SCENARIO CLUSTERING AND DYNAMIC PROBABILISTIC RISK ASSESSMENT

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Diego Mandelli, M.S.

Graduate Program in Nuclear Engineering

The Ohio State University

2011

Dissertation Committee:

Tunc Aldemir, Adviser

Alper Yilmaz

Richard Denning

Umit Catalyurek

ABSTRACT

The recent trend to use a best estimate plus uncertainty (BEPU) approach to nuclear reactor safety analysis instead of the traditional conservative approach can produce very large amounts of data. Hence, the need for methodologies able to handle high volumes of data in terms of both cardinality (due to the high number of uncertainties included in the analysis) and dimensionality (due to the complexity of systems) arises.

Clustering methodologies offer powerful tools that can help the user to identify groups of scenario that are representative of the original data set and, thus, can reduce the effort involved in data analysis. Scenario clustering aims to: a) identify the scenarios that have a similar behavior (i.e., identify the most evident classes), b) decide for each event sequence to which cluster it belongs (i.e., classification), and c) perform the analysis of each cluster. The main objective of this dissertation is to show how it is possible to accomplish these three objectives by using clustering methodologies to the scenarios generated by safety analysis codes.

Several clustering algorithms are developed, evaluated and compared using different types of data sets. Mode-seeking clustering algorithms such as Mean-Shift are proven to be well suited for the scenario analysis. The Mean-Shift algorithm is a kernel-based, non-parametric density estimation technique that is used to find the modes of an unknown distribution, which correspond to regions with highest data density. The

obtained cluster centers represent the most representative scenarios from the original data set and the analysis can be now carried out on the smaller set of representative scenarios.

The specific types of data under consideration are those generated using the dynamic event tree (DET) approach for nuclear power reactor transients which are described by a large set of state variables (i.e., temperature, pressure of specific nodes in the plant simulator) and information regarding the status of specific components/systems. Several examples are presented in order to illustrate the applications of clustering algorithms to data generated by DET. In addition, pre-processing of the raw data and data reduction techniques are described and compared.

Dedicated to Cheri, my family and friends

ACKNOWLEDGMENTS

This dissertation could not have been written without the support and friendship found in Ohio and left in Italy. The love of family and friends provided my inspiration and was my driving force. It has been a long journey and completing this work is definitely a high point in my academic career. I could not have come this far without the assistance of many individuals and I want to express my deepest appreciation to them.

I would like to gratefully and sincerely thank my advisor prof. Aldemir for his guidance, understanding, patience, and most importantly, his friendship during my graduate studies. His mentorship was paramount in providing a well rounded experience consistent with my long-term career goals.

I would like to thank Prof Yilmaz for his assistance and guidance in getting my research started on the right foot and providing me with the foundation of my dissertation topic.

I would also like to thank Prof. Denning and Prof. Çatalyürek for their ideas and the precious observations to this research project.

I thank the whole OSU Nuclear Engineering Program, for the knowledge and skills I gained while studying and working here that helped me to complete this Ph.D. dissertation work.

I thank my parents, Giuseppe and Letizia, for their faith in me and allowing me to be as ambitious as I wanted.

Finally, and most importantly, I would like to thank Cheri. Her support, understanding and patience have been invaluable for me.

I hope that this work makes all of you proud.

VITA

June 25, 1978	Born - Brescia, Italy
2004	Laurea Nuclear Engineering, Politecnico di Milano
2005-2008	Graduate Research Associate, The Ohio State University.
2008	M.S. Nuclear Engineering, The Ohio State University.
2008-present	Graduate Research Associate, The Ohio State University.

PUBLICATIONS

Research Publications

T. Aldemir, S. Guarro, D. Mandelli, J. Kirschenbaum, P. Bucci, M. Yau, E. Ekici, M. Stovsky, S. A. Arndt, ''A Benchmark System for Comparing Reliability Modeling Approaches for Digital Instrumentation and Control Systems'', *Nuclear Technology*, **165**, no.1, pp. 53-65 (2008).

T. Aldemir, S. Guarro, D. Mandelli, J. Kirschenbaum, L. A. Mangan, P. Bucci, M. Yau, E. Ekici, D.W. Miller, X. Sun, S. A. Arndt, ''Probabilistic Risk Assessment Modeling of Digital Instrumentation and Control Systems Using Two Dynamic Methodologies'', *Reliability Engineering and Safety Systems (RESS)*, **95**, pp. 1011-1039 (2010).

T. Aldemir, M. P. Stovsky, J. Kirschenbaum, D. Mandelli, P. Bucci, L. A. Mangan, D. W. Miller, X. Sun, E. Ekici, S. Guarro, M. Yau, B. Johnson, C. Elks, S. A. Arndt, ''NUREG/CR-6942: Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments'', U.S. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research (2007).

T. Aldemir, S. Guarro, J. Kirschenbaum, D. Mandelli, L. A. Mangan, P. Bucci, M. Yau, B. Johnson, C. Elks, E. Ekici, M. P. Stovsky, D. W. Miller, X. Sun, S. A. Arndt, Q. Nguyen, J. Dion, ''NUREG/CR-6985: A Benchmark Implementation of Two Dynamic Methodologies for the Reliability Modeling of Digital Instrumentation and Control Systems'', U.S. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research (2009).

D. Mandelli, A. Yilmaz, T. Aldemir, ''Scenario Analysis and PRA: Overview and Lessons Learned'', *ESREL 2011 Conference*, (2011).

D. Osborn, D. Mandelli, T. Aldemir, ''A Dynamic Level 2 PRA Using ADAPT-MELCOR'', *ESREL 2011 Conference*, (2011).

D. Mandelli, A. Yilmaz, T. Aldemir, ''Clustering Scenarios on Manifolds'', *Proceedings of the American Nuclear Society (ANS)*, (2011).

D. Mandelli, A. Yilmaz, T. Aldemir, ''Data Processing Methodologies Applied to Dynamic PRA: an Overview'', *Proceedings of PSA 2011 Conference*, (2011).

D. Mandelli, A. Yilmaz, T. Aldemir, R. Denning, ''Scenario Aggregation and Analysis via Mean-Shift Methodology'', *in Proceedings for Probabilistic Safety Assessment and Management 2010 (PSAM 10).*

D. Mandelli, A. Yilmaz, T. Aldemir, ''Scenario Aggregation in Dynamic PRA Uncertainty Quantification'', *Proceedings of the American Nuclear Society (ANS)*, **102**, pp. 246-249 (2010).

D. Mandelli, K. Metzroth, A. Yilmaz, R. Denning, T. Aldemir, ''Probabilistic Clustering for Scenario Analysis'', *Proceedings of the American Nuclear Society (ANS)*, pp. 371-374 (2010).

D. Mandelli, A. Yilmaz, K. Metzroth, T. Aldemir, R. Denning, ''Scenario Aggregation and Analysis via Mean-Shift Methodology'', *Proceedings of ICAPP 2010*, pp. 990-994 (2010).

T. Aldemir, D. Mandelli, L. Mangan, D. Miller, M. Stovsky, X. Sun, S. Guarro, M. Yau, P. Bucci, J. Kirschenbaum, B. Johnson, C. Elks, E. Ekici, S. Arndt, ''Dynamic Reliability Modeling Of Digital Instrumentation And Control Systems In Nuclear Power Plants'', *Proceedings of NPIC-HMIT*, (2009).

F. Di Maio, M. Stasi, E. Zio, D. Mandelli, T. Aldemir, ''Identification of Faults in a Level Control Dynamic System'', *Proceedings of NPIC-HMIT*, (2009).

D. Mandelli, J. Kirschenbaum, L. Mangan, E. Ekici, T. Aldemir, ''Modeling of Communications in the Safety Assessment of Nuclear Power Plants'', *Proceedings of the American Nuclear Society (ANS)*, **99**, pp. 473-475 (2008).

D. Mandelli, T. Aldemir, J. Kirschenbaum, P. Bucci, D. W. Miller, M. Stovsky, E. Ekici, S. A. Arndt, ''A Benchmark System for the Reliability Modeling of Digital Instrumentation and Control Systems'', *Proceedings for PSAM 9 Conference* (2008).

D. Mandelli, T. Aldemir, P. Bucci, L. A. Mangan, J. Kirschenbaum, M. Stovsky, E. Ekici, S. A. Arndt, ''Markov/CCMT Modeling of the Benchmark System and Incorporation of the Results into an Existing PRA'', *Proceedings for PSAM 9 Conference* (2008).

D. Mandelli, P. Zhang, T. Aldemir, R. Denning, ''A Risk-Informed Approach in the Design of a Molten Salt Reactor'', *Proceedings of the American Nuclear Society (ANS)*, **96**, pp. 299-301 (2007).

T. Aldemir, D. Mandelli et al., ''A Benchmark System for the Assessment of Reliability Modeling Methodologies for Digital Instrumentation and Control Systems in Nuclear Plants,'', *Proceedings NPIC-HMIT* (2006).

T. Aldemir, D. Mandelli et al., ''Incorporation of Markov Reliability Models for Digital Instrumentation and Control Systems into Existing PRAs'', *Proceedings NPIC-HMIT* (2006).

D. Mandelli, T. Aldemir, E. Zio, ''An Event Tree/Fault Tree/Embedded Markov Model Approach for the PSAM-8 Benchmark Problem Concerning a Phased Mission Space Propulsion System'', *Proceedings for PSAM 8 Conference* (2006).

FIELDS OF STUDY

Major Field: Nuclear Engineering

Studies in : Probabilistic Safety Assessment

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Description

A recent trend in the nuclear power engineering field is the implementation of heavily computational and time consuming algorithms [1, 2, 3] and codes [4, 5, 6] for both design and safety analysis. In particular, the new generation of system analysis codes aim to embrace several phenomena such as thermo-hydraulic, structural behavior, system dynamics and human behavior, as well as uncertainty quantification and sensitivity analyses associated with these phenomena[1]. The use of dynamic probabilistic risk assessment (PRA) methodologies allows a systematic approach to uncertainty quantification.

Dynamic methodologies in PRA [7] account for possible coupling between triggered or stochastic events through explicit consideration of the time element in system evolution, often through the use of dynamic system models (simulators). They are usually needed when the system has more than one failure mode, control loops, and/or

[1]LWR Sustainability Program (INL/EXT-07-13543, ''Strategic Plan for Light Water Reactor Research and Development,'' Idaho National Laboratory, November 2007).

hardware/process/software/human interaction [8]. Dynamic methodologies are also capable of modeling the consequences of epistemic[2] and aleatory[3] uncertainties [9].

Dynamic PRA methods include Dynamic Logical Analytical Methodology (DYLAM) [10], Dynamic Event Tree Analysis Method (DETAM) [2], ADS [11], Analysis of Dynamic Accident Progression Trees (ADAPT) [1, 12], Sequence Diagrams (ESDs) [13], Petri Nets [14], Dynamic Flowgraph Methodology (DFM) [15], Discrete Dynamic Event Trees (DDET) [16], Markov/Cell-to-Cell Mapping Technique [17] and Monte Carlo Dynamic Event Tree (MCDET) [18]. The list is not exhaustive and only provides some samples of dynamic PRA methods. A more comprehensive discussion of dynamic methods is given in [8].

The DYLAM, DETAM, ADS and ADAPT are among methodologies that use dynamic event trees (DETs) to account for aleatory uncertainties. ADAPT can also account for epistemic uncertainties within the DET framework. A DET is an expansion on traditional static event trees (ETs), and seeks to incorporate timing and process relationships into the stochastic system model. Static ETs have a fixed and predetermined event sequence defined by the analyst.

Figure 1.1 shows a simplified ET for a large break loss of cooling accident (LOCA). In order to reach a safe state of the plant, the reactor protection system trips the reactor and performs the cooling of the reactor through the emergency cooling system (ECCS). A failure in any of these two systems will cause core damage (CD). Note that in the ET pictured in Fig. 1.1, the sequencing of events (and accompanying branchings) are already pre-fixed in the system logic designed by the analyst.

[2]Uncertainties that derive from some level of ignorance, or incomplete information, of the system or the surrounding environment.

[3]Uncertainties due to the inherent variation associated with the physical system or the environment under consideration.

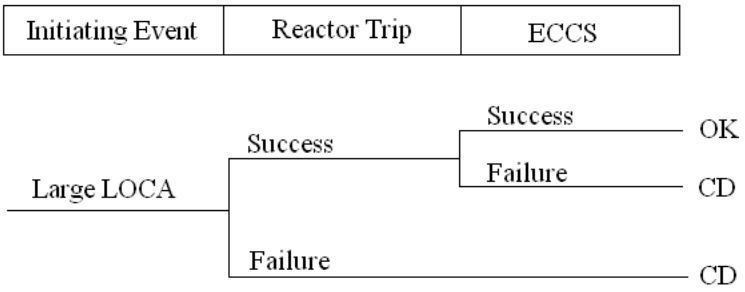

Figure 1.1: Example of event tree.

The DETs are generated by the direct coupling between the a dynamic model of the plant constructed using plant simulators such as RELAP [4], MELCOR [5] and the stochastic behavior of system components (including software/firmware), parameters and human action. The branching conditions in a DET are generated by user specified rules, such as activation/non-activation upon demand of components or correct/faulty crew action depending on specific plant conditions. The plant simulator evaluates the temporal behavior of the plant and determines the timing and nature of each branch. Subsequently, each system included in the analysis (e.g., reactor protection system, emergency core cooling system) is demanded according to its control logic (e.g., when a setpoint is reached) and not in a predefined order.

The major challenges in using DETs (as well as other dynamic methodologies) are the heavier computational and memory requirements compared to the classical ET analysis. This is due to the fact that each branch generated can contain time evolutions of a large number of variables (about 50,000 data channels are present in MELCOR) and a large number of scenarios can be generated from a single initiating

event (possibly on the order of hundreds or even thousands). Such large amounts of information are usually very difficult to organize in order to identify the main trends in scenario evolutions and the main risk contributors for each initiating event[4] [20].

1.2 Objectives and scope

This dissertation addresses this problem of data analysis by clustering the scenarios into groups (or clusters) and analyzing the properties of the scenarios contained in each cluster by:

1. identifying the scenarios that have a ''similar'' behavior (i.e. identify the most evident clusters), and,

2. deciding for each sequence which cluster it belongs (i.e., classification).

Data analysis is still a relatively young branch of computer science. Especially after the development and the wide spread use of workstations, it has been possible to develop more advanced and precise simulation models albeit with the potential to generate huge quantities of data.

Clustering methodologies [21] offer convenient tools to post-process large data sets that include a large variety of information (e.g., transient profiles, component states, human performances). The idea of clustering can be summarized as the process of finding partitions of the original data set and characterizing each partition by a representative data point. With regard to the output of safety analysis codes (such as RELAP, MELCOR), each element of the data set is a scenario (or, equivalently,

[4]Note that also in the early PRA analysis in NUREG-1150, the problem of data analysis has been faced and post processing of the scenarios generated for each of the three PRA levels has been performed using classification methods [19]. Chapter 2 describes in details how this classification has been performed.

a transient) which represents a unique simulation of the system considered under specified conditions.

In this dissertation, a systematic clustering approach is developed using the Mean-Shift algorithm which capable of handling large volume of scenarios generated by DETs.

The volume of data is not the only concern that needs to be addressed. The following are also of concern:

1. Different types of attributes (e.g., time, process variables, component variables)
2. Clusters with arbitrary shapes
3. Noise and outliers
4. Interpretability and usability

As indicated in [22], three important factors need to be considered in scenario analysis:

- Type of data
- Type of analysis
- Type of variables chosen to characterize each scenario

The data generated by dynamic methodologies such as DETs for the analysis of nuclear power plants are usually inhomogeneous (i.e., both discrete and continuous) due to the fact that they contain

- the temporal description of the state variables of each node of the simulator (e.g., temperature, pressure, level or concentration of particular elements), and,

- the status of system components, both hardware and software (e.g., aperture of a valve or status of a digital control system), and sub-systems (e.g., Emergency Core Cooling System) of the plant under consideration.

While the former data type is generally continuous, the latter is typically discrete.

Regarding the type of analysis, it is possible to group the set of scenarios into two possible modes when dealing with nuclear transients:

- *End State Analysis* which groups scenarios into clusters based on the end state of the scenarios (e.g., NUREG-1150 [23])
- *Transient Analysis* which groups scenarios into clusters based on their time evolution [22].

Lastly, it is possible to characterize each scenario based on

- the status of a set of components [24], and,
- the temporal behavior of a set of state variables [22] (e.g., node pressure, temperature).

This dissertation focuses on the clustering of scenarios using the time evolution of state variables to characterize each scenario. Although targeting scenarios generated by DETs, the methodology developed can be applied to any data set.

1.3 Annotated bibliography

From the literature it is possible to organize clustering methodologies into two classes based on their approach to the clustering problem (see Fig. 1.2):

- Hierarchical (see Fig. 1.3)

- Partitional (see Fig. 1.4)

Hierarchical algorithms organize data into a structure according to a proximity matrix in which each element *(j, k)* is some measure of the similarity (or distance) between the items to which row j and column k correspond. Usually, the final result of these algorithms is a binary tree, also called dendrogram, in which the root of the tree represents the whole data set and each leaf is a data point. An example of hierarchical clustering is pictured in Fig. 1.3 [25] for a data set containing seven points in a 2-dimensional space. The hierarchical clustering algorithm is also described in detail in Section 5.1.

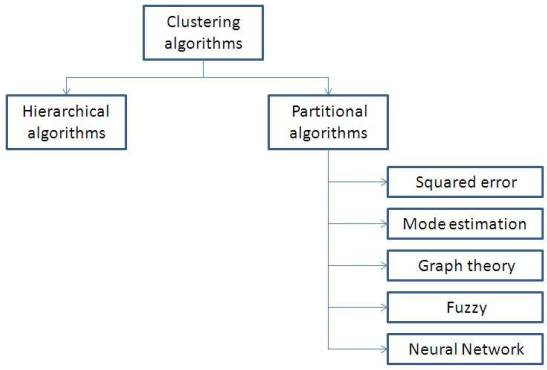

Figure 1.2: Taxonomy of the clustering techniques [25].

On the other hand, partitional clustering seeks for a single partition of the data set as shown in Fig. 1.4 instead of nested sequence of partitions as shown for the hierarchical methodologies. Partitional clustering methodologies can be divided furthermore into five classes:

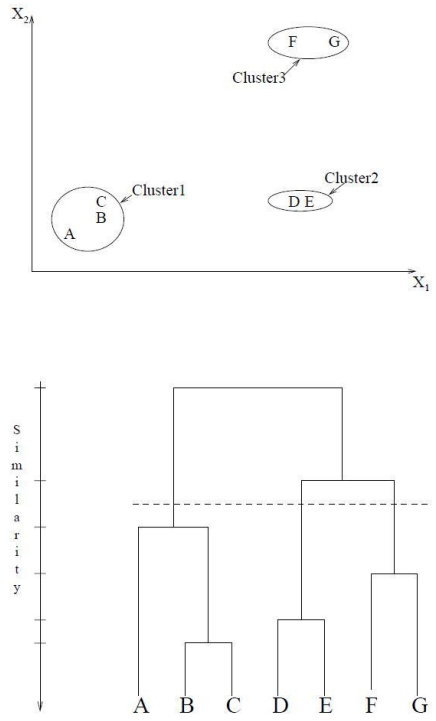

Figure 1.3: Example of hierarchical clustering [25].

Squared error: Squared error algorithms assign each point to the cluster whose center (also called centroid) is nearest. The center is the average of all the points in the cluster, that is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster. The most famous and used methodology is the K-Means algorithm [26] which is described in detail in Section 5.2.

Fuzzy: Fuzzy clustering is very similar to K-Means but each point has a degree of belonging to every cluster, as in fuzzy logic, rather than belonging completely to just one cluster [27]. Thus, points on the edge of a cluster may be in the

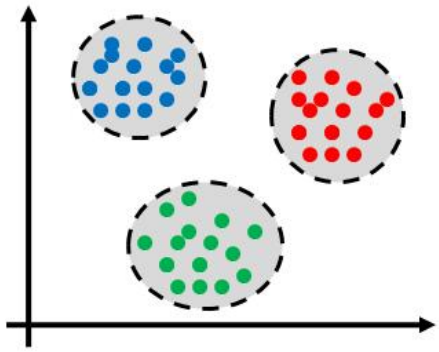

Figure 1.4: Example of partitional clustering.

cluster to a lesser degree than points in the center of cluster. For each point $\vec{x}$ there is a coefficient $u_k(\vec{x})$ which gives the degree of being in the k^{th} cluster. An example of fuzzy clustering methodology is the Fuzzy C-Means algorithm which is described in detail in Section 5.3.

Mode Seeking: These methodologies are based on the assumption that the distribution of the points in the state space can be described through a probability density function (pdf). The goals is to find the modes, i.e., the regions in the state space with higher data densities. An example of this kind of methodology is the Mean-Shift methodology [28] which is described in detail in Chapter 6.

Graph Theoretical: Graph theory based methodologies aim to build a graph of the data set often called Minimal Spanning Tree [29, 30]. Clusters are determined by deleting the the longest edges of the graph as also shown in Figure 1.5 for a limited set of data. Conceptually this approach is very similar to the hierarchical one.

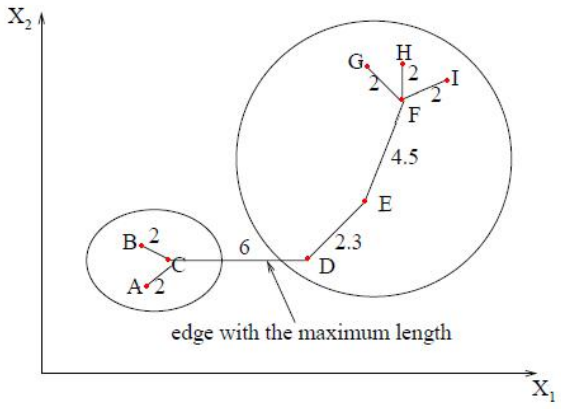

Figure 1.5: Example of graph theory clustering process.

Neural Network: Neural network methodologies are essentially inspired by the biological neural network [31]. The learning process associated with the training of an artificial neural network (ANN) allows to associate patterns (input variables) to clusters (output nodes) through a series of weights that are updated at each iteration.

In the nuclear industry, intelligent transient analysis algorithms able to analyze nuclear transients have been developed in recent years using methodologies mainly based on fuzzy logic [32] and artificial neural networks. Politecnico di Milano [24, 33] and the Paul Scherrer Institut (PSI) [34] have developed methodologies based on fuzzy clustering. Note that the these works implement classification algorithms and not clustering ones. Classification implies that clusters (or equivalently, classes) have been set a priori by the user and the algorithm performs the scenario-to-cluster membership. In clustering, however, the algorithm determines the clusters based on a set of similarity rules specified by the user.

It is important to highlight that the need for techniques capable to group scenarios has been faced also in the classic PRA environment such as in the NUREG-1150 [23]. The number of scenarios generated by ETs depends on the number of branching conditions which can be quite large. In that respect, a classification methodology has been used in NUREG-1150 in order to reduce the original data set to a more manageable size. Chapter 2 shows how this classification has been performed.

1.4 Contributions of this dissertation

This work has been structured in the following manner:

- Chapter 2 gives an overview on how the classification has been performed in NUREG-1150 [23].
- Chapter 3 gives a description of clustering applied to scenario analysis.
- Chapter 4 shows how the data are pre-processed before performing the clustering. This chapter also introduces the problem of dimensionality reduction and the need to reduce the number of variables that describe each scenario. Both linear and non-linear methodologies are shown and results are compared.
- Chapter 5 compares several clustering methodologies and shows the decision process that leads to the selection of the Mean-Shift algorithm as the most suited for the scope of this dissertation.
- Chapter 6 shows in detail the Mean-Shift algorithm and how it is implemented. An example is given using the simple level controller model described in [35].
- Chapter 7 shows the results of the clustering algorithm described in Chapter 6 applied to several DET data sets.

- Chapter 8 presents a summary of the work and the conclusions that can be drawn from the analysis reported in Chapter 7. In addition, ideas and opportunities for future work are listed.

CHAPTER 2

SCENARIO ANALYSIS IN CLASSICAL PRA

PRA is a process that is commonly used to evaluate the potential risks of a system. The objectives of this process are to determine three main goals: a) the failures that can occur, b) the likelihood for these failures to occur, and, c) the consequences that these failures will have on the system. PRA applied nuclear power plants has been developed since 1974 when the U.S. Nuclear Regulatory Commission (NRC) published the first milestone in the safety analysis community, the Reactor Safety Study WASH-1400 [36], followed in more recent years by the Severe Accident Risks: An Assessment for Five U.S. Nuclear Power Plants (NUREG-1150[5] [23].

Classical safety analysis described in NUREG-1150 is divided into 3 levels:

Level 1: Starting from an initiating event (e.g., station black-out or loss of coolant accident) the analysis is carried out until the reactor core damage condition is reached.

Level 2: Starting from a situation of core damage, the analysis is carried out until containment is breached (radioactive release occurs).

[5]Actually, NUREG-1150 simply provides the results of the analysis for 5 US power plants. NUREG-4550 shows in much more detail the overall methodology used in NUREG-1150.

Level 3: Starting from a situation of radioactive release outside the containment the analysis evaluates the effects of the release on the population and on the environment.

For each level, the corresponding ET is built. Branching conditions (e.g., activation of specific systems) are specified by the user and branch probabilities may be determined by fault trees, depending on the system configuration.

Due to the complexity of nuclear power plants, the number of branching conditions is very high and, hence each level generates a large number of scenarios and, thus, authors of NUREG-1150 originally faced the problem of managing this large amount of data.

The problem has been addressed by performing the following at the end of each analysis:

1. Characterize each scenario by considering the status of a specific subset of systems

2. Define a priori a set of classes (i.e. groups of scenarios)

3. Perform the scenario-to-class membership (binning)

4. Continue into the next level by considering the classes chosen in Step 2

This process is usually called classification instead of clustering.

In NUREG-1150 , eight characteristics based on the status of particular systems have been considered for the Level 1 analysis of the Zion plant:

1. Status of reactor cooling system (RCS)

2. Status of Emergency cooling system (ECCS)

3. Containment heat removal

4. AC Power

5. Contents of Reactor Water Storage Tank (RWST)

6. Heat removal from steam generators

7. Cooling of reactor coolant pump (RCP) seals

8. Status of containment fan coolers

All the scenarios generated in Level 1 analysis that lead to core damage and with frequency higher than 10^{-9} have been characterized by these eight characteristics listed above. Note that no information on the transient temporal behavior has been considered but only the component status which characterizes each scenario.

The scenarios have, then, been grouped into five classes:

1. Station Blackout

2. LOCAs

3. Transients

4. Steam generator tube rupture (SGTR)

5. Event V

where each group has been characterized by the mean, median and $5^{th} - 95^{th}$ percentile of the frequencies.

After the Level 2 analysis has been performed, leading to so called Accident Progression Even Trees or APETs [23], a similar binning process is performed. Twelve characteristics are used to specify these Accident Progression Bins (APBs):

1. Time of containment failure
2. Periods of which sprays operate
3. Occurrence of core-concrete Interactions
4. RCS pressure before vessel breach
5. More of vessel breach
6. Steam generator tube rupture
7. Amount of core available for CCI
8. Fraction of Zirconium oxidized in-vessel
9. Fraction of the core in high pressure melt ejection
10. Size or type of containment failure
11. Number of large holes in the RCS after vessel breach
12. Time of core damage

Five APBs have been defined:

1. Early containment failure
2. Late containment failure

3. Alpha

4. Bypass

5. No containment failure

Results are summarized by expressing the conditional probabilities of accident progression in terms of these bins as shown in Fig. 2.1.

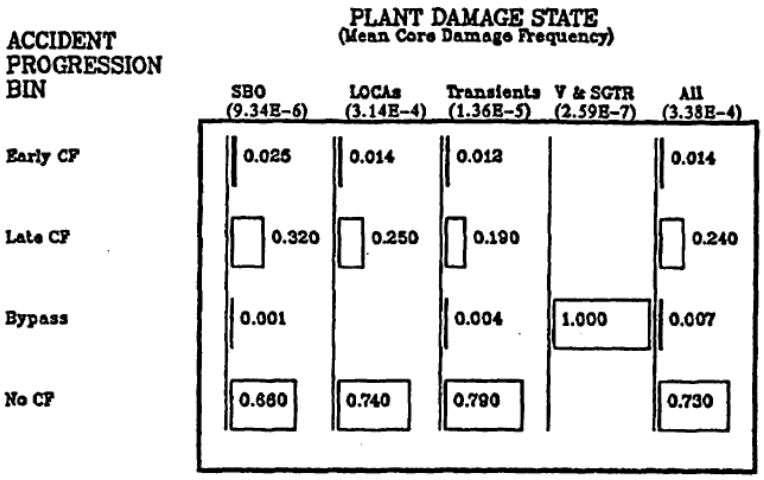

Figure 2.1: Example of binning results for Zion Plant [23].

CHAPTER 3

CLUSTERING: AN OVERVIEW

A loose definition of clustering is the process of organizing objects into groups whose members are, in some way, similar. A cluster is therefore a collection of objects which are similar to each other and are dissimilar to the objects belonging to other clusters [37].

Figure 1.4 shows an elementary example of partitional clustering [25] applied to a 2-dimensional set of data. Here it is possible to identify the 3 clusters into which the data can be divided. The similarity criterion is distance. Two or more objects belong to the same cluster if they are ‘‘close’’ according to a specified distance. The approach of using distance metrics to clustering is called distance-based clustering and is used in this work.

The notion of distance implies that the data points lay in a metric space [38]:

Definition 1 (Metric Space)**.** *A metric space is a space X provided with a function* d*:* $f : X \times X \to \mathbb{R}$ *satisfying the following properties* $\forall \vec{x}, \vec{y} \in X$ *:*

- $d(\vec{x}, \vec{y}) \geqslant 0$
- $d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x})$

- $d(\vec{x}, \vec{y}) \leqslant d(\vec{x}, \vec{z}) + d(\vec{z}, \vec{y})$

The function $d(\vec{x}, \vec{y})$ is usually called the distance function. In a 2-dimensional Euclidean space ($\mathbb{R}^2$), the distance between points can be calculated using the Pythagorean theorem which is the direct application of the Euclidean distance and is a special case of the most general Minkowski distance $d_2(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ between two points $\vec{x} = (x_1, x_2)$ and $\vec{y} = (y_1, y_2)$ in $\mathbb{R}^2$.

In the literature [38], it is possible to find several types of distances other than the Euclidean and the Minkowski distance as shown in Table 3.1. The approach of using distance metrics is called distance-based clustering and will be used in this dissertation.

Table 3.1: Summary of the commonly used measures [38].

Measure	Form
Minkowski distance	$d_n(\vec{x}, \vec{y}) = (\sum_{k=1}^{\delta} \lvert x_k - y_k \rvert^n)^{\frac{1}{n}}$
Euclidean distance	$d_2(\vec{x}, \vec{y}) = (\sum_{k=1}^{\delta} \lvert x_k - y_k \rvert^2)^{\frac{1}{2}}$
Taxicab distance	$d_1(\vec{x}, \vec{y}) = \sum_{k=1}^{\delta} \lvert x_k - y_k \rvert$
Supremum distance	$d_0(\vec{x}, \vec{y}) = max_k \lvert x_k - y_k \rvert$
Mahalanobis distance	$d_M(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})$

From a mathematical viewpoint, the concept of clustering [37] aims to find a partition $\mathbf{C} = \{C_1, \ldots, C_l, \ldots, C_L\}$ of the set of I scenarios $\mathbf{X} = \{\vec{x_1}, \ldots, \vec{x_i}, \ldots, \vec{x_I}\}$

where each scenario $\vec{x_i}$ is represented as a δ-dimensional vector (see Section 4.1). Each C_l $(l = 1, \ldots, L)$ is called a cluster. The partition $\mathbf{C}$ of $\mathbf{X}$ is given as follows[6]:

$$\begin{cases} \mathbf{C}_l \neq \varnothing, l = 1, \ldots, L \\ \\ \bigcup_{l=1}^{L} \mathbf{C}_l = \mathbf{X} \end{cases} \tag{3.1}$$

The clustering process presented in this dissertation consists of the following steps [19]:

1. *Variable selection.* This first step consists of the identification of the variables that are considered useful for characterization of the data. For a transient in a nuclear plant, these variables may be both process variables such as temperature, pressure or level in specific points of the system and hardware/software/firmware states. The choice of the variables of interest therefore specifies the dimensions of the state space (see Chapter 4).

2. *Transient representation format.* The chosen variables may be different in both nature (e.g., temperature and pressure) and scale (i.e., the range of these variables might differ in terms of order of magnitude). Hence, it may be necessary to perform a scaling operation on the chosen variables. This work uses the Principal Component Analysis (PCA) [39] as a tool to process the data before clustering is performed (see Chapter 4).

3. *Clustering algorithm design.* The clustering algorithm along with metric and other parameters of the algorithm are chosen (see Chapter 5).

[6]In most clustering algorithms each scenario belongs to only one cluster. However this is not always the case. In fuzzy clustering methodologies [33] a scenario may be allowed to belong to more than one cluster with a degree of membership $u_{i,j} \in [0, 1]$ which represents the member coefficient of the j scenario for the i^{th} cluster and satisfies the following properties:
$\sum_{i=1}^{K} u_{i,j} = 1$, and $\sum_{j=1}^{N} u_{i,j} < N, \forall j$

4. *Clustering.* The clustering algorithm is applied to the data set and cluster centers are determined (see Chapter 6).

5. *Post-processing.* The cluster centers are converted back into the original format of the data.

6. *Interpretation of the results.* Given the cluster centers obtained in the previous step, the goal is to provide the user with meaningful insight into the original data set. Further classification analysis may be required in order to gain such a meaningful insight (see Chapter 7).

Table 3.2 gives a summary of the data analysis methodologies that have been used in PRA and shows how the clustering methodology described in this work (i.e., Ohio State University in Table 3.2) fits in.

Table 3.2: Summary of the data analysis methodologies applied to PRA.

Methodology	Type	Data	Timing	Metric
Nureg-1150	Classification	System Components	No	No
Politecnico di Milano	Classification	System Components	Yes	Distance
Paul Scherer Institut	Classification	System Components	Yes	Distance
Ohio State University	Clustering	State Variables	Yes	Distance

CHAPTER 4

DATA PRE-PROCESSING AND DIMENSIONALITY REDUCTION

Before performing clustering, it is often necessary to pre-process the data (see Chapter 3). This chapter describes in detail the pre-processing step which includes the following:

- choice of the data representation
- variable reduction
- data normalization and transformation

Section 4.1 shows how each scenario is represented. In Section 4.2, data normalization and data transformation are discussed. Section 4.3 shows how the dimensionality of the data sets (i.e., the product of the variables chosen to represents the scenarios and the number of times these variables have been sampled) can be reduced. For very complex systems this dimensionality is extremely high which leads to high computational cost in the clustering process. In that respect, reduction of the dimensionality of the data set would be needed in most practical applications. Section 4.4 and 4.5, respectively, show the linear and the non-linear algorithms that can be used for data reduction.

4.1 Data Representation

Since the temporal evolution of each scenario is typically described by the time evolution of all system state variables (e.g., pressure and temperature at a computational node), each scenario $\vec{x_i}$ $(i = 1, \ldots, I)$ is represented by M state variables x_{im} $(m = 1, \ldots, M)$ plus time t (ranging from 0 to T) as the state vector

$$\vec{x_i} = [x_{i1}(t_1), \ldots, x_{iM}(t_1), \ldots, x_{i1}(t_K), \ldots, x_{iM}(t_K)] \tag{4.1}$$

where $x_{im}(t_k)$ corresponds to the value of the variable x_m (e.g., temperature, pressure at a computational node) sampled at time t_k (e.g., $t_1 = 0$ and $t_K = T$) for scenario i. Note that the dimensionality δ of each scenario is $\delta = M \cdot K$ and can be extremely high for complex systems[7] (i.e., large number of state variables and large number of samples). When dealing with transient codes such as RELAP [4] or MELCOR [5] the state space would theoretically include all the variables of all the discretization nodes (on the order of tens of thousands). Thus, a smaller set of variables of interest needs to be chosen.

As indicated in Chapter 3, distance metric is employed to measure the similarity (or, equivalently, the dissimilarity) of data points. In this study, the Euclidean distance $d(\vec{x_i}, \vec{x_j})$ is used as a measure of the similarity between two data points $\vec{x_i}$ and $\vec{x_j}$:

$$d(\vec{x_i}, \vec{x_j}) = (\sum_{d=1}^{M \cdot K} |x_i(d) - x_j(d)|^2)^{\frac{1}{2}} \tag{4.2}$$

[7]Note that it is assumed here that all the simulation have same time length (i.e., $t = 0, \ldots, T$). This is not often the case, especially for simulations generated by DET. Several options have been evaluated in order to overcome this limitation by extending scenarios that ended at time $t' < T$ using:

- the last simulated value of the scenario,
- arbitrary constant value (e.g., 0), or,
- an arbitrary decreasing function

from t' to T.

4.2 Data Normalization and Data Transformation

The expression of $d(\vec{x_i}, \vec{x_j})$ in Eq. 4.2 assumes that each data point $\vec{x_i}$ (see Eq. 4.1) is lying in a multidimensional space characterized by a set of orthogonal axes. However, due to the possible correlation among the chosen variables, the assumption of orthogonality is not often justified.

This problem can be solved in two ways:

1. By using the original set of axes and generalized metrics such as the Mahalanobis distance (see Table 3.1)

2. By transforming the set state space using principal component analysis (PCA) [39]

The Mahalanobis distance [19] is a distance measure introduced in 1936 by P.C. Mahalanobis and it differs for the Euclidean metric in that that it takes into consideration the correlation of the data set and is independent of the scale of the measurements[8]. Given two vectors $\vec{x}$ and $\vec{y}$, the Mahalanobis distance $d_M(\vec{x}, \vec{y})$ is defined as:

$$d_M(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})} \tag{4.3}$$

where S is the covariance matrix of the data set. When the axis of these vectors are orthogonal to each other, then S is a diagonal matrix and the Mahalanobis distance results in the normalized Euclidean distance:

$$d_M(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{\delta} \frac{(x_i - y_i)}{\sigma_i^2}} \tag{4.4}$$

[8]The need for an independent scale of measurements is another relevant issue when dealing with similarity measures as well.

where σ_i is the standard deviation of the i-th variable over the sample set.

On the other hand, PCA aims to transform the original set of δ possibly oblique coordinate axes into a new set of δ orthogonal axis allowing the use of any of the metrics shown in Table 3.1. This new set of δ orthogonal axes can be obtained by finding the eigenvectors of the covariance matrix S and the data points can be projected into the new coordinate system. Since the Euclidean distance has been chosen as metric, PCA have been implemented in order to project the original data set into a coordinate system having orthogonal axes.

Another issue that arises when dealing with nuclear transients is the fact that the variables of interest may differ in units (e.g., pressure, level or temperature), as well as ranges. This problem can be solved in two ways:

- normalize each dimension into the $[0, 1]$ interval, or
- normalize each dimension by dividing it by its standard-deviation.

4.3 Dimensionality Reduction: an Overview

As indicated in Section 4.1, the dimensionality δ of each data point (i.e., each scenario) is equal to the product of the number of variables (i.e., M) chosen to represent each scenario multiplied by the number of times each variable has been sampled. In order to reduce the computational time due to the high data dimensionality, the use of dimensionality reduction techniques was to reduce the number of variables M[9].

The raw data generated by DET methodologies contain the temporal behavior of a vast set of variables (e.g., temperature, pressure). These variables are often heavily

[9]Other possible options are to reduce the number of sample instants K or to observe the local properties of the covariance matrix S.

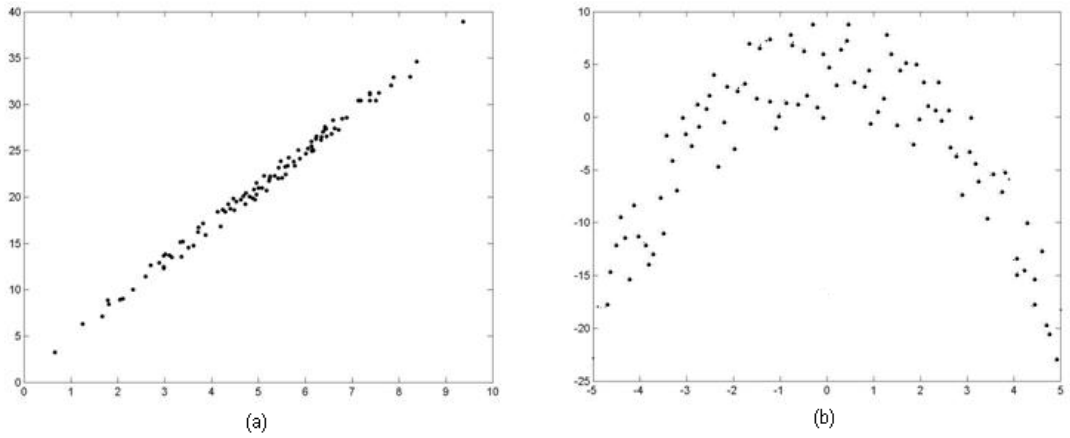

Figure 4.1: Example of linear (a) and non-linear (b) correlation among 2 variables.

correlated and, consequently, the information contained in the set of M variables comprising the full state space can be condensed to a set of N variables where $N < M$. The objective of the dimensionality reduction process is to determine those N variables by finding the correlations among the original M variables[10].

Linear algorithms, such as PCA [39] or multidimensional scaling (MDS) [40], have the advantage that they are easier to implement but they can only identify linear correlation among variables. On the other hand, methodologies such as Local Linear Embedding [41] and ISOMAP [42] are more computationally intensive but they are able to identify non-linear correlations.

Figure 4.1 shows two examples of linear and non-linear correlations. In both cases points are distributed in a 2-dimensional space (i.e., characterized by 2 variables: x, y) but they are lying in a 1-dimensional space.

[10]Note that those N variables are not necessarily a subset of the original M variables but, more likely, a combination of those M variables.

Dimensionality reduction is the process of finding a bijective mapping function $\mathfrak{F}$

$$\mathfrak{F} : \mathbb{R}^D \mapsto \mathbb{R}^d \text{ (where } d < D) \tag{4.5}$$

which maps the data points from the D-dimensional space into a reduced d-dimensional space (i.e. embedding on a manifold) in such a way that the distances between each point and its neighbors are preserved. In our applications $D = M + 1$, i.e. M state variables plus time t.

4.4 Dimensionality Reduction: Linear Algorithms

This section describes the two most important algorithms for dimensionality reduction:

1. PCA (see Section 4.4.1), and,

2. MDS (see Section 4.4.2).

4.4.1 Principal Component Analysis (PCA)

The main idea behind PCA [39] is to perform a linear mapping of the data set onto a lower dimensional space such that the variance of the data in the low-dimensional representation is maximized.

This is accomplished by determining the eigenvectors and their corresponding eigenvalues of the data covariance matrix[11] S. The eigenvectors that correspond to the largest eigenvalues (i.e., the principal components) can be used as a set of basis functions. Thus, the original space is reduced to the space spanned by a few eigenvectors.

[11]Given a data set in form of a vector Z, rows correspond to data dimensions (D) and columns correspond to data observations (Λ), the covariance matrix S is determined as: $S = \frac{1}{\Lambda - 1} Z'Z$.

Figure 4.2 shows an example of dimensionality reduction using PCA for a data set distributed in a 2-dimensional space. After performing the eigenvalue-eigenvector decomposition of the covariance matrix, the algorithm chooses the eigenvector having the largest eigenvalue (i.e., λ_1) as subspace to project the original data.

The algorithm is very easy to implement but, on the other hand, PCA is not able to identify non-linear correlations of more complex data sets.

Figure 4.2: Example of dimensionality reduction using PCA (reduction from D=2 to d=1).

4.4.2 Multidimensional Scaling (MDS)

Multidimensional scaling [40] is a popular technique used to analyze the properties of data sets. The scope of this methodology is to find a set of dimensions that preserve distances between data points.

This is performed by:

1. Creating dissimilarity matrix $D = [d_{ij}]$ where d_{ij} is the distance between two points x_i and x_j.

2. Finding the hyper-plane that preserves the dissimilarity matrix D (i.e., the *nearness* of points)

As in PCA analysis, the algorithm can be easily implemented but it is not able to identify non-linear correlations of more complex data sets.

4.5 Dimensionality Reduction: Non Linear Algorithms

Since PCA and MDS are linear algorithms and, thus, limited to the data sets which include only linear correlations among variables it was decided to consider non-linear algorithms such as those based on manifold analysis. In particular, the ISOMAP [42] algorithm have been considered.

The ISOMAP[12] algorithm provides a simple method for estimating the intrinsic geometry of a data manifold based on a rough estimate of each data point's neighbors on the manifold. ISOMAP is one representative of isometric mapping methods, and extends MDS by incorporating the geodesic distances[13] (distance along the manifold) imposed by a weighted graph. ISOMAP is distinguished by its use of the geodesic distance induced by a neighborhood graph embedded in the classical scaling. The algorithm is implemented by the following two steps:

1. Estimate the geodesic distance between points in inputs using shortest-path distances on the data set's k nearest neighbor[14]. The connectivity of each data point in the neighborhood graph is defined as its nearest k Euclidean neighbors in the high-dimensional space.

2. Use MDS to find points in low-dimensional Euclidean space whose interpoint distances match the the distances found in Step 1.

[12]http://isomap.stanford.edu/

[13]In graph theory, the distance between two vertices in a graph is the number of edges in a shortest path connecting them. This is also known as the geodesic distance.

[14]ISOMAP defines the geodesic distance to be the sum of edge weights along the shortest path between two nodes.

Section 7.5 shows dimensionality reduction results using ISOMAP to a large data set generated by a DET methodology such as ADAPT [12].

CHAPTER 5

COMPARISON OF CLUSTERING METHODOLOGIES

This chapter describes in detail the following clustering methodologies that have been investigated:

- Hierarchical (see Section 5.1)
- K-Means (see Section 5.2)
- Fuzzy C-Means (see Section 5.3)
- Mode-Seeking (see Section 5.4)

Section 5.5 shows how these 4 methodologies have been tested using three different data sets and how Mode-Seeking methodology is chosen as the most effective for the scope of this dissertation. Section 5.6 gives a short overview of the most relevant clustering methodologies for high dimensionality data as possible future research options.

5.1 Hierarchical methodologies

These methodologies organize the data set into a hierarchical structure according to a proximity matrix. Each element $d(i, j)$ of this matrix contains the distance

between the the i^{th} and the j^{th} cluster center. The final results of this technique is a tree commonly called a dendrogram (see Fig. 1.3). This kind of representation has the advantages of providing a very informative description and visualization of the data structure even for high values of dimensionality.

The procedure to determine the dendrogram for a data set of I points in an δ-dimensional space is the following:

1. Start the analysis with a set of I clusters (i.e., each point is considered as a cluster).

2. Determine the proximity matrix M (dimension: $I \times I$): $M(i,j) = d(\vec{x_i}, \vec{x_j})$ where $\vec{x_i}$ and $\vec{x_j}$ are the position of the i^{th} and the j^{th} cluster.

3. For each point p find the closest neighbor q from the proximity matrix M

4. Combine the points p and q

5. Repeat Steps 2, 3 and 4 until all the points of the data set are in the same cluster

The advantage of this kind of algorithm is the nice visualization of the results that show the underlying structure of the data set. However, the computational complexity for most of the hierarchical algorithm is of the order of $\mathcal{O}(I^2)$ (where I is the number of points in the data set).

5.2 *K*-Means

K-Means clustering algorithms belong to the more general family of Squared Error algorithms. The goal is to partition I data points $\vec{x_i}$ $(i = 1, \ldots, I)$ into K clusters

in which each data point maps to the cluster with the nearest mean. The stopping criterion is to find the global minimum of the error squared function χ defined as:

$$\chi = \sum_{i=1}^{K} \sum_{x_j \in C_i} |\vec{x_j} - \vec{\mu_i}|^2 \tag{5.1}$$

where $\vec{\mu_i}$ is the centroid (i.e., the center) of the cluster C_i.

The procedure to determine the centroids $\vec{\mu_i}$ of K clusters $(C_1, \ldots, C_K)$ is the following:

1. Start with a set of K random centroids distributed in the state space
2. Assign each pattern to the the closest centroid
3. Determine the new K centroids according to the point-centroid membership

$$\mu_i = \frac{1}{N_i} \sum_{\vec{x_j} \in C_i} \vec{x_j} \tag{5.2}$$

where N_i corresponds to the number of of data points in the i^{th} cluster.

4. Repeat Steps 2 and 3 until convergence is met (i.e., until a minima of the χ function is reached)

K-Means algorithm is one of the most popular and used methodologies also due to the fact that is very easy to implement and the computational time is directly proportional to the cardinality of data points (i.e., $\mathcal{O}(I)$ where I is the number of data points). The main disadvantage is that the algorithm is sensitive to the choice of the initial partition and may converge to a local minimum of the error squared function [21]. Another disadvantage of this algorithm is that is only able to identify

clusters having spherical or ellipsoidal geometry. Thus, K-Means is not able to identify clusters of points having arbitrary shapes. Moreover, the number of cluster K to be obtained is specified by the user prior the clustering process.

5.3 Fuzzy C-Means

Fuzzy C-Means clustering is a clustering methodology that is based on fuzzy sets and, hence, it allows a data point to belong to more that one cluster [27, 43]. Similar to the K-Means clustering, the objective is to find a partition of C fuzzy centers to minimize the function J defined as following:

$$J = \sum_{i=1}^{I} \sum_{j=1}^{C} u_{ij}^{m} |\vec{x_i} - \vec{\mu_j}|^2 \tag{5.3}$$

where:

- $u_{ij}^{m} \in [0, 1]$ is the membership coefficient of the data point $\vec{x_i}$ for the j^{th} cluster having centroid $\vec{\mu_j}$,
- $m \in [0, \infty)$ is the fuzzification parameter (usually set to $m = 2$), and,
- μ_j is the centroid of the j^{th} cluster center

The procedure to determine the centroids (or, equivalently, cluster centers) $\vec{\mu_j}$ $(j = 1, \ldots, C)$ of C clusters is the following:

1. Initialize the $U = [u_{ij}^{m}]$ matrix
2. Calculate the set of C centroids as following:

$$\mu_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \tag{5.4}$$

3. Update the matrix $U = [u_{ij}^m]$ as following:

$$u_{ij}^m = \frac{1}{\sum_{k=1}^{C} (\frac{|x_i - \mu_j|}{|x_i - \mu_k|})^{\frac{2}{m-1}}} \tag{5.5}$$

4. Repeat Steps 2 and 3 until convergence, i.e. if $|U^{(K+1)} - U^{(K)}| < \epsilon$

Fuzzy C-Means clustering is very similar to the K-Means. As seen for the K-Means, Fuzzy C-Means can also converge to a local minima of the convergence criterion function [27]. Like K-Means, it is not able to identify cluster of points having arbitrary shapes but only clusters having ellipsoidal or spherical geometry and the number of clusters C to be obtained is specified by the user prior the clustering process. Fuzzy C-Means algorithms can be useful when the boundaries among clusters are ambiguous and not well defined.

5.4 Mode-Seeking

Mode-seeking approaches look at the density distribution of data points lying in a metric space. Clusters are viewed as regions of the space with high point density separated by regions of low point density. Clusters can be identified by searching for regions of high density, called modes.

For the comparison a Mode-seeking algorithm referred to as the Mean-Shift [44] has been chosen and it is extensively described in Chapter 6.

The advantage of this class of algorithms is that they are able to identify clusters with arbitrary shapes and, hence, they are not limited to topological figures such as spheres or ellipsoids. Moreover, compared to K-Means (see Section 5.2) and Fuzzy C-Means (see Section 5.3), the number of clusters is not specified a-priori by the user

but it is the algorithm that determines this number based on the areas with higher point concentration.

5.5 Comparison of Methodologies

In order to compare the four methodologies listed above, three different data sets have been selected:

1. A set of 300 points grouped in 3 spherical clusters (see Fig. 5.1(a)),

2. A set of 200 points distributed in 2 rings (see Fig. 5.1(b)),

3. A set that considers scenarios that have been generated for the analysis of a pressurized water reactor. The initiating event investigated was that of a station blackout (SBO) and the MELCOR code was linked to the ADAPT tool to determine the evolution of each DET scenario. The simulations using MELCOR model the transient from the occurrence of the SBO through the core melting phase and up to the point of containment failure and release of radionuclides to the environment. All the 104 scenarios generated in this DET led to containment failure at some point in the scenario evolution. For the purposes of this paper, we choose 4 state variables of interest (see Fig. 5.2):

   (a) Core water level [m]: L,

   (b) System pressure [Pa]: P,

   (c) Intact core fraction [%]: CF,

   (d) Fuel temperature [K]: T.

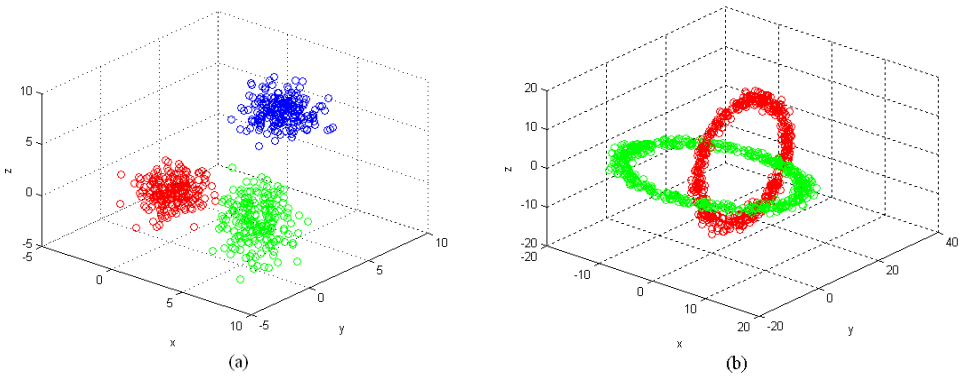

Figure 5.1: Representation of the Data set 1 (a) and 2 (b).

Each state variable has been sampled 100 times, which gives an accurate description of all the 104 transients. As described in Section 4.1, each scenario $\vec{x_i}$ is a multidimensional vector:

$$\vec{x_i} = [L(1), \ldots, L(100), P(1), \ldots, P(100), CF(1), \ldots, CF(100), T(1), \ldots, T(100))] \tag{5.6}$$

In order to compare the results for the methodologies presented in Sections 5.1 through 5.4, for Data sets 1 and 2, the cluster centers obtained from each methodology are compared with the original ones. Tables 5.1 and 5.2 show the comparison of the cluster centers for K-Means, Fuzzy C-Means and Mean-Shift.

For Data set 1 it is possible to note a general agreement between the three methodologies (Table 5.1). However, Table 5.2 shows major disagreements for the K-Means and the Fuzzy C-Means methodologies. The reason for these discrepancies for the two methodologies is due to the fact that the shape of the two clusters is not spherical or ellipsoidal and, thus, both the K-Means and the Fuzzy C-Means

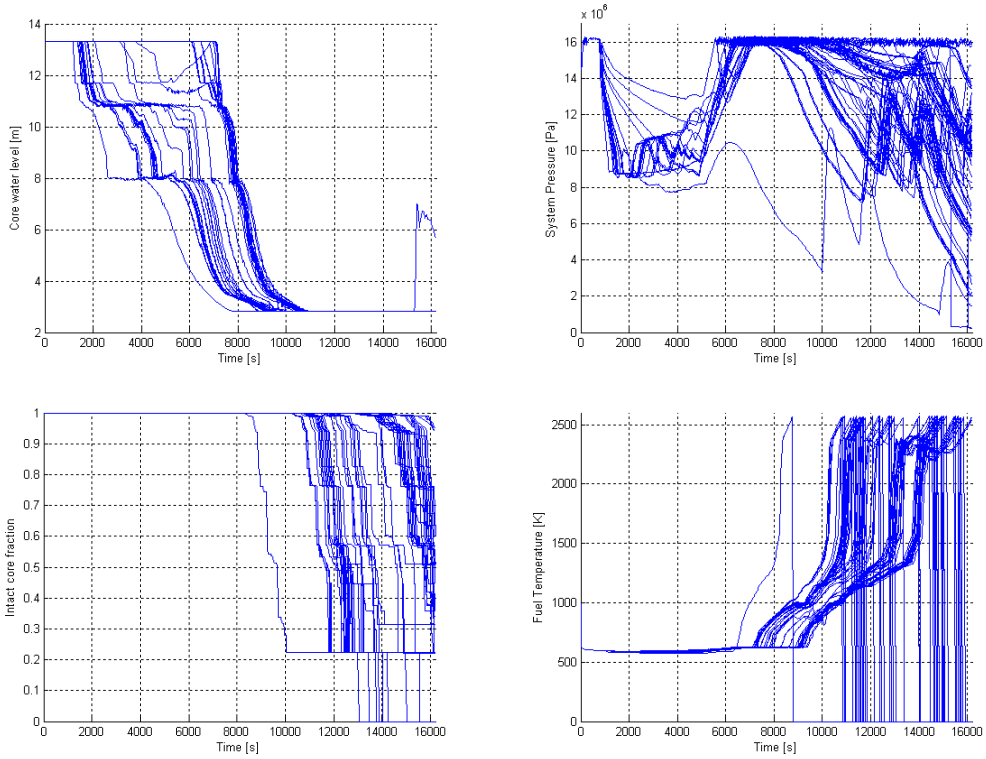

Figure 5.2: Representation of the Data set 3.

Table 5.1: Cluster centers comparison for Data set 1.

Original	K-Means	Fuzzy C-Means	Mean-Shift
(0,0,0)	(-0.0173,0.018, -0.034)	(-0.019,0.003,-0.031)	(-0.012, 0.0055,-0.036)
(6,0,0)	(5.9510,-0.001,-0.059)	(6.38,0.128,-0.201)	(6.16,0.075,-0.043)
(3,6,6)	(2.974,6.01,6.118)	(2.51,5.4878,5.98)	(3.02,5.893,6.011)

Table 5.2: Cluster centers comparison for Data set 2.

Original	K-Means	Fuzzy C-Means	Mean-Shift
(0,0,0)	(0.142,3.632,1.01)	(-0.737,3.866,0.422)	(0.101, -0.0732,0.127)
(0,5,0)	(0.573,15.59,-0.311)	(0.246,15.4894,-0.342)	(0.121,5.182,-0.095)

algorithms have problems to find 2 clusters with more complex shapes such as the two rings pictured in Fig. 5.1(b).

Hierarchical clustering for both Data sets 1 and 2 has also been performed. Figure 5.3 shows the dendrogram for Data set 1 where it is possible to identify the 3 clusters and the hierarchical structure based on the reciprocal distance of the points. Regarding Data set 2, the algorithm was not able to identify the two clusters for the same reason presented for K-Means and Fuzzy C-Means.

For Data set 3, the exact solution is not available and so one methodology has been chosen as a comparison reference using the scenarios shown in Fig. 5.2. Mean-Shift methodology has proven to be more flexible in terms of identifying clusters with arbitrary shapes and, consequently, it has been decided to use the cluster centers obtained from Mean-Shift as a reference. Clustering using Mean-Shift has been performed with value of bandwidth equal to 20 obtaining 8 cluster centers pictured in

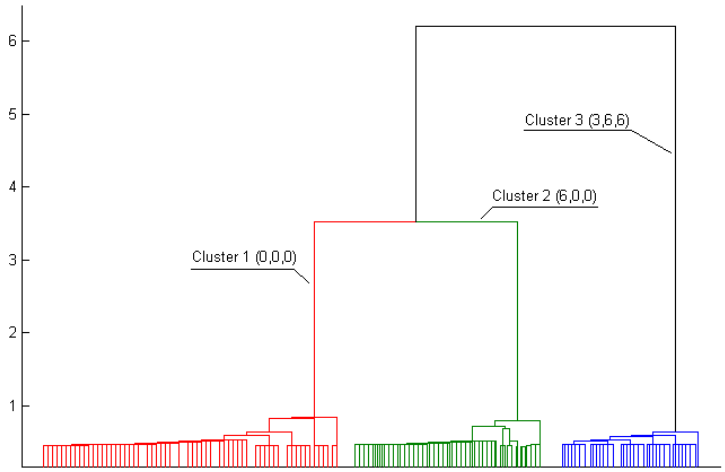

Figure 5.3: Dendrogram derived from the Hierarchical clustering algorithm for Data set 1.

Fig. 5.4. Then, clustering of the third data set using K-Means and Fuzzy C-Means has been performed using the number of clusters obtained with the Mean-Shift as input (i.e., 8). Results are shown in Fig.s 5.5 and 5.6 for K-Means and Fuzzy C-Means, respectively.

Five out of 8 clusters had notable differences in terms of both cluster centers (pictured in Fig. 5.5 and 5.6) and scenario memberships. From the comparison of the results shown for the Data sets 1 and 2, the 8 clusters obtained using the Mean-Shift methodology have geometrical shapes that cannot be modeled using K-Means and Fuzzy C-Means. However, all the three methodologies were able to identify outliers which are scenarios that belong to clusters having only very few elements.

Hierarchical clustering has the advantage that it is able to show the distribution of the data points through a dendrogram. This is useful when the dimensionality of the data points is greater than 3 and hence, it is not easy to graphically visualize

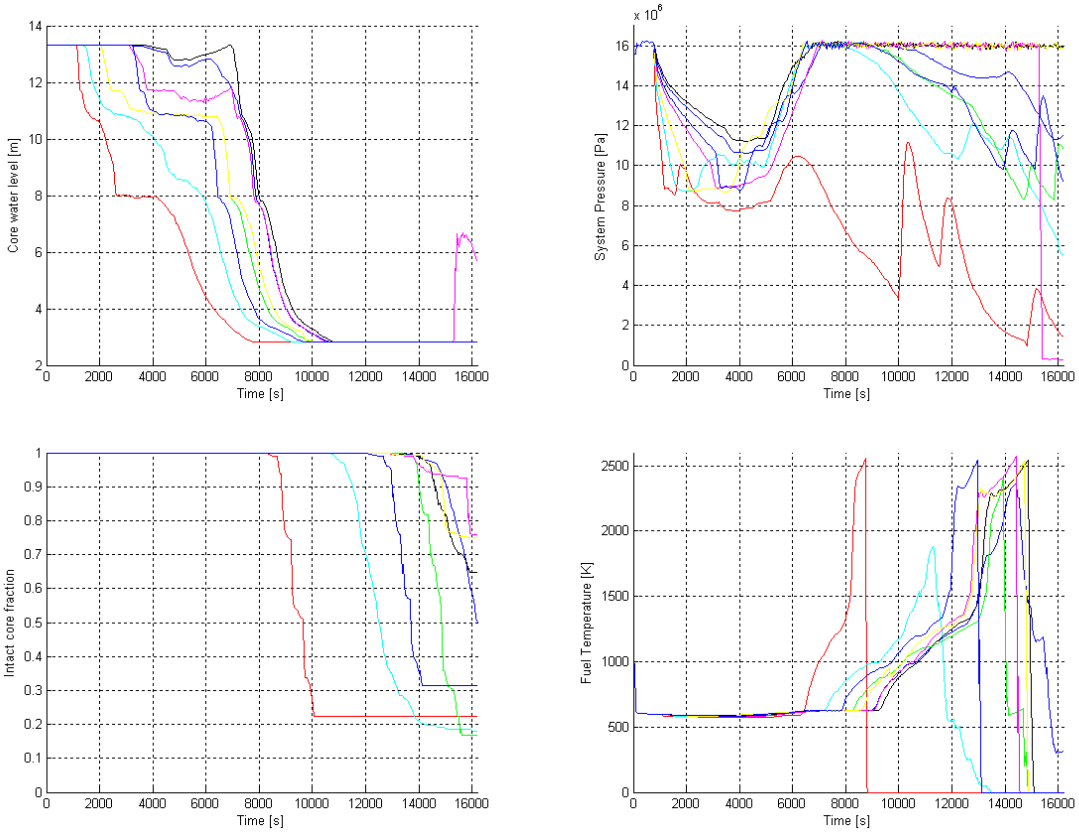

Figure 5.4: Cluster centers obtained using Mean-Shift for Data set 3.

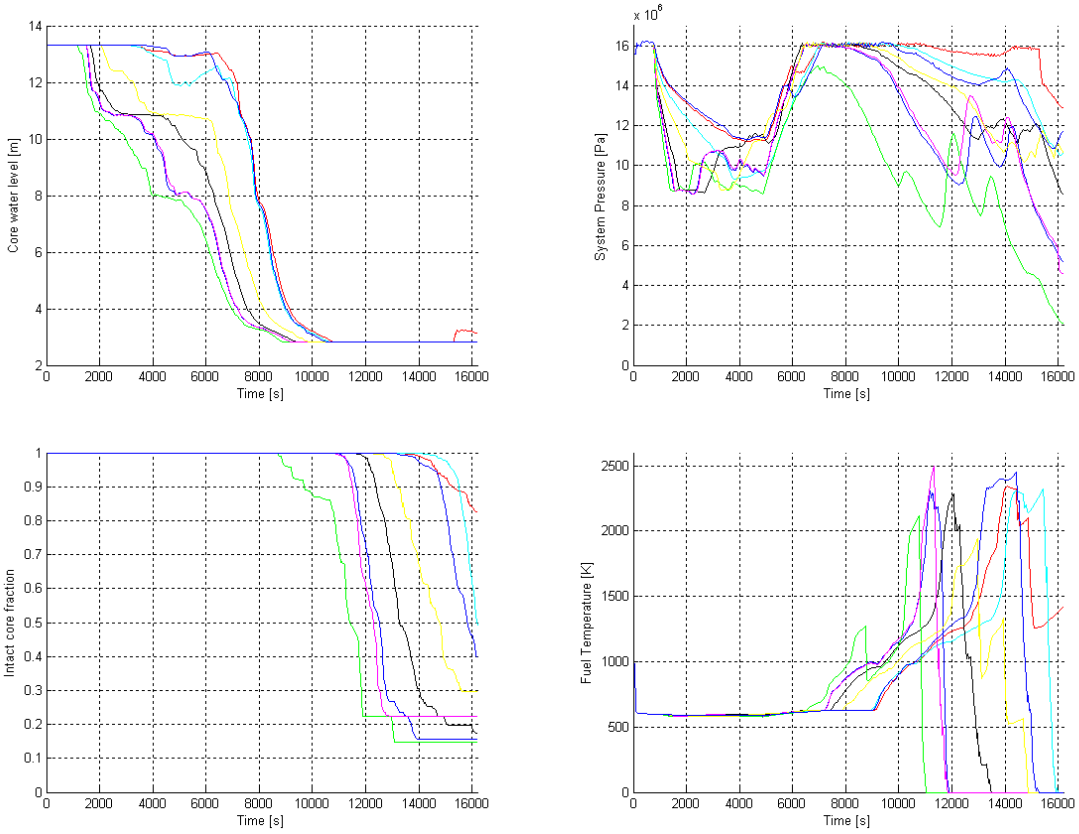

Figure 5.5: Cluster Centers obtained using K-Means for Data set 3.

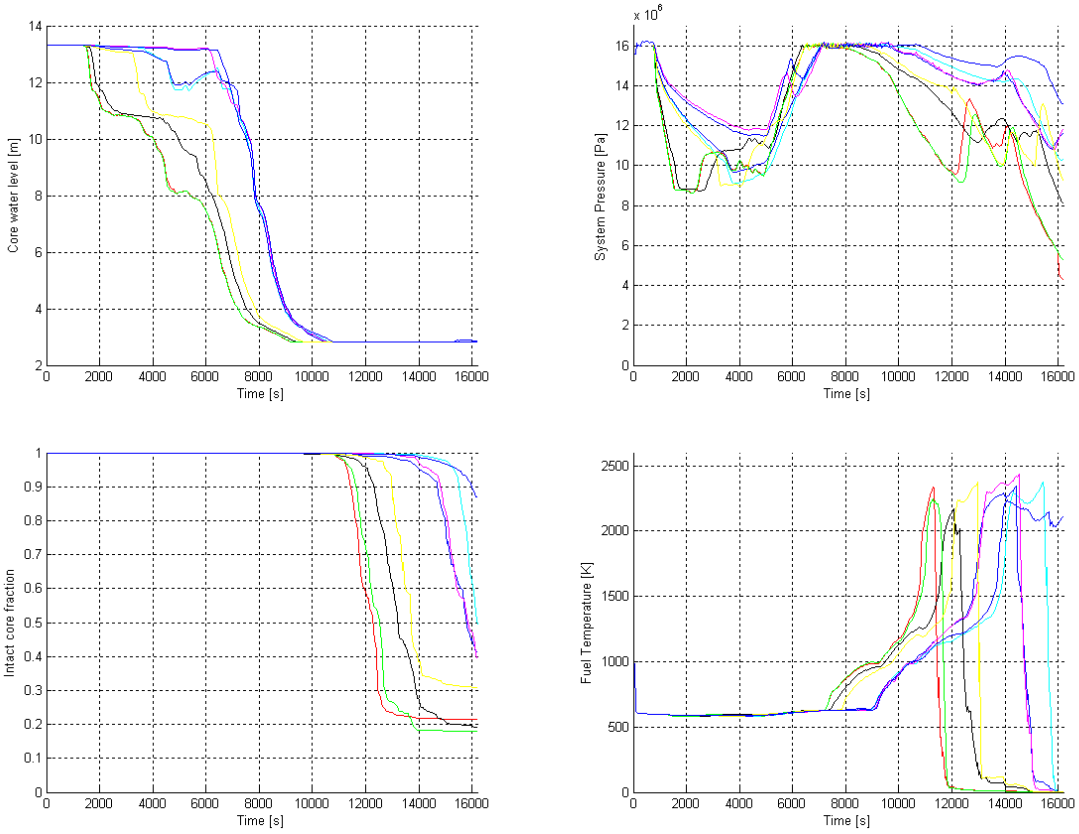

Figure 5.6: Cluster Centers obtained using Fuzzy C-Means for Data set 3.

the data set distribution. However, this ability is lost when the data points are distributed in clusters having complex geometries. K-Means and Fuzzy C-Means methodologies have the same disadvantage since they are able to identify mainly spherical or ellipsoidal cluster of points. However, Mean-Shift algorithm is able to overcome this limitation.

5.6 High dimensionality

An issue that has been emerging in recent years is the increase in the dimensionality of data sets. It started to appear in medical applications for the analysis of genetic data. In the literature, [45] gives a very comprehensive description of the problems when dealing with high-dimensional data and approaches towards this challenging aspect of clustering. Kriege in [45] introduces the fact that with increasing values of dimensionality δ, the set of data points becomes more sparse and, hence, it is harder to define clusters as a partition of a subset of data points.

Moreover, for algorithms that implement metric distances (see Table 3.1) as measure of similarity, another issue arises: depending on the distribution of the data points in the state space and given an arbitrarily chosen point, the relative difference of the distances of the closest and the farthest point (d_{min} and d_{max} respectively) tends to 0:

$$\lim_{\delta \to \infty} \frac{d_{max} - d_{min}}{d_{min}} = 0 \tag{5.7}$$

This would imply that as the dimensionality δ increases, distances between points becomes uniform.

The last issue is more of a geometric effect that can be explained by considering the ratio of the volume of the sphere with radius r and the volume of a cube in

an Euclidean metric space[15]. This ratio tends to 0 as the dimensionality increases. Hence, it seems that as the dimensionality increases the distance measures become meaningless.

As indicated in Section 4.1, each transient is represented by a multidimensional vector where the dimension would be equal to the number of the variables (e.g., temperature, pressure or level of particular components of the plant) chosen to represent the scenarios multiplied by the number of times these variables have been sampled over time. Hence, for any level of safety analysis of a nuclear power plant (or even part of it) that has been carried out over a mission time of the order of hours (or even days), the dimensionality of each data point would be very high.

In order to solve the clustering problem for high dimensional data sets using a Mean-Shift algorithm (or any of the ones listed above) two options would be to: a) act on the pre-processing of the raw data as indicated in Section 4.3, or, b) employ more advanced algorithms able to deal with high dimensionality data.

Only recently studies have been performed towards clustering of high dimensionality data sets and several algorithms have been proposed, including

- PROCLUS [46]
- FINDIT [47]
- COSA [48]
- CLIQUE [49]
- ENCLUS [50]

[15]This issue is also commonly known as ‘‘curse of dimensionality’’.

- MAFIA [51]
- P3C [52]

This dissertation focuses attention on dimensionality reduction techniques prior to data clustering rather than using algorithms that are more suited for having high dimensional data.

CHAPTER 6

MEAN-SHIFT METHODOLOGY

This Chapter describes in detail the theoretical basis of the Mean-Shift algorithm (Section 6.1 and 6.2). Section 6.3 shows an initial testing of clustering using the Mean-Shift algorithm applied to the data set generated by a DET for the analysis of a simple level controller [35]. Section 6.4 shows how the algorithm has been optimized for multi-core processors using both Matlab and C++.

6.1 Theoretical basis

The Mean-Shift algorithm [44] is a non-parametric iterative procedure that can be used to assign each point to one cluster center through a set of local averaging operations [44]. The local averaging operations provide empirical cluster centers within the locality and define the vector which denotes the direction of increase for the underlying unknown density function (see Section 5.4).

The underlying idea is to treat each point $\vec{x_i}$ $(i = 1, \ldots, I)$ of the dataset as an empirical probability distribution function using kernel $K(\vec{x}) : \mathbb{R}^{M \cdot K} \rightarrow \mathbb{R}$. This multivariate kernel density resides in a multidimensional space where regions with high data density (i.e., modes) correspond to local maxima of the density estimate

$f_I(\vec{x})$ [53] (see Fig. 6.1) defined by

$$f_I(\vec{x}) = \frac{1}{Ih^d} \sum_{i=1}^{I} K\left(\frac{\vec{x} - \vec{x_i}}{h}\right), \tag{6.1}$$

where $\vec{x} \in \mathbb{R}^{M \cdot K}$ and h is often referred as the bandwidth associated with the kernel.

The kernel in Eq. 6.1 serves as a weighting function [53] associated with each data point and is expressed as:

$$K(\vec{x}) = c_k k(\|\vec{x}\|^2), \tag{6.2}$$

where $k(x) : [0, \infty] \rightarrow \mathbb{R}$ is referred as the *kernel profile* and c_k is a normalization constant. The profile satisfies the following properties:

- $k(x)$ is non negative
- $k(x)$ is non increasing (i.e., $k(a) \geq k(b)$ if $a < b$)
- $k(x)$ is piecewise continuous and $\int_0^\infty k(x)\, dx < \infty$

In order to estimate the data points with highest probability from an initial estimate (i.e., the modes of $f_I(\vec{x})$), consider the gradient of the density function $\nabla_x f_I(\vec{x}) = 0$ [44] where

$$\begin{aligned} \nabla_x f_I(\vec{x}) &= \frac{2c_k}{Ih^{d+2}} \sum_{i=1}^{I} (\vec{x} - \vec{x_i}) k'\left(\|\frac{\vec{x} - \vec{x_i}}{h}\|^2\right) \\ &= \underbrace{\frac{2c_k}{Ih^{d+2}} \left(\sum_{i=1}^{I} g\left(\|\frac{\vec{x} - \vec{x_i}}{h}\|^2\right)\right)}_{A} \underbrace{\left(\frac{\sum_{i=1}^{I} \vec{x} g\left(\|\frac{\vec{x} - \vec{x_i}}{h}\|^2\right)}{\sum_{i=1}^{I} g\left(\|\frac{\vec{x} - \vec{x_i}}{h}\|^2\right)} - \vec{x}\right)}_{B}, \end{aligned} \tag{6.3}$$

which points in the direction of the increase in kernel density estimate. The kernel $K(\vec{x})$ is also referred to as the shadow of $G(\vec{x}) = c_g g(\|\vec{x}\|^2)$ [54] where c_g, similar to c_k, is a normalization constant and $g(x)$ is the derivative of $k(x)$ over x, i.e., $g(x) = k'(x)$. In Eq. 6.3 the first term denoted as A is a scalar proportional to the density estimate

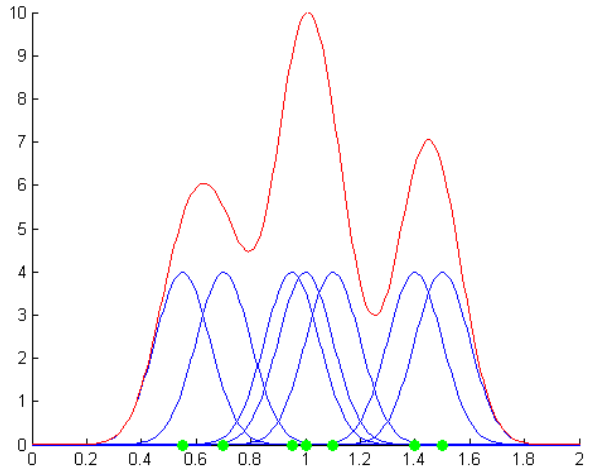

Figure 6.1: Density function.

computed with the kernel $G(\vec{x})$ and does not provide information regarding where the mode resides. Unlike A, the vector quantity B, which is the second term in Eq. 6.3, is difference between the weighted mean

$$m(\vec{x}) = \frac{\sum_{i=1}^{I} \vec{x} g(\| \frac{\vec{x}-\vec{x_i}}{h} \|^2)}{\sum_{i=1}^{I} g(\| \frac{\vec{x}-\vec{x_i}}{h} \|^2)}. \tag{6.4}$$

and the initial estimate $\vec{x}$. This term points in the direction of local increase in density using kernel $G(\vec{x})$, hence provides a means to find the mode of the density. Note that all points used to compute a particular mode are considered to reside in the same cluster.

Since each each data point $\vec{x_i}$ (or scenario) is considered as an empirical probability distribution function, this consideration allows to include in the scenario clustering analysis also the possible uncertainty associated with each scenario.

6.2 The Mean-Shift Algorithm

In order to show how the clustering is performed, consider an example system whose dynamics can be described by 3 variables that include time t, pressure p and temperature T. Also, assume that a DET analysis quantifying the impact of uncertainties in the system yields the trajectories shown in Fig. 6.2. The $S(t)$ plane in Fig. 6.2 represents the state of these scenarios at a particular time instant.

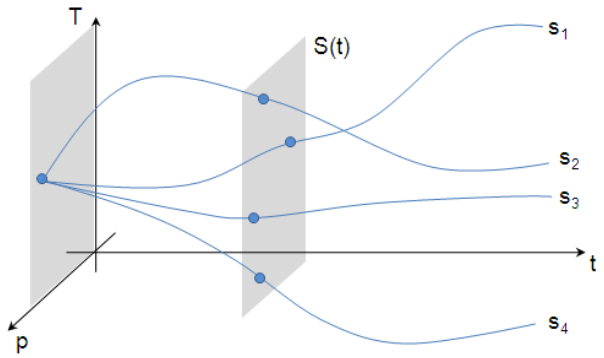

Figure 6.2: Representation of example scenarios generated by a DET in a 3-dimensional space.

Note that the dataset on the $S(t)$ plane consists of points distributed in a two dimensional space $\mathbb{R}^2$. Starting from an arbitrary data point (e.g., point S_A in Fig. 6.3), the algorithm defines the locality using a circular region centered around this point (or a hyperspherical region depending on the number of dimensions). The radius of this region is equal to the bandwidth h of the chosen kernel (see Section 6.1). The objective is to consider points residing in this region for estimating their weighted average which corresponds to the center of mass of these points (point $m(S_A)$ in

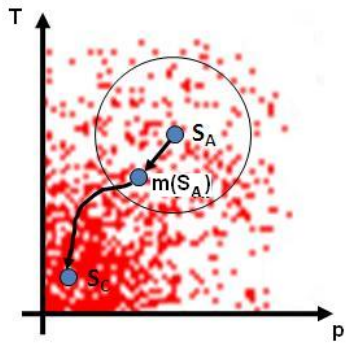

Figure 6.3: Determination of a cluster center in a 2-dimensional space using a Mean-Shift algorithm.

Fig. 6.3) where

$$m(S_A) = \frac{\sum_{i=1}^{I} \vec{x_i} g(\| \frac{S_A - \vec{x_i}}{h} \|^2)}{\sum_{i=1}^{I} g(\| \frac{S_A - \vec{x_i}}{h} \|^2)}. \tag{6.5}$$

The weighted average or the center of mass $m(s_A)$ is used as the new position for S_A for the next iteration, such that the density in the new center of mass is always higher than its previous position. Convergence is reached when the distance between the new center of mass and the old one is below a fixed threshold[16] (point S_C in Fig. 6.3). Upon reaching this stopping condition:

- point S_C is considered the center of a cluster, and,
- the original point S_A is uniquely associated to the cluster centered by point S_C.

During this process, several derivative kernels $G(\vec{x})$ can be used as indicated in [28], including

[16]Usually the threshold is a fixed value chosen to be a small fraction of h (typically $\frac{h}{100}$ or $\frac{h}{1000}$)

$$\text{Flat Kernel: (Fig. 6.4(a))}\ G(\vec{x}) = \begin{cases} 1 & \text{if } \|\vec{x}\| \leqslant h \\ 0 & \text{if } \|\vec{x}\| > h \end{cases} \tag{6.6}$$

$$\text{Cone Kernel: (Fig. 6.4(b))}\ G(\vec{x}) = \begin{cases} (1 - \|\vec{x}\|) & \text{if } \|\vec{x}\| \leqslant h \\ 0 & \text{if } \|\vec{x}\| > h \end{cases} \tag{6.7}$$

$$\text{Gauss Kernel: (Fig. 6.4(c))}\ G(\vec{x}) = e^{-\frac{\|\vec{x}\|^2}{h^2}} \tag{6.8}$$

As indicated earlier, the purpose of $G(\vec{x})$ is to assign different weights to different points during the estimation of the center of mass. Kernels such as the Gauss and the Cone kernel assign higher weights to the points located at the center of kernel which implies that the calculated center of mass is biased toward the center of the kernel. Thus, the distance between two consecutive calculated positions of the center of mass is smaller if Gauss or Cone kernels are used compared to the uniform kernel. Consequently, convergence can be reached faster using kernels such as the Gauss or the Cone ones.

When this procedure is repeated for all the points in the data set it is possible to obtain

- the center of all the clusters and the list of all the points that belong to that specific cluster, and,
- the cluster to which each point belongs (as mentioned earlier, each point belongs to one cluster only).

Figure 6.5 shows the flow diagram of the proposed approach. Starting from a dataset generated by the DET tool, the user selects variables of interest (Feature

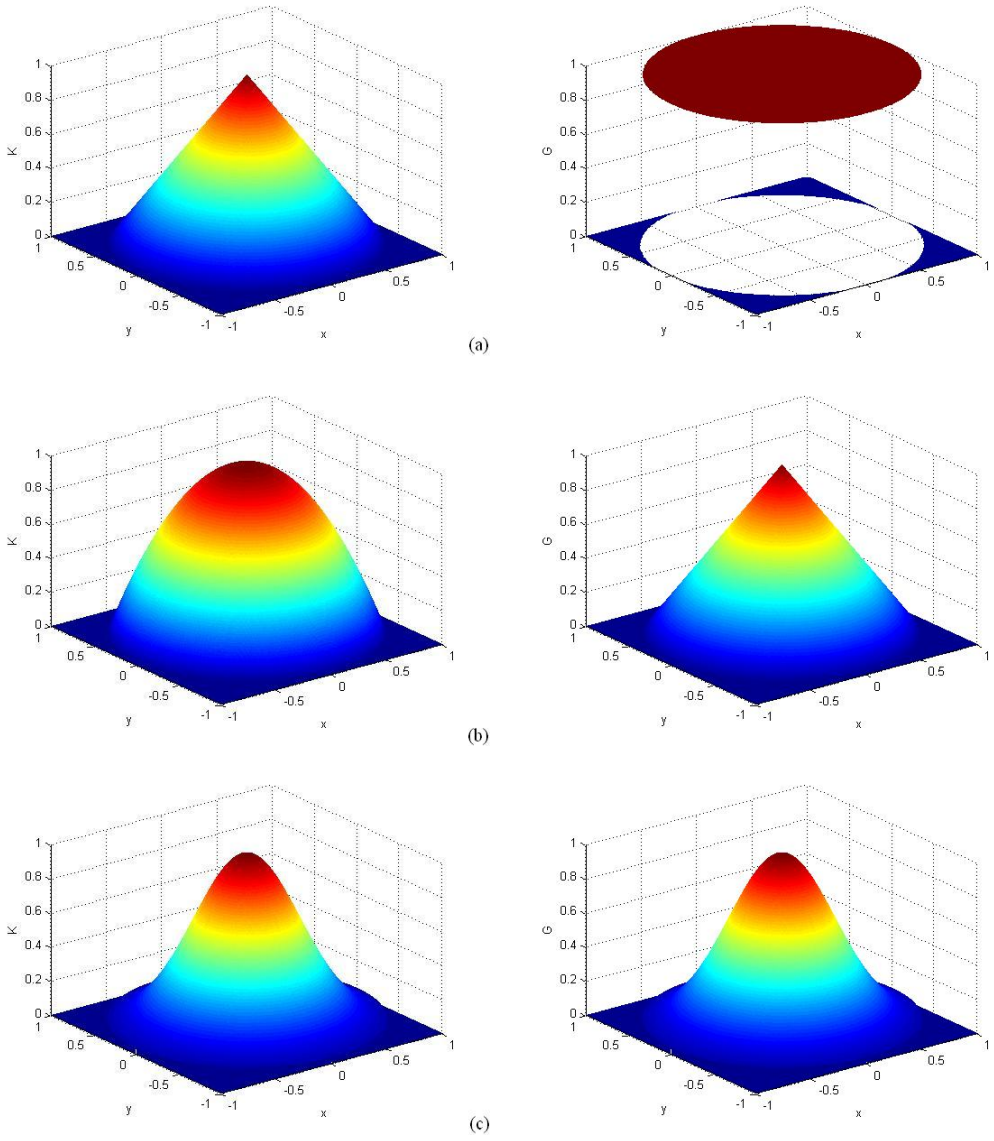

Figure 6.4: Graphical representation of the 2-D $K(\vec{x})$ kernels (left) and the corresponding $G(\vec{x})$ kernel.

Selection in Fig. 6.5). Following the choice of bandwidth, type of kernel, and the distance metric, the clustering is iteratively performed (Clustering in Fig. 6.5). Sensitivity of the grouping of scenarios can be then examined for different types of kernels, values of bandwidth and metrics (Post-Processing in Fig. 6.5).

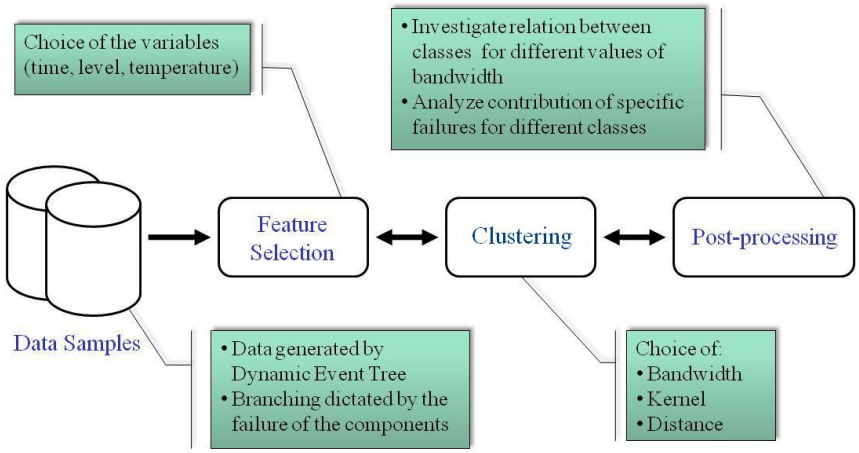

Figure 6.5: General Scheme of the Mean-Shift Methodology algorithm.

6.3 Water level controller analysis

As a simple system to illustrate the deployment of the algorithm shown in Fig. 6.5, the heated tank example presented in [55] and shown in Fig. 6.6 is used here.

The liquid level L is actively controlled through the actuation of three components: two inlet pumps and one outlet valve, hereafter called Units 1, 2 and 3, respectively. Each unit is a multi-state component operating either correctly ON or OFF, stuck ON or stuck OFF. At time $t = 0$, the system is assumed to be in its nominal state (ON,OFF,ON), with nominal values of $T = 30.93C$ of the liquid temperature and

$L = 7m$. The temperature T is assumed to directly affect the failure rates λ of the components as given by [55]:

$$\lambda(T) = \frac{b_1 e^{-b_d(T-20)} + b_2 e^{-b_d(T-20)}}{b_1 + b_2} \overline{\lambda(i)} \tag{6.9}$$

where b_i and $\overline{\lambda(i)}$ are constant and characteristic of each component.

A power source heats up the fluid to keep it at the nominal temperature. The liquid level is kept between 6 and 8 m through the control laws reported in [17] (also see Table 6.1). Two possible Top Events that need to be considered are Low Level ($L < 4m$) and High Level ($L > 10m$).

Table 6.1: Water level controller control laws.

Case	Controller 1	Controller 2	Controller 3
$6m \leq L \leq 8m$	ON	OFF	ON
$L \leq 6m$	ON	ON	OFF
$L \geq 8m$	OFF	OFF	ON

For testing the algorithm described in Section 6.2, the variables that describe the evolution of each scenario were chosen to be the temperature T, level L and time t. Thus the state space is three-dimensional.

In the DET analysis, the branching is dictated by the failure of the three active components (i.e. Units 1, 2 and 3 in Fig. 6.6). The Mean-Shift algorithm has been applied to the set of transients generated in [56] and shown in Fig 6.7.

Each transient contains information about the time evolution of temperature and level and the status of the three controllers as well. Every transient has been converted into a vector $\vec{x_i}$ as shown in Eq. 4.1 with

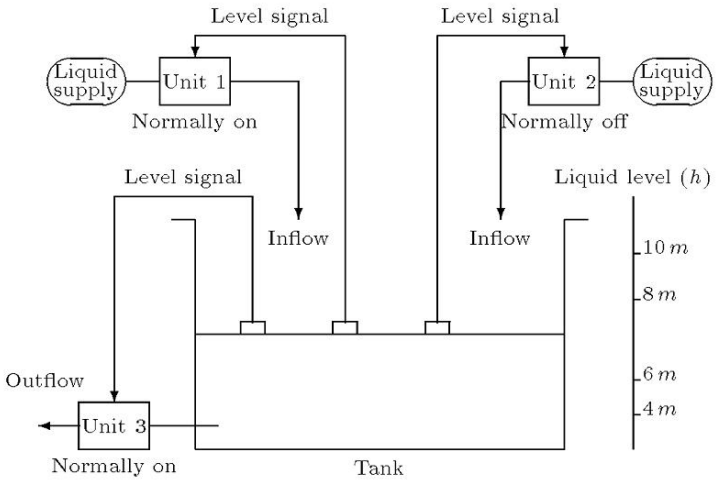

Figure 6.6: Scheme of the water heated tank.

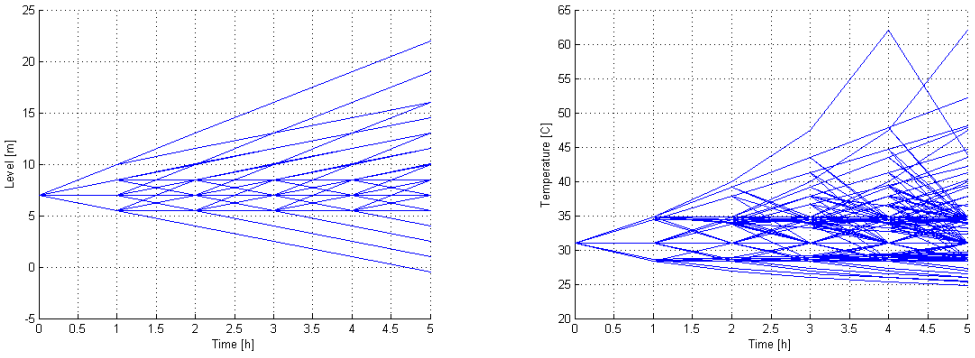

Figure 6.7: Plots of the scenarios generated by DET for the level controller.

- $M = 2$ (i.e., 2 state variables: temperature and level),
- time is ranging from 0 to 5 hours (i.e., $T = 5h$), and,
- level, temperature sampled every hour (i.e., K=5): $t_1 = 0h$, $t_2 = 1h$, $t_3 = 2h$, $t_4 = 3h$, $t_5 = 4h$ and $t_6 = 5h$.

Thus, each scenario is characterized by a vector having dimensionality equal to 10 ($M \cdot K = 10$).

Clustering was performed for the scenarios leading to Low Level and High Level separately. Figures 6.8, 6.9 and 6.10 show the cluster centers (i.e., the representative scenarios) for the these Top Events. As can be seen from the figures, the clustering process becomes more refined by decreasing the value of the bandwidth and the number of clusters obtained increases. Asymptotically, the number of clusters equals the number of scenarios with decreasing bandwidth.

6.4 Parallel implementation

The Mean-Shift algorithm has been developed initially using Matlab. It was found to be more appropriate to rewrite the algorithm and design it for parallel computing in a more suitable form using C++.

In the literature it is possible to find two major approaches for parallel programming applied to C++:

- OpenMP [57]
- MPI [17]

[17] "MPI: The Complete Reference" by Snir, Otto, Huss-Lederman, Walker, and Dongarra, MIT Press (http://switzernet.com/people/emin-gabrielyan/060708-thesis-ref/papers/Snir96.pdf)

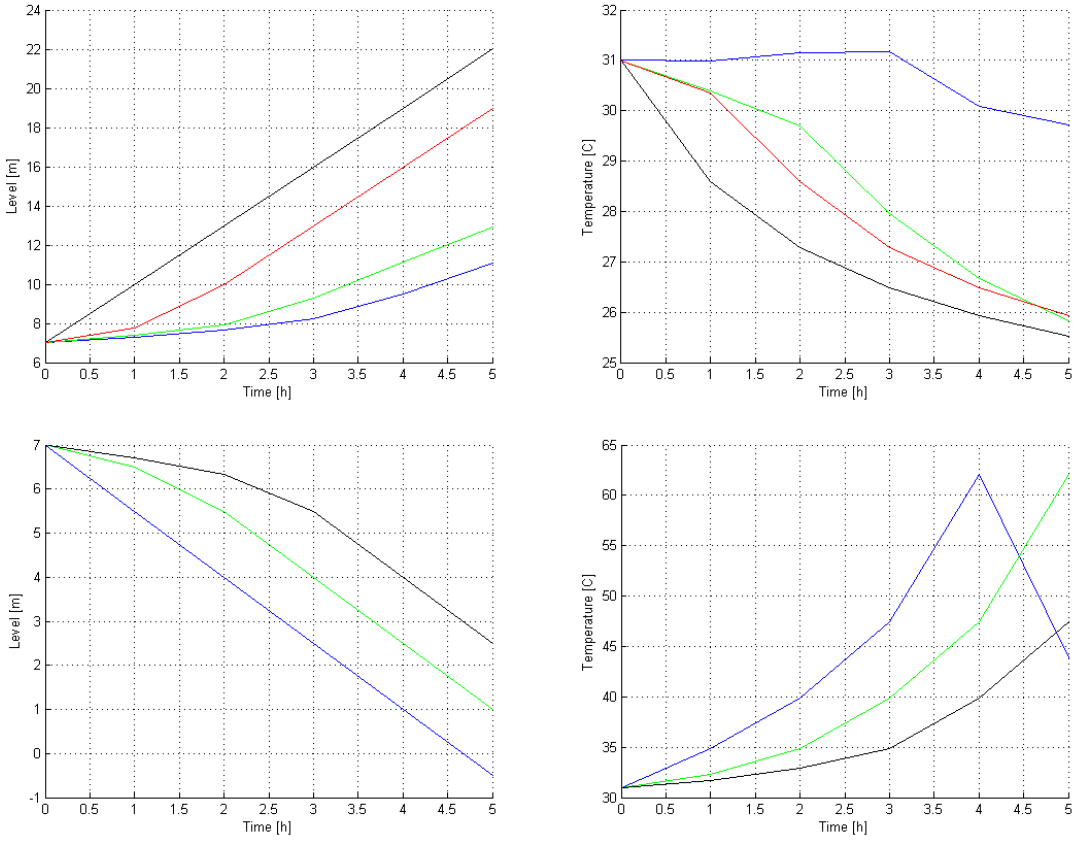

Figure 6.8: Cluster centers for High Level (top) and Low Level (bottom): $h = 11$.

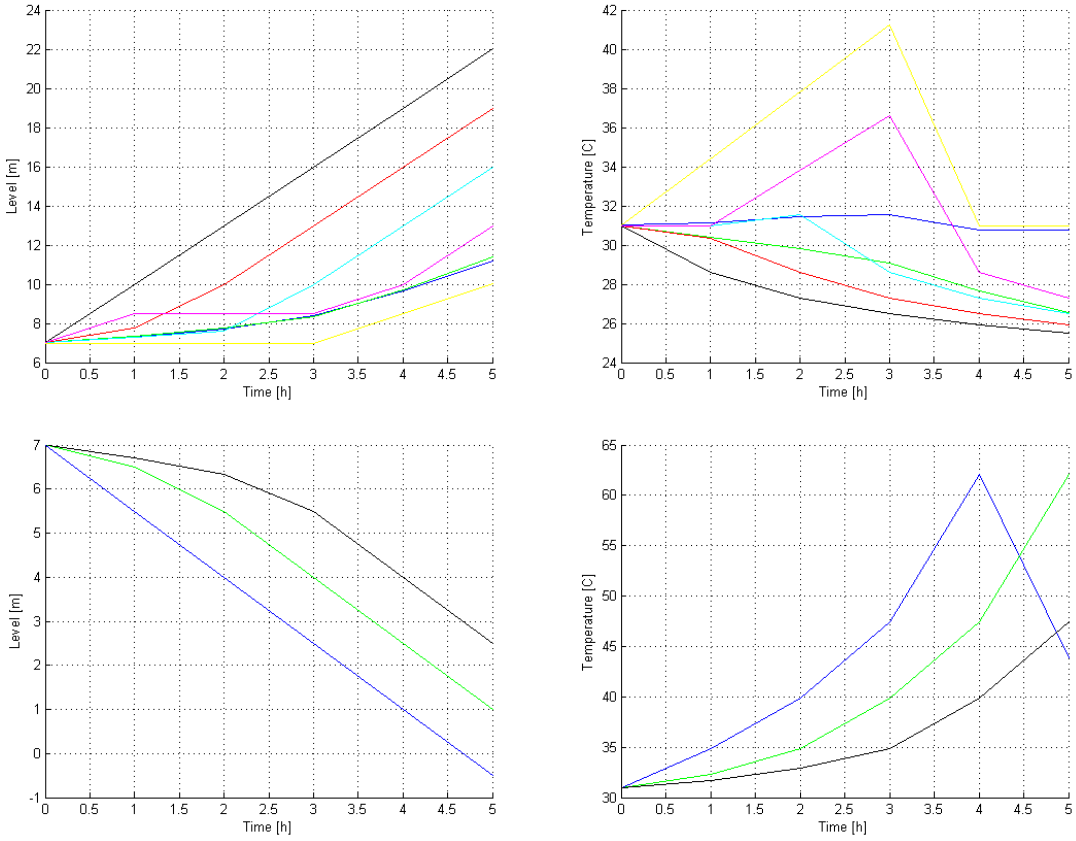

Figure 6.9: Cluster centers for High Level (top) and Low Level (bottom): $h = 9$.

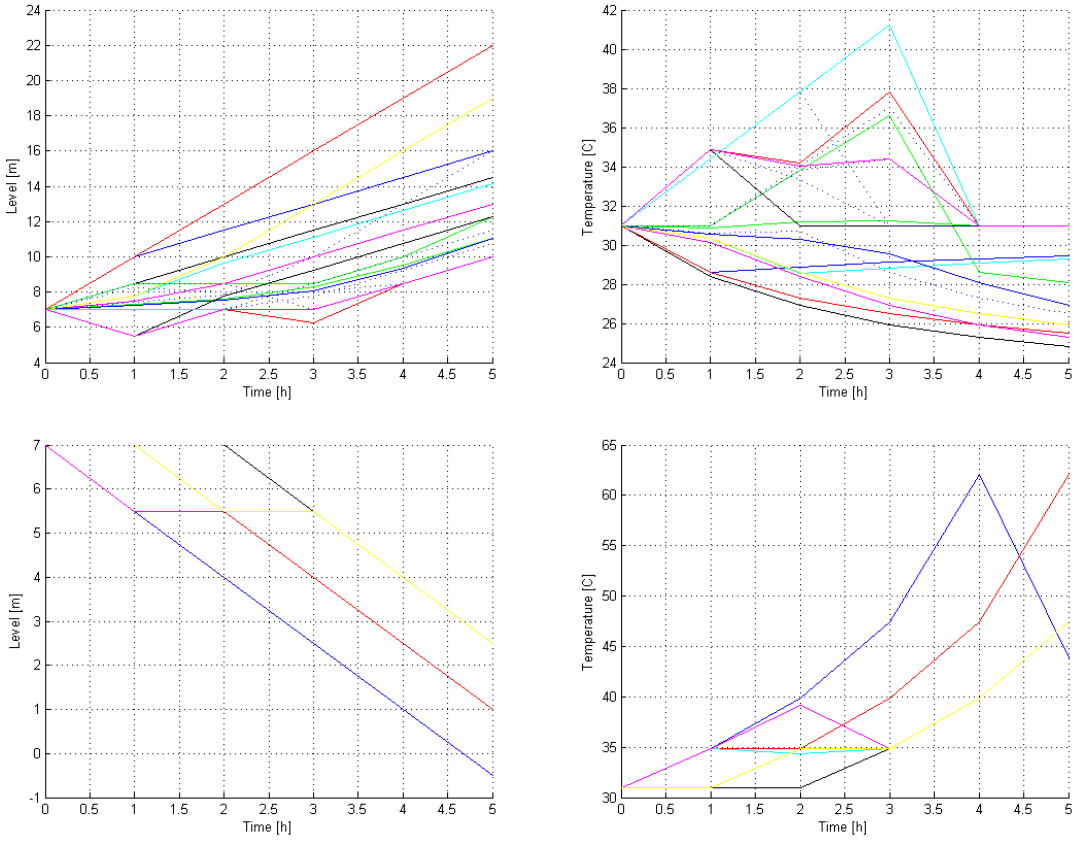

Figure 6.10: Cluster centers for High Level (top) and Low Level (bottom): $h = 7$.

For the scope of this dissertation the use of directives of OpenMP are more suited due to fact that OpenMP supports shared memory computation. OpenMP (Open Multi-Processing) is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran on many architectures, including Unix and Microsoft Windows platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior. OpenMP is an implementation of multi-threading, a method of parallelization whereby the master ''thread'' (a series of instructions executed consecutively) ''forks'' a specified number of slave ''threads'' and a task is divided among them. The threads then run concurrently, with the runtime environment allocating threads to different processors.

The Mean-Shift algorithm described in Section 6.2 is performed for each data point in a single thread. During this process the algorithm initially looks at all the points within the bandwidth. Thus, a shared memory parallel implementation is a natural choice. Such an implementation of the Mean-Shift algorithm in C++ using OpenMP directives is performed in two steps:

1. For each thread, implement the algorithm of Section 6.2 for each data point separately to find the center of the cluster associated with the data point

2. Determine the global cluster centers from the set of centers determined in Step 1

The code is presented in Appendix B while results are shown in Section 7.6.

CHAPTER 7

LEVEL 2 PRA ANALYSIS

This chapter presents the results obtained by the algorithm described in Section 6.2 and shown in Appendix A for four different cases:

1. Sodium-Cooled Fast Reactor (SFR) [58] recovery after an aircraft crash scenario (see Section 7.1)
2. Zion plant [23] during a station blackout (see Section 7.2 and Section 7.4)
3. Analysis of the pump seal leakage model (see Section 7.3)

Sections 7.5 and 7.6 show results of the dimensionality reduction algorithm described in Section 4.5 and the parallel implementation of Mean-Shift described in Section 6.4 respectively, for the data set of Section 7.3.

7.1 SFR Aircraft Crash Analysis

After the analysis of the water level controller presented in Section 6.3, a more complex system, the reactor vessel auxiliary cooling system (RVACS) of the sodium-cooled fast reactor [58] has been analyzed. The RVACS is schematically shown in Fig. 7.1. The RVACS is a passive decay-heat removal system that removes heat by

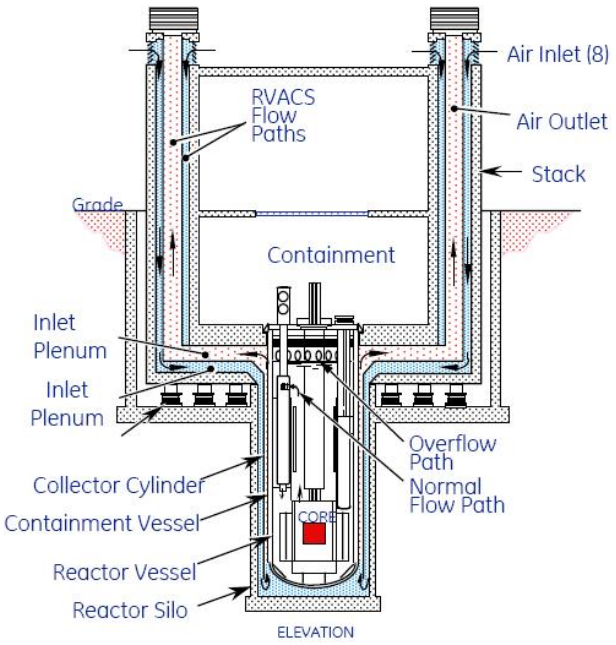

Figure 7.1: RVACS system applied to SFR.

natural circulation of air in the gap between the vessel and a duct surrounding the vessel. With this system, the reactor decay heat is released to the atmosphere through four towers or stacks.

The Analysis of Dynamic Accident Progression Trees (ADAPT) tool [1] has been used here as the DET generator tool while the system dynamics are modeled using RELAP5 [4]. At time zero with the plant operating at 100% power, an aircraft crashes into the plant. Three of the four towers are assumed to be destroyed, producing debris that blocks the air passages (hence, impeding the possibility to remove the decay heat). The reactor trips, off-site power is lost, the pump trips and coasts down.

A recovery crew and heavy equipment are used to remove the debris. Figure 7.2 illustrates the strategy that is followed by the crew in reestablishing the capability of the RVACS to remove the decay heat. Crew arrival and tower recovery times are stochastic variables and can have any value within the specified bands. Several assumptions have been made for the purpose of the analysis:

- A tower is assumed to have no heat removal capacity until the rubble has been removed. At that point it is assumed to regain full capacity
- There is a one hour period following the crash in which a fire is being extinguished
- There is a uniform probability of work being initiated between one and nine hours after the crash
- The workers remove debris from one tower at a time
- After work begins on a tower there is a minimum time of two hours to recover the tower
- There is a uniform distribution of recovery between two and ten hours. The team then moves on to the next tower
- The recovery time of each tower is assumed to be independent of the other towers

As indicated above, the uncertainties in crew arrival time and tower recovery have been modeled by assigning to each one a uniform probability distribution function [59]. Example branching times were chosen to correspond to the probabilities 0.001, 0.25, 0.5, 0.75, and 1.0 on the corresponding cumulative distribution function. When one of

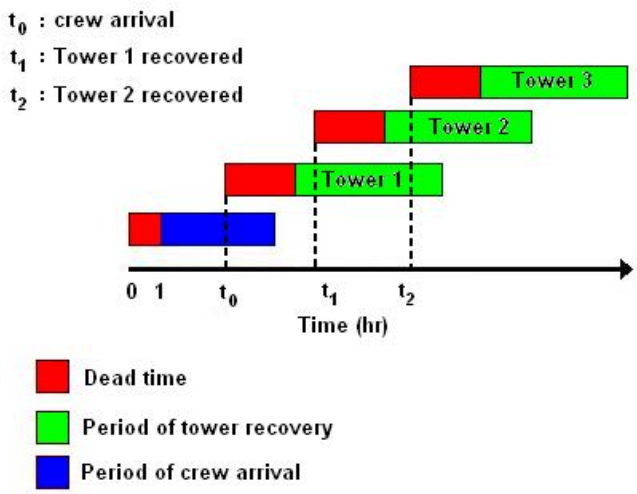

Figure 7.2: Crew recovery strategy for the aircraft crash scenario.

these branching times is reached in the RELAP simulation, a bifurcation in the system evolution occurs in which one branch represents the realization of the specific event (e.g., crew arrival) and the other branch represents non-realization of the specific event. The latter branch then continues until the time corresponding to the next branching point is reached on the cumulative distribution function.

Only one Top Event has been considered: temperature T of the core reaches the limit of 1000 K, associated with clad failure by eutectic formation. Figure 7.3 shows the temporal behavior of the temperature of the core for all the 610 scenarios generated by ADAPT. Mission time for this system analysis has been fixed to $2 \times 10^5 s$[18].

Each transient includes information about:

[18]In the original data set, scenarios that reach a core temperature of 1000 K are stopped even though they did not reach $2 \times 10^5 s$. Consequently, the time length of scenarios may change depending if they have reached 1000 K or not. For those scenarios that reach 1000 K before $2 \times 10^5 s$, it has been decided to extend in time these scenario up to $2 \times 10^5 s$ with the last value simulated.

- Time profile of core temperature
- Crew arrival time
- Recovery time of tower i ($i = 1, 2, 3$)

Clustering analysis of the data set pictured in Fig. 7.3 has been performed using the values of temperatures sampled at specific time instants. Each scenario was represented by $K = 68$ time points sampled uniformly over a time horizon of $T = 2 \times 10^5 s$.

Table 7.1 shows the number of clusters obtained for different values of bandwidth h. For very large values of h (e.g., $h = 4$) the algorithm determines a single cluster which contains all the 610 scenarios. On the other hand, for very small values of h (e.g., $h = 0.001$) the algorithm determines 610 clusters; each of them contains a single scenario. For both of these cases no additional information can be extracted from the clusters since the clusters reflect exactly the original data set.

Table 7.1: Number of clusters obtained for different values of bandwidth h.

h	Number of clusters
4.0	1
3.0	2
2.0	4
1.5	8
1.0	22
0.5	96
0.1	300
0.01	308
0.001	610

Figure 7.4 shows the cluster centers obtained for $h = 1.5$. Figure 7.4 also shows, for each of the 8 cluster centers, the cluster probability and the fraction of scenarios

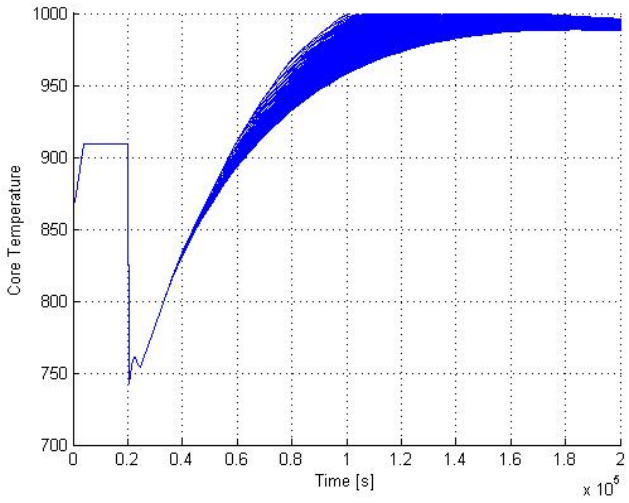

Figure 7.3: Graphical representation of the scenarios generated by ADAPT for the aircraft crash scenario.

that belong to that cluster. Cluster probability is determined by summing all the probabilities of the scenarios contained in that cluster. Figure 7.5 shows the cluster centers (i.e., the representative scenarios) in black lines and the scenarios belonging to that cluster in red lines.

At this point it is possible to analyze the properties of the clusters individually instead of the full data set. In this respect, a second analysis has been performed for each of the eight obtained clusters by evaluating the distribution of the crew arrival time (red) and the recovery time of tower 1 (blue), 2 (green) and 3 (magenta) as function of time for the scenarios belonging to each cluster. Figure 7.6 shows, for each of the 8 clusters, the distribution of the crew arrival time and tower recovery time for all the scenarios belonging to that cluster.

Figures 7.5 and Fig. 7.6 indicate the following:

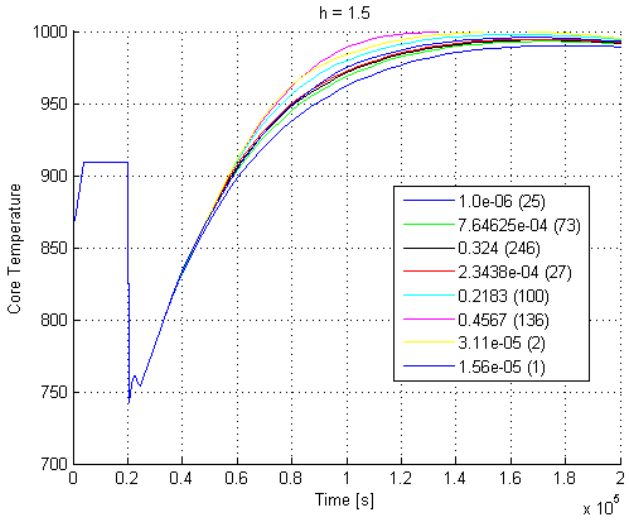

Figure 7.4: Cluster centers for the RVACS system for $h = 1.5$. The numbers in the legend indicate the cluster probability and, in parenthesis, the number of scenarios that fall in each cluster.

- Scenarios contained in Clusters 1 and 4 are characterized by very early crew arrival (red bars in Fig. 7.6 located at $2.5 \times 10^4 s$) and a rapid sequence of towers recovery which allow to cool the core rapidly (all towers are recovered before $8.5 \times 10^4 s$ for all scenarios contained). As shown in Fig. 7.5, the scenarios included in both clusters lead to adequate core cooling (i.e., the maximum core temperature do not reach $1000K$) and, as expected, a rapid recovery of the towers is sufficient condition for the safety of the plant

- Scenarios contained in Cluster 2 are characterized as well by an early crew arrival (red bar in Fig. 7.6 located at $2.5 \times 10^4 s$) and a rapid recovery of the first tower (blue bars in Fig. 7.6 located between $3.5 \times 10^4 s$ and $4.5 \times 10^4 s$). However, the recovery of the remaining two towers is not as rapid (green and

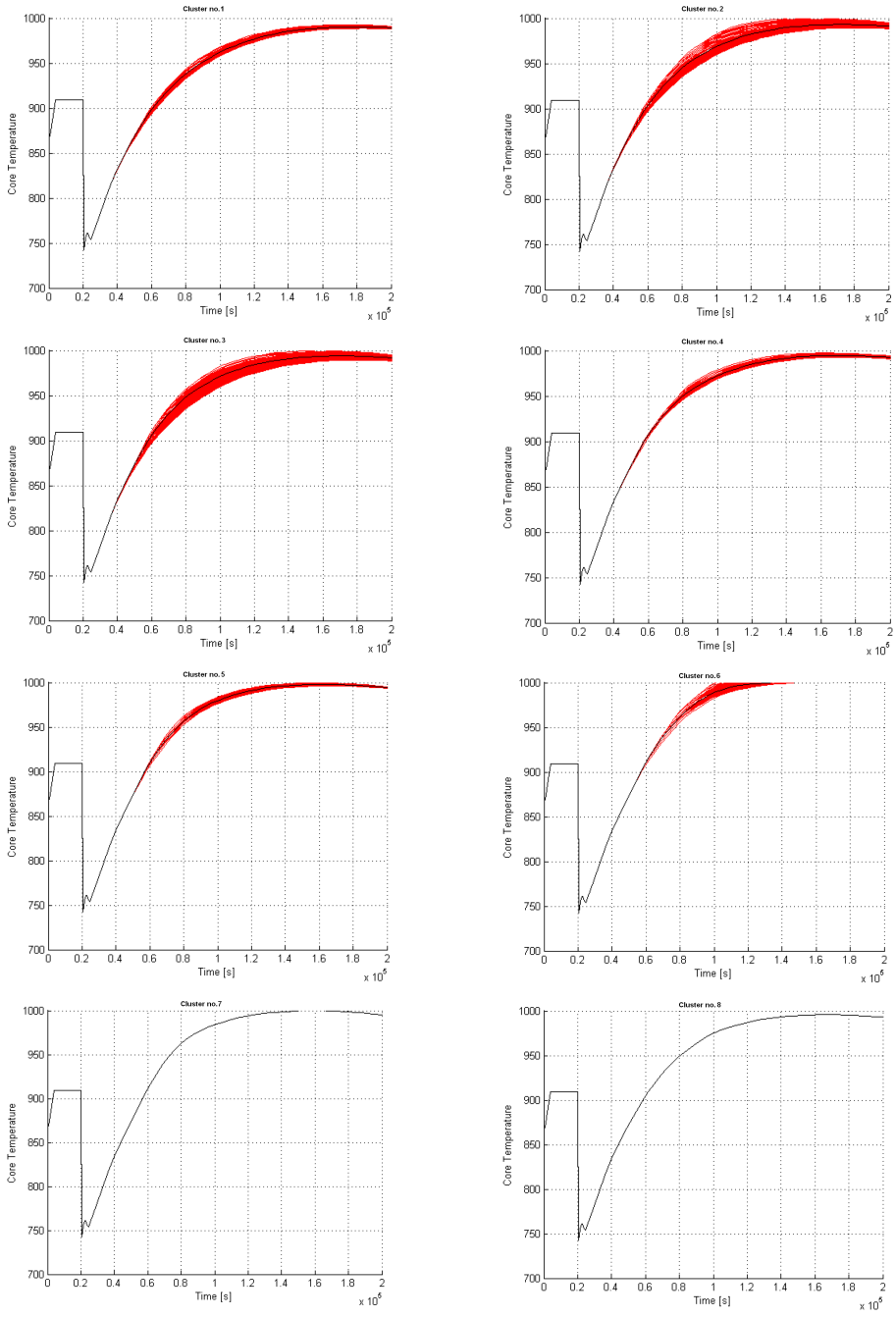

Figure 7.5: Cluster centers (black lines) and scenarios associate to it (red lines) for each cluster.

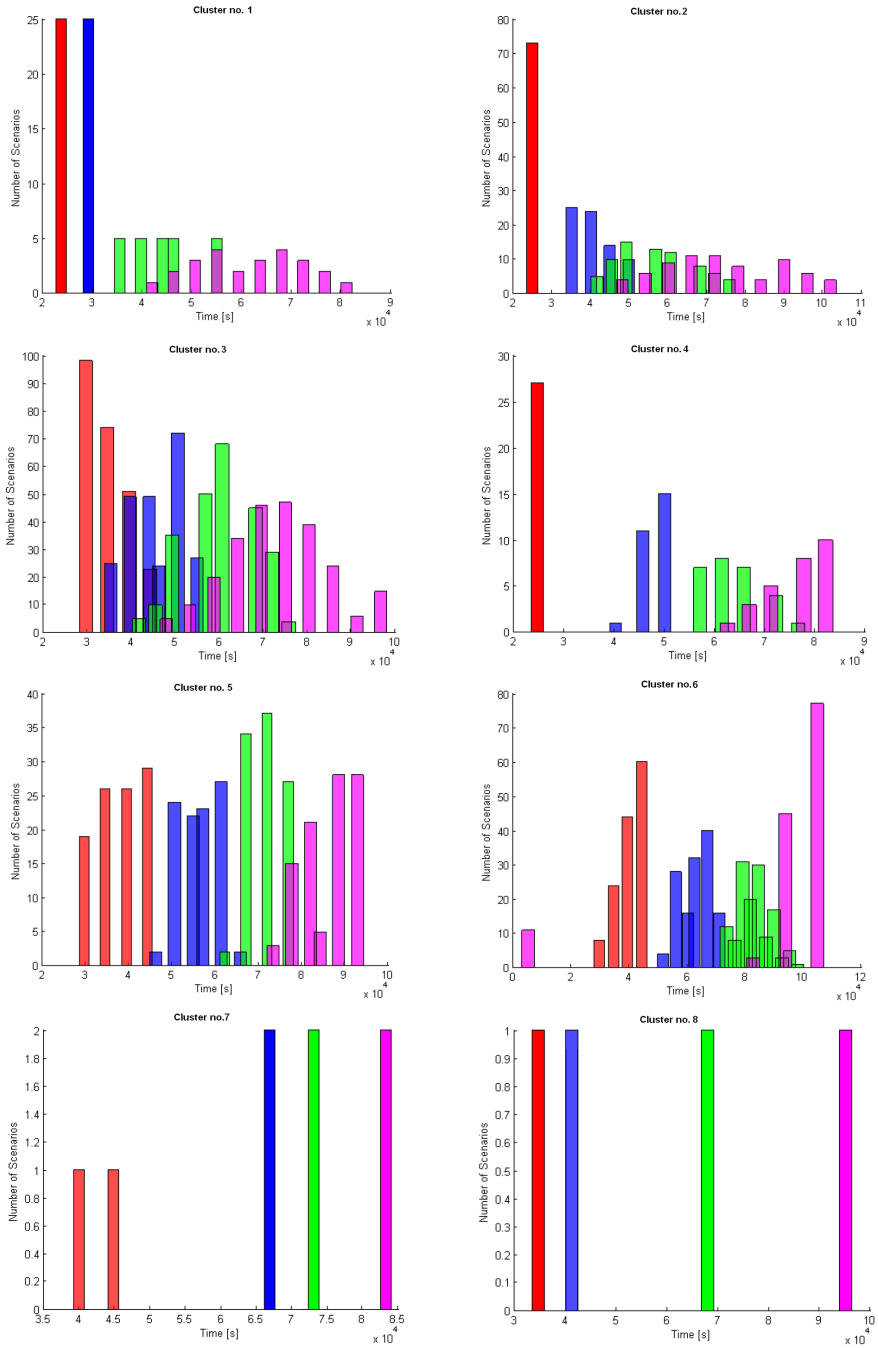

Figure 7.6: Distribution of crew arrival time (red) and the recovery time of tower 1(blue), 2(green) and 3(magenta).

purple bars in Fig. 7.6). Since this cluster contains scenarios that lead to both adequate core cooling and core damage (see Fig. 7.5 where some scenarios reach the limit of $1000K$ and others are below it), it is possible to conclude that early crew arrival and recovery of the first tower is not sufficient condition to reach the safe state of the plant.

- Scenarios contained in Cluster 3 and 5 lead to both sufficient core cooling and core damage (see Fig. 7.5 where some scenarios reach the limit of $1000K$ and so are below it). This is due to the fact that crew arrives on the field considerably late (red bars in Fig. 7.6 located between $3.0 \times 10^4 s$ and $4.5 \times 10^4 s$) followed by a rapid recovery of the three towers (blue, green and purple bars in Fig. 7.6). However, the rapid recovery is not sufficient to avoid core damage.

- Cluster 6 is composed exclusively of scenarios that lead to core damage (see Fig. 7.5 where all scenarios reach the limit of $1000K$). Moreover, it also contains all the scenarios characterized by the non recovery of the third tower (purple bar located at $0s$ in Fig. 7.6). As observed for Clusters 3 and 5, crew arrives very late on site (red bars in Fig. 7.6 located between $3.0 \times 10^4 s$ and $4.5 \times 10^4 s$) and hence the tower recovery strategy is not sufficient for adequate core cooling.

- Clusters 7 and 8 contain a very small number of scenarios (i.e., 2 and 1 scenario respectively). Scenarios included in Cluster 7 are characterized by a late crew arrival (see red bars in Fig. 7.6), a late recovery of the first tower (see blue bars in Fig. 7.6) and a fast recovery of the following towers (see green and purple bars in Fig. 7.6). This action allows to sufficiently cool the core but temperature profile for these scenarios is very close to the limit temperature (i.e.,

max temperature of the core for the scenarios in Fig. 7.5 is $999K$). However, the scenario included in Cluster 8 is characterized by an early crew arrival and recovery of the first tower (see red and blue bars in Fig. 7.6). The recovery of the remaining towers is not particularly fast (see green and purple bars in Fig. 7.6) but it avoids to reach the limit temperature.

In conclusion, the clustering of the data set was able to identify similarities between scenarios that are leading to both core damage and to adequate core cooling. These similarities were identified in Clusters 2, 3 and 5. By analyzing the similarities in scenarios contained in these three clusters it was possible to identify the following:

- Early crew arrival and early recovery of the first tower is not sufficient condition to adequate core cooling; a late recovery of the remaining two towers leads to core damage
- Late crew arrival time does not lead to core damage but the fast recovery of the 3 towers can be sufficient to provide adequate core cooling.

7.2 Zion Plant Analysis: Station Blackout (1)

The initiating event investigated in this section is the station-blackout (SBO) at a U.S. Pressurized Water Reactor (PWR) and the MELCOR code [5] was linked to the ADAPT tool [12] to determine the evolution for each DET scenario. The simulations using MELCOR as the system code model the transient from the occurrence of the initial condition through the core-melting phase of the accident and up to point of containment failure and release of radionuclides to the environment. For this case, two branching conditions were considered:

1. Creep rupture of major reactor coolant system (RCS) components (hot leg, pressurizer surge line, and steam generator tubes)

2. Failure of the containment vessel

For each of these phenomena, distributions were developed [1] which gave the probability of occurrence of a phenomenon as a function of certain system variables. For the first branching condition (creep rupture) a probability distribution was developed [60] on the value of the component lifetime fraction which would lead to failure. For the second branching condition (containment failure), a distribution was developed [60] which gave the probability of containment failure as a function of containment pressure.

Since the ADAPT methodology is discrete in nature, it is necessary to discretize the branching condition distributions [1]. Each branching condition distribution was discretized into seven points, namely, for each of these branching conditions, discrete probability points were selected from the appropriate cumulative distribution functions (CDFs), and the physical values corresponding to these probability values were used as branching criteria. For each branching condition, discrete probability points of 1%, 5%, 25%, 50%, 75%, 95%, and 99% were chosen.

All the 176 scenarios generated in this DET led to containment failure at some point in the scenario evolution. With regards to creep rupture, failure of the surge line dominated this failure mode, with surge line failure also occurring in all scenarios. The DET analysis also showed the potential failure of steam generator tubes before failure of the surge line occurred. However, in this case, steam generator tube rupture was modeled as the failure of a single steam generator tube and this failure did not result in sufficient reactor coolant system depressurization to preclude future failures. As a

result steam generator tube rupture had a limited effect on the risk in this scenario as it resulted in little to no containment bypass. Hence, the major contributor to the release of radionuclides to the environment is the over-pressurization and failure of the containment structure (containment failure modes such as basmat melt-through and other energetic containment failure events are not modeled here).

For this data set, four variables have been chosen to represent each scenario:

1. Average core coolant temperature (x_1)
2. Primary system pressure (x_2)
3. Containment temperature (x_3)
4. Containment pressure (x_4)

Since this scenario deals with both the core melting phase and the containment failure phase of the accident, it was felt that variables should be chosen which represent these phenomena. Variables 1 and 2 are surrogates for the progression core melt and the integrity of the reactor coolant system. Variables 3 and 4 relate to the integrity of the containment.

As indicated in Section 4.1, each scenario $\vec{x_i}$ has been represented by a vector as following:

$$\vec{x_i} = [x_1(t_1), x_2(t_1), x_3(t_1), x_4(t_1), \ldots, x_1(t_K), x_2(t_K), x_3(t_K), x_4(t_K)] \tag{7.1}$$

In each simulation, the set of four variables were sampled every 500 seconds (on mission time of $t_K = 12 \cdot 10^4 s$, $t_1 = 0s$). Since h is a parameter which must be defined

by the user before performing the clustering analysis, clustering has been performed for different values of h and the impact of the choice of h on the fidelity of the risk results has been examined.

In our approach to clustering analysis, a parameter that needs to be selected by the user is the bandwidth h. In numerical analysis a similar issue arises when the size of the mesh grid needs to be decided and, in such a case, the objective function is represented by the ''difference'' between the analytical and the numerical solutions.

For this study, the objective function F is represented using the CDF for the containment failure as function of time. In particular, this is accomplished by evaluating the difference between area underneath the CDF for the full data set ($CDF_{FullData}(t)$) and the CDF for the obtained clusters ($CDF_{Clusters}(t)$):

$$F = \left| \frac{\int_0^{12\cdot10^4} CDF_{FullData}(t)\,dt - \int_0^{12\cdot10^4} CDF_{Clusters}(t)\,dt}{\int_0^{12\cdot10^4} CDF_{FullData}(t)\,dt} \right| \tag{7.2}$$

In this application, F was selected to be below 5%. Hence, the chosen value of bandwidth h is such that $F \leqq 5\%$.

Clustering was performed for different values of h and compared the CDF for containment failures for each case with the CDF obtained from the full set of data. The timing of containment failure has a major impact on severe accident consequences. Figure 7.7 shows the CDF for containment failure as a function of time for the full data set (no clustering) and for the four different values of h.

The results show a convergence of the risk results generated from the aggregated data to the raw DET results as the h decreases (see Fig. 7.8). For $h = 14$ (4 clusters), the distribution is not well captured in the period before 60,000s. However, the analysis computed with $h = 13$ and $h = 11$ does a much better job of approximating

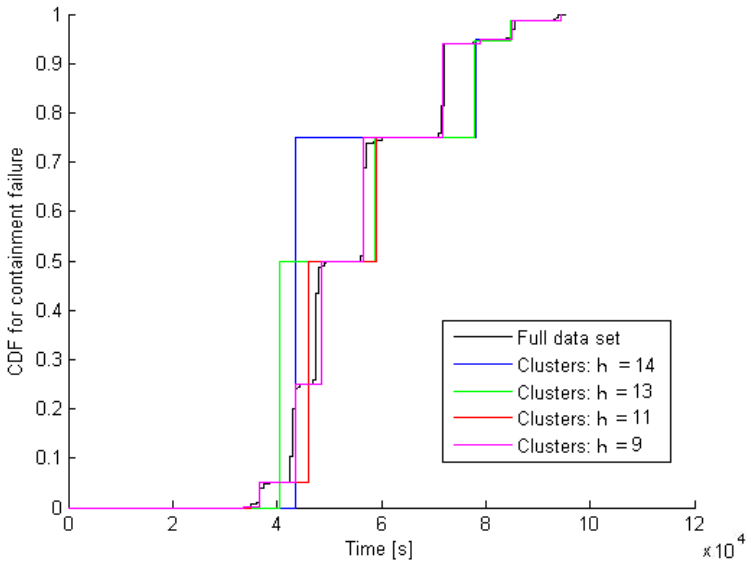

Figure 7.7: CDF for containment failure for the whole set of data compared to the ones obtained from the clustering process for different values of bandwidth h.

the distribution resulting from the raw data. Finally, the analysis performed using $h = 9$ (15 clusters) results in a distribution of containment failure time which closely approximates the distribution resulting from the raw DET data ($F \leqq 5\%$). Figure 7.9 shows a plot of the representative scenarios for the case where $h = 9$. These results show that the clustering scheme reduced the number of scenarios to consider from 176 down to 15 clusters (each of which can be analyzed separately) while still capturing the consequence distribution well.

7.3 Zion Plant Analysis: pump seal leakage

The purpose of this section is to analyze the impact of the pump seal leakage on the scenarios generated by ADAPT. Three different models have been considered and

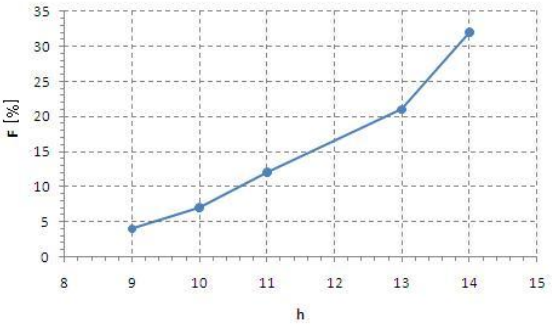

Figure 7.8: Objective function F (Eq. 7.2) as function of the bandwidth h.

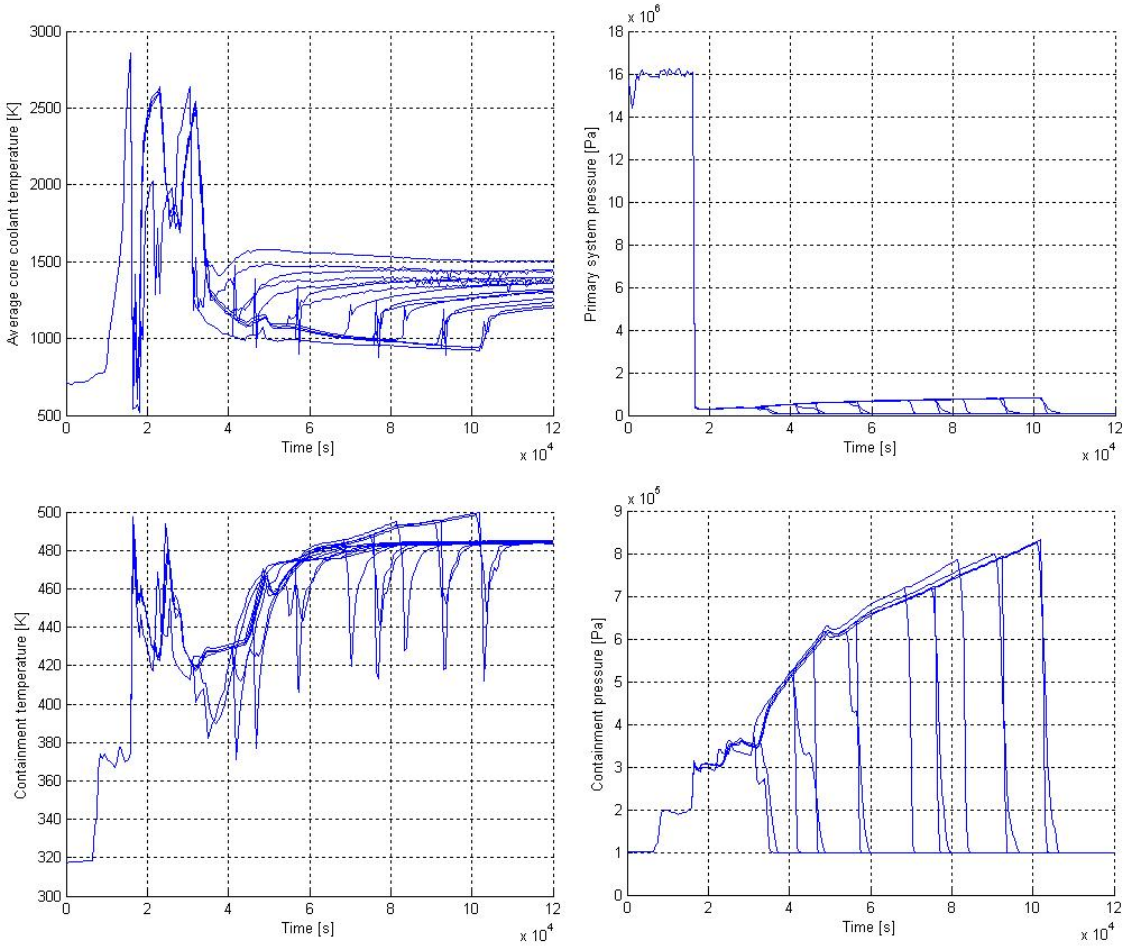

Figure 7.9: Cluster centers obtained for $h = 9$.

for each of these a DET analysis for a station black out of the Zion power plant has been performed. As a notation the three DET analysis are called experiments 135, 136 and 137. The goal of the clustering analysis is to identify differences between the three data sets by observing differences in the cluster centers obtained by Mean-Shift algorithm.

In all the three cases each scenario is represented by 8 variables:

- Seal LOCA flow rate [gpm]
- Hydrogen mass generated [kg]
- Core water level [m]
- System Pressure [Pa]
- Core vapor temperature [K]
- Hot leg vapor temperature [K]
- Intact core fraction [%]
- Fuel Temperature [K]

As indicated in Table 7.2, the number of scenarios differs among the three experiments. This is due to the fact that differences in the pump seal leakage model induces deviations in the dynamics of the system causing different branchings and, thus, a different number of scenarios generated.

A plot of the scenarios for all the three experiments are given in Figs. 7.10, 7.11 and 7.12.

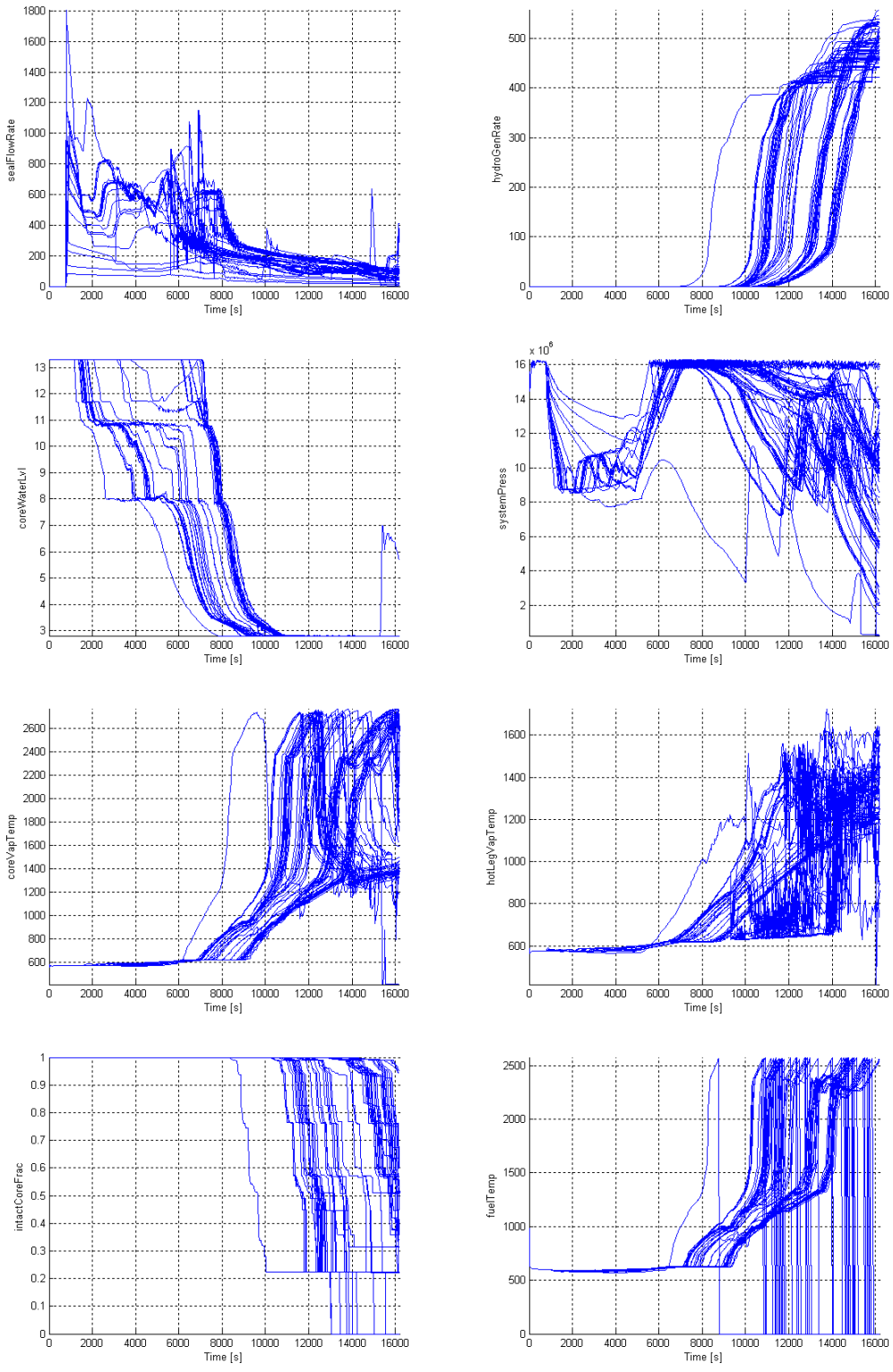

Figure 7.10: Plots of the scenarios for experiment 135.

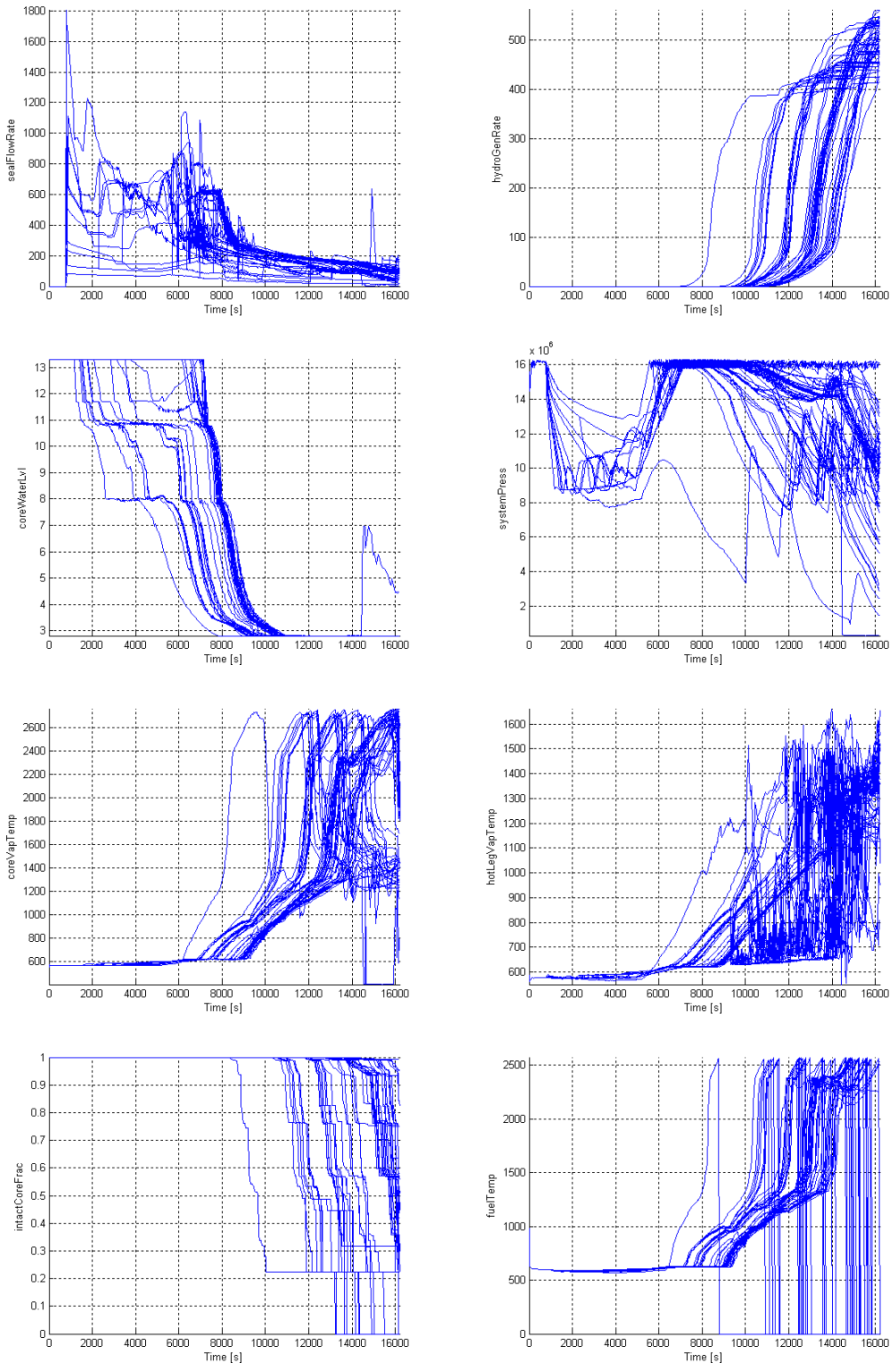

Figure 7.11: Plots of the scenarios for experiment 136.

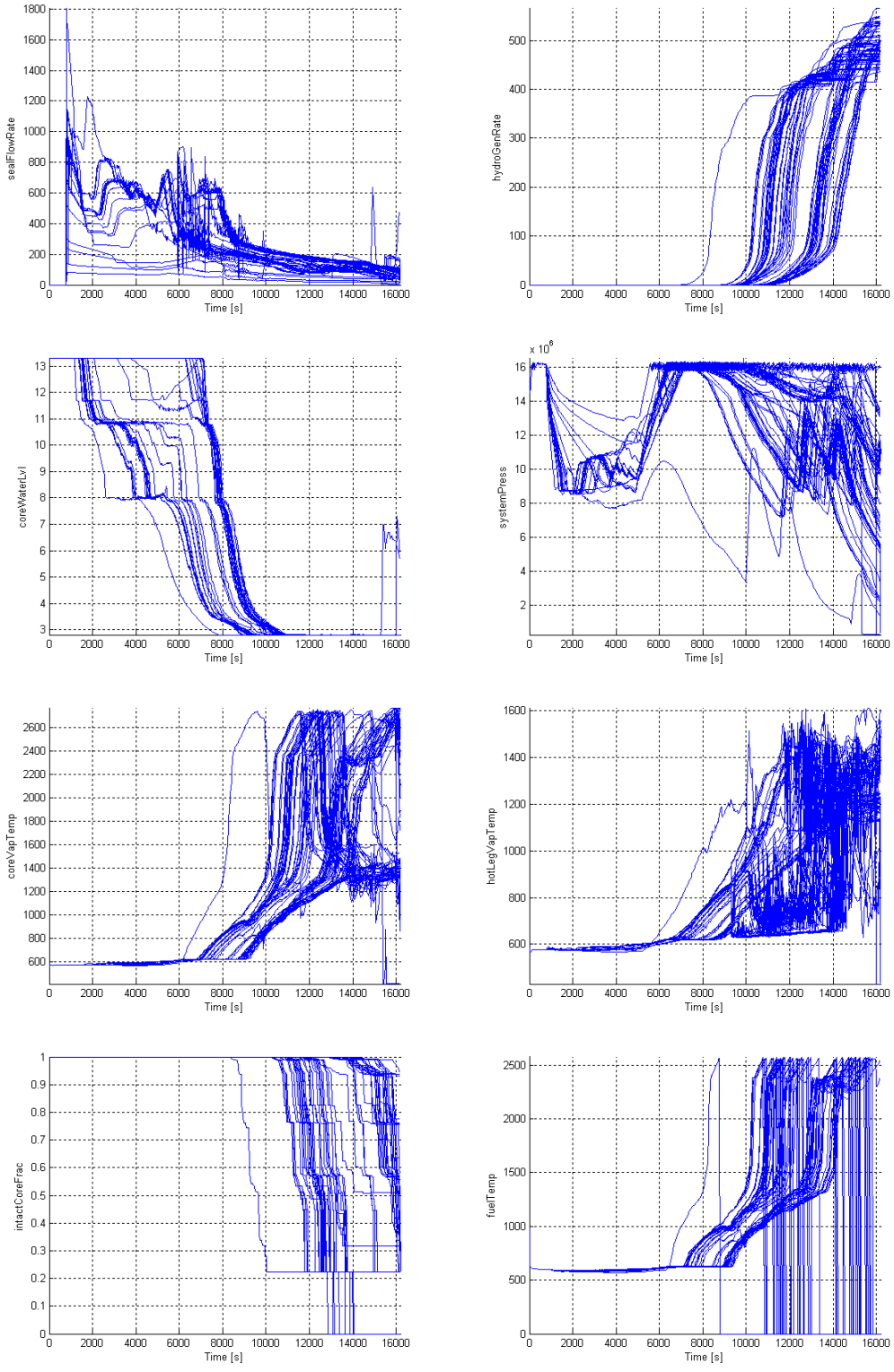

Figure 7.12: Plots of the scenarios for experiment 137.

Table 7.2: Number of scenarios contained in each experiment.

Experiment	Number of scenarios generated by ADAPT
135	104
136	70
137	101

All the 8 variables have been sampled 400 times (thus K in Eq. 4.1 is equal to 400). Hence each scenario is represented a data point having a dimensionality of $400 \cdot 8 = 3200$.

Clustering has been performed using a value of bandwidth $h = 20$ for all the three experiments and the results are shown in Fig s. 7.13, 7.14 and 7.15. Note the the cluster centers shown in Fig s. 7.13, 7.14 and 7.15 are drawn by including a shaded bar around each cluster center. The shaded bar indicates the how the scenarios belonging to the clusters are spread around the cluster center[19]. Thus, these shaded bars are not constant in time but they change depending on the temporal behavior of the scenarios belonging to the clusters.

The following differences are observed among the three data sets:

- Cluster 1: Figures 7.16(a), 7.16(b) and 7.16(c) have this cluster in common which is composed of a single scenario
- Cluster 2: Figures 7.16(a), 7.16(b) and 7.16(c) have this cluster in common but in Fig. 7.16(b) the shaded bar is narrower
- Clusters 3 and 4: Figures 7.16(a), 7.16(b) and 7.16(c) have in common these clusters composed of a single scenario.

[19]These shaded areas are equivalent to the error bars of a data point.

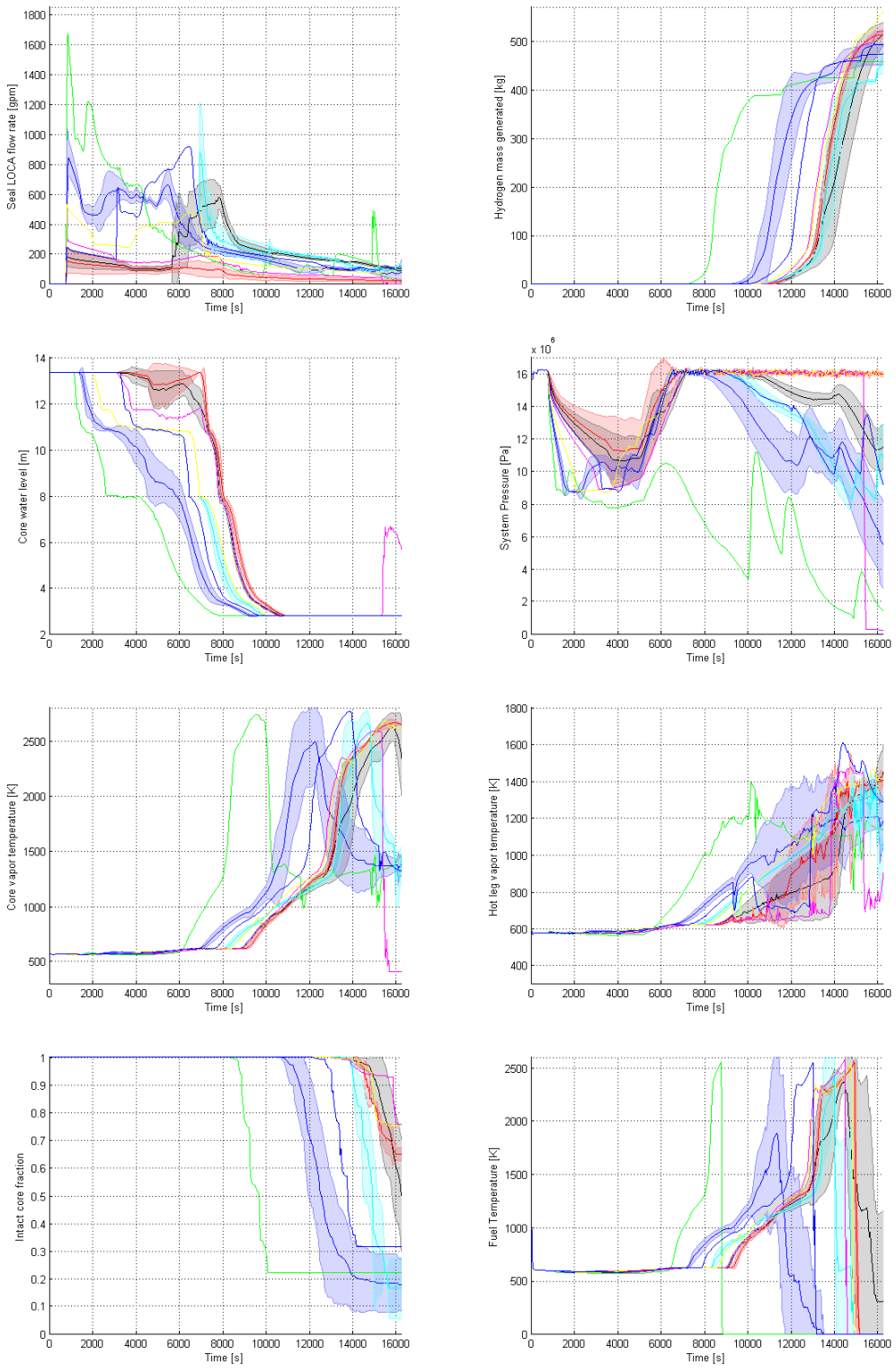

Figure 7.13: Clusters for experiment 135. Lines denote cluster centers, shaded region indicates how the scenarios contained spread around the cluster center. Numbers denote clusters identifiers.

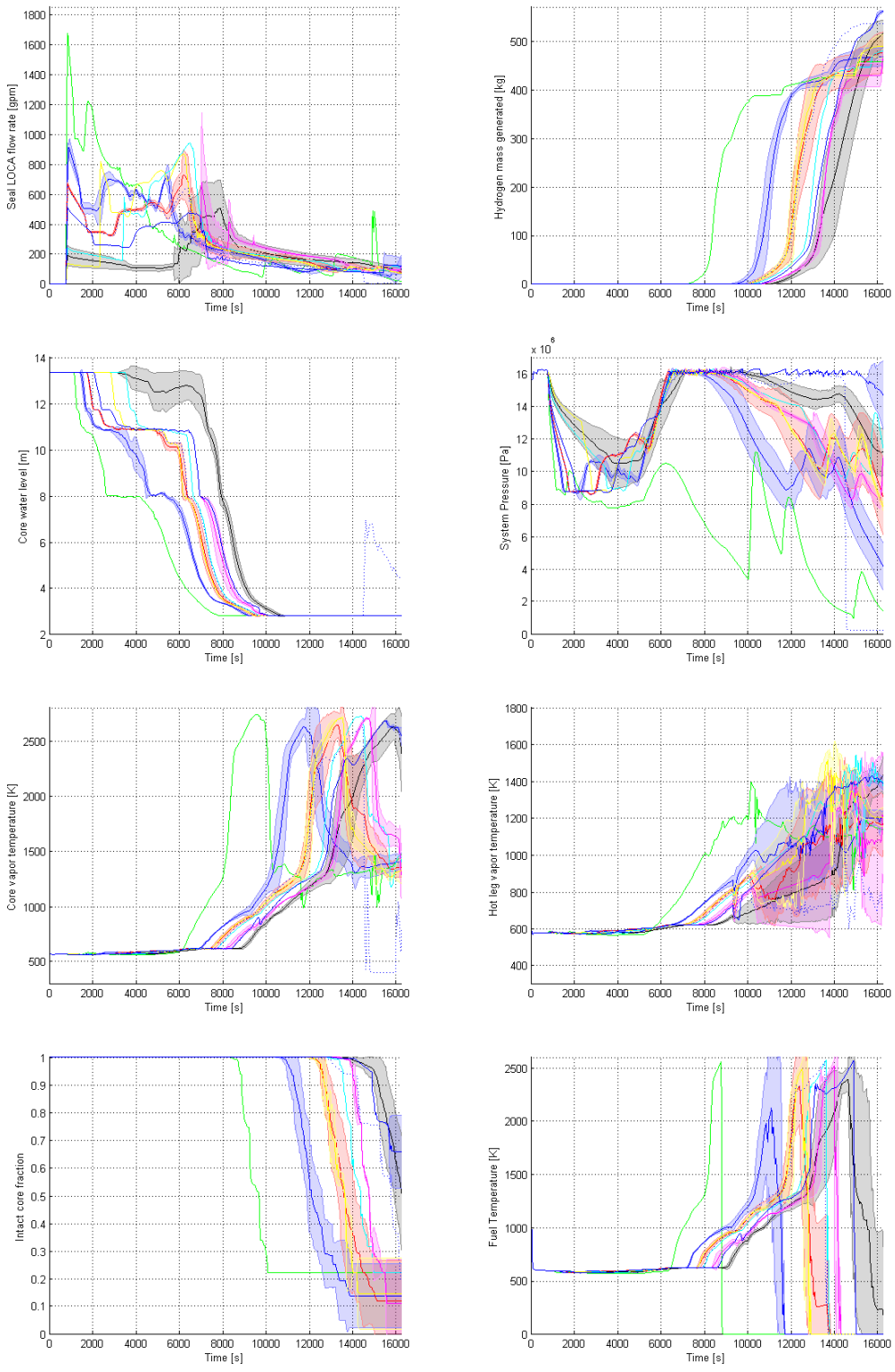

Figure 7.14: Clusters for experiment 136. Lines denote cluster centers, shaded region indicates how the scenarios contained spread around the cluster center. Numbers denote clusters identifiers.

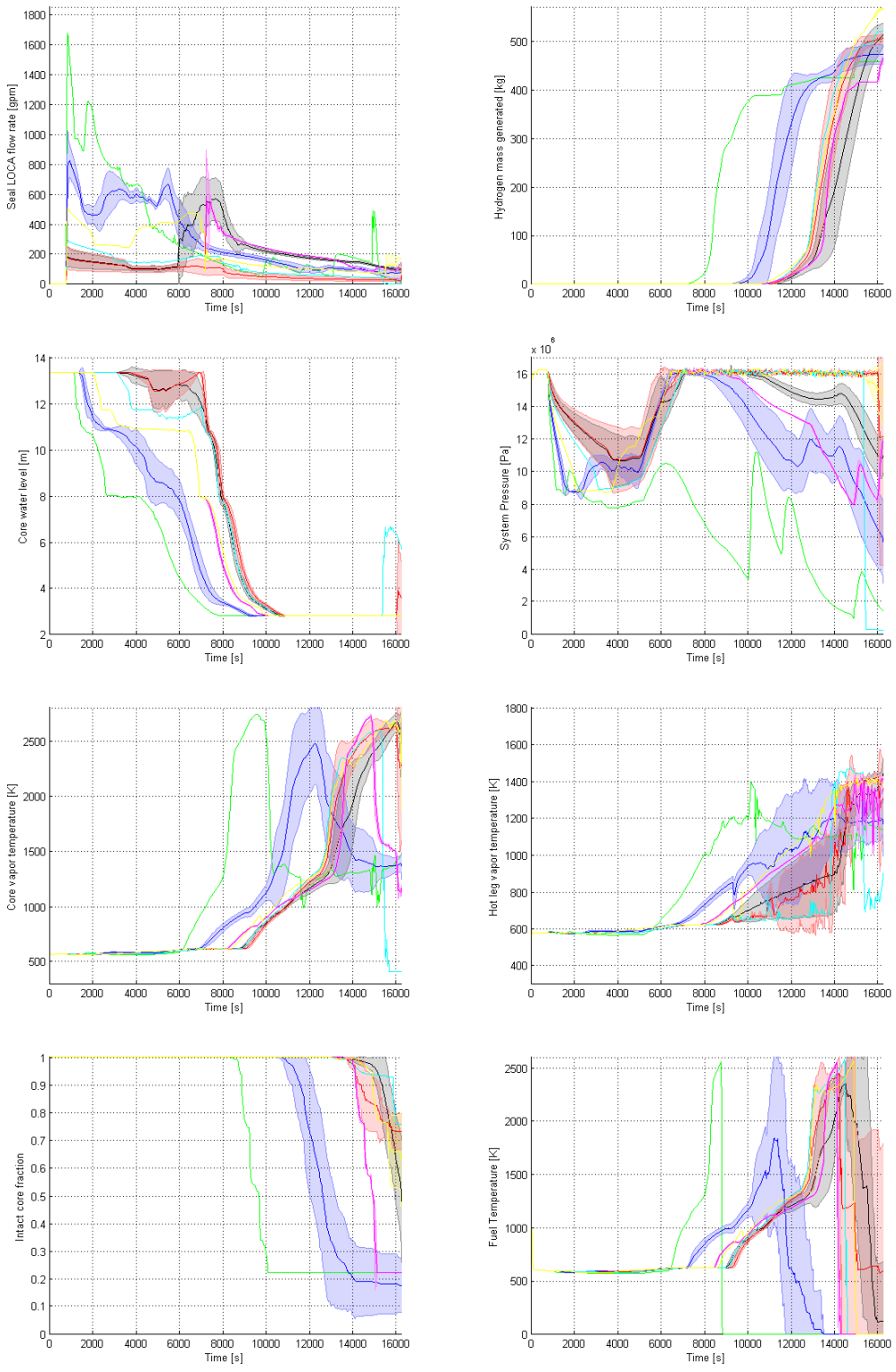

Figure 7.15: Clusters for experiment 137. Lines denote cluster centers, shaded region indicates how the scenarios contained spread around the cluster center. Numbers denote clusters identifiers.

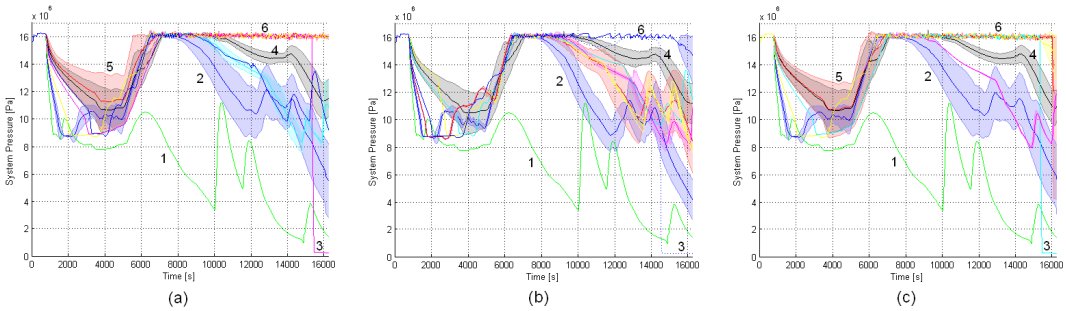

Figure 7.16: Clusters obtained from 3 different models of pump seal leakage. Lines denote cluster centers, shaded region indicates how the scenarios contained spread around the cluster center. Numbers denote clusters identifiers.

- Cluster 5: This cluster is in common in Fig. 7.16(a) and Fig. 7.16(c) while is not present in case (b).

It is also noted that in the region marked as 6, Fig. 7.16(a) and Fig. 7.16(c) are very similar while in Fig. 7.16(b) there is only one cluster scenario which includes scenarios that after 6000 s are characterized by stable system pressure at $16 \cdot 10^6$ Pa. The above results show how the clustering process allows identification of differences arising from different modeling assumptions in a relatively simple manner.

In order to investigate how the number of sampling instants K chosen in Eq. 4.1 affects the scenario-to-cluster memberships, the sampling times were varied and the resulting clusters were compared with the clusters obtained with $K = 400$ sampling instants as described above. As can be seen from Eq. 4.1, by changing K the structure of the data changes the structure of the state space itself. Thus, for different values of K the following steps were carried out:

1. Cluster the data set of Experiment 105 with the new data format

2. Find the value of h which maintains the same number of clusters (i.e., 8)

3. Compare the scenario-to-cluster membership to the original one obtained with $K = 400$

Several values of K have been tested and Table 7.3 summarize the results. From Table 7.3 it is possible to note that for $K > 100$ the scenario-to-cluster membership is preserved. On the other side, for $K < 100$ the data representation is too coarse and the resulting data partition is not preserved, i.e., several scenarios are associated with the wrong clusters as can be seen from Table 7.3.

Table 7.3: Summary of the cluster-to-scenario membership obtained for different values of K

K	Cluster-to-scenario membership	% of scenarios in wrong cluster
400	Preserved	0
200	Preserved	0
150	Preserved	0
100	Preserved	0
75	Not Preserved	3
50	Not Preserved	8
20	Not Preserved	12
10	Not Preserved	21

7.4 Zion Plant Analysis: Station Blackout (2)

The purpose of this section is to show another application of the clustering algorithm developed for analysis of data sets generated by DET. Due to the fact that the simulation mission time is fixed by the user, some scenarios might not reach the a Top Event (e.g., core damage) only because the end of the simulation was reached before they were able to reach that Top Event. Thus, as part of the scenario analysis,

it could be fruitful to identify such, scenarios based on the similarities with other scenarios that reach that Top Event before the end of the simulation.

In order to show how to perform this kind of analysis for a large data set, a data set similar to the one used in Section 7.2 was considered. The scenarios under consideration still originate from a SBO of the Zion plant but consist of 2225 scenarios with a total of 844 GB of data. Based on an engineering judgement of their interdependence, each scenario was represented using 22 state variables (e.g., pressure P and temperature T of specific nodes in the plant simulator) as listed in Table 7.4 as opposed to the 8 used to represent SBO scenarios described in Section 7.2, with subsequent reduction of the full dataset from 844 GB to 400 MB of data.

Table 7.4: State variables chosen for the Zion dataset (2).

id	Name	id	Name
1	Pressurizer P [Pa]	12	Fuel cladding T in cell 519 [K]
2	Pressurizer vapor T [K]	13	Pressurizer surge line structure T [K]
3	Core volume 342 vapor T [K]	14	Steam generator tube structure T [K]
4	Core volume 352 vapor T [K]	15	Hot leg structure T [K]
5	Core volume 362 vapor T [K]	16	Containment P [Pa]
6	Core volume 347 vapor T [K]	17	Containment air T [K]
7	Core volume 382 vapor T [K]	18	Intact core fraction
8	Fuel cladding T in cell 105 [K]	19	Steam generator P [Pa]
9	Fuel cladding T in cell 207	20	Steam generator vapor T [K]
10	Fuel cladding T in cell 310 [K]	21	Total hydrogen production [K]
11	Fuel cladding T in cell 413 [K]	22	Core water level [m]

Clustering was performed using different values of h. Table 7.5 shows the number of clusters obtained for these different values of h. In order to decide the optimal value of h, the CDF of core damage as function of time of both the original and the

Table 7.5: Number of clusters obtained as function of h.

h	Number of clusters
40	1
30	2
25	6
20	19
15	32
0.1	2225

clustered data set have been compared. The clustered dataset obtained for $h = 20$ was adequately representative of the original data set. Figure 7.17 shows the CDF for core damage for both the original dataset and the clustered dataset for $h = 20$. The discrepancy between the two CDFs around 14,000 s - 15,000 s originate from the scenarios that do not lead to core damage (non-failure scenarios) due to their termination based on the mission time chosen (15,000 s) that are clustered with those that do lead to core damage (failure scenarios) within the mission time. Since the clustering was based on the similarity between scenario trajectories in the state-space (i.e. by Eq. 4.1), the discrepancy indicates that the non-failure scenarios would have also led to core damage if the mission time was longer. Table 7.6 shows the fraction of scenarios that lead to core damage for the 19 clusters with $h = 20$.

Figure 7.18 shows the cluster centers obtained for $h = 20$ and plotted for core water level and system pressure.

From Table 7.6 it is noted that the majority of the clusters contain either scenarios that all lead to core damage (failure scenarios) or scenarios that none lead to core damage (non-failure scenarios). However, Clusters 1, 2, 13 and 16 contained both of these types of scenario. As indicated above, these clusters contain both types of

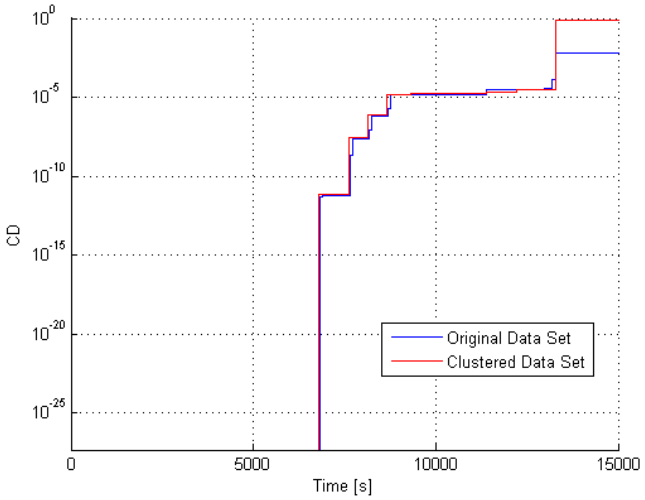

Figure 7.17: Comparison of the CDF of core damage for the original data set and the clustered data.

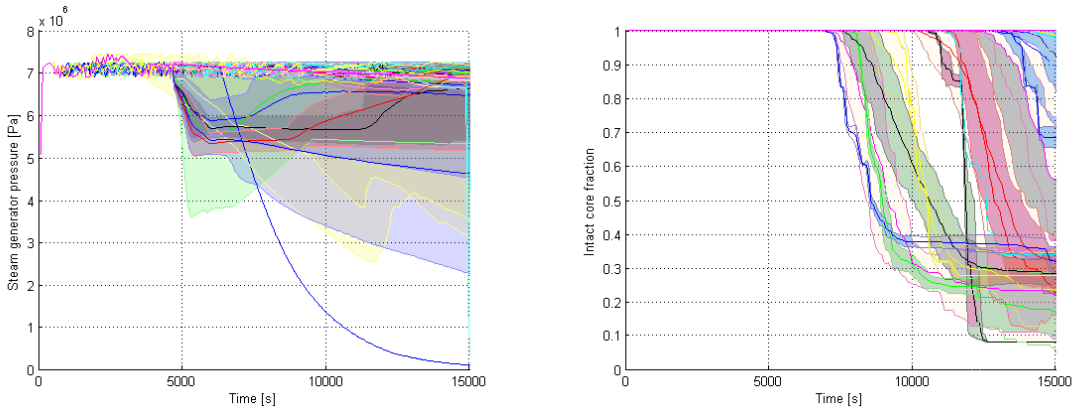

Figure 7.18: Cluster centers obtained for $h = 20$.

Table 7.6: Analysis of the clusters obtained for $h = 20$

Cluster id	Scenarios	Scenarios that lead to CD
1	132	98
2	321	28
3	24	24
4	631	0
5	27	0
6	6	6
7	43	43
8	3	3
9	5	5
10	108	108
11	150	150
12	44	44
13	304	147
14	75	75
15	124	124
16	127	7
17	63	63
18	12	12
19	26	0

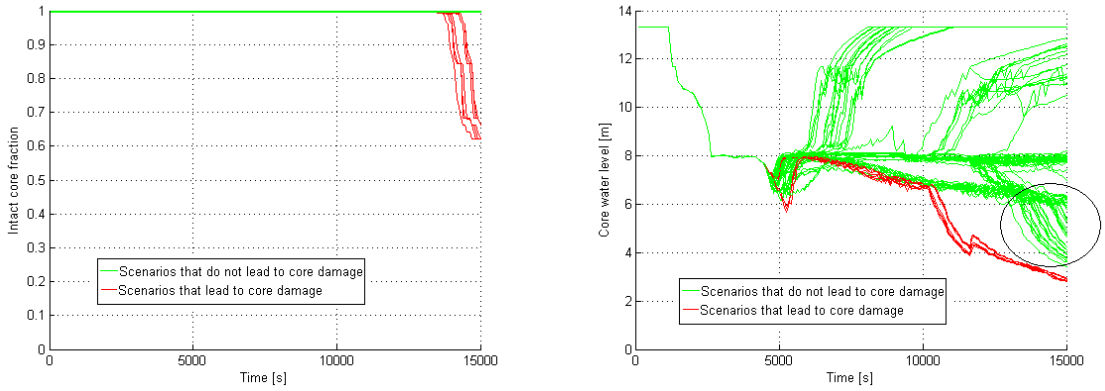

Figure 7.19: Example of scenarios contained in Cluster 16 that lead to core damage (red lines and that do not lead to core damage (green lines)

scenarios since the non-failure scenarios did not lead to core damage only because the simulation ended before they reached this state. Figure 7.19 illustrates this conclusion graphically for scenarios contained in Cluster 16. From Fig. 7.19 it is possible to identify a set of 26 scenarios (highlighted in the circle for the plot of water level) that have similar behavior to scenarios that lead to core damage and which would have led to core damage if the mission time was longer.

7.5 Dimensionality reduction results

The ISOMAP algorithm described in Section 4.5 has been applied to the data set of Experiment 135 described in Section 7.3 in order to evaluate how the dimensionality reduction process performs for complex data sets.

The state space of the system described in Section 7.3 is composed of 8 state variables (i.e., $M = 8$) and, hence, $D = 9$. The overall number of data points distributed in this 9-dimensional space is 100 sampling points/scenario $\cdot$ 104 scenarios

=10,400. By applying the ISOMAP algorithm it has been possible to determine a new data set which has the same number of points but with a reduced number of dimensions from $D = 9$ to $d = 6$.

In order to validate the new data set against the reduced one, the clusters obtained from the two data sets have been compared. While the same number of clusters were obtained from both the original and reduced data sets and each cluster contained the same number of scenarios for the high- and low-dimensional cases, some differences were observed in the cluster centers for Clusters 1, 2 and 3 (highlighted in bold in Table 7.7).

Table 7.7: Comparison of cluster centers obtained from the original and the reduced data sets. Entries denote scenario identifiers.

Cluster #	Original data set ($D = 9$)	Reduced data set ($d = 6$)
1	**62**	**60**
2	**13**	**12**
3	**20**	**21**
4	7	7
5	29	29
6	34	34
7	59	59
8	12	12

When the Euclidean distances between the scenarios in each pair (62,60), (13,12) and (20,21) were determined it was found that these differences were very small. The differences are possibly due to the fact that the dimensionality reduction process described in Section 4.5 may change slightly the geometrical distribution of the original data-set. Figure 7.17 shows the cluster centers and cluster envelopes obtained from both the original and the reduced data sets using two sample state variables (core

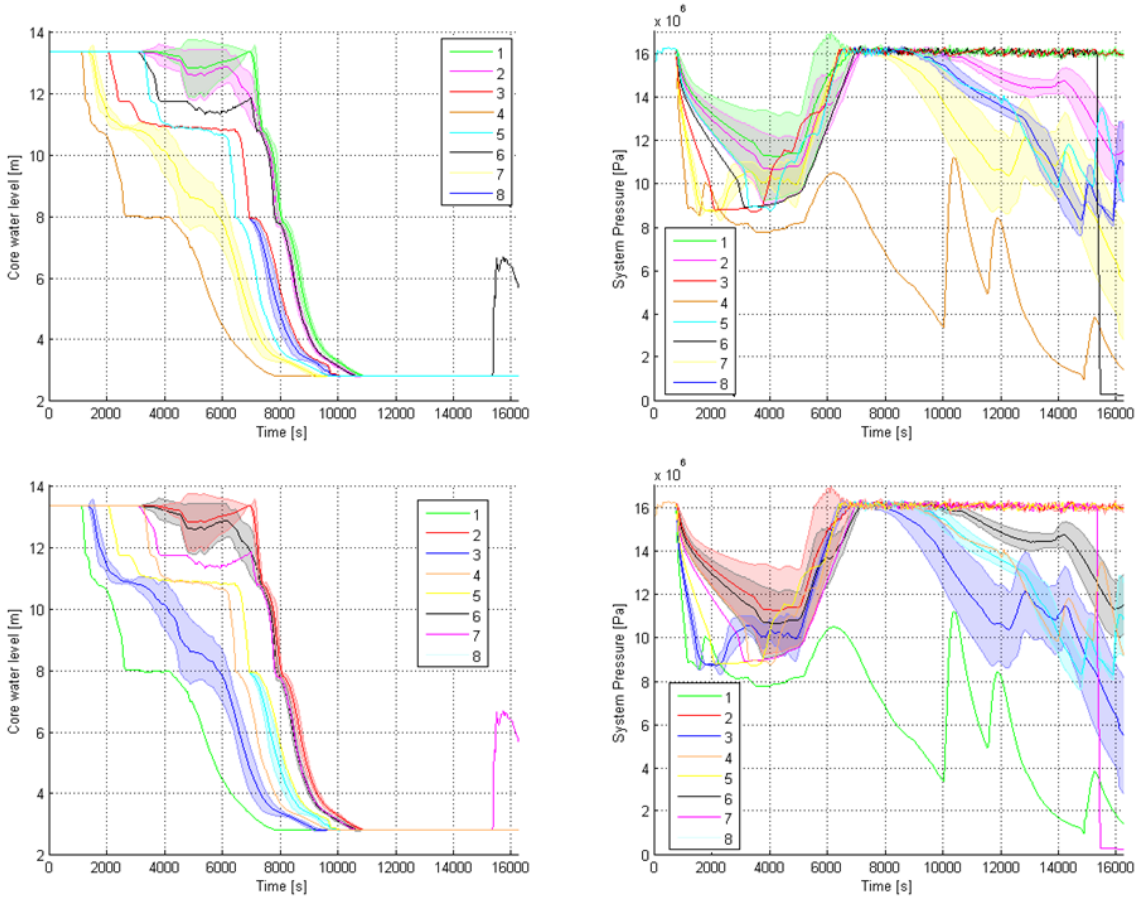

Figure 7.20: Plots of the cluster centers and cluster envelopes derived from all contained within. The top two figures are for cluster obtained from the original data set and the bottom figures are from the reduced data set.

water level and system pressure), Fig. 7.17 shows that not only the cluster centers obtained from the original and the reduced data sets are similar, but their envelopes are as well.

As a last remark, this dimensionality reduction implied an overall reduction in computational time in the clustering process of about 30%.

7.6 Parallel implementation results

In order to evaluate the performance of the parallel implementation of the clustering methodology, two cases were considered:

- Parallel implementation of Mean-Shift using Matlab
- Parallel implementation of Mean-Shift using OpenMP

Two different data sets were used for the evaluation:

- Data set described in Section 7.2 (60 MB size)
- Data set described in Section 7.4 (400 MB size)

The scope of this analysis was to investigate if the parallel implementation can be effective at reducing the computational time of the clustering process with multi-core processors. Clustering was performed on the same machine for 1, 2 and 4 core configurations[20]. Computational times as function of the number of cores used are shown in Fig. 7.21.

As expected, Fig. 7.21 shows that the computational time strongly decreases when a higher number of cores is employed both by using the Matlab and C++ implementation of the Mean-Shift algorithm. Moreover, Fig. 7.21 shows that the C++ version of the algorithm is slightly faster than the Matlab one.

[20]These series of tests were performed on a machine equipped with a quad-core processor.

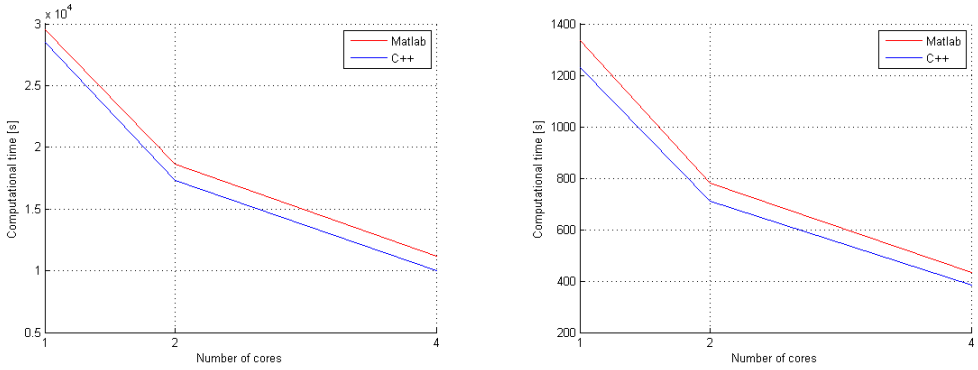

Figure 7.21: Computational time as function of number of cores used for the data set presented in Section 7.2 (left) and Section 7.4 (right).

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

The chapter summarizes the work presented in this dissertation (see Section 8.1) and indicates possible areas for future research developments (see Section 8.2).

8.1 Conclusions

The data sets generated by DETs contain a vast quantity of information. Such large data sets may not be easy to analyze and the retrieval of valuable information may be a difficult task.

This dissertation presents a possible approach for scenario analysis based on the Mean-Shift algorithm. The idea is to group the scenarios into clusters considering not only their final state (i.e., OK state or Fail state) but also how they reach the final state (i.e., the entire time evolution).

Each scenario is characterized by a set of state variables (e.g., temperature, level, pressure) of specific nodes of the plant simulator sampled at specific time instants, normalized and transformed using PCA. Other methodologies that are able to reduce the dimensionality of the data set such as ISOMAP have been also presented. The application of ISOMAP showed promising results in terms of number of state variables that can be used to represent each scenario.

Several clustering methodologies were investigated (e.g., hierarchical, K-Means, Fuzzy C-Means and Mean-Shift) using various data sets with the Mean-Shift proving to be most flexible in terms of ability to identify clusters having arbitrary shape and outliers (i.e., clusters with a very small fraction of scenarios contained in it). Each obtained cluster is characterized by its own representative scenario (cluster center) and by the subset of scenarios that belong to that cluster. Cluster centers give indications regarding the most relevant trends of the overall DET analysis. Several data sets generated by DETs were analyzed using the Mean-Shift methodology.

The work showed the methodology presented is able to simplify the analysis of large sets of transients generated by dynamic PRA methods. Grouping scenarios into clusters can be helpful to identify trends and evaluate their characteristics. Such a grouping is also valuable to identify differences between DET data sets generated for different system configurations.

As a final remark, the application of the methodology presented in this dissertation is not only relevant for the post-processing of data sets generated by dynamic methodologies. In fact, any time a large data set of multi-dimensional functions (e.g., flux profiles) need to be analyzed, the user may find the algorithm developed in this dissertation a valuable tool for the analysis.

8.2 Future Work

As a result of the work presented in this dissertation the following items are recommended as directions for future work:

Algorithm Implementation: The next step is to attach the clustering algorithm to a DET generation code such as ADAPT. The idea is to incorporate data

preprocessing, data clustering and clusters visualization as a tool that would automatically analyze the data sets generated by the DET algorithm.

Data Analysis: Although coming from realistic initiating events and branching conditions for real nuclear power plants, the data sets analyzed in Chapter 7 are still limited in terms of both dimensionality (i.e., number of variables that represent each scenario) and cardinality (i.e., number of scenarios). A good test of the performances of the Mean-Shift clustering algorithm would be to analyze data sets generated for complex systems where a higher number of branching conditions is implemented in the DET. PRA applied to nuclear power plants may include several levels of analysis (i.e., Level 1, 2 and 3) which are usually carried out by different codes/algortihms. In this study, the DET analysis was limited to Level 1 and 2. One possible application of clustering would be to identify the representative scenarios obtained by the clustering algorithm from the Level 1-2 analysis and perform the Level 3 analysis on this reduced set of representative scenarios. In case it is required to perform the Level 3 analysis for all the scenarios generated for the Level 1-2 analysis, it would be useful to perform clustering on the complete data sets which include information about both plant state and off-site consequences.

Data Pre-Processing: As shown in Chapters 4 and 6, the purpose of the data pre-processing is dual:

- Transform the data in suitable form such that it is possible to measure a distance between scenarios
- Reduce the number of variables that represent each scenario

While both of these objectives have been investigated in Chapter 4, it is important to investigate the latter further with the use of non-linear dimensionality reduction algorithms. In particular, it would be useful to apply the algorithms presented in Chapter 6 for larger data sets in order to identify the correlations between variables.

APPENDIX A

MEAN-SHIFT ALGORITHM (MATLAB)

The command

```
[clustCentNorm,data2cluster,cluster2data,numberOfClusters,probabilities]
   = MS2Expr(signals,prob,BW,NScen,NDim);
```

performs the clustering of the matrix `signals`. The matrix `signals` contains the full data set that need to be evaluated and its dimensions are[21] $I \times \delta$. Other input parameters are:

- `prob`: array $1 \times I$ which contains the probability for all I scenarios
- `BW`: bandwidth h
- `NScen`: number of scenarios I
- `NDim`: dimensionality δ

Output parameters are:

- `numberOfClusters`: number of clusters γ obtained
- `clustCentNorm`: cluster centers, i.e. matrix having dimensions $\gamma \times \delta$

[21] I is equal to the number of scenarios while δ corresponds to the dimensionality of each scenario, see the data representation format described in Section 4.1.

- `data2cluster`: array having dimension $1 \times I$ where each element $data2cluster(i)$ $(i = 1, \ldots, I)$ contains the scenario-to-cluster membership for scenario i

- `cluster2data`: cell having dimension $1 \times \gamma$ where each element $cluster2data(j)$ $(j = 1, \ldots, \gamma)$ contains the scenarios included in cluster j

- `probabilities`: array having dimension $1 \times \gamma$ where each element $probabilities(j)$ $(j = 1, \ldots, \gamma)$ contains the the probability of cluster j

The Matlab function `MS2Expr` that performs the clustering is the following:

```
function [clustCentNew,data2cluster,cluster2data,numberOfClusters,
    probabilities] = MS2Expr(data, prob,bandWidth, NScen, NDim)
%            Gauss kernel employed
%            data2cluster employed
%            cluster2data employed

% ---INPUT---
% data            - data matrix (N Scen, Dimensions)
% bandWidth       - bandwidth parameter (scalar)
% NScen           - Number of scenarios i.e. number of points
% NDim            - Dimensions of the vector that describe each scenario
% ---OUTPUT---
% clustCent       - is locations of cluster centers (numDim x numClust)
% data2cluster    - for every data point which cluster it belongs to
    (numPts)
% cluster2data    - for every cluster which points are in it (numClust)
% numberOfClusters - number of clusters generated

stopThresh = bandWidth/1000; % Threshold for convergence

numberOfClusters = 0;
data2cluster = zeros(NScen,1);
clusterCent = zeros(1,NDim);


for i=1:NScen % Run MS for all the points...

    probe = data(i,:);

    PosOld=probe;
    MX = stopThresh*1.1;
```

```
    while (MX > stopThresh) % MS core: Determine m(x) until it fall below
        the threshold

        PosNew = zeros(1,NDim);

        den = 0;

        for j=1:NScen % find all the point within the sphere with
            radius=bandwith/2

            pointIn=data(j,:);

            modX = norm(PosOld-pointIn);

            if(modX < bandWidth/2)

                PosNew= PosNew + pointIn* exp(-(modX*modX)/bandWidth^2);
                den= den + exp(-(modX*modX)/bandWidth^2);

%                 den= den + 1;
%                 PosNew= PosNew + pointIn;
            end
        end

        if(den==0) % no points within BW:add new cluster to the list of
            cluster centers
            clusterCent(numberOfClusters+1,:)=PosOld;
            numberOfClusters = numberOfClusters + 1;
            MX=0;
            data2cluster(i)=numberOfClusters;
        else
            PosNew= PosNew/den;
            MX=norm(PosOld-PosNew);
            PosOld = PosNew;
        end

    end

    count=0;
    for m=1:numberOfClusters
        if (norm(clusterCent(m,:)-PosOld)<(bandWidth/3))
            clusterCent(m,:)=0.5*(clusterCent(m,:)+PosOld);
            data2cluster(i,1)=m;
            count=count+1;
```

```
            break
        end
    end

    if(count==0) % add new cluster to the list of cluster centers
        clusterCent(numberOfClusters+1,1:NDim)=PosOld;
        numberOfClusters = numberOfClusters + 1;
        data2cluster(i,1)=numberOfClusters;
    end

end

cluster2data = cell(numberOfClusters,1);

for x=1:numberOfClusters
    cluster2data{x}=[];
end

for y=1:NScen
    temp= cluster2data{data2cluster(y)};
    l=length(temp);
    temp(l+1)=y;
    cluster2data{data2cluster(y)}=temp;
end

probabilities = zeros(numberOfClusters,1);

for i=1:NScen
    probabilities(data2cluster(i))=probabilities(data2cluster(i))+prob(i);
end

clustCentNew = zeros(numberOfClusters,NDim);

for i=1:numberOfClusters
    a=cluster2data{i};
    for j=1:length(a)
        clustCentNew(i,:)=clustCentNew(i,:)+ data(a(j),:);
    end
    clustCentNew(i,:)=clustCentNew(i,:)/length(a);
end

end
```

APPENDIX B

MEAN-SHIFT ALGORITHM (C++)

This appendix contains the C++ code that performs the clustering using Mean-Shift optimized for parallel computing suing OpenMP directives:

```
/*
 *  MeanShift.cpp
 *  Created on: Aug 9, 2010
 *      Author: Diego
 */

#include <iostream>
#include <fstream>
#include <string>  // in order to use string type
#include "cluster.h"  // in order to use the cluster class
#include <math.h>
#include <vector>
#include <stdlib.h>
#include <cmath>

using namespace std;

void MeanShiftOperator(double *NewPosition, double *point, double **data,
    double h, int card, int dim);
double LpNorm(double p, double x[], int NDim);
int FindClosestCentroid (vector<cluster> clusterSet, double NewCentroid[],
    int p, int dim);

int main (){

    // Variable definitions
    int cardinality = 2;      // Number of scenarios
    int dimensionality = 2;   // Number of dimensions for each scenario
```

```
    // Access data and store it in a 2D array //
    double **data = new double*[cardinality];
    double **centroid = new double*[cardinality];
    double *pointIn = new double[dimensionality];
    double datapoint;
    double BW = 4;
    int p=2;

    for(int j = 0; j < cardinality; j++) {
        data[j] = new double[dimensionality];
        centroid[j] = new double[dimensionality];
    }

    ifstream fi;
    fi.open("data.txt");

    if (fi.fail()) {
        fprintf(stderr, "cannot open file data.txt\n");
        exit(1);
    }

    for(int i = 0; i < dimensionality; i++)
        for(int j = 0; j < cardinality; j++) {
            fi>> data[i][j];
        }
    // End data input session

    // Perform clustering //

    // Initialize the set of clusters (ClusterSet.size() gives size of
        vector)
    vector<cluster> ClusterSet;

#pragma omp parallel for
    for(int i = 0; i < cardinality; i++) //Perform MSM for each data point
    {
        // perform MeanShift for point i and get the centroid for each point

        MeanShiftOperator(centroid[i], data[i], data, BW, cardinality,
            dimensionality);
    }

    for(int i = 0; i < cardinality; i++) //Perform MSM for each data point
    {
```

```
        // update cluster centroid

        int check = FindClosestCentroid (ClusterSet, centroid[i], p,
            dimensionality);

        if(check==-1)
        {
            cluster temp(dimensionality, centroid[i], i);
            ClusterSet.push_back(temp);
        }
        else
        {
            ClusterSet[check].addPoint(centroid[i], i, dimensionality);
        }
    }
    // End clustering //

    return 0; // part of main
}

void MeanShiftOperator(double *NewPosition, double *point, double **data,
    double h, int card, int dim)
{

    double p=2;               // Norm type
    double epsilon = h*0.01; // Convergence parameter
    double den=0;
    double modX=0;

    double m_x=0; // initialize m_x: new position - old position
    double OldPosition [dim];

    for (int i=0; i<dim; i++)
            OldPosition[i]=point[i];

    double diff[dim];

        for (int j=0; j<dim; ++j)
            NewPosition[j] = 0.0;

    do
    {
            for (int i=0; i<card; i++) // find all the point within the
                sphere with radius=bandwith/2
```

```
            {
                double *pointIn=data[i];

                for (int j=0; j<dim; j++)
                    diff[j] = OldPosition[j]-pointIn[j];

                modX = LpNorm(p,diff,dim);

                if (modX < h/2)
                {
                    for (int j=0; j<dim; j++)
                        NewPosition[j] += pointIn[j] *
                            exp(-(modX*modX)/(h*h));

                    den = den + exp(-(modX*modX)/(h*h));

                    den= den + 1;

                    for (int j=0; j<dim; j++)
                        NewPosition[j] /= den;
                }
            }

            for (int j=0; j<dim; j++)
                diff[j]=OldPosition[j]-point[j];

            m_x = LpNorm(p,diff,dim);

            for (int j=0; j<=dim; j++)
                OldPosition[j] = NewPosition[j];

    } while (m_x > epsilon);


}


double LpNorm(double p, double x[], int NDim)
{
    // Determine the p-norm of an NDim-dimensional vector x
    double norm=0;
    double temp=0;

    if (p==0)   // L infinite
    {
```

```
        for (int i=0; i<NDim; i++)
        {
            temp = abs(x[i]);

            if (temp>norm)
                norm=temp;
        }
    }
    else       // Lp
    {
        for (int i=0; i<NDim; i++)
        {
            norm += pow(abs(x[i]),p);
        }
        norm = (pow(norm,1/p));
    }

    return (norm);
}

int FindClosestCentroid (vector<cluster> clusterSet, double NewCentroid[],
    int p, int dim)
{
    // Find the closest centroid to NewCentroid and return the position of
        that point
    int answer = -1;
    double modX;
    double distanceFromMinimum = 999;
    double diff[dim];

    for (int i=0; i<clusterSet.size(); i++)
    {
        for (int j=0; j<dim; j++)
        {
            diff[j] = NewCentroid[j]-clusterSet.at(i).getCentroid()[j];
        }
        modX = LpNorm(p,diff,dim);

        if (modX<distanceFromMinimum)
        {
            distanceFromMinimum = modX;
            answer = i;
        }
    }
```

```
    return (answer);
}
```

```
#include <vector>
using namespace std;

class cluster{
private:
    int dimensionality;
    int cardinality;
    double *centroid;
    vector <int> datapointsID;

public:
    cluster()
    {
        centroid = NULL;
    }

    cluster(const cluster &in);

    cluster(int dimensions, double center[], int pointID);

    ~cluster()
    {
        if (!centroid) delete [] centroid;
    }

    int getDimensionality ();
    int getCardinality ();
//    void setNew (int dimensions, double center[], int pointID);
    void addPoint (double NewCentroid[], int pointID, int dimensions);
    double* getCentroid ();
};


cluster::cluster(const cluster &in) : datapointsID(in.datapointsID)
{

    dimensionality = in.dimensionality;
    centroid = new double[dimensionality];

    for (int i=0; i<dimensionality; i++)
        centroid[i] = in.centroid[i];
}
```

```
cluster::cluster(int dimensions, double center[], int pointID)
{
    // this method add a new cluster
    dimensionality = dimensions;
    cardinality = 1;

    centroid = new double[dimensionality];

    for (int i=0; i<dimensions; i++)
        centroid[i] = center[i];

    datapointsID.push_back(pointID);
}

void cluster::addPoint (double NewCentroid[], int pointID, int dimensions)
{
    // this method add a new point to an existing cluster and update the
        cluster center

    for (int i=0; i<dimensions; i++)
    {
        centroid[i] =
            (centroid[i]*cardinality+NewCentroid[i])/(cardinality+1);
    }

    //dimensionality does not change;
    cardinality++;

    datapointsID.push_back(pointID);
}

double* cluster::getCentroid()
{
    return centroid;
}

int cluster::getCardinality ()
{
    return cardinality;
}

int cluster::getDimensionality ()
{
```

```
    return dimensionality;
}
```

BIBLIOGRAPHY

[1] A. Hakobyan, T. Aldemir, R. Denning, S. Dunagan, D. Kunsman, B. Rutt, and U. Catalyurek, ''Dynamic generation of accident progression event trees,'' *Nuclear Engineering and Design*, vol. 238, no. 12, pp. 3457 -- 3467, 2008.

[2] C. Acosta and N. Siu, ''Dynamic event trees in accident sequence analysis: application to steam generator tube rupture,'' *Reliability Engineering and System Safety*, vol. 41, pp. 135--154, 1993.

[3] T. Aldemir, M. Stovsky, J. Kirschenbaum, D. Mandelli, P. Bucci, L. Mangan, D. Miller, X. Sun, E. Ekici, S. Guarro, M. Yau, B. Johnson, C. Elks, and S. Arndt, *NUREG/CR-6942: Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments*. Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, Washington, DC, 2007.

[4] RELAP5-3D Code Development Team, *RELAP5-3D Code Manual*. 2005.

[5] R. O. Gauntt, *MELCOR Computer Code Manual, Version 1.8.5, Vol. 2, Rev. 2*. Sandia National Laboratories, NUREG/CR-6119.

[6] D. I. Chanin, H. N. Jow, and J. A. Rollstin, *MELCOR Accident Consequence Code System(MACCS)*. Sandia National Laboratories, NUREG/CR-4691, Vols. 1-3, SAND86-1562, 1990.

[7] N. Siu, ''Risk assessment for dynamic systems: an overview,'' *Reliability Engineering and System Safety*, vol. 43, no. 1, pp. 43--73, 1994.

[8] T. Aldemir, D. Miller, M. Stovsky, J. Kirschenbaum, P. Bucci, A. Fentiman, L. Mangan, and S. Arndt, *NUREG/CR 6901: Current state of reliability modeling methodologies for digital systems and their acceptance criteria for nuclear power plant assessments*. Division of Fuel, Engineering and Radiological Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, Washington, DC, 2006.

[9] W. L. Oberkampf, J. C. Helton, C. A. Joslyn, S. F. Wojtkiewicz, and S. Ferson, ''Challenge problems: uncertainty in system response given uncertain parameters,'' *Reliability Engineering and System Safety*, vol. 85, no. 1-3, pp. 11 -- 19, 2004.

[10] A. Amendola and G. Reina, ''Dylam-1, a software package for event sequence and consequence spectrum methodology,'' in *EUR-924, CEC-JRC. ISPRA: Commission of the European Communities*, 1984.

[11] K. S. Hsueh and A. Mosleh, ''The development and application of the accident dynamic simulator for dynamic probabilistic risk assessment of nuclear power plants,'' *Reliability Engineering and System Safety*, vol. 52, pp. 297--314, 1996.

[12] B. Rutt, U. Catalyurek, A. Hakobyan, K. Metzroth, T. Aldemir, R. Denning, S. Dunagan, and D. Kunsman, ''Distributed dynamic event tree generation for reliability and risk assessment,'' in *Challenges of Large Applications in Distributed Environments*, pp. 61--70, IEEE, 2006.

[13] S. Swaminathan and C. Smidts, ''The mathematical formulation of the event sequence diagram framework,'' *Reliability Engineering and System Safety*, vol. 65, no. 65, pp. 103--118, 1999.

[14] J. L. Peterson, ''Petri nets,'' *Computing Surveys*, vol. 9, no. 3, pp. 223--252, 1977.

[15] S. Guarro, M. Yau, and M. Motamed, *NUREG/CR-6465: Development of Tools for Safety Analysis of Control Software in Advanced Reactors*. Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, Washington, DC, 1996.

[16] S. Marchand, B. Tombuyses, and P. Labeau, ''DDET and monte carlo simulation to solve some dynamic reliability problems,'' in *in Proceeding on Probabilistic Safety Assement and Management (PSAM4)*, no. 4, pp. 2055--2060, 1998.

[17] T. Aldemir, ''Utilization of the cell-to-cell mapping technique to construct markov failure models for process control systems,'' in *Proceedings of Probabilistic Safety Assessment and Management: PSAM1*, pp. 1431--1436, Elsevier, New York, 1991.

[18] E. Hofer, M. Kloos, B. Krzykacz-Hausmann, J. Peschke, and M. Woltereck, ''An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties,'' *Reliability Engineering and System Safety*, vol. 77, pp. 229--238, 2002.

[19] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[20] E. Zio, ''Reliability engineering: Old problems and new challenges,'' *Reliability Engineering and System Safety*, vol. 94, no. 2, pp. 125--141, 2009.

[21] A. K. Jain, K. Dubes, and C. Richard, *Algorithms for clustering data*. Upper Saddle River, NJ (USA): Prentice-Hall, Inc., 1988.

[22] D. Mandelli, T. Aldemir, A. Yilmaz, K. Metzroth, and R. Denning, ''Scenario aggregation in dynamic probabilistic risk assessment,'' *Draft for Reliability Engineering and System Safety*, 2010.

[23] U.S.NRC, *NUREG 1150 - Severe accident risks: an assessment for five U.S. nuclear power plants*. Division of Systems Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, Washington, DC, 1990.

[24] E. Zio and P. Baraldi, ''Identification of nuclear transients via optimized fuzzy clustering,'' *Annals of Nuclear Energy*, vol. 32, no. 10, pp. 1068 -- 1080, 2005.

[25] A. K. Jain, M. N. Murty, and P. J. Flynn, ''Data clustering: A review,'' *ACM Comput. Surv.*, no. 31, pp. 264--323, 1999.

[26] J. B. MacQueen, ''Some methods for classification and analysis of multivariate observations,'' in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (L. M. L. Cam and J. Neyman, eds.), vol. 1, pp. 281--297, University of California Press, 1967.

[27] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[28] Y. Cheng, ''Mean shift, mode seeking, and clustering,'' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790--799, 1995.

[29] H. van Groenewoud, ''A cluster analysis based on graph theory,'' *Plant Ecology*, vol. 29, no. 2, pp. 115--120, 1974.

[30] C. T. Zhan, ''Graph theoretical methods for detecting and describing gestatlt clusters,'' *IEEE Transaction on Computers*, vol. 1, no. 20, pp. 68--86, 1971.

[31] A. K. Sharma, S. Sheikh, I. Pelczer, and G. C. Levy, ''Classification and clustering using neural networks,'' *Journal of Chemical Information and Computer Sciences*, pp. 1130--1139, 1994.

[32] J. Reifman, ''Survey of artificial intelligence methods for detection and identification of component faults in nuclear power plants,'' in *Nuclear Technology*, pp. 76--97, 1997.

[33] E. Zio and F. D. Maio, ''Processing dynamic scenarios from a reliability analysis of a nuclear power plant digital instrumentation and control system,'' *Annals of Nuclear Energy*, vol. 36, pp. 1386--1399, 2009.

[34] D. Mercurio, L. Podofillini, E. Zio, and V. Dang, ''Identification and classification of dynamic event tree scenarios via possibilistic clustering: Application to a steam generator tube rupture event,'' *Accident Analysis and Prevention*, vol. 41, p. 11801191, 2009.

[35] T. Aldemir, ''Computer-assisted markov failure modeling of process control systems,'' *IEEE Transactions on Reliability*, vol. 36, pp. 133--144, 1987.

[36] U. N. R. Commission., *WASH 1400 - Reactor Safety Study - An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants*. Division of Systems Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, Washington, DC, 1975.

[37] X. Rui and Ii, ''Survey of clustering algorithms,'' *IEEE Transactions on Neural Networks*, vol. 16, pp. 645--678, May 2005.

[38] B. Mendelson, *Introduction to Topology*. New York (NY), USA: Dover Publications, 1990.

[39] I. T. Jolliffe, *Principal Component Analysis*. Springer, second ed., October 2002.

[40] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag New York, 2005.

[41] S. T. Roweis and L. K. Saul, ''Nonlinear dimensionality reduction by locally linear embedding,'' *Science*, vol. 290, pp. 2323--2328, 2000.

[42] J. B. Tenenbaum, V. de Silva, and J. C. Langford, ''A global geometric framework for nonlinear dimensionality reduction,'' *Science*, vol. 290, pp. 2319--2323, 2000.

[43] J. Dunn, ''A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters,'' *Journal of Cybernetics*, vol. 3, pp. 32--57, 1973.

[44] K. Fukunaga and L. Hostetler, ''The estimation of the gradient of a density function, with applications in pattern recognition,'' *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32--40, 1975.

[45] H. P. Kriegel, P. Kröger, and A. Zimek, ''Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering,'' *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, pp. 1--58, 2009.

[46] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, ''Fast algorithms for projected clustering,'' in *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 61--72, ACM, 1999.

[47] W. Kyoung-Gu, L. Jeong-Hoon, K. Myoung-Ho, and L. Yoon-Joon, ''FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting,'' *Information and Software Technology*, vol. 46, no. 4, pp. 255 -- 271, 2004.

[48] J. H. Friedman and J. J. Meulman, ''Clustering objects on subsets of attributes,'' *Journal of the Royal Statistic Society: Series B (Statistical Methodology)*, vol. 66, no. 4, pp. 815--849, 2004.

[49] R. Agrawal, J. Gehrke, D. Gunopulos, and Raghavan, ''Automatic subspace clustering of high dimensional data for data mining applications,'' in *Proceedings of ACM SIGMOD*, pp. 94--105, 1998.

[50] C.-H. Cheng, A. W. Fu, and Y. Zhang, ''Entropy-based subspace clustering for mining numerical data,'' in *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York (NY) USA), pp. 84--93, ACM, 1999.

[51] H. Nagesh, S. Goil, and A. Choudhary, ''Adaptive grids for clustering massive data sets,'' in *Proceedings of the SIAM Data Mining Conference*, 2001.

[52] G. Moise, J. Sander, and M. Ester, ''Robust projected clustering,'' *Knowl. Inf. Syst.*, vol. 14, no. 3, pp. 273--298, 2008.

[53] T. Cacoullos, ''Estimation of a multivariate density,'' *Annals of the Institute of Statistical Mathematics*, vol. 18, no. 1, pp. 179--189, 1966.

[54] Y. A. Sheikh, E. Khan, and T. Kanade, ''Mode-seeking by medoidshifts,'' in *Eleventh IEEE International Conference on Computer Vision (ICCV 2007)*, no. 1, October 2007.

[55] B. Tombuyses and T. Aldemir, ''Continuous cell-to-cell mapping,'' *Journal of Sound and Vibration*, vol. 202, no. 3, pp. 395 -- 415, 1997.

[56] P. Bucci, J. Kirschenbaum, L. A. Mangan, T. Aldemir, C. Smith, and T. Wood, ''Construction of event-tree/fault-tree models from a markov approach to dynamic system reliability,'' *Reliability Engineering & System Safety*, vol. 93, no. 11, pp. 1616 -- 1627, 2008.

[57] R. Chandra, *Parallel programming in OpenMP*. Morgan Kaufmann Publishers Inc., 2001.

[58] T. K. Kim, W. S. Yang, C. Grandy, and R. N. Hill, ''Core design studies for a 1000 MWt advanced burner reactor,'' in *Proceedings of PHYSOR 2008, Interlaken (Switzerland)*, September 2008.

[59] R. Winningham, K. Metzroth, T. Aldemir, and R. Denning, ''Passive heat removal system recovery following an aircraft crash using dynamic event tree analysis,'' in *Proceeding of American Nuclear Society (ANS)*, vol. 100, pp. 461--462, 2009.

[60] A. Hakobyan, *Severe Accident Analysis Using Dynamic Accident Progression Event Trees*. PhD thesis, The Ohio State University, 2006.