from ParameterSequence with rate (sigma) of 0.
      keepindex=find(str2num(char(Data.AutoData.ParameterSequence(:,4))));
      keep=cell(length(keepindex),6);
      for n=1:length(keepindex)
         keep(n,:)=Data.AutoData.ParameterSequence(keepindex(n),:);
      end
      Data.AutoData.ParameterSequence=keep;
      %Saves each scenario file. Leading zeros are used where necessary to
```

```
        %make incremental numbered name 5 digits. This can be expanded for
        %larger sets.
        savefile=cell2mat([cellstr(Folder) '\' cellstr(Name) num2str(i,'%06d') '.mat']);
        save(savefile,'Data');
    end
end
delete(h)

if SG(2)==1
    h = waitbar(0,'Creating set of scenarios for SG condition 2');
    folder2=strcat(Folder,'2');
    copyfile(Folder,folder2,'f')
    dirlis2=dir(folder2);
    lengthname=length(dirlis2(3,1).name);
    ntosafeguard2=size(dirlis2,1);
    for i=3:size(dirlis2,1)
        waitbar(i/size(dirlis2,1),h)
        oldname2=dirlis2(i,1).name;
        newname2=dirlis2(i,1).name;
        newname2(lengthname-3:lengthname+2)='_2.mat';
        movefile(strcat(folder2,'\',oldname2),strcat(folder2,'\',newname2))
    end
    dirlis2=dir(folder2);
    delete(h)
end

if SG(3)==1
    h = waitbar(0,'Creating set of scenarios for SG condition 3');
    folder3=strcat(Folder,'3');
    copyfile(Folder,folder3,'f')
    dirlis3=dir(folder3);
    lengthname=length(dirlis3(3,1).name);
    ntosafeguard3=size(dirlis3,1);
    for i=3:size(dirlis3,1)
        waitbar(i/size(dirlis3,1),h)
        oldname3=dirlis3(i,1).name;
        newname3=dirlis3(i,1).name;
        newname3(lengthname-3:lengthname+2)='_3.mat';
        movefile(strcat(folder3,'\',oldname3),strcat(folder3,'\',newname3))
    end
    dirlis3=dir(folder3);
    delete(h)
end

if SG(4)==1
    h = waitbar(0,'Creating set of scenarios for SG condition 4');
    folder4=strcat(Folder,'4');
    copyfile(Folder,folder4,'f')
    dirlis4=dir(folder4);
    lengthname=length(dirlis4(3,1).name);
    ntosafeguard4=size(dirlis4,1);
    for i=3:size(dirlis4,1)
        waitbar(i/size(dirlis4,1),h)
        oldname4=dirlis4(i,1).name;
        newname4=dirlis4(i,1).name;
```

```
        newname4(lengthname-3:lengthname+2)='_4.mat';
        movefile(strcat(folder4,'\',oldname4),strcat(folder4,'\',newname4))
    end
    dirlis4=dir(folder4);
    delete(h)
end

if SG(2)==1
    h = waitbar(0,'Modifying scenarios for SG condition 2');
    for n=3:ntosafeguard2
        waitbar(n/ntosafeguard2,h)
        load(strcat(folder2,'\',dirlis2(n,1).name))
        for a=1:length(Data.AutoData.ParameterSequence(:,2))
            %Sets safeguards for electro-refiner if there is diversion.
            if strcmp(Stages(18,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item(i),'2',1)
                        Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for ESFR SF Disassembly if there is diversion.
            if strcmp(Stages(16,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item(i),'2',1)
                        Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for Chopping if there is diversion.
            if strcmp(Stages(17,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item(i),'2',1)
                        Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for TRU Extraction if there is diversion.
            if strcmp(Stages(20,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item(i),'2',1)
                        Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for Pin Fabrication if there is diversion.
            if strcmp(Stages(22,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item(i),'2',1)
                        Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item(i)=[];
                    end
```

```
    end
end

%Sets safeguards for Storage Basket: Fresh Fuel if there is diversion.
if strcmp(Stages(2,1),Data.AutoData.ParameterSequence(a,3))
    for i=size(Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item,2):-1:1
        if strncmpi(Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item(i),'2',1)
            Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item(i)=[];
        end
    end
end

%Sets safeguards for LWR Spent Fuel Storage if there is diversion.
if strcmp(Stages(1,1),Data.AutoData.ParameterSequence(a,3))
    for i=size(Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item,2):-1:1
        if strncmpi(Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item(i),'2',1)
            Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item(i)=[];
        end
    end
end

%Sets safeguards for Staging/Washing Area if there is diversion.
if strcmp(Stages(9,1),Data.AutoData.ParameterSequence(a,3))
    for i=size(Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item,2):-1:1
        if strncmpi(Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item(i),'2',1)
            Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item(i)=[];
        end
    end
end

%Sets safeguards for Staging/Washing Transfer if there is diversion.
if strcmp(Stages(11,1),Data.AutoData.ParameterSequence(a,3))
    for i=size(Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item,2):-1:1
        if strncmpi(Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item(i),'2',1)
            Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item(i)=[];
        end
    end
end

%Sets safeguards for SF & NF Storage Cell if there is diversion.
if strcmp(Stages(12,1),Data.AutoData.ParameterSequence(a,3))
    for i=size(Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item,2):-1:1
        if strncmpi(Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item(i),'2',1)
            Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item(i)=[];
        end
    end
end

%Sets safeguards for SF & NF Transfer if there is diversion.
if strcmp(Stages(14,1),Data.AutoData.ParameterSequence(a,3))
    for i=size(Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item,2):-1:1
        if strncmpi(Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item(i),'2',1)
            Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item(i)=[];
        end
    end
```

```
end

%Sets safeguards for LWR SF Disassembly if there is diversion.
if strcmp(Stages(15,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item(i),'2',1)
         Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for U-product Processing if there is diversion.
if strcmp(Stages(19,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item(i),'2',1)
         Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Product Preparation if there is diversion.
if strcmp(Stages(21,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item(i),'2',1)
         Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Assembly if there is diversion.
if strcmp(Stages(23,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item(i),'2',1)
         Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Transfer if there is diversion.
if strcmp(Stages(5,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item(i),'2',1)
         Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Transfer Port if there is diversion.
if strcmp(Stages(3,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item(i),'2',1)
         Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end
```

```
    %Sets safeguards for ESFR Reactor if there is diversion.
    if strcmp(Stages(4,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item(i),'2',1)
                Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item(i)=[];
            end
        end
    end

    %Sets safeguards for ESFR Transfer if there is diversion.
    if strcmp(Stages(8,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item(i),'2',1)
                Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for ESFR Transfer Port if there is diversion.
    if strcmp(Stages(7,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item(i),'2',1)
                Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for SF & NF Transfer Port if there is diversion.
    if strcmp(Stages(13,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item(i),'2',1)
                Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for Storage Basket: Spent Fuel if there is diversion.
    if strcmp(Stages(6,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item(i),'2',1)
                Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item(i)=[];
            end
        end
    end

    %Sets safeguards for Staging/Washing Transfer Port if there is diversion.
    if strcmp(Stages(10,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item(i),'2',1)
                Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end
end
```

```
        save(strcat(folder2,'\',dirlis2(n,1).name),'Data')
    end
    delete(h)
end

if SG(3)==1
    h = waitbar(0,'Modifying scenarios for SG condition 3');
    for n=3:ntosafeguard3
        waitbar(n/ntosafeguard3,h)
        load(strcat(folder3,'\',dirlis3(n,1).name))
        for a=1:length(Data.AutoData.ParameterSequence(:,2))
            %Sets safeguards for electro-refiner if there is diversion.
            if strcmp(Stages(18,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item(i),'3',1)
                        Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for ESFR SF Disassembly if there is diversion.
            if strcmp(Stages(16,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item(i),'3',1)
                        Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for Chopping if there is diversion.
            if strcmp(Stages(17,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item(i),'3',1)
                        Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for TRU Extraction if there is diversion.
            if strcmp(Stages(20,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item(i),'3',1)
                        Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for Pin Fabrication if there is diversion.
            if strcmp(Stages(22,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item(i),'3',1)
                        Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end
```

```
%Sets safeguards for Storage Basket: Fresh Fuel if there is diversion.
if strcmp(Stages(2,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item(i),'3',1)
         Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Spent Fuel Storage if there is diversion.
if strcmp(Stages(1,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Staging/Washing Area if there is diversion.
if strcmp(Stages(9,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Staging/Washing Transfer if there is diversion.
if strcmp(Stages(11,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for SF & NF Storage Cell if there is diversion.
if strcmp(Stages(12,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for SF & NF Transfer if there is diversion.
if strcmp(Stages(14,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end
```

```
%Sets safeguards for LWR SF Disassembly if there is diversion.
if strcmp(Stages(15,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for U-product Processing if there is diversion.
if strcmp(Stages(19,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Product Preparation if there is diversion.
if strcmp(Stages(21,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Assembly if there is diversion.
if strcmp(Stages(23,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Transfer if there is diversion.
if strcmp(Stages(5,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Transfer Port if there is diversion.
if strcmp(Stages(3,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item(i),'3',1)
         Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end
```

```
    %Sets safeguards for ESFR Reactor if there is diversion.
    if strcmp(Stages(4,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item(i),'3',1)
                Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item(i)=[];
            end
        end
    end

    %Sets safeguards for ESFR Transfer if there is diversion.
    if strcmp(Stages(8,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item(i),'3',1)
                Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for ESFR Transfer Port if there is diversion.
    if strcmp(Stages(7,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item(i),'3',1)
                Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for SF & NF Transfer Port if there is diversion.
    if strcmp(Stages(13,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item(i),'3',1)
                Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for Storage Basket: Spent Fuel if there is diversion.
    if strcmp(Stages(6,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item(i),'3',1)
                Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item(i)=[];
            end
        end
    end

    %Sets safeguards for Staging/Washing Transfer Port if there is diversion.
    if strcmp(Stages(10,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item(i),'3',1)
                Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end
end
```

```
        save(strcat(folder3,'\',dirlis3(n,1).name),'Data')
    end
    delete(h)
end

if SG(4)==1
    h = waitbar(0,'Modifying scenarios for SG condition 4');
    for n=3:ntosafeguard4
        waitbar(n/ntosafeguard4,h)
        load(strcat(folder4,'\',dirlis4(n,1).name))
        for a=1:length(Data.AutoData.ParameterSequence(:,2))
            %Sets safeguards for electro-refiner if there is diversion.
            if strcmp(Stages(18,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item(i),'4',1)
                        Data.DiversionInfo(1,14).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for ESFR SF Disassembly if there is diversion.
            if strcmp(Stages(16,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item(i),'4',1)
                        Data.DiversionInfo(1,12).Safeguards.Approaches(1,2).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for Chopping if there is diversion.
            if strcmp(Stages(17,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item(i),'4',1)
                        Data.DiversionInfo(1,13).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for TRU Extraction if there is diversion.
            if strcmp(Stages(20,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item(i),'4',1)
                        Data.DiversionInfo(1,15).Safeguards.Approaches(1,2).Item(i)=[];
                    end
                end
            end

            %Sets safeguards for Pin Fabrication if there is diversion.
            if strcmp(Stages(22,1),Data.AutoData.ParameterSequence(a,3))
                for i=size(Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item,2):-1:1
                    if strncmpi(Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item(i),'4',1)
                        Data.DiversionInfo(1,17).Safeguards.Approaches(1,1).Item(i)=[];
                    end
                end
            end
```

```
%Sets safeguards for Storage Basket: Fresh Fuel if there is diversion.
if strcmp(Stages(2,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item(i),'4',1)
         Data.DiversionInfo(1,1).Safeguards.Approaches(1,2).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Spent Fuel Storage if there is diversion.
if strcmp(Stages(1,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,1).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Staging/Washing Area if there is diversion.
if strcmp(Stages(9,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,6).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Staging/Washing Transfer if there is diversion.
if strcmp(Stages(11,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,8).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for SF & NF Storage Cell if there is diversion.
if strcmp(Stages(12,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,9).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for SF & NF Transfer if there is diversion.
if strcmp(Stages(14,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,11).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end
```

```
%Sets safeguards for LWR SF Disassembly if there is diversion.
if strcmp(Stages(15,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,12).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for U-product Processing if there is diversion.
if strcmp(Stages(19,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,15).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Product Preparation if there is diversion.
if strcmp(Stages(21,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,16).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for Assembly if there is diversion.
if strcmp(Stages(23,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,18).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Transfer if there is diversion.
if strcmp(Stages(5,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,3).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end

%Sets safeguards for LWR Transfer Port if there is diversion.
if strcmp(Stages(3,1),Data.AutoData.ParameterSequence(a,3))
   for i=size(Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item,2):-1:1
      if strncmpi(Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item(i),'4',1)
         Data.DiversionInfo(1,2).Safeguards.Approaches(1,1).Item(i)=[];
      end
   end
end
```

```
    %Sets safeguards for ESFR Reactor if there is diversion.
    if strcmp(Stages(4,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item(i),'4',1)
                Data.DiversionInfo(1,2).Safeguards.Approaches(1,2).Item(i)=[];
            end
        end
    end

    %Sets safeguards for ESFR Transfer if there is diversion.
    if strcmp(Stages(8,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item(i),'4',1)
                Data.DiversionInfo(1,5).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for ESFR Transfer Port if there is diversion.
    if strcmp(Stages(7,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item(i),'4',1)
                Data.DiversionInfo(1,4).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for SF & NF Transfer Port if there is diversion.
    if strcmp(Stages(13,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item(i),'4',1)
                Data.DiversionInfo(1,10).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end

    %Sets safeguards for Storage Basket: Spent Fuel if there is diversion.
    if strcmp(Stages(6,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item(i),'4',1)
                Data.DiversionInfo(1,3).Safeguards.Approaches(1,2).Item(i)=[];
            end
        end
    end

    %Sets safeguards for Staging/Washing Transfer Port if there is diversion.
    if strcmp(Stages(10,1),Data.AutoData.ParameterSequence(a,3))
        for i=size(Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item,2):-1:1
            if strncmpi(Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item(i),'4',1)
                Data.DiversionInfo(1,7).Safeguards.Approaches(1,1).Item(i)=[];
            end
        end
    end
end
```

```
        save(strcat(folder4,'\',dirlis4(n,1).name),'Data')
    end
    delete(h)
end

h = waitbar(0,'Moving files');
if SG(2)==1
    copyfile(folder2,Folder,'f')
    rmdir(folder2,'s')
    waitbar(1/4,h)
end

if SG(3)==1
    copyfile(folder3,Folder,'f')
    rmdir(folder3,'s')
    waitbar(2/4,h)
end

if SG(4)==1
    copyfile(folder4,Folder,'f')
    rmdir(folder4,'s')
    waitbar(3/4,h)
end
delete(h)

global Info;
global InfoFolder;
global infofilename;

dirlist=dir(Folder);
dirlist(1)=[];
dirlist(1)=[];
dirlist=dirlist';
num_scen=size(dirlist,2);
Info.Names=cell(num_scen,1);
Info.Rates=struct;
Info.Elements=Elements';
Info.Stages=Stages;
sa=length(Matrix);
if sum(SG)==2
    Matrix=[Matrix,Matrix];
end
if sum(SG)==3
    Matrix=[Matrix,Matrix,Matrix];
end
if sum(SG)==4
    Matrix=[Matrix,Matrix,Matrix,Matrix];
end

Info.Matrix=Matrix';
Info.SG=SG;
SGCond=ones(sa,1);
if Info.SG(2)==1
    lsgcond=length(SGCond);
    SGCond(lsgcond+1:sa*2)=2;
```

```
end
if Info.SG(3)==1
   lsgcond=length(SGCond);
   SGCond(lsgcond+1:lsgcond+sa)=3;
end
if Info.SG(4)==1
   lsgcond=length(SGCond);
   SGCond(lsgcond+1:lsgcond+sa)=4;
end
Info.SGCond=SGCond;

h = waitbar(0,'Creating Info File');
for n=1:num_scen
   waitbar(n/num_scen,h)
   load(strcat(Folder,'\',dirlist(1,n).name))
   Info.Names(n,1)=cellstr(dirlist(1,n).name);
   fieldname=(dirlist(1,n).name);
   fieldname=fieldname(1:end-4);
   prolifsteps=size(Data.AutoData.ParameterSequence,1);
   Info.Rates.(fieldname)=cell(prolifsteps,2);

   for i=1:prolifsteps
      Info.Rates.(fieldname)(i,1)=Data.AutoData.ParameterSequence(i,3);
      Info.Rates.(fieldname)(i,2)=Data.AutoData.ParameterSequence(i,4);
   end
end
delete(h)

infofilename=strcat(Name,'_info.mat');
%Save info file
savefile=strcat(InfoFolder,infofilename);
save(savefile,'Info');
msgbox('Done creating scenario files and info file.')
return

% Help dialog.
function pushbutton5_Callback(hObject, eventdata, handles)
msgbox(sprintf(['This interface allows creation of PRCALC input files.\n' ...
   '1. Choose a series name, with no spaces.\n' ...
   '2. Enter a set of diversion rates, separating them with spaces.\n' ...
   '3. Choose additional safeguard conditions.\n' ...
   '4. Choose stage elements for diversion.\n' ...
   '5. Calculate the approximate number of scenarios.\n' ...
   '6. Choose a folder where the input files and info file will be created.\n' ...
   '7. Create input files & info file.\n' ...
   '8. Query number of cores available.\n' ...
   '9. Enter number of batches to use.\n' ...
   '10. Create batch files.\n' ...
   '11. Run PRCALC.']))
return

% Returns number of cores available for parallel running.
function pushbutton9_Callback(hObject, eventdata, handles)
global cores;
cores=java.lang.Runtime.getRuntime().availableProcessors;
```

```
msgbox(cat(2, {'There are this many cores available: '},num2str(cores)))
return

% Set number of batches.
function edit3_Callback(hObject, eventdata, handles)

global Batches;
Batches=1;
Batches=str2double(get(hObject,'String'));
return

% Set number of batches.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
return

% Creates batch files.
function pushbutton10_Callback(hObject, eventdata, handles)
global Batches;
global cores;
global Folder;
global Elements;
global Stages;
global InfoFolder;
global Name;
global infofilename;
direc=pwd;
cd(InfoFolder)
batchname=cell(1,Batches);
filename=zeros(1,Batches);
dirlisting=dir(Folder);
dirlisting(1)=[];
dirlisting(1)=[];
ntobatch=size(dirlisting,1);
BatchAssign=zeros(ntobatch,1);

for i=1:ntobatch
   BatchAssign(i)=ceil((i/ntobatch)*Batches);
end

for i=1:Batches
   batchname(i)=cellstr(strcat(Name,'_batch_',num2str(i),'.m'));
   fullbatchname(i)=cellstr(strcat(InfoFolder,Name,'_batch_',num2str(i),'.m'));
end

h = waitbar(0,'Saving Batch Files');
for n=1:Batches
   waitbar(n/Batches,h)
   sfile=fopen(cell2mat(batchname(n)), 'w');
   for i=ceil(1+(n-1)*(ntobatch/Batches)):ceil((ntobatch/Batches)*n)
      fprintf(sfile,'%s',Folder);
      fprintf(sfile,'%s','\');
```

```
        fprintf(sfile,'%s',dirlisting(i,1).name);
        fprintf(sfile,'\n');
    end
    fclose(sfile);
end
delete(h)
load(infofilename)
Info.BatchAssign=BatchAssign;
Info.Batches.Cores=cores;
Info.Batches.Name=batchname';
Info.Batches.Path=InfoFolder;
ConfirmFile=cell(4,1);
for i=1:length(Info.Batches.Name)
    Info.ConfirmFile(i)=cellstr(strcat(infofilename(1:end-4),num2str(i)));
end
%Save info file
savefile=strcat(InfoFolder,infofilename);
save(savefile,'Info');
cd(direc)
msgbox('Done creating batch files.')
return

function launch_a_bat_file(batFile)
startInfo = System.Diagnostics.ProcessStartInfo('cmd.exe', sprintf('/c "%s"', batFile));
startInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;  %// if you want it invisible
proc = System.Diagnostics.Process.Start(startInfo);
if isempty(proc)
    error('Failed to launch process');
end
while true
    if proc.HasExited
        break
    end
    pause(.1);
end

% Runs PRCALC
function pushbutton11_Callback(hObject, eventdata, handles)
global InfoFolder;
global Info;
global infofilename;
global OSUPRGUI_dir;
global Batches;
cd(InfoFolder)
tic;
load(strcat(InfoFolder,infofilename))
BatchName=Info.Batches.Name;

listname=strcat(infofilename(1:end-4),'_list.bat');
fid = fopen(listname, 'wt');
for i=1:Batches
    listcontents=char(strcat('matlab -nosplash -sd "',OSUPRGUI_dir,'" -r
"PRCALCDistRun("",",infofilename,"",',"",InfoFolder,"',',num2str(i),')""));
    if i~=Batches
        fprintf(fid,'%s\n',listcontents);
```

```
    end
    if i==Batches
        fprintf(fid,'%s',listcontents);
    end
end
fclose(fid);
launch_a_bat_file(listname)

for i=1:length(Info.Batches.Name)
    while exist(char(Info.ConfirmFile(i)),'file')~=2
        pause(15)
    end
end

load(strcat(InfoFolder,infofilename))

for i=1:length(BatchName)
    resultsname(i)=strcat(BatchName(i),'at');
end

numruns=length(Info.Names);
PRData=zeros(8,numruns);
Facility=cell(1,numruns);
Scenario=cell(1,numruns);
indexc=0;

for i=1:length(BatchName)
    load(char(resultsname(i)))
    for n=1:size(PRRecordData,2)
        indexc=indexc+1;
        PRData(1,indexc)=PRRecordData(1,n);
        PRData(2,indexc)=PRRecordData(2,n);
        PRData(3,indexc)=PRRecordData(3,n);
        PRData(4,indexc)=PRRecordData(4,n);
        PRData(5,indexc)=PRRecordData(5,n);
        PRData(6,indexc)=PRRecordData(6,n);
        PRData(7,indexc)=PRRecordData(7,n);
        PRData(8,indexc)=PRRecordData(8,n);
        Facility(1,indexc)=PRRecordFacility(1,n);
        Scenario(1,indexc)=PRRecordScenario(1,n);
    end
end

PRRecordData=PRData;
PRRecordFacility=Facility;
PRRecordScenario=Scenario;
Info.PRResults.PRRecordData=PRRecordData';
Info.PRResults.PRRecordFacility=PRRecordFacility';
Info.PRResults.PRRecordScenario=PRRecordScenario';
savefile=strcat(InfoFolder,infofilename);
save(savefile,'Info');
toc
return
```

```
% Loads a previously created set of scenarios to be run.
function pushbutton12_Callback(hObject, eventdata, handles)
global InfoFolder;
global infofilename;
[infofilename,InfoFolder]=uigetfile;
return
```

Appendix B: OSUPR Clustering & Analysis Interface Code

The MATLAB code behind the OSUPR clustering & analysis interface is given in this appendix. It was created using the GUIDE tool in MATLAB, and is seen in Fig. 24:

```
function varargout = OSUPR_cluster(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @OSUPR_cluster_OpeningFcn, ...
   'gui_OutputFcn',  @OSUPR_cluster_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

% Written by Zachary Jankovsky, 2012-2014.
% The Ohio State University

function OSUPR_cluster_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

global datasize;
global Type;
global method;
method=[0 0 0];
global dimensionality;
dimensionality=3;
global k;
global bandwidth;
global repetition;
repetition=20;
global OSUPRGUI_dir;
OSUPRGUI_dir=pwd;
```

```
addpath(OSUPRGUI_dir)
return;

function varargout = OSUPR_cluster_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
global datasize;
global name;
global folder;
[name, folder]=uigetfile;
load(strcat(folder,name));
Info.PRResults.Norm=struct;
Info.FOM=struct;
Info.PRResults.Norm=Info.PRResults.PRRecordData;
DetectMin=min(Info.PRResults.Norm(:,1),[],1);
DetectMax=max(Info.PRResults.Norm(:,1),[],1);
FailMin=min(Info.PRResults.Norm(:,2),[],1);
FailMax=max(Info.PRResults.Norm(:,2),[],1);
TimeMin=min(Info.PRResults.Norm(:,4),[],1);
TimeMax=max(Info.PRResults.Norm(:,4),[],1);
SuccessMin=min(Info.PRResults.Norm(:,7),[],1);
SuccessMax=max(Info.PRResults.Norm(:,7),[],1);

datasize=length(Info.PRResults.Norm(:,2));
%Populating standardized output variable:
for i=1:datasize
   Info.PRResults.Norm(i,1)=((Info.PRResults.Norm(i,1))-DetectMin)/(DetectMax-DetectMin);
   Info.PRResults.Norm(i,2)=((Info.PRResults.Norm(i,2))-FailMin)/(FailMax-FailMin);
   Info.PRResults.Norm(i,3)=Info.PRResults.Norm(i,3);
   Info.PRResults.Norm(i,4)=((Info.PRResults.Norm(i,4))-TimeMin)/(TimeMax-TimeMin);
   Info.PRResults.Norm(i,5)=Info.PRResults.Norm(i,5);
   Info.PRResults.Norm(i,6)=Info.PRResults.Norm(i,6);
   Info.PRResults.Norm(i,7)=((Info.PRResults.Norm(i,7))-SuccessMin)/(SuccessMax-SuccessMin);
   Info.PRResults.Norm(i,8)=Info.PRResults.Norm(i,8);
end

%Zac Jankovsky, 2013, The Ohio State University
%OSUPR FOM Extension

%Calculates figure of merit for a scenario according to the following
%criteria:
%FOM=a1*PSM+a2*PDM+a3*PFM+a4*PTM+a5*MTI+a6*PCM
%PSM: probability of success
%PDM: probability of detection
%PFM: probability of technical failure
%PTM: proliferation time
%MTI: material type index from PRCALC
%PCM: proliferation cost, 1 for reprocessing, 0 for no reprocessing
%a1=1.0;
%a2=-1.0;
%a3=-0.5;
%a4=0.6;
%a5=1.0;
%a6=0.2;
```

```
%PSM=Info.PRResults.PRRecordData(:,7);
%PFM=Info.PRResults.PRRecordData(:,2);
%PDM=Info.PRResults.PRRecordData(:,1);
%MTI=Info.PRResults.PRRecordData(:,3);
%PTM=-0.289*log(Info.PRResults.PRRecordData(:,4))+1.7401;
%for i=1:datasize
%    if Info.PRResults.PRRecordData(i,4)>416
%        PTM(i)=0;
%    end
%    if Info.PRResults.PRRecordData(i,4)<13
%        PTM(i)=0;
%    end
%end
%FOM=a1*PSM+a2*PDM+a3*PFM+a4*PTM+a5*MTI;
%Info.FOM.PSM=PSM;
%Info.FOM.PFM=PFM;
%Info.FOM.PDM=PDM;
%Info.FOM.MTI=MTI;
%Info.FOM.PTM=PTM;
%Info.FOM.FOM=FOM;

%FOMMin=min(Info.FOM.FOM(:,1),[],1);
%FOMMax=max(Info.FOM.FOM(:,1),[],1);
%for i=1:datasize
%    Info.FOM.FOM(i)=((Info.FOM.FOM(i))-FOMMin)/(FOMMax-FOMMin);
%end
return;

%Toggles Mean-shift method.
function radiobutton1_Callback(hObject, eventdata, handles)
global method;
method(1)=0;
if get(hObject,'Value')
   method(1)=1;
else
   method(1)=0;
end
return

%Toggles adaptive Mean-shift method.
function radiobutton2_Callback(hObject, eventdata, handles)
global method;
method(2)=0;
if get(hObject,'Value')
   method(2)=1;
else
   method(2)=0;
end
return

%Toggles K-means method.
function radiobutton3_Callback(hObject, eventdata, handles)
global method;
method(3)=0;
if get(hObject,'Value')
```

```
    method(3)=1;
else
    method(3)=0;
end
return

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Sets bandwidth for mean-shift clustering.
function edit3_Callback(hObject, eventdata, handles)
global bandwidth;
bandwidth=str2num(get(handles.edit3,'String'));
return

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% Runs mean-shift algorithm as written by Diego Mandelli.
% Adds clustering results to Info file.
function pushbutton2_Callback(hObject, eventdata, handles)
global Info;
global dimensionality;
global bandwidth;
global datasize;
tic
%Each datapoint represents exactly one case, so the probability for each is
%equal and they add up to 1:
prob=zeros(1,datasize);
for u=1:datasize
    prob(1,u)=1/datasize;
end
%Creates matrix to be used in clustering, of size datasize x
%dimensionality. 3 dimensions will typically be Detection Probability,
%Proliferation Time, and Success Probability:
PR_Cluster=zeros(datasize,dimensionality);
if dimensionality==3
    for i=1:datasize
        %Detection Probability:
        PR_Cluster(i,1)=Info.PRResults.Norm(i,1);
        %Proliferation Time:
        PR_Cluster(i,2)=Info.PRResults.Norm(i,4);
        %Success Probability:
        PR_Cluster(i,3)=Info.PRResults.Norm(i,7);
    end
end
if dimensionality==4
    for i=1:datasize
        %Detection Probability:
        PR_Cluster(i,1)=Info.PRResults.Norm(i,1);
        %Proliferation Time:
```

```
        PR_Cluster(i,2)=Info.PRResults.Norm(i,4);
        %Success Probability:
        PR_Cluster(i,3)=Info.PRResults.Norm(i,7);
        %Figure of Merit:
        PR_Cluster(i,4)=Info.FOM.FOM(i);
    end
end

%Calling the clustering function:
[clustCentNorm,data2cluster,cluster2data,numberOfClusters,probabilities]...
    =MS2Expr(PR_Cluster,prob,bandwidth,datasize,dimensionality);
%Add cluster results to information file:
Info.MeanShiftInfo=struct;
Info.MeanShiftInfo.data2cluster=data2cluster;
Info.MeanShiftInfo.Bandwidth=bandwidth;
Info.MeanShiftInfo.Dimensionality=dimensionality;
Info.MeanShiftInfo.DataSize=datasize;
Info.MeanShiftInfo.NumberOfClusters=numberOfClusters;
Info.MeanShiftInfo.clustCentNorm=clustCentNorm;
Info.MeanShiftInfo.cluster2data=cluster2data;
Info.MeanShiftInfo.Probabilities=probabilities;

global name;
global folder;
%Save info file
savefile=[folder name];
save(savefile,'Info');
toc
msgbox('Done running Mean-shift.')
return

%Sets K for K-means clustering.
function edit5_Callback(hObject, eventdata, handles)
global k;
k=str2num(get(handles.edit5,'String'));
return

function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% Runs K-means algorithm as included in MATLAB.
% Adds clustering results to Info file.
function pushbutton3_Callback(hObject, eventdata, handles)
global k;
global Info;
global datasize;
global dimensionality
global repetition;

%Creates matrix to be used in clustering, of size datasize x
%dimensionality. 3 dimensions will typically be Detection Probability,
%Proliferation Time, and Success Probability:
PR_Cluster=zeros(datasize,dimensionality);
```

```
if dimensionality==3
   for i=1:datasize
      %Detection Probability:
      PR_Cluster(i,1)=Info.PRResults.Norm(i,1);
      %Proliferation Time:
      PR_Cluster(i,2)=Info.PRResults.Norm(i,4);
      %Success Probability:
      PR_Cluster(i,3)=Info.PRResults.Norm(i,7);
   end
end
if dimensionality==4
   for i=1:datasize
      %Detection Probability:
      PR_Cluster(i,1)=Info.PRResults.Norm(i,1);
      %Proliferation Time:
      PR_Cluster(i,2)=Info.PRResults.Norm(i,4);
      %Success Probability:
      PR_Cluster(i,3)=Info.PRResults.Norm(i,7);
      %Figure of Merit:
      PR_Cluster(i,4)=Info.FOM.FOM(i);
   end
end

[data2cluster,clustCentNorm]=kmeans(PR_Cluster,k,'Start','cluster','Replicates',repetition);

%Add cluster results to information file:
Info.KmeansInfo=struct;
Info.KmeansInfo.data2cluster=data2cluster;
Info.KmeansInfo.numberOfClusters=k;
Info.KmeansInfo.Dimensionality=dimensionality;
Info.KmeansInfo.DataSize=datasize;
Info.KmeansInfo.clustCentNorm=clustCentNorm;
Info.KmeansInfo.Repetitions=repetition;

global name;
global folder;
%Save info file
savefile=[folder name];
save(savefile,'Info');
msgbox('Done running K-means.')
return

% Runs adaptive mean-shift algorithm as implemented in "Mean shift based
%clustering in high dimensions: A texture classification example."
%B. Georgescu, I. Shimshoni, P. Meerin the proceedings of ICCV 2003.

% Adds clustering results to Info file.
function pushbutton4_Callback(hObject, eventdata, handles)
global name;
global folder;
global Info;
global datasize;
global dimensionality;
global OSUPRGUI_dir;
```

```
%Creates matrix to be used in clustering, of size datasize x
%dimensionality. 3 dimensions will typically be Detection Probability,
%Proliferation Time, and Success Probability:
PR_Cluster=zeros(datasize,dimensionality);
if dimensionality==3
   for i=1:datasize
      %Detection Probability:
      PR_Cluster(i,1)=Info.PRResults.Norm(i,1);
      %Proliferation Time:
      PR_Cluster(i,2)=Info.PRResults.Norm(i,4);
      %Success Probability:
      PR_Cluster(i,3)=Info.PRResults.Norm(i,7);
   end
end
if dimensionality==4
   for i=1:datasize
      %Detection Probability:
      PR_Cluster(i,1)=Info.PRResults.Norm(i,1);
      %Proliferation Time:
      PR_Cluster(i,2)=Info.PRResults.Norm(i,4);
      %Success Probability:
      PR_Cluster(i,3)=Info.PRResults.Norm(i,7);
      %Figure of Merit:
      PR_Cluster(i,4)=Info.FOM.FOM(i);
   end
end

filename=strcat(name(1:end-4),'.txt');
file=fopen(filename,'w');
fclose(file);
file = fopen(filename, 'a');
fprintf(file,num2str(datasize));
fprintf(file,' ');
fprintf(file,num2str(dimensionality));
fprintf(file,'\n');
fclose(file);
dlmwrite(filename,PR_Cluster ,'-append','delimiter',' ');

space={' '};
phrase=strcat('fams 0 0',space,num2str(datasize),space,filename(1:end-4),space,'"',folder,'\"');
phrase=cell2mat(phrase);
%cd(OSUPRGUI_dir)
system(phrase)
%cd(folder)
famsoutname=strcat('out_',filename);
adap=dlmread(famsoutname);
clustCentNorm=unique(adap,'rows');
data2cluster=zeros(length(adap),1);

for i=1:length(adap)
   for j=1:length(clustCentNorm)
      if mean(adap(i,:)==clustCentNorm(j,:))==1
         data2cluster(i)=j;
      end
   end
```

```
end

%Add cluster results to information file:
Info.FAMSInfo=struct;
Info.FAMSInfo.data2cluster=data2cluster;
Info.FAMSInfo.numberOfClusters=length(clustCentNorm);
Info.FAMSInfo.Dimensionality=dimensionality;
Info.FAMSInfo.DataSize=datasize;
Info.FAMSInfo.clustCentNorm=clustCentNorm;

%Save info file
savefile=[folder name];
save(savefile,'Info');
msgbox('Done running adaptive Mean-shift.')

return

%Displays cluster centroids for the chosen clustering method.
function pushbutton5_Callback(hObject, eventdata, handles)
global Type;
global Info;

if str2num(Type)==1
   centroids_to_display=cell(length(Info.MeanShiftInfo.clustCentNorm),1);
   for i=1:length(Info.MeanShiftInfo.clustCentNorm)
      centroids_to_display(i)=cellstr(mat2str(Info.MeanShiftInfo.clustCentNorm(i,:)));
   end
   centroid_header='DP (normalized) PT (normalized) PS (normalized)';
   centroids_cell=cell(length(Info.MeanShiftInfo.clustCentNorm)+1,1);
   centroids_cell(1)=cellstr(centroid_header);
   for i=2:length(Info.MeanShiftInfo.clustCentNorm)+1
      centroids_cell(i)=strcat(centroids_to_display(i-1));
   end
end

if str2num(Type)==2
   centroids_to_display=cell(length(Info.KmeansInfo.clustCentNorm),1);
   for i=1:length(Info.KmeansInfo.clustCentNorm)
      centroids_to_display(i)=cellstr(mat2str(Info.KmeansInfo.clustCentNorm(i,:)));
   end
   centroid_header='DP (normalized) PT (normalized) PS (normalized)';
   centroids_cell=cell(length(Info.KmeansInfo.clustCentNorm)+1,1);
   centroids_cell(1)=cellstr(centroid_header);
   for i=2:length(Info.KmeansInfo.clustCentNorm)+1
      centroids_cell(i)=strcat(centroids_to_display(i-1));
   end
end

if str2num(Type)==3
   centroids_to_display=cell(length(Info.FAMSInfo.clustCentNorm),1);
   for i=1:length(Info.FAMSInfo.clustCentNorm)
      centroids_to_display(i)=cellstr(mat2str(Info.FAMSInfo.clustCentNorm(i,:)));
   end
   centroid_header='DP (normalized) PT (normalized) PS (normalized)';
```

```
    centroids_cell=cell(length(Info.FAMSInfo.clustCentNorm)+1,1);
    centroids_cell(1)=cellstr(centroid_header);
    for i=2:length(Info.FAMSInfo.clustCentNorm)+1
        centroids_cell(i)=strcat(centroids_to_display(i-1));
    end
end

msgbox(centroids_cell)
return

%Chooses the clustering method to investigate.
function edit6_Callback(hObject, eventdata, handles)
global Type;
Type=get(hObject,'String');
return

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Chooses a cluster number to investigate.
function edit7_Callback(hObject, eventdata, handles)
global ClustNum;
ClustNum=get(hObject,'String');
return

function edit7_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Creates plots for a chosen cluster.
function pushbutton6_Callback(hObject, eventdata, handles)
global ClustNum;
global Info;
global Type;
global datasize;
global name;
global folder;
clustnum=str2num(ClustNum);
type=str2num(Type);

%PRCALC Outputs
if type==1
    w=1;
    for i=1:datasize
        if Info.MeanShiftInfo.data2cluster(i)==clustnum
            clust_out_hist(w,:)=Info.PRResults.PRRecordData(i,:);
            w=w+1;
        end
    end
end
```

```
if type==2
   w=1;
   for i=1:datasize
      if Info.KmeansInfo.data2cluster(i)==clustnum
         clust_out_hist(w,:)=Info.PRResults.PRRecordData(i,:);
         w=w+1;
      end
   end
end
if type==3
   w=1;
   for i=1:datasize
      if Info.FAMSInfo.data2cluster(i)==clustnum
         clust_out_hist(w,:)=Info.PRResults.PRRecordData(i,:);
         w=w+1;
      end
   end
end

%PRCALC Inputs
if type==1
   w=1;
   for i=1:datasize
      if Info.MeanShiftInfo.data2cluster(i)==clustnum
         clust_in_hist(w,:)=Info.Matrix(i,:);
         clust_SG_hist(w,:)=Info.SGCond(i,:);
         w=w+1;
      end
   end
end
if type==2
   w=1;
   for i=1:datasize
      if Info.KmeansInfo.data2cluster(i)==clustnum
         clust_in_hist(w,:)=Info.Matrix(i,:);
         clust_SG_hist(w,:)=Info.SGCond(i,:);
         w=w+1;
      end
   end
end
if type==3
   w=1;
   for i=1:datasize
      if Info.FAMSInfo.data2cluster(i)==clustnum
         clust_in_hist(w,:)=Info.Matrix(i,:);
         clust_SG_hist(w,:)=Info.SGCond(i,:);
         w=w+1;
      end
   end
end

%Histograms
space={' '};

%Outputs
```

```
out_hist_labels=cell(1,3);
out_hist_labels(1)=cellstr('Probability of Detection');
out_hist_labels(2)=cellstr('Proliferation Time');
out_hist_labels(3)=cellstr('Probability of Success');
h=figure;
hist(clust_out_hist(:,1),100)
xlabel('Probability of Detection')
ylabel('Number of Scenarios')
title(strcat(out_hist_labels(1),space,'Cluster',space,num2str(clustnum)))
saveas(h, strcat(char(out_hist_labels(1)),'_',num2str(type),'_',num2str(clustnum)), 'png')
close(h)
clear h
h=figure;
hist(clust_out_hist(:,4),100)
xlabel('Proliferation Time')
ylabel('Number of Scenarios')
title(strcat(out_hist_labels(2),space,'Cluster',space,num2str(clustnum)))
saveas(h, strcat(char(out_hist_labels(2)),'_',num2str(type),'_',num2str(clustnum)), 'png')
close(h)
clear h
h=figure;
hist(clust_out_hist(:,7),100)
xlabel('Probability of Success')
ylabel('Number of Scenarios')
title(strcat(out_hist_labels(3),space,'Cluster',space,num2str(clustnum)))
saveas(h, strcat(char(out_hist_labels(3)),'_',num2str(type),'_',num2str(clustnum)), 'png')
close(h)
clear h
h=figure;
hist(Info.PRResults.PRRecordData(:,1),100)
xlabel('Probability of Detection')
ylabel('Number of Scenarios')
title(strcat(out_hist_labels(1),space,'Entire Set'))
saveas(h, strcat(char(out_hist_labels(1))), 'png')
close(h)
clear h
h=figure;
hist(Info.PRResults.PRRecordData(:,4),100)
xlabel('Proliferation Time')
ylabel('Number of Scenarios')
title(strcat(out_hist_labels(2),space,'Entire Set'))
saveas(h, strcat(char(out_hist_labels(2))), 'png')
close(h)
clear h
h=figure;
hist(Info.PRResults.PRRecordData(:,7),100)
xlabel('Probability of Success')
ylabel('Number of Scenarios')
title(strcat(out_hist_labels(3),space,'Entire Set'))
saveas(h, strcat(char(out_hist_labels(3))), 'png')
close(h)
clear h

%Inputs
histstage=Info.Stages(Info.Elements==1);
histstages=strrep(histstage,':','_');
```

```
for i=1:size(Info.Matrix,2)
   hist(clust_in_hist(:,i))
   xlabel('Rate (\sigma)')
   ylabel('Number of Scenarios')
   title(strcat(histstages(i),' - Cluster ',num2str(clustnum)))
   set(gca,'XTickLabel',[unique(clust_in_hist(:,1))])
   set(gca,'XTick',[unique(clust_in_hist(:,1))])
   saveas(gcf, strcat(char(histstages(i)),'_',num2str(type),'_',num2str(clustnum)), 'png')
   close
end

for i=1:size(Info.Matrix,2)
   hist(Info.Matrix(:,i))
   xlabel('Rate (\sigma)')
   ylabel('Number of Scenarios')
   title(strcat(histstages(i),' - Entire Set'))
   set(gca,'XTickLabel',[unique(Info.Matrix(:,1))])
   set(gca,'XTick',[unique(Info.Matrix(:,1))])
   saveas(gcf, strcat(char(histstages(i))), 'png')
   close
end

hist(clust_SG_hist(:,1))
xlabel('Safeguard Condition')
ylabel('Number of Scenarios')
title(strcat('Safeguard Conditions - Cluster ',num2str(clustnum)))
set(gca,'XTickLabel',[1 2 3 4])
set(gca,'XTick',[1 2 3 4])
saveas(gcf, strcat('Safeguard Conditions','_',num2str(type),'_',num2str(clustnum)), 'png')
close

hist(Info.SGCond(:,1))
xlabel('Safeguard Condition')
ylabel('Number of Scenarios')
title('Safeguard Conditions - Entire Set')
set(gca,'XTickLabel',[1 2 3 4])
set(gca,'XTick',[1 2 3 4])
saveas(gcf, strcat('Safeguard Conditions_Entire Set'), 'png')
close
return
```