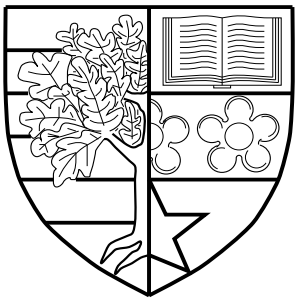
DATA MINING CLUSTERING OF HIGH DIMENSIONAL DATABASES WITH EVOLUTIONARY ALGORITHMS

by

Ioannis Sarafis



Submitted for the Degree of
Doctor of Philosophy
at Heriot-Watt University
on Completion of Research in the
School of Mathematical and Computer Sciences
August 2005

I hereby declare that the work presented in this thesis was carried out by myself at Heriot-Watt University, Edinburgh, except where due acknowledgement is made, and has not been submitted for any other degree.

Ioannis Sarafis (Candidate)

Dr Phil Trinder and Dr Ali Zalzala (Supervisors)

Date

†

Ω πανύμνητε Μῆτερ, η τεκοῦσα τὸν πάντων

Αγίων Αγιώτατον Λόγον·

δεξαμένη τήν νῦν προσφοράν, απὸ πάσης ρῦσαι

συμφορᾶς απαντας· καὶ τῆς μελλούσης λύτρωσαι

κολάσεως, τοὺς συμβοῶντας·

†

"...Εσθιόντων δὲ αυτῶν λαβὼν ο Ιησοῦς τὸν αρτον καὶ

ευλογήσας εκλασε καὶ εδίδου τοῖς μαθηταῖς καὶ ειπε· Λάβετε

Φάγετε· τοῦτό εστι τὸ σῶμά μου· καὶ λαβὼν τὸ ποτήριον

καὶ ευχαριστήσας εδωκεν αυτοῖς λέγων· Πίετε εξ αυτοῦ

πάντες· τοῦτο γάρ εστι τὸ αιμά μου τὸ τῆς καινῆς

διαθήκης τὸ περὶ πολλῶν εκχυνόμενον εις αφεσιν αμαρτιῶν..."

—Μόνον αὐτὰ ποὺ δέχονται ἁγιασμό, αὐτὰ μόνον ἁγιάζονται. Ὅπως· Τὸ νερό, δέχεται ἁγιασμό, καὶ γίνεται Ἁγιασμός. Τὰ σύρτα δὲν δέχονται ἁγιασμό.

Ἡ φέτα, μὲ θαῦμα γίνεται ψωμί. Ἡ ἀκαθαρσία, δὲν δέχεται ἁγιασμό. Ἐπομένως, ὁ διάβολος, ὁ ἀντίχριστος, ὅταν εἶναι στὴν ταυτότητάς, ἢ στὸ χέρι, ἢ στὸ μετάλλιάς, μὲ τὸ σύμβολό του, δὲν ἁγιάζονται, μὲ τὸ νὰ βάλουμε καὶ ἕνα Σταυρό.

Ἔχουμε τὴν δύναμη τοῦ Τιμίου Σταυροῦ, τοῦ Ἁγίου Συμβόλου, τὴν Θεία χάρη τοῦ Χριστοῦ, μόνον ὅταν ἁρμούμεθα στὸ Ἅγιο Σφράγισμα τοῦ Βαπτίσματος, ποὺ ἀπαρνούμεθα τὸν σατανά, καὶ συντασσώμεθα στὸν Χριστό, καὶ δεχόμεθα τὸ Ἅγιο Σφράγισμα· "Σφραγὶς δωρεᾶς Πνεύματος Ἁγίου".—

Ὁ Χριστὸς νὰ μᾶς δίνῃ καλὴ φώτιση. Ἀμήν.—

Ἅγιον Ὄρος Κουτλουμουσιανὸ Κελλὶ "Παναγούδα" Σαββάτο Ἀ' Νηστειῶν 1987.

—Μὲ πολὺ πόνο καὶ ἀγάπη Χριστοῦ,

Μοναχὸς Παΐσιος

To my family

"my father **Αλέξανδρος**, *my mother* **Κατίνα**,
and my brother **φίλιππος**"

Abstract

Driven by advances in data collection and storage, increasingly large and high dimensional datasets are being stored. Without special tools, human analysts can no longer make sense of such enormous volumes of data. Hence, intelligent data mining (DM) techniques are being developed to semi-automate the process of mining nuggets of hidden knowledge, and extract them in forms that can be readily utilised in areas such as decision support. Clustering high dimensional data is especially challenging due to the inherent sparsity of the dataspace. Evolutionary algorithms (EAs) are a promising technique for DM clustering as population-based searches have intrinsic search parallelism, their stochastic nature avoids local optima and recovers from poor initialisation.

This thesis investigates the use of evolutionary algorithms to effectively and efficiently mine clusters from massive and high dimensional numerical databases. The fundamental question addressed by this thesis is: *can a stochastic search cluster large high dimensional datasets, and extract knowledge that conforms to the important requirements for DM clustering*? Experimental results on both artificial and real-world datasets lead us to conclude that it can.

The thesis proposes a novel EA methodology for DM clustering with the following three phases. Firstly, a sophisticated quantisation algorithm ($\mathcal{TSQ}$: $\mathcal{T}$wo $\mathcal{S}$tage $\mathcal{Q}$uantisation) imposes a uniform multi-dimensional grid onto the dataspace to reduce the search combinations. $\mathcal{TSQ}$ quantises the dataspace using a novel statistical analysis that reflects the local data distribution. It determines an appropriate grid resolution that enables the discrimination of clusters, while preserving accuracy and acceptable computational cost. Secondly, a novel EA ($\mathcal{NOCEA}$: $\mathcal{N}$on-$\mathcal{O}$verlapping $\mathcal{C}$lustering with $\mathcal{E}$volutionary $\mathcal{A}$lgorithms) discovers high quality clustering rules using several novel semi-stochastic genetic operators, an integer-valued encoding scheme, and a simple data coverage maximisation fitness function. Both $\mathcal{TSQ}$ and $\mathcal{NOCEA}$ rely on a novel statistical analysis ($\mathcal{UDA}$: $\mathcal{U}$niform-region $\mathcal{D}$iscovery $\mathcal{A}$lgorithm) identifying flat density regions ($\mathcal{U}$-regions) in univariate histograms. $\mathcal{U}$-regions detected in orthogonal uni-dimensional projections are "signatures" of clusters being embedded in higher dimensional spaces. Thirdly, a post-processing simplification phase that removes irrelevant dimensions (subspace clustering) and assembles the clusters. The thesis also explores task parallelism for several genetic operations to improve scalability when the data to be mined is large and high dimensional.

$\mathcal{NOCEA}$ is a generic and robust clustering algorithm that meets the key DM clustering criteria. The following properties of $\mathcal{NOCEA}$ are demonstrated on both benchmark artificial datasets, and in a substantial real-world case study clustering the seismic activity associated with the active crustal deformation along the African-Eurasian-Arabian tectonic plate boundary. $\mathcal{NOCEA}$ produces interpretable output in the form of *disjoint* and *axis-aligned hyper-rectangular* clustering rules with *homogeneous* data distribution; the output is minimised for ease of comprehension. $\mathcal{NOCEA}$ has the ability to discover homogeneous clusters of arbitrary density, geometry, and data coverage. $\mathcal{NOCEA}$ effectively treats high dimensional data, e.g. 200 dimensions, and it effectively identifies subspace clusters being embedded in arbitrary subsets of dimensions. $\mathcal{NOCEA}$ has near linear scalability with respect to the database size (number of records), and both data and cluster dimensionality. $\mathcal{NOCEA}$ has substantial potential for task parallelism, e.g. reaching a speed up of 13.8 on 16 processors. $\mathcal{NOCEA}$ produces similar quality results irrespective initialisation and order of input data. $\mathcal{NOCEA}$ is exceptionally resistant to background noise. Finally, $\mathcal{NOCEA}$ has minimal requirements for *a priori* knowledge, and does not presume any canonical distribution of the input data.

Acknowledgements

I will eternally give praise to ***Jesus Christ***, my Lord and Savior, And His Mother, the Most Holy, Pure, Blessed, and Glorious Lady, the ***Theotokos*** and ***Ever−Virgin Mary***, the ***Mother of God***, for always keeping me and my family under Their protection.

I will be eternally grateful for the endless love and affection bestowed upon me from my family, my father $\mathcal{A}\lambda\acute{\varepsilon}\xi\alpha\nu\delta\rho\varsigma$, my mother $\mathcal{K}\alpha\tau\acute{\imath}\nu\alpha$, and my brother $\phi\acute{\imath}\lambda\imath\pi\pi\varsigma$.

I am indebted to Dr *Phil W. Trinder*, my primary supervisor, for the guidance, invaluable encouragement and patience he gave me during the course of my PhD research. I heartily thank Dr Trinder for initially getting the opportunity and the necessary funds of conducting research at Heriot-Watt University, And more importantly for the hand-holding done in the notoriously difficult writing-up stage.

I would like to thank my second supervisor Dr *Ali Zalzala*, for his strong support and genuine enthusiasm for my work throughout the project. I am particularly grateful to Dr Zalzala for acquiring a licence to use the *EOS* software by BT - *British Telecommunication plc* - for my research, And for ensuring funds to sponsor my participation in international conferences that were held in USA.

I would like to thank the computer-support staff - *Ross MacIntyre*, *Iain McCrone*, *Steve Salvini*, and *Donald Pattie* - and the financial officer *Christine McKenzie*, in the Department of Computing at Heriot-Watt University for providing valuable assistance.

I gratefully acknowledge the Heriot-Watt University partial scholarship that made this degree possible, The provision of the *EOS* software by BT's - *British Telecommunication plc* - Intelligent Systems Laboratory, And the donation of the earthquake dataset by ANSS - *American Advanced National Seismic System.*

I would also like to acknowledge the invaluable comments and suggestions on kernel theory by Professor Chris Jones from the Open University.

Many thanks must go to my good friends Giorgos Markoyiannakis and Abyd AlZain for their patience and support whilst waiting for me to complete this thesis.

I heartily thank $\mathcal{I}\varepsilon\rho o\mu\acute{o}\nu\alpha\chi o\varsigma$ $\mathcal{X}\rho\imath\sigma\tau\acute{o}\delta o\upsilon\lambda o\varsigma$ $\mathcal{A}\gamma\imath o\rho\varepsilon\acute{\imath}\tau\eta\varsigma$ ($\mathcal{A}\gamma\gamma\varepsilon\lambda\acute{o}\gamma\lambda o\upsilon$), And the Holy Hesyhasterion "PANAGIA, H FOVERA PROSTASIA", Megali Panagia,Chalkidiki, Greece, for permitting me to reproduce (in page iii) part of their book '$\mathcal{O}$ $\Gamma\acute{\varepsilon}\rho\omega\nu$ $\Pi\alpha\ddot{\imath}\sigma\imath o\varsigma$'.

I would like to thank the members of my PhD viva panel, Dr Nick Taylor and Dr Alex Freitas for the very constructive comments on how to improve my thesis.

Finally I am also grateful to many others, all of whom can not be named.

Contents

List of Tables

List of Figures

Part I

Context

Chapter 1

Introduction

1.1 Ethos

This Thesis explores the use of evolutionary algorithms (EAs) to effectively and efficiently perform data mining (DM) clustering on massive and high dimensional numerical databases.

The advantages of EAs as underlying search mechanisms for multi-dimensional optimisation problems are often cited and recited. The fundamental question addressed by this thesis is: *can a stochastic search cluster high dimensional datasets, and extract knowledge that conforms to the important requirements for DM clustering*? From the experimental results reported in the thesis this seems to be true for a wide variety of artificial and real-world datasets.

Knowledge Discovery in Databases: The importance of collecting data related to business or scientific activities to achieve competitive advantage is widely recognised. Powerful systems for collecting and managing data in large databases are widely used commercially. Driven by these technological advances, increasingly large, complex, and high dimensional datasets are being stored in databases. Human analysts with no special tools can no longer make sense of enormous volumes of data that require processing in order to make informed business decisions. Hence, a new generation of techniques and tools are needed with the ability to intelligently and (semi-)automatically analyse the mountains of data for nuggets

of useful knowledge. These techniques are the subject of *Knowledge Discovery in Databases* (*KDD*) [31, 40, 51]. *Data Mining* (*DM*), the core stage of KDD, refers to the application of intelligent techniques to extract hidden knowledge from data.

Data Mining Clustering: *Clustering* is a descriptive DM task seeking to identify homogeneous groups of data points based on the values of their attributes [30, 37, 58, 62]. Clustering high dimensional datasets is an especially challenging task because of the inherent sparsity of the data space and the fact that different correlations may occur in different subsets of dimensions (subspaces) in different data neighbourhoods. Emerging DM applications place specific requirements on clustering techniques such as effective treatment of high dimensionality, end-user comprehensibility of the results, good scalability with database size and dimensionality, the ability to discover clusters of arbitrary geometry, size and density, detection of features relevant to clustering, noise tolerance, insensitivity to initialisation and order of data input, handling of various data types, and minimal requirements for domain knowledge [40, 51].

Evolutionary Algorithms: *Evolutionary Algorithms* (*EA*s) are optimisation techniques inspired by the abstractions of *Darwinian* evolution [11, 12, 47, 71]. In nature, individuals best suited to competition for scare resources survive. Evolving to adapt to a changing environment is essential for all species. The driving force of evolution is the combination of natural *selection* and *genetics*. Natural selection leads to the survival and reproduction of the *fittest* organisms, while natural genetics are the mechanisms that introduce random variation in the population, e.g. cross breeding and mutation.

EAs are iterative and stochastic processes that utilise the collective learning of a population of individuals. An individual represents a potential solution in some problem space through a suitable coding. Each individual is assigned, by means of a fitness function, a measure of its performance with respect to the target problem. Individuals are selected for reproduction with a probability proportional to

their fitness. New points in the search space are sampled using various genetic operators, such as crossover and mutation. To maintain a fixed-size population a replacement policy selects the best individuals for survival in the next generation. The whole process is repeated until some termination criterion is satisfied. Since highly fit individuals have more chances of surviving and attracting mates their characteristics conveying a high fitness will spread, and eventually dominate successive generations.

EAs are a promising technique for DM clustering as population-based searches have intrinsic search parallelism, their stochastic nature avoids local optima and recovers from poor initialisation. Additionally, EAs do not make presumptions about the problem space, and they have no prerequisites on any type of auxiliary information, except of course the fitness function. From an implementation point of view, EAs are highly parallel procedures and can be easily and conventionally used in parallel systems. For instance, since EAs are made up from several tasks involving a group of individuals rather than the entire population, several processors can work simultaneously on the same task, thereby improving scalability.

Parallel Processing: *Parallel Processing* aims to speed-up the execution time of a program by sharing the work to be done amongst several processors [43]. To achieve a substantial reduction in execution time the sequential program must be decomposed into several threads, independent units of computation, that are simultaneously executed on different processors. Introducing the concept of threads means that mechanisms for generating threads, synchronising threads, communicating data between threads, and terminating threads have to be established. Clearly, these aspects of a parallel program are significantly more complex than those of a sequential program.

Due to their population-based nature, EAs are generally considered as slow compared to more conventional optimisation techniques that operate on a single solution at a time. Hence, to establish the practicality of an EA-based clustering system for high-dimensional datasets, it is vital to introduce parallelism [43, 44].

1.2 Contributions

Thesis Scope: The thesis addresses the challenging area of DM clustering on realistic datasets by combining techniques from three main areas of computer sciences, as depicted in figure 1.1 (shadowed area). In particular, the thesis investigates how to use EAs with parallel computing architectures to effectively and efficiently mine homogeneous clusters from massive high dimensional databases.

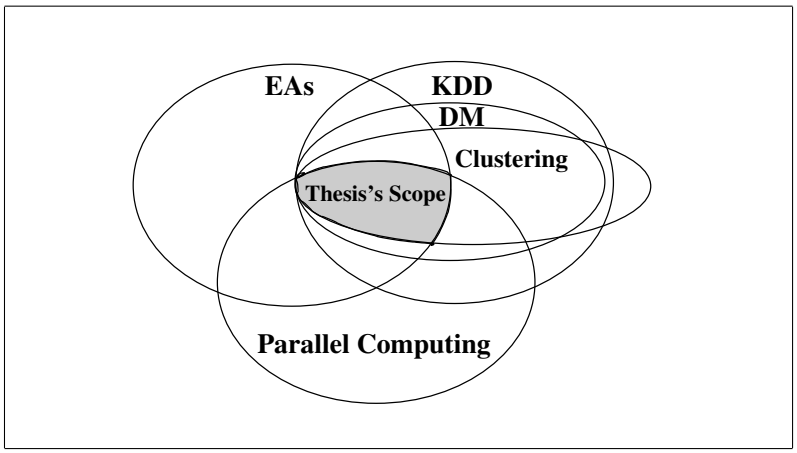


Figure 1.1: Thesis Scope (Shadowed Area)

Core Idea: The thesis proposes a novel framework for DM clustering that utilises the powerful search mechanism of EAs along with task parallelism to mine *disjoint* and *axis-aligned hyper-rectangular* clustering *rules* with *homogeneous* data distribution from *large* and *high* dimensional *numerical* databases.

Contributions: The thesis makes the following contributions in the fields of DM clustering and EAs:

- A novel rule-based clustering EA ($\mathcal{NOCEA}$), described in Chapter 5 and [87, 88, 89], that:
  - Yields interpretable output being minimised for ease of comprehension.
  - Discovers highly-homogeneous clusters of arbitrary shape, density, size, and data coverage. The space enclosed by a homogeneous cluster has a quasi-uniform distribution of data points.
  - Treats high dimensionality effectively, i.e. it can easily discriminate clusters in very sparse high dimensional spaces.

- Performs effective subspace clustering, i.e. automatic detection of clusters being embedded in arbitrary subspaces of high dimensional data.
- Produces similar quality results irrespective of initialisation and order of data input.
- Is exceptionally resistant to the presence of increased background noise.
- Scales linearly with the database size (number of records), data and cluster dimensionality.
- Does not presume any canonical distribution for the input data.
- Has minimal requirements for domain knowledge, e.g. it automatically determines the optimal number of clusters and the subspace where each cluster is embedded, on the fly.
- Operates always on the full dimensional space that guards against artifacts formed by the joint projection of multiple clusters in lower dimensional spaces.
- Introduces a simple non-distance or density based clustering criterion.
- Traverses the search space stochastically avoiding easily local optima.

• A novel two-stage statistical quantisation algorithm ($\mathcal{TSQ}$), described in Chapter 4 and [89], that:

- Automates the construction of a uniform multi-dimensional grid onto the dataspace to reduce the search combinations for $\mathcal{NOCEA}$.
- Determines an appropriate grid resolution that enables the discrimination of clusters, while preserving accuracy and acceptable computational cost.
- Combines standard statistical techniques, e.g. Kernel Density Estimation, with new sophisticated heuristics that reflect the local distribution of data.
- Is unbiased towards coarse or fine grid resolutions; the optimal resolution is dictated by the underlying data distribution.

- A novel histogram analysis algorithm ($\mathcal{UDA}$), described in Chapter 3, identifying flat density regions ($\mathcal{U}$-regions) in univariate histograms. $\mathcal{U}$-regions detected in uni-dimensional orthogonal projections are essential for both data *quantisation* (Chapter 4) and *clustering* (Chapter 5), as they are the "signatures" of clusters existing in higher dimensional spaces. $\mathcal{UDA}$ consists of three elaborate statistical tests that can detect $\mathcal{U}$-regions of arbitrary density even under highly noisy conditions.

- A parallel processing architecture for $\mathcal{NOCEA}$, described in section 5.13 and [89], that:

  - Explores task parallelism for the most expensive genetic operations.
  - Delivers a substantial speed up of 13.8 on 16 processors.

- New scientific knowledge and understanding about the distribution, dynamics, and evolution of seismic activity has been acquired by clustering earthquakes that occurred along the African-Eurasian-Arabian plate boundary in a spatio-temporal-magnitude space, as described in Chapter 7 and [89]. The discovered knowledge was exploited to compile improved hazard maps. Furthermore, it can aid seismologists in gaining a deeper insight into the phenomenon and allow them to improve the reliability of their estimates.

- Contributions, described in Chapter 5, have also been made in the field of evolutionary computation including an elaborate integer-valued representation scheme, novel mutation and recombination operators, as well as advanced task-specific operators, i.e. generalisation and homogeneity. In short, mutation and recombination enable $\mathcal{NOCEA}$ to traverse the enormous data space in a cost-effective fashion. Generalisation delivers end-user comprehensibility and simplification of the clustering results by making rules as generic as possible. It also reduces computation by minimising the length, i.e. number of rules, of individuals. Homogeneity manipulates the candidate rules to ensure that the space enclosed by them has homogeneous data distribution.

1.3 Organisation

The remainder of the thesis is made up of four parts. Initially, there is an introductory part describing the *context* of the thesis and related work. Then, there is a part to present the *system design and implementation* of the clustering, quantisation, and post-processing simplification algorithms. The next part presents a thorough *evaluation* of the proposed methodology both on artificial and real-world datasets. Finally there is *concluding* part that summarises the results reported throughout the thesis.

Part I Context

Chapter 2 introduces the computational background to the thesis - a detailed survey of DM clustering techniques, and a brief discussion regarding the foundations of EAs. The aims of the research in this thesis are also established.

Part II System Design and Implementation

Chapter 3 describes a novel statistical analysis methodology ($\mathcal{U}$*niform-region* $\mathcal{D}$*iscovery* $\mathcal{A}$*lgorithm* or $\mathcal{UDA}$) for identifying cleanly separable $\mathcal{U}$-regions in univariate frequency histograms. A $\mathcal{U}$-region is defined as a set of contiguous bins whose histogram values exhibit only a marginal variation. $\mathcal{U}$-regions detected in univariate projections are signatures of clusters existing in higher dimensional spaces [5, 7, 72, 74]. $\mathcal{UDA}$ combines standard histogram smoothing techniques (i.e. Kernel Density Estimation) with new heuristics that perform a fine localised analysis of the data distribution. $\mathcal{UDA}$ is exceptionally resistant to noise and local data artifacts. The $\mathcal{U}$-regions identified by $\mathcal{UDA}$ are extensively used for both data *quantisation* (Chapter 4) and *clustering* (Chapter 5).

Chapter 4 describes in detail the two-stage statistical quantisation algorithm ($\mathcal{TSQ}$) that automatically imposes a multi-dimensional grid structure onto the data space; the quantised data are subsequently clustered by $\mathcal{NOCEA}$. $\mathcal{TSQ}$

reduces the search combinations for $\mathcal{NOCEA}$, and addresses well the classical trade-off between grid resolution that greatly determines the quality of the clustering results, and computational complexity. $\mathcal{TSQ}$ computes an appropriate uniform bin width for each dimension using both gross and detailed statistical analysis of uni-dimensional density histograms. $\mathcal{TSQ}$ is unbiased towards any resolution; the optimal resolution is dictated by the underlying data distribution.

Chapter 5 - the core part of the thesis, gives a detailed description of the novel evolutionary algorithm ($\mathcal{NOCEA}$) for DM clustering that has been developed and implemented in this thesis. The discussion focuses on the rationale and the design of the novel genetic operators that have been devised to search the enormous data space in a cost-effective fashion and to deliver high quality knowledge to end-users.

The discussion continues with the description of two post-processing simplification algorithms that have been developed to facilitate the interpretation of the discovered knowledge. The first algorithm groups the pieces of knowledge, i.e. clustering rules, that have been previously found by $\mathcal{NOCEA}$ into clusters, while the second algorithm performs subspace clustering, that is, the detection of relevant features to the clustering of data.

Chapter 5 concludes with a detailed description of a parallel processing architecture ($p\mathcal{NOCEA}$) that introduces task parallelism for the most computationally expensive genetic operations of $\mathcal{NOCEA}$. The discussion focuses on three important aspects of parallel computing, namely, architecture, granularity and load balancing. It is shown, in chapter 7, that for large and high dimensional datasets with multitudinous clusters, the parallelisation of all genetic operations despite the increased communication-coordination overhead gives the best speedup, e.g. 13.8 on 16 processors

Part III Evaluation

Chapter 6 reports an experimental evaluation of $\mathcal{NOCEA}$ on artificial datasets. Thorough investigation into $\mathcal{NOCEA}$'s performance on benchmark datasets verifies its exceptional properties for DM clustering as listed in section 1.2.

Chapter 7 presents a thorough evaluation of $\mathcal{NOCEA}$ on a challenging real-world problem; a detailed cluster analysis of earthquakes that occurred along the African-Eurasian-Arabian tectonic plate boundary. The experimental results, as in Chapter 6, suggest that $\mathcal{NOCEA}$ is a generic and robust clustering algorithm that meets the desiderata for realistic DM clustering. Furthermore, new scientific knowledge about the distribution of the seismic activity along that area has been acquired, and reported in interpretable form aiding seismology to better understand the phenomenon. Finally, the Chapter concludes with an extensive efficiency and effectiveness performance evaluation on a combination of massive synthetic and real-world datasets. The scalability results show a nearly linear dependency on the database size, data and cluster dimensionality, as well as substantial potential for high levels of parallelism, reaching a speed up of 13.8 on 16 processors.

Part IV Conclusion

Chapter 8 summarises the original contributions to knowledge that this thesis has made, and concludes that evolutionary algorithms have substantial potential as underlying search mechanisms for real-world data mining clustering applications. Further research directions for improving effectiveness, efficiency and knowledge interpretability are also identified.

1.4 Authorship

Unless otherwise stated, the work reported throughout this Thesis including the following research publications was done *primarily* by the author with limited contributions by his supervisors Dr. P. W. Trinder and Dr. A. M.S. Zalzala.

1. I. A. Sarafis, P. W. Trinder and A.M.S Zalzala: $\mathcal{NOCEA}$: *A Rule-based Evolutionary Algorithm for Efficient and Effective Clustering on Massive High-dimensional Databases.* International Journal of Applied Soft Computing (*ASOC*), Elsevier Science, 2005. (Invited Paper) (To Appear)

2. I. A. Sarafis, P. W. Trinder and A.M.S Zalzala: *Towards Effective Subspace Clustering with an Evolutionary Algorithm.* Proceedings of the *IEEE* Congress on Evolutionary Computation (*CEC03*), pp 797–806, Canberra, Australia, 2003.

3. I. A. Sarafis, P. W. Trinder and A.M.S Zalzala: *Mining Comprehensive Clustering Rules With an Evolutionary Algorithm.* Proceedings of Genetic and Evolutionary Computation Conference (*GECCO03*), pp 2301–2312, Chicago, USA, 2003. (Nominated for Best Paper Award)

4. I. A. Sarafis, A.M.S Zalzala and P. W. Trinder: *A Genetic Rule-based Data Clustering Toolkit.* Proceedings of the *IEEE* Congress on Evolutionary Computation (*CEC02*), pp 1238–1243, Honolulu, USA, 2002.

The following research paper was a joint work between the author, A.M.S Zalzala and A. Alzain.

5. A.M.S. Zalzala, A. AlZain and I. A. Sarafis: *A Data Mining Tool Using An Intelligent Processing System with a Clustering Application.* Proceedings of the 5th International Conference of Adaptive Computing in Design and Manufacturing, (*ACDM02*), Devon, United Kingdom, 2002.

Chapter 2

Background

Capsule

This chapter discusses several approaches towards clustering for data mining (DM) applications. It starts with motivating the use of intelligent techniques to automate the process of knowledge discovery in large databases. Then it presents specific requirements that emerging DM applications place on clustering techniques. The main part of this chapter focuses on surveying the state-of-the-art clustering techniques, and identifying their limitations. The foundations of evolutionary algorithms are also discussed. The chapter concludes with a focused survey of the use of EAs for clustering.

2.1 Introduction

The first stage in the design of a new clustering algorithm is *a*) to understand the important requirements for realistic DM clustering, *b*) to ascertain the advantages and disadvantages of the state-of-the-art clustering algorithms, and *c*) to identify the areas with the greatest potential for addressing the limitations of existing techniques.

After approximately one year surveying the author had adequate experience of the working principles of modern DM clustering techniques as well as the foundations of EAs. The impressions gained from this short exposure to DM research lead the author to believe that, current clustering techniques do not address all

of the important requirements for DM clustering, although considerable work has been done in addressing each requirement separately. Furthermore, after decades of claiming that EAs are powerful optimisation techniques well-suited for problems with large and complex search spaces, no EA-based system has adequately exploited the advantages (section 2.4.11) of EAs to tackle realistic large-scale clustering problems.

This Chapter reviews the literature addressing DM clustering, and justifies the author's impressions discussed in the preceding paragraphs. The remainder of this Chapter is structured as follows. Section 2.2 motivates the use of intelligent techniques for Knowledge Discovery in large and high dimensional Databases (KDD), outlines the steps of KDD process, and finally lists common data mining (DM) functionalities in KDD. Section 2.3 outlines the challenges and important requirements for real-world DM clustering, and provides a focused survey of the state-of-the-art clustering algorithms, including partitioning (section 2.3.4), hierarchical (section 2.3.5), density-based (section 2.3.6), grid-based (section 2.3.7), subspace-clustering (section 2.3.8), and some other approaches (section 2.3.9). Section 2.4 briefly discusses the principles of Evolutionary Algorithms (EAs) (sections 2.4.1-2.4.9), compares EAs with conventional optimisation techniques (section 2.4.10). The Chapter concludes with section 2.5 that provides a focused survey of the use of EAs for clustering.

2.2 Knowledge Discovery in Databases

Driven by advances in data collection and storage, increasingly large and complex datasets are being stored in massive and high dimensional databases. Such enormous volumes of data clearly overwhelm traditional manual methods of data analysis such as spreadsheets and ad-hoc queries. These methods can create informative reports from data, but can not analyse the contents of these reports to focus on important knowledge. *Knowledge Discovery in Databases* or *KDD* automates the process of finding relationships and patterns in raw data from large databases and delivers results that can be either utilised in automated decision

support systems or assessed by a human analyst [31, 40, 51].

Questions such as the following would probably be answered if information hidden among terabytes of data can be found explicitly and utilised.

- What is the response probability of a certain customer to a planned promotion?
- Will this customer default on a loan or pay back on schedule?
- What medical diagnosis should be assigned to this patient?
- How large are the peak loads of a telephone or energy network going to be?

Modelling the investigated system, discovering relations that connect variables in a database are the essence of KDD. Modern computer-based KDD systems self-learn from the previous history of the investigated system, formulating and testing hypotheses about the rules that the system obeys. When concise and valuable knowledge about the system of interest has been discovered, it can be incorporated into some decision support system that helps the manager to make wise and informed business decisions. KDD is an interdisciplinary field involving database systems, statistics, machine learning, visualisation, etc [51].

2.2.1 Steps of KDD

The KDD process is both *iterative* and *interactive* involving a number of steps as shown in figure 2.2 [31]. Often the output of a step is fed back into preceding step(s), and typically multiple iterations are required to extract high-quality knowledge. The interaction of the system with a domain expert through the monitoring of the loop is also necessary to ensure the usefulness and accuracy of the results.

In particular, the KDD process consists of the following steps:

1. **Selection and Integration:** The data to be mined may reside in different and heterogeneous data sources. Thus, the first step involves selecting the data relevant to the analysis task from various databases and integrating them into a coherent data store.

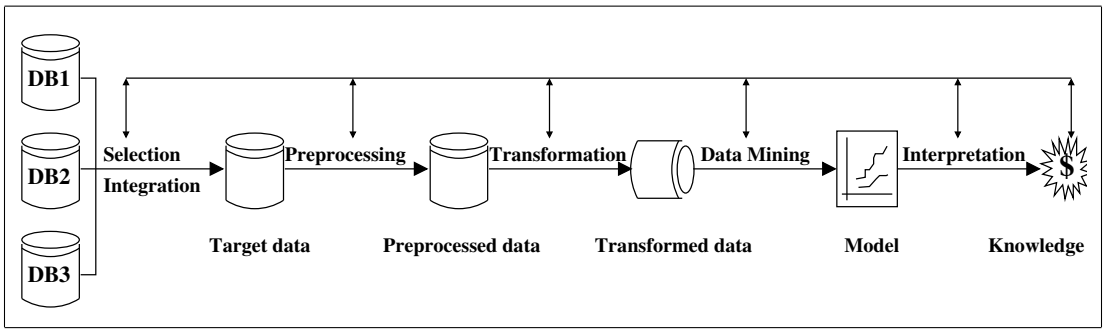


Figure 2.2: The KDD Process

2. **Preprocessing:** Raw data may have erroneous or missing values. Erroneous values are corrected or removed while missing values are supplied or predicted.

3. **Transformation:** The data are transformed into representations appropriate for mining tasks.

4. **Data Mining:** *Data Mining* (*DM*) is the *core* step of the KDD process referring to the application of intelligent techniques to extract the hidden knowledge from the transformed data.

5. **Interpretation and Evaluation:** A data mining system has the potential to generate a large number of patterns but only a small fraction of them may be of interest. Thus, appropriate metrics are required to evaluate the interestingness of the results. Advanced visualization techniques would facilitate when interpreting/evaluating the discovered knowledge.

2.2.2 Data Mining Tasks

DM tasks can be broadly classified into two categories: *descriptive* and *predictive* [51]. Descriptive tasks infer a profile that characterise the general properties of the data, while predictive tasks perform inference in the current data to make predictions. There are many and diverse data mining tasks including: *classification*, *regression*, *association analysis*, *clustering*, *time series analysis*, outlier analysis [31, 40, 51].

Classification: *Classification*, a well-studied data mining task, is learning a set of functions that classify a data item into one of several predefined classes [53]. Classification is often referred to as *supervised learning* because the classes are determined before examining the data. Each data instance consists of two parts, a set of predictor attribute values and a goal attribute value where the latter is drawn from a small discrete domain. During classification the dataset is divided into two mutually exclusive subsets, the training and testing set. In the training set the values of both the predictor attributes and goal attribute are available to the algorithm to learn a relationship between them that is used subsequently to predict the class label of the data in the test set. The maximisation of the classification accuracy rate in the test set is the main goal of learning.

Regression: *Regression* or *prediction*, a type of classification, is the task of learning patterns from examples and using the developed model to predict future values of the target variable. Whereas classification predicts categorical labels, regression models continuous-valued functions predicting numerical values.

Association Analysis: *Association analysis* is the discovery of association rules, which can be viewed as a model that identifies specific types of data associations [51]. These associations are often used in the retail sales management to identify items that are frequently purchased together. One also searches for directed association rules identifying the best product to be offered with a current selection of purchased products.

Time Series Analysis: A time series is an ordered sequence of values of an attribute at equally spaced time intervals. There are three basic tasks performed in time series analysis: *a*) to obtain an understanding of the underlying forces and structure, e.g. regularities or trends, that produced the observed data *b*) to find similarities or correlations between different time series and *c*) to fit a model and proceed to forecasting, monitoring or even feedback and feed-forward control.

Clustering: *Clustering* is an unsupervised learning task where one seeks to identify a finite set of categories termed clusters to describe the data [58, 62, 37, 30]. Unlike classification that analyses class-labelled instances, clustering has no training stage, and is usually used when the classes are not known in advance. A similarity metric is defined between items of data, and then similar items are grouped together to form clusters. Often, the attributes providing the best clustering should be identified as well. The grouping of data into clusters is based on the principle of *maximising the intraclass similarity* and *minimising the interclass similarity*. Properties about the clusters can be analysed to determine cluster profiles, which distinguish one cluster from another. A new data instance is classified by assignment to the closest matching cluster, and is assumed to have characteristics similar to the other data in the cluster.

2.2.3 Thesis's Data Mining Task

The research in the Thesis addresses the challenging unsupervised-learning task of *clustering*. When dealing with large and high dimensional databases, clustering can be viewed as a type of data *compression* or *summarisation*; the detailed data within the database are abstracted and compressed to a smaller set of class descriptions, one for each class, that summarise the characteristics of the data in each cluster. Although the idea of approximating a group of similar data points using its cluster descriptor loses fine details, it is very useful especially for large and high dimensional datasets as it provides a simplification of the underlying data distribution, and it also helps to uncover hidden nuggets of knowledge.

2.3 Cluster Analysis

Cluster analysis is a descriptive data analysis task that aims to find the intrinsic structure of a collection of data points (or objects) by partitioning them into homogeneous clusters based on the values of their attributes. Often, a similarity metric is defined between data objects, and then similar objects are grouped together to form clusters. Unlike classification, clustering is unsupervised learning since it does not require assumptions about category labels that tag objects with prior identifiers. Since there is no universal definition of clustering, there is no universal measure with which to compare clustering algorithms. Clustering methods have been extensively studied in a wide variety of disciplines including psychology and other social sciences, biology, statistics, pattern recognition, information retrieval, machine learning, and DM [17, 30, 31, 37, 40, 51, 52, 58, 59, 62].

The Goal of Clustering: To illustrate the goal of clustering consider figures 2.3 and 2.4 showing the distribution of the seismic activity in mainland USA, projected onto two different subspaces, [*Longitude* × *Latitude*] and [*Longitude* × *Latitude* × *Depth*], respectively. From figure 2.3 it is evident that earthquakes are not evenly distributed throughout USA, but rather the vast majority of events are highly concentrated in the west coast along the infamous *Saint Andreas* faulting zone [1, 2]. More informative structures are revealed in figure 2.4 because in the augmented subspace [*Longitude* × *Latitude* × *Depth*] correlations along the Depth axis are not flattened as in figure 2.3. However, as discussed in subsequent sections, not always all dimensions increase the clustering tendency of the data.

The goal of clustering is to summarise the underlying distribution of data for the purposes of improved understanding. For very low dimensional data, e.g. up to 4 dimensions, with well-separated clusters, the use of visualisation techniques aids users to better understand the distribution of data. However, the old expression, "*a picture is worth a thousand words*" rapidly vanishes when examining the structure of data with increasing dimensionality. Thereby, robust clustering algorithms operating in the full-dimensional space, with the ability to produce output that can be easily assimilated by end-users, are needed.

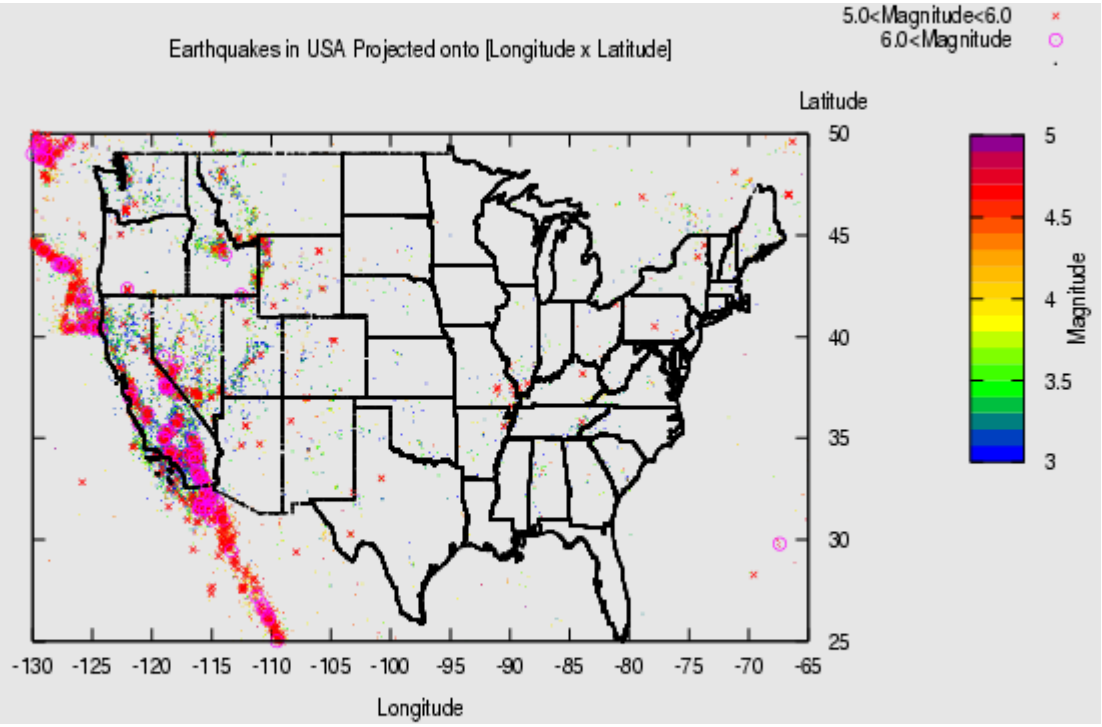


Figure 2.3: Seismic Activity in USA Projected in [*Longintude* × *Latitude*]

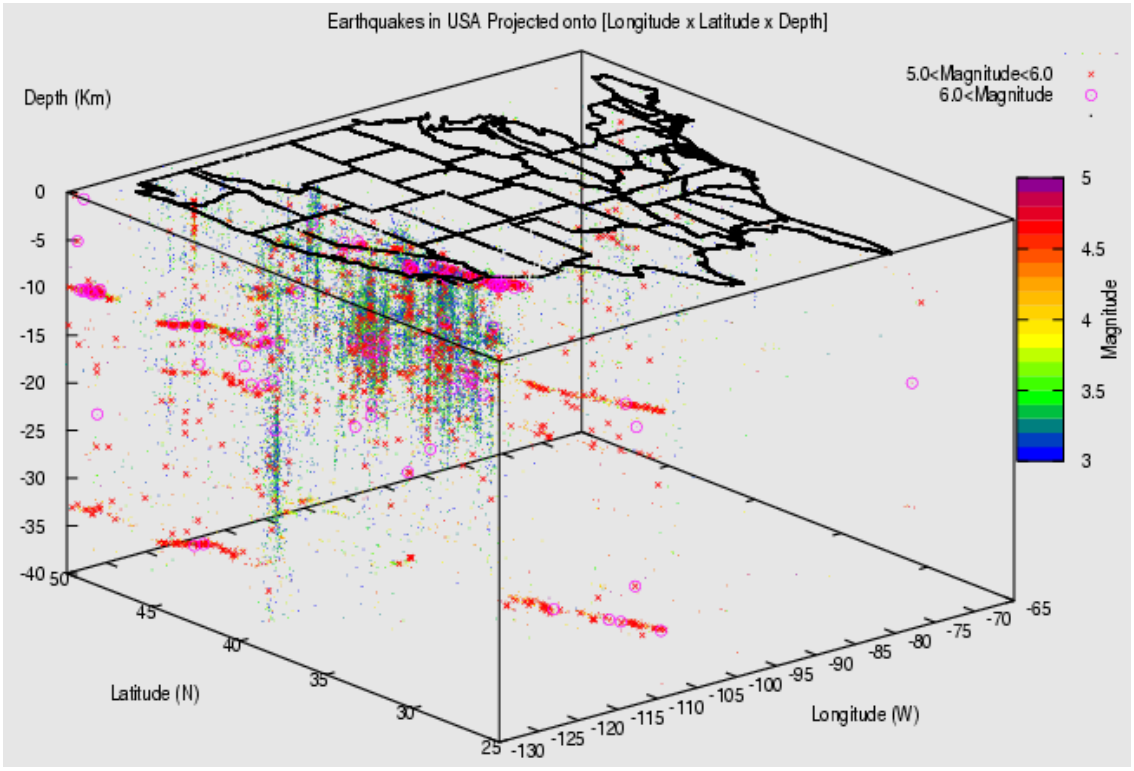


Figure 2.4: Seismic Activity in USA Projected in [*Longintude* × *Latitude* × *Depth*]

2.3.1 Definitions and Notation

Let $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_d\}$ be a set of attributes with bounded, totally ordered numerical domains and $\mathcal{F} = \mathcal{A}_1 \times ... \times \mathcal{A}_d$ a d-dimensional *feature* or *data space*, where $\mathcal{A}_1$,...,$\mathcal{A}_d$ are the *features*, *attributes*, *variables*, or *dimensions* of $\mathcal{F}$, and d denotes the dimensionality of $\mathcal{F}$. The input, $N \times d$ *pattern matrix* $\mathcal{P} = \{p_1, ..., p_N\}$ consists of a set of d-dimensional *records* that are also known as *patterns*, *points*, *objects*, *tuples*, *items*, *examples* or *cases*, while N denotes the *size* of $\mathcal{P}$. Each data point p_i, is a vector containing d numerical values $p_i = [p_{i1}, ..., p_{id}]$ such that p_{ij} is drawn from the domain $[a_j, b_j]$ of $\mathcal{A}_j$ attribute. In current databases the diversity of kinds of data is higher than ever before, e.g. continuous, categorical, temporal, multimedia, transactional [51]. The thesis focuses on clustering databases with real-valued attributes. The most popular method to measure the proximity $d(i, j)$ between two points p_i and p_j on $\mathcal{F}$ is the Euclidean distance: $d(i, j) = \sqrt{|p_{i1} - p_{j1}|^2 + ... + |p_{id} - p_{jd}|^2}$ [58]. The *clustering criterion* that is often expressed via a cost function or some other type of transition rule, assesses the quality of a given partitioning. *Fuzzy* clustering techniques assign to each data point a fractional degree of membership in each cluster, while *crisp* or *hard* clustering methods create disjoint partitions where each data point belongs exclusively to an individual cluster. This thesis addresses crisp clustering only.

2.3.2 Challenges in Data Mining Clustering

Clustering in high dimensional spaces is a very hard problem due to the *curse of dimensionality* phenomenon [15], and the presence of *irrelevant features* [7].

The Curse of Dimensionality: It was Richard Bellman who apparently originated the phrase "*...It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days...*" [14]-(page 97), to emphasise the impossibility of optimizing a function of many variables by a brute force search on a discrete multidimensional grid. This is because the number of grids points increases exponentially with dimensionality, i.e. the number of problem variables.

With the passage of time, the curse of dimensionality has come to refer to any problem in data analysis that results from a large number of variables.

For clustering purposes, the most relevant aspects of the curse of dimensionality are the impact of increasing dimensionality on point proximity and density. In particular, distance-based clustering techniques depend critically on the measure of distance, and require that the objects within clusters are, in general, closer to each other than to objects in other clusters. Density-based clustering algorithms require that the point density within clusters must be significantly higher than the surrounding noise regions.

In moderate-to-high dimensional spaces *almost all pairs of points are about as far away as average* [4, 18] and *the density of points inside a fixed-volume region is about as the average*. Under such circumstances the data are "lost in space" and the effectiveness of clustering algorithms that depend critically on the measure of distance or density, degenerate rapidly with increasing dimensionality [55].

Irrelevant Features - Subspace Clustering: Often, it maybe that not all dimensions are relevant - the data are binded along such dimensions, to a given cluster. The presence of irrelevant features reduces any clustering tendency in the data. Intuitively, if all irrelevant features are pruned away, the points in each cluster come closer to one another, making easier the discrimination of clusters using a distance or density based criterion. However, feature selection techniques are susceptible to a substantial loss of information because *different types of inter-attribute correlations may occur in different subsets of dimensions in different localities of the data* [6, 7]. Therefore, it is vital for any clustering algorithm to operate on the full dimensional space.

Requirements for DM Clustering: Emerging DM applications place specific requirements on clustering techniques such as [51]:

1. **Handling High Dimensionality:** Often, complex real-world concepts are accompanied by a large number of features. As a result of the sparsely filled space - the number of available points can not grow exponentially with the

dimensionality, there is often a very poor discrimination between clusters in the full dimensional space.

2. **Irrelevant Features - Subspace Clustering:** Often, especially in high dimensional spaces, not all dimensions are relevant - the data are binded along such dimensions, to a given cluster. It is vital for a clustering method to be able to detect clusters being embedded in subspaces possibly formed by different combinations of dimensions in different data localities.

3. **Scalability:** The massive datasets, both in size and dimensionality, associated with DM applications require highly scalable clustering algorithms. Sampling and parallelisation can be potentially used to improve scalability.

4. **Clusters of Arbitrary Shape, Size, Density, and Data Coverage:** Distance-based clustering algorithms tend to find spherical clusters with similar size and density. It is important to develop clustering algorithms that can detect clusters of arbitrary shape, size, density, and data coverage. This would help us gain a deeper insight into the different correlations between the features which, in turn, can greatly facilitate other steps of KDD, e.g. decision making processes.

5. **Interpretability of the Results:** Even the most advanced visualisation techniques do not work well in high dimensional spaces, simply because the human eye-brain system is able to perform rough clustering only up to three dimensions. Therefore it is essential to produce cluster descriptors that can be easily assimilated by an end-user, e.g. *IF-THEN* rules, decision trees.

6. **Insensitivity to Noise:** Most real-world databases contain noise and outliers that do not fit nicely into any of the clusters. The quality of the clustering results must not be affected by the presence of noise and outliers.

7. **Insensitivity to Initialisation and Order of Input:** It is vital to develop clustering algorithms that produce similar quality results irrespective of the initialisation phase and the order in which input data are processed.

8. **Minimal Requirements for Domain Knowledge:** Clustering algorithms should have minimal requirements of auxiliary domain knowledge to determine the input parameters, since the former is rarely complete and consistent. Furthermore, the quality of the results must be relatively insensitive to the input settings. Finally, the clustering algorithm must not presume any canonical data distribution for the input data.

9. **Handling of Different Types of Features:** Given the diversity of kinds of data being stored in the current databases, e.g. numerical, categorical, multimedia, real-world applications may require clustering data consisting of a mixture of data types.

2.3.3 Classification of Clustering Algorithms

Clustering algorithms for metric spaces can be broadly classified into five, possibly overlapping, types and they are discussed in the following subsections together with specific algorithms: *partitioning* (section 2.3.4), *hierarchical* (section 2.3.5), *density-based* (section 2.3.6), *grid-based* (section 2.3.7), *subspace-clustering* (section 2.3.8), and some other (section 2.3.9) methods [17, 52, 58, 59].

In short, partitioning algorithms attempt to determine k clusters that optimise a certain, often distance-based, criterion function. Hierarchical algorithms create a hierarchical decomposition of the database that can be presented as a dendrogram. Density-based algorithms search for dense regions in the data space that are separated from one another by low density noise regions. Grid-based methods quantise the search space into a finite number of disjoint cells and then operate on the quantised space. Subspace clustering attempts to identify, in addition to the clusters, the subspace of dimensions in which each cluster is embedded.

2.3.4 Partitioning Clustering Algorithms

Partitioning clustering attempts to decompose a set of N objects into k clusters such that the partitions optimise a certain criterion function. Each cluster is represented by the centre of gravity (or centroid) of the cluster, e.g. *k-means*, or by the closest instance to the gravity centre (or medoid), e.g. *k-medoids*. Typically, k seeds are randomly selected and then a relocation scheme iteratively reassigns points between clusters to optimise the clustering criterion. The minimisation of the *square-error criterion* [58] - sum of squared Euclidean distances of points from their closest cluster representative point, is the most commonly used. A serious drawback of partitioning algorithms is that they suffer from a combinatorial explosion due to the number of possible solutions. In particular, the number of all possible partitions $P(n,k)$ that can be derived by partitioning n patterns into k clusters is [42, 60]:

$$\boxed{P(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\begin{pmatrix} k \\ i \end{pmatrix}(i)^n} \tag{2.1}$$

An example in [58] shows that having to partition $n = 10$ patterns into $k = 4$ clusters the total number of different partitions is $P(10,4) = 34105$. However, for $n = 19$ and $k = 4$, $P(19,4)$ becomes huge, approximately 11,259,666,000.

Some representative examples of partitioning methods are:

K-MEANS

k-means is perhaps the most popular clustering method in metric spaces [54, 58, 68]. Initially k cluster centroids are selected at random. *k-means* then reassigns all the points to their nearest centroids and recomputes centroids of the newly assembled groups. The iterative relocation continues until the criterion function, e.g. square-error, converges. Despite its wide popularity, *k-means* is very sensitive to noise and outliers since a small number of such data can substantially influence the centroids. Other weaknesses are sensitivity to initialisation, entrapments into local optima, poor cluster descriptors, inability to deal with clusters of arbitrary shape, size and density, reliance on user to specify the number of clusters.

K-MEDOIDS

Unlike *k-means*, in the *k-medoids* or *Partitioning Around Medoids* (*PAM*) [62] method a cluster is represented by its medoid that is the most centrally located object (pattern) in the cluster. Medoids are more resistant to outliers and noise compared to centroids. PAM begins by selecting randomly an object as medoid for each of the k clusters. Then, each of the non-selected objects is grouped with the medoid to which it is the most similar. PAM then iteratively replaces one of the medoids by one of the non-medoids objects yielding the greatest improvement in the cost function. Clearly, PAM is an expensive algorithm as regards finding the medoids, as it compares each medoid with the entire dataset at each iteration of the algorithm.

Expectation Maximisation - EM

Instead of representing a cluster with a single point, the *expectation maximisation (EM)* algorithm [69] represents each cluster using a probability distribution. EM is an example of fuzzy clustering where each object is assigned a certain degree of membership to each cluster. Similar to k-means and k-medoids, EM iteratively modifies the membership of each object until a likelihood-based criterion function converges. EM is frequently entrapped into local optima.

Clustering Large Applications - CLARA

CLARA [62] is an implementation of PAM in a subset of the dataset. It draws multiple samples of the dataset, applies PAM on samples, and then outputs the best clustering out of these samples.

Clustering Large Applications based on Randomised Search - CLARANS

CLARANS [75] combines the sampling techniques with PAM. The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of k medoids. The clustering obtained after replacing a medoid is called the neighbour of the current clustering. CLARANS selects a node and compares it to a user-defined number of neighbours searching for a local minimum. If a better neighbour is found having lower square error, CLARANS moves

to the neighbour's node and the process starts again; otherwise the current clustering is a local optimum. If the local optimum is found, CLARANS starts with a new randomly selected node in search for a new local optimum.

Advantages and Disadvantages of Partitioning Clustering

In short, the advantages and disadvantages of partitioning clustering methods are:

- **Advantages**
  - Relatively scalable and simple.
  - Suitable for datasets with compact spherical clusters that are well-separated
- **Disadvantages**
  - Severe effectiveness degradation in high dimensional spaces as almost all pairs of points are about as far away as average; the concept of distance between points in high dimensional spaces is ill-defined
  - Poor cluster descriptors
  - Reliance on the user to speficy the number of clusters in advance
  - High sensitivity to initialisation phase, noise and outliers
  - Frequent entrapments into local optima
  - Inability to deal with non-convex clusters of varying size and density

2.3.5 Hierarchical Algorithms

Unlike partitioning methods that create a single partition, *hierarchical algorithms* produce a nested sequence (or *dendrogram*) of clusters, with a single all-inclusive cluster at the top and singleton clusters of individual points at the bottom [17, 52, 58, 59]. The hierarchy can be formed in top-down (*divisive*) or bottom-up (*agglomerative*) fashion and need not necessarily be extended to the extremes. The merging or splitting stops once the desired number of clusters has been

formed. Typically, each iteration involves merging or splitting a pair of clusters based on a certain criterion, often measuring the proximity between clusters. Hierarchical techniques suffer from the fact that previously taken steps (merge or split), possibly erroneous, are irreversible. Some representative examples are:

AGNES and DIANA

AGglomerative NESting (*AGNES*) [62] and *Divisive ANAlysis* (*DIANA*) [62] are two earlier bottom-up and top-down hierarchical clustering methods, respectively. In both AGNES and DIANA, the similarity or dissimilarity between clusters is computed using the distance between the cluster representative points, e.g. centroids or closest points. Both methods are irreversible and use over-simplified static rules to split or merge clusters which may lead to low quality clustering. Finally, they do not scale well since the decision to merge or split needs to examine and evaluate many combinations of clusters.

CURE

Clustering Using REpresentatives (*CURE*) [48] is an agglomerative method introducing two novelties. First, clusters are represented by a fixed number of well-scattered points instead of a single centroid. Second, the representatives are shrunk toward their cluster centres by a constant factor. At each iteration, the pair of clusters with the closest representatives are merged. The use of multiple representatives allows CURE to deal with arbitrary-shaped clusters of different sizes, while the shrinking dampens the effects of outliers and noise. CURE uses a combination of random sampling and partitioning to improve scalability.

CHAMELEON

CHAMELEON [61] improves the clustering quality by using more elaborate merging criteria compared to CURE [61]. Initially, a graph containing links between each point and its k-nearest neighbours [58] is created. Then a graph-partitioning algorithm recursively splits the graph into many small unconnected sub-graphs. During the second phase, each sub-graph is treated as an initial sub-cluster and

an agglomerative hierarchical algorithm repeatedly combines the two most similar clusters. Two clusters are eligible for merging only if the resultant cluster has similar inter-connectivity and closeness to the two individual clusters before merging. Due to its dynamic merging model CHAMELEON is more effective than CURE in discovering arbitrary-shaped clusters of varying density. However, the improved effectiveness comes at the expense of computational cost that is quadratic in the database size.

BIRCH

Balanced Iterative Reducing and Clustering Using Hierarchies (*BIRCH*) [99] introduces a novel hierarchical data structure, *CF-tree*, for compressing the data into many small sub-clusters and then performing clustering with these summaries rather than the raw data. Sub-clusters are represented by compact summaries, called *cluster-features* (*CF*) that are stored in the leafs. The non-leaf nodes store the sums of the *CF*s of their children. A *CF-tree* is built dynamically and incrementally, requiring a single scan of the dataset. An object is inserted in the closest leaf entry. Two input parameters control the maximum number of children per non-leaf node and the maximum diameter of sub-clusters stored in the leafs. By varying these parameters, BIRCH can create a structure that fits in main memory. Once the *CF-tree* is built, any partitioning or hierarchical algorithms can use it to perform clustering in main memory. BIRCH is reasonably fast, but has two serious drawbacks: data order sensitivity and inability to deal with non-spherical clusters of varying size because it uses the concept of diameter to control the boundary of a cluster.

Advantages and Disadvantages of Hierarchical Clustering

In short, the advantages and disadvantages of hierarchical clustering methods are:

- **Advantages**
  - Embedded flexibility regarding the level of granularity.

- Well suited for problems involving point linkages, e.g. taxonomy trees.

- **Disadvantages**
  - Inability to make corrections once the splitting/merging decision is made.
  - Lack of interpretability regarding the cluster descriptors.
  - Vagueness of termination criterion.
  - Prohibitively expensive for high dimensional and massive datasets.
  - Severe effectiveness degradation in high dimensional spaces due to the curse of dimensionality phenomenon (see section 2.3.2) [4, 18].

2.3.6 Density-based Clustering Algorithms

Density-based clustering methods group neighbouring objects into clusters based on local density conditions rather than proximity between objects [17, 31, 51, 52]. These methods regard clusters as dense regions being separated by low density noisy regions. Density-based methods have noise tolerance, and can discover non-convex clusters. Similar to hierarchical and partitioning methods, density-based techniques encounter difficulties in high dimensional spaces because of the inherent sparsity of the feature space, which in turn, reduces any clustering tendency. Some representative examples of density-based clustering algorithms are:

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (*DBSCAN*) [32] seeks for *core* objects whose ϵ-neighbourhood (ϵ:radius) contains at least *MinPts* points. A set of core objects with overlapping ϵ-neighbourhoods define the skeleton of a cluster. Non-core points lying inside the ϵ-neighbourhood of core objects represent the boundaries of the clusters, while the remaining are noise. DBSCAN can discover arbitrary-shaped clusters, is insensitive to outliers and order of data input, while its complexity is $O(N^2)$. If a spatial index data structure is used the complexity can be improved up to $O(N \log N)$. DBSCAN breaks down in high-dimensional spaces and is very sensitive to the input parameters ϵ and *MinPts*.

OPTICS

Ordering Points To Identify the Clustering Structure (*OPTICS*) [8], an extension of DBSCAN to adapt to local densities, builds an augmented ordering of data and stores some additional distance information, allowing the extraction of all density-based clustering for any lower value of the radius ϵ. OPTICS has the same complexity as that of DBSCAN.

DENCLUE

DENsity-based CLUStEring (*DENCLUE)* [56] uses an *influence function* to describe the impact of a point about its neighbourhood while the overall density of the data space is the sum of influence functions from all data. Clusters are determined using *density attractors*, local maxima of the overall density function. To compute the sum of influence functions a grid structure is used. DENCLUE scales well ($O(N)$), can find arbitrary-shaped clusters, is noise resistant, is insensitive to the data ordering, but suffers from its sensitivity to the input parameters. The curse of dimensionality phenomenon (see section 2.3.2) heavily affects DENCLUE's effectiveness. Moreover, similar to hierarchical and partitioning techniques, the output, e.g. labelled points with cluster identifier, of density-based methods can not be easily assimilated by humans.

Advantages and Disadvantages of Density-based Clustering

In short, the advantages and disadvantages of density-based clustering are:

- **Advantages**
  - Discovery of arbitrary-shaped shaped clusters with varying size
  - Resistance to noise and outliers
- **Disadvantages**
  - High sensitivity to the setting of input parameters
  - Poor cluster descriptors

- Unsuitable for high-dimensional datasets because of the curse of dimensionality phenomenon (see section 2.3.2)

2.3.7 Grid-based Clustering Algorithms

Grid-based clustering techniques quantise the feature space into a multi-dimensional grid structure on which all operations for clustering are performed. The aggregated statistical information stored in the grid cells is then exploited by the clustering algorithm for fast processing. Due to data aggregation, the running time is typically independent of the database size, yet dependent on only the number of cells, which in turn, grows exponentially with the dimensionality. However, as the dimensionality increases more points map into individuals cells, thus making the use of summarised information decreasingly relevant to clustering. Some representative grid-based clustering techniques are:

WaveCluster

WaveCluster [93], a multi-resolution clustering algorithm, maps the data into a user-specified multi-dimensional grid, it then applies a wavelet transformation to the original feature space and, finally, finds connected dense regions in the transformed space. A wavelet transform is a signal processing technique that decomposes a signal into different frequency sub-bands. The high-frequency parts of a signal correspond to cluster boundaries, while low frequency high amplitude parts correspond to the clusters' interiors. The wavelet model can be applied to d-dimensional signals by applying a one-dimensional wavelet transformation d times. Convolution with an appropriate kernel function results in a transformed space where the clusters become more distinguishable. WaveCluster conforms to many requirements of DM clustering, such as detection of clusters at varying levels of accuracy, discovery of arbitrary-shaped clusters of varying density and size, insensitivity to noise and data ordering and linear scalability with dataset size. Both the effectiveness and efficiency of WaveCluster degrade rapidly with increasing dimensionality.

STING

STatistical INformation Grid-based method (*STING*) [98] uses a hierarchical technique to divide the feature space into rectangular cells. Each cell at level i is partitioned into a fixed number, k, of cells at the next lower level and so on. The database is scanned once, and statistical information related to the points inside each cell, e.g. mean, min, max, variance, distribution type is stored. The cells are populated in bottom-up fashion. Statistical information of higher level cells can easily be computed from the parameters of the lower level cell. The aggregated information stored in the cells is used for efficient query processing. To perform clustering on such a data structure, the user must first supply the density level. Then a breadth-first tree traversal is used to find relevant regions with sufficient density, until the bottom layer is reached. The regions of relevant cells are then returned as clusters. Once the tree is built, clustering at a certain density level is fast and query-independent, while the updating of the tree can be made in an incremental fashion. Balancing grid granularity and accuracy is not a trivial task. Since STING ignores the spatial relationship between the children and their neighbouring cells for construction of the parent cell, the resulting clusters are isothetic - the clusters are either horizontal or vertical.

Advantages and Disadvantages of Grid-based Clustering

In short, the advantages and disadvantages of grid-based clustering methods are:

- **Advantages**
  - Fast processing time that is typically independent of the number of data points, yet dependent on only the number of cells in the quantised space.
  - Resistance to the presence of noise and outliers
  - Insensitivity to the order of data input and initialisation
  - Discovery of arbitrary-shaped clusters, e.g. WaveCluster
  - Relatively comprehensible clustering output
- **Disadvantages**

- Computationally expensive for high dimensional datasets
- Aggregation in high dimensional spaces becomes meaningless because the vast majority of points map into different cells; Hence all grid-based clustering methods are only suitable for very low dimensional datasets.

2.3.8 Subspace Clustering Algorithms

Often, especially in high dimensional spaces, it maybe that not all dimensions are relevant - the data along relevant dimensions are bounded, to a given cluster. *Subspace clustering* seeks, in addition to the clusters, the subspace of relevant dimensions in which each cluster is embedded [5, 7, 23, 74]. Examining every possible subspace for clusters is infeasible as their number grows exponentially with dimensionality. Relying on the user to specify the subspace(s) is not a good choice since domain knowledge is rarely complete and consistent. Applying dimensionality reduction techniques, e.g. Principal Components Analysis [45], also has drawbacks. First, the new dimensions - linear combinations of original attributes, are hard to interpret, making it hard to understand clusters in relation to the original data space. Second, these techniques are not effective in identifying clusters that may exist in different subspaces, possibly overlapping. Recently, there have been proposed some extensions of existing subspace clustering methods to tackle scalability and effectiveness, but their analysis is out of the scope of the thesis. A comprehensive and comparative discussion regarding the recent advances in subspace clustering can be found in [79]. Some representative subspace clustering algorithms are:

CLIQUE

CLustering In QUEst - developed by the data mining research at IBM Almaden - (*CLIQUE*) [7] is the pioneer bottom-up search method that combines grid and density based clustering to locate dense clusters in subsets of dimensions. Initially, a user-specified uniform grid is imposed onto the data space. A k-dimensional unit defined by the Cartesian product of one interval from each of the k dimensions is *dense* if the number of points inside it exceeds an input threshold.

Clusters are unions of connected dense units. CLIQUE utilises the downward closure property of density to reduce search combinations: *if a k-dimensional unit is dense, then so are its projections in (k-1)-dimensional space*. Thereby, candidate k-D dense units are formed by self-jointing dense units found in (k-1)-D spaces. The joint condition is that units share the *first* (k-2) dimensions. Clusters are formed by combining dense units using a greedy growth scheme. The hyper-rectangular clusters are reported using Disjunctive Normal Form (DNF) expressions. CLIQUE does not presume any canonical distribution for the data, is insensitive to noise and data ordering, finds arbitrary-shaped clusters of varying size and more importantly, produces highly interpretable cluster descriptors. It scales linearly with the size of input and quadratically with the data and cluster dimensionality. CLIQUE operates on a regular, static, user-defined grid with very coarse resolution. The coarse resolution is adopted for two reasons: a) to enable locating dense units in high dimensional spaces, and b) to reduce computation that grows exponentially with the number of bins. Hence, CLIQUE places more emphasis on finding relatively dense clusters than on capturing their shapes with precision.

MAFIA

Merging of Adaptive Finite Intervals (And is more than a clique) (*MAFIA*) [74] extends CLIQUE by introducing adaptive bins that are constructed semi-automatically based on the data distribution to improve scalability and cluster quality. Moreover, MAFIA examines more subspaces compared to CLIQUE, since it combines dense units sharing *any* (k-2) dimensions rather than the *first* (k-2) dimensions as in CLIQUE. The introduction of adaptive bins yields an average of two orders of magnitude speedup as compared to CLIQUE. Finally, MAFIA scales linearly with database size and dimensionality, but not surprisingly its execution time increases quadratically with the dimensionality of the clusters.

ENCLUS

ENtropy-based CLUStering (*ENCLUS*) [23], another extension of CLIQUE, uses

static uniform grids and performs subspace clustering based on the concept of entropy [45] rather than density. In short, subspaces with clusters typically have lower entropy than subspaces without clusters. Additionally, entropy decreases as the density of cluster units increases. Pruning is accomplished using the downward closure property of entropy, which is similar to the density downward closure property. ENCLUS shares much of the flexibility of CLIQUE, but similar to CLIQUE is very sensitive to the user-specified resolution of the grid.

Advantages and Disadvantages of Subspace Clustering Algorithms

In short, the advantages and disadvantages of subspace clustering methods are:

- **Advantages**
  - Relatively effective treatment of high dimensionality
  - Automatic detection of irrelevant features
  - Comprehensible cluster descriptors
  - Insensitivity to the order of data input, noise/outliers, and initialisation
  - Detection of arbitrary-shaped clusters of varying size and density
  - Good scalability with database size
  - No presumption of any canonical distribution for the input data
- **Disadvantages**
  - Computation increases exponentially with dimensionality.
  - To reduce computational complexity while locating adjacent dense cells in high dimensional spaces, *grid-based techniques adopt coarse resolutions at the expense of the accuracy of the clustering results* [7, 23, 74].

2.3.9 Other Clustering Methods

This section discusses briefly two important clustering algorithms that do not fit directly in any of the well defined clustering methods.

PROCLUS

PROjected CLUStering (*PROCLUS*) [5], the first top-down subspace clustering algorithm, attempts to find the so-called *projected clusters*. A projected cluster is a subset $\mathcal{C}$ of points together with a subset of dimensions $\mathcal{D}$, such that the orthogonal projection of $\mathcal{C}$ onto $\mathcal{D}$ is a tight cluster. Both the number of clusters k and the average cluster dimensionality l must be specified by the user *a priori*. Initially, k well scattered points are chosen to serve as medoids for the clusters using a greedy algorithm. Then a hill-climbing iterative process relocates the k-medoids as well as the subset of dimensions associated with each medoid aiming to minimise the distance between data points and the nearest medoid. Each cluster is represented as a non-overlapping set of points with its associated medoid and subspace. Due to the distance-based approach, PROCLUS is biased toward clusters that are hyper-spherical in shape. Moreover, although the subspaces where the clusters are embedded may be different, the dimensionality of subspaces must be similar.

ORCLUS

ORiented projected CLUSter generation (*ORCLUS*) [6] is an extension of PROCLUS [5] seeking non-axis aligned subspaces. Similar to PROCLUS, ORCLUS requires the number of clusters k and the average cluster dimensionality l to be specified by the user beforehand. Each cluster is represented by a set of points, its centroid, and an l-dimensional orthogonal system with eigenvectors having the least spread. ORCLUS is suitable for datasets where there are strong inter-attribute correlations, but is computationally very demanding due mainly to the computation of the covariance matrices, where the latter are used to decide which orthogonal system fits better the data of a cluster.

OptiGrid

Optimal Grid Clustering (*OptiGrid*) [55] seeks dense clusters using a data partitioning scheme based on divisive recursion by an irregular multi-dimensional grid. The grid is constructed using cutting hyperplanes passing through areas of

low density. This is done to minimise the partitioning of individual clusters and to discriminate distinct clusters as much as possible. Although non axis-aligned cutting planes are discussed in [55] the evaluation results are based on orthogonal projections. OptiGrid uses kernel density estimation to approximate the density of points [92, 97]. Due to the divisive recursion fashion by which the grid is built, OptiGrid does not face the problem of combinatorial search such as CLIQUE or MAFIA. OptiGrid is very sensitive to the setting of the input parameters.

O-Cluster

O-Cluster [72] combines a novel active sampling technique with an axis-parallel partitioning strategy to identify continuous areas of high density in the dataspace. O-Cluster uses a statistical test to validate the quality of orthogonal cutting planes, and operates on a small buffer containing a random sample of the dataset. Active sampling ensures that partitions are provided with additional data if more information is required to validate a given cutting plane. Partitions without ambiguities are frozen and the data associated with them are removed from the active buffer. Due to active sampling O-Cluster can handle efficiently large datasets that do not fit in the main memory. For each partition, a hyper-rectangular region, O-Cluster analyses orthogonal density histograms for valid cutting planes using an elaborate statistical test. Valid cutting planes pass through low density valleys surrounded on both sides by regions of significantly high density. Unlike, OptiGrid, only a single cutting plane is applied at time. O-Cluster creates a binary clustering tree where the disjoint leaves are regions with flat or unimodal density. Despite the appealing efficiency and effectiveness of O-Cluster, e.g. good scalability, noise tolerance, interpretable cluster descriptors, there is a serious drawback: *once a splitting operation has been performed no corrections are possible*. Another limitation of O-Cluster is the fact that it uses classical density histograms, which are extremely sensitive to the selection of bin width, especially for partitions with small numbers of points. From a practical point of view, in relatively sparse partitions the resulting histograms are jagged, making the discovery of unimodal density regions difficult.

2.4 Evolutionary Algorithms

2.4.1 Optimisation Problems

Optimisation is the process of finding feasible solution(s), i.e. solution(s) satisfying the problem constraints (if any), that optimally solve the target problem. Let f be a *fitness function* that ranks solutions to a given problem P with respect to their quality. Further, let S be the search space of all possible solutions to P. Optimising P involves finding an $x \in S$ satisfying:

$$\boxed{\begin{array}{l} f(x) \leq f(y)\ \forall y \in S \text{ if } f \text{ is to be } \textit{minimised}, \text{ and} \\ f(x) \geq f(y)\ \forall y \in S \text{ if } f \text{ is to be } \textit{maximised} \end{array}} \tag{2.2}$$

A solution x satisfying condition 2.2 is referred to as the *global optimum*. If S is a subset of $\mathbb{R}^d$ then P reduces to a *numerical optimisation problem*, where d is the dimensionality of the problem. Typically, the search space of large scale real-world problems is enormous, making enumeration or brute-force traversal impossible. Thereby, a variety of *heuristic techniques* have been suggested to surmount these difficulties. Heuristic techniques seek good, i.e. near-optimal, solutions at a reasonable computational cost without being able to guarantee either optimality, or even in many cases to state how close to optimality a particular solution is. One heuristic approach that has received increasing interest over the last three decades is *evolutionary algorithms*.

2.4.2 Biological Basis of Evolutionary Algorithms

Evolutionary Algorithms (*EA*s) or *Evolutionary Computation* (*EC*) - these terms are used interchangeably throughout the thesis - are stochastic search methods that mimic the metaphor of natural biological evolution. Over many generations natural populations evolve according to the principles of *natural selection* and *survival of the fittest*, first clearly stated by Charles Darwin in "The Origin of Species". In nature, individuals best suited to competition for scarce resources survive. Evolving to adapt to a changing environment is essential for the members of any species. A recently compiled two-volume textbook [11, 12] provides an excellent introduction to the field of *EA*s.

2.4.3 Basic Terminology

The terminology used in EA studies is borrowed from biology. All living organisms or individuals consist of cells containing a set of *chromosomes*, which serve as a "blueprint" for the organism. A chromosome contains a number of *genes*, each encoding a particular characteristic, e.g. a decision variable in optimisation problems. *Alleles* is the set of different states that a gene can express. Each gene is located at a certain position in the chromosome, the *locus*. If the chromosomes are arranged in pairs the organism is called *diploid* otherwise *haploid*. Most EAs use single-chromosome haploid representations. The *phenotype* is the expressed physical behaviour and morphology of the organism while the *genotype* is the particular set of genes in the given organism. Although an improving fitness manifests itself in many ways, e.g. animals run faster and increase their intelligence, viruses develop ever more effective ways of penetrating their host's defences, the ultimate measure of evolutionary fitness is simply the success of an organism in passing on its genes to viable offspring - or reproducing. In an EA context, the fitness of an individual depends on the performance of the phenotype and is usually determined by a fitness function. The problem parameters along with their domains constitute the *search space*, which is the collection of all possible solutions to the problem.

2.4.4 The Skeleton of an Evolutionary Algorithm

The skeleton of a typical EA is outlined in figure 2.5. EAs are iterative and stochastic processes that operate on a population of individuals. An individual encodes a potential solution to the target problem. An initial population of individuals P(0) is generated at random or heuristically. Each individual is then evaluated by a fitness function. The fitness scalar value is the quantitative information the algorithm uses to guide the search. Then some of these individuals are *selected* for *reproduction* and copied to the mating buffer C(t). Individuals are usually selected with a probability proportional to their fitness, which ensures that fitter individuals have more chances of reproducing. Selection and reproduction alone can not sample any new point in the search space. Therefore,

various genetic operators, usually recombination and mutation, are applied to the individuals in the mating buffer C(t), producing *offspring* C′(t). Next, the newly formed offspring are evaluated. The two populations of parents P(t-1) and children C′(t) are then merged to form a temporary population. Since most EAs maintain a fixed-sized population a *replacement* policy selects the appropriate number of individuals from the temporary population to create the new population P(t). This is normally achieved by replacing the worst individuals in the population, however, many different choices are available. The whole process is repeated until some termination criterion is satisfied, e.g. reaching a maximum number of generations or finding an acceptable solution by some criterion.

```
BEGIN
    t ← 0
    P(t) ← initialisation();
    evaluation(P(t));
    WHILE(termination_condition ≠ true) DO
       BEGIN
          t ← t + 1;
          C(t) ← selection_reproduction(P(t-1));
          C′(t) ← recombination(C(t));
          C′(t) ← mutation(C′(t));
          evaluation(C′(t));
          P(t) ← selection_replacement(C′(t), P(t-1));
       END
END
```

Figure 2.5: Skeleton of an Evolutionary Algorithm

2.4.5 Variants of Evolutionary Algorithms

Numerous variants of EAs have been proposed since the early attempts more than fifty years ago. This section gives a short review of EAs to highlight their similarities and differences. EAs can be broadly classified into three mainstreams, *Genetic Algorithms* (*GA*s) [57], *Evolution Strategies* (*ES*) [84, 91] and *Evolutionary Programming* (*EP*) [41]. Despite some functional differences, all these paradigms are based on the same biological principle, the survival of the fittest. Bäck [11] identifies three basic properties shared by all EA paradigms:

- **Population-based Search:** EAs operate on a population of individuals instead of a single solution. An individual is an abstraction of a living organism and

represents a potential solution in some problem space through a suitable coding. Individuals may also incorporate internal control parameters which are themselves subject to adaptation during the evolutionary search, e.g. ES.

- **Stochastic Search:** At each generation a new set of descendants are generated by stochastic processes intended to model natural mutation and recombination.

- **Fitness Selection:** Each individual is evaluated by a fitness function that measures its performance in solving the target problem. The fitness determines the probability of an individual in surviving and attracting mates. By favouring fit individuals, the most promising areas of the search space are explored.

Genetic Algorithms

The basic principles of GAs were first laid down rigorously by Holland [57] during the seventies, and are well described in many texts [11, 12, 25, 47, 71]. Unlike EP and ES, canonical GAs encode each decision variable in a binary substring of specific length depending on the required accuracy. However, there has been a general trend away from binary codings within the GA research community toward other representations, e.g. floating point [71]. One reason for the wide popularity of binary coding is because of its universality, which allowed for a uniform set of operators making GAs a general-purpose optimisation technique. Stochastic variation is usually introduced through crossover and mutation. Real-valued GAs use task specific variation operators [11, 12, 27].

In a typical binary-coded GA, during recombination two individuals (strings of bits) are selected from the mating pool at random and fragments of genetic material are exchanged between them to create the offspring. In *one-point crossover* operation a cross site along the string is randomly chosen and all bits after that point are swapped. Another popular form of recombination is the *two-point crossover*, which is performed by choosing two cross sites randomly and swapping the corresponding segments from the two parents between the two cross sites. Unlike one and two-point crossover which are segment-based, *Uniform Crossover* (*UC*) operates in the bit-level by exchanging individual bits between the parents

with a fixed and position independent probability, often set to 0.5. Mutation is achieved by flipping bits at random with a very small probability, e.g. 0.001.

A commonly used stochastic selection scheme in GAs is *proportional* selection, also known as *roulette-wheel* selection where the probability of an individual being selected for the next generation is directly proportional to its fitness. The main drawback with proportional selection is that a few good, but not necessarily near global optimum, individuals with a high fitness can quickly take over the population because they will be chosen more often than other individuals. Another popular stochastic selection scheme is the so-called *tournament selection*. A pool of ts (often called tournament size) individuals is picked at random from the current population and the fittest is copied into the new population. These choices are independent so individuals may be chosen more than once. Tournaments are performed to select the individual with the highest fitness among the individuals in the pool. The winning individual is copied to the new population pool and the process is repeated until the new population has been filled up. The larger the value of tournament size, the higher the selection pressure. If ts is set to one the selection is totally random. A commonly used value for tournament size is two, which is also referred to as a binary-tournament selection. When using stochastic selection schemes, such as tournament selection, an elitism strategy is often used. *Elitism* (also referred to as an elitist strategy) was introduced by De Jong and the idea is to keep the k best performing individuals in the population by protecting them from for example crossover, mutation, and stochastic selection. Without the elitism scheme the best individuals could accidentally be removed from the population forcing the EA to rediscover the solutions once more. Typically, only the best individual is protected from deletion ($k = 1$).

Evolutionary Programming

EP was introduced by L. Fogel during the early sixties [41]. Generally, EP bears many similarities to ES although they were developed independently. Both paradigms apply normally distributed mutations to real-valued solution vectors to generate offspring. The mutation variances may also be encoded in the chromosome, and self-adapted similar to ES. In strict EP crossover is not utilised.

In the basic EP model, μ parents generate λ offspring by mutation resulting in a total of $(\mu + \lambda)$ individuals that are evaluated with regard to the specified fitness function. Afterwards, the individuals in the current $(\mu + \lambda)$ population are selected to become parents in the next generation. Competition for survival is achieved in a stochastic manner. Each individual is compared with k randomly chosen (with replacement) opponents from the population. If the individual has a higher fitness than the opponent it receives a win, thus each individual can at most receive k wins. The individuals with the highest number of wins are selected for the next generation. Finally, the whole process of mutation-selection is repeated until a termination condition is fulfilled.

Evolution Strategies

ES were introduced by Rechenberg and Schwefel during the sixties [10, 84, 91]. Unlike binary-coded GAs, in ES candidate solutions are represented as real-valued vectors. In addition to the decision variables, the chromosome includes a set of strategy parameters specifying the variance of mutation for each variable or variable combinations. ES employ normally distributed mutation to modify the real-valued vector of the decision variables. Recent ES use either uniform or some blending-type, e.g. *arithmetic*, recombination scheme. In arithmetic crossover the offspring is generated as a weighted mean of the value of each gene (decision variables) of the parents, while in uniform crossover each decision variable is chosen from on the parents at random. There are two deterministic selection strategies that are commonly termed plus (+) and comma (,). The abbreviation (μ+λ)ES denotes an *ES* where μ parents generate λ offspring, and then the μ best individuals from the intermediate population of (μ+λ) individuals are selected for survival. The (μ, λ)ES scheme places all the bias in the offspring selection stage: μ parents produce λ offspring ($\lambda > \mu$), and the best μ offspring are selected to replace the parent population. The strategy parameters controlling the variance of mutation are modified either in a predetermined fashion or are subjected to self-adaptation through evolutionary optimisation as the decision variables [12, 82].

2.4.6 Individual Representation

Critical to any EA's performance is the choice of encoding for the candidate solutions, which is inherent to the nature of the target problem. This is because, the encoding scheme influences the design of the variation operators, and determines the size of the search space being sampled by the EA. Various encoding schemes have been devised to represent the decision variables [11, 12]. In most GA studies each variable is encoded in a binary string of a specific length depending on the required accuracy [47, 57]. However, it has been shown that for some real-valued numerical optimisation problems, floating-point representations outperform binary coding because they are faster, more precise and more consistent [71]. The coding of variables in strings makes the search space discrete, which implies a certain loss of precision in the accuracy of the solution. Longer strings can potentially reduce this loss but excessive string lengths would slow down the GA. A major drawback of conventional binary encoding is the presence of *Hamming cliffs* associated with certain strings where the transition to a neighbouring solution requires the alteration of many bits. To tackle Hamming cliffs one can possibly use Gray coding where any two consecutive numbers differ always by one bit only in their binary encoding. However, in both conventional binary and Gray coding, a bit change in any arbitrary location may cause a large change in the decoded value. ES and EP methods typically operate directly on the real-valued vector of the problem variables, which is usually accompanied by a set of strategy parameters specifying the variance of mutation.

2.4.7 Selection

Selection, a key process in EAs, does not create any new solution, but instead emphasises highly fit individuals in the population based on fitness criteria [11]. Searching a large and complex space involves a trade-off between *exploring* all possible solutions and *exploiting* knowledge accumulated from previously visited points of the search space to find the global optimum. If the focus is solely on exploitation, the population may prematurely converge into a local optimum. In

contrast, when strong emphasis is placed on exploration, the optimisation behaves similarly to a random search, which means that the EA might never converge to a good solution or the convergence will be very slow. Using the population-based search, EAs combine both exploration and exploitation at the same time through various selection and variation operators, respectively. Furthermore, EAs allow explicit control over exploitation to exploration by varying the parameters of the corresponding operators.

Parent selection allocates reproductive opportunities to each individual and can be either *deterministic* or *probabilistic* [28]. Most of the selection schemes control either the proportion of the population that reproduces or the distribution of reproductive opportunities or both. The most popular selection scheme is *tournament selection* where tournaments are played between k randomly drawn solutions and the best is copied in the mating pool. It is recommended to use stochastic selection to preserve diversity in the population by occasionally choosing not so good solutions [27].

2.4.8 Mutation

Mutation operators act on a single individual at a time by replacing the value of a gene with another, randomly generated value, leading to deleterious, neutral, or beneficial changes in the performance of the individual [11, 12, 47, 71]. The mutation probability is commonly set to a very small value in GAs, although significantly larger values are often used in EP and ES. From an exploratory viewpoint, mutation is important because it can introduce into an individual a gene value that is not even present in the current population, providing thus escapes from local optima. In binary-coded GAs, mutation flips the value of a gene with a very small probability. It is treated as a "background" operator, supporting the recombination operator, by ensuring that all possible combinations of allele values of the search space are accessible to the GA. Mutation disregards semantic linkages among genes in the chromosome in the sense that the positions in the string to undergo mutation and the new values for the mutated genes are

determined at random regardless of what happens at other positions in the string. ES and EP methods use normally distributed mutations to modify the real-valued vector of the decision variables. Usually, the probability of mutating a variable is inversely proportional to the number of variables. The strategy parameters that control the variance of the normal distributions may also be subjected to evolutionary optimisation in a separate search space.

2.4.9 Recombination

The aim of recombination or crossover in EAs is to combine the best characteristics of highly fit individuals in the hope of creating even better solutions [11, 12, 47, 71]. Offspring are created by selecting randomly parent solutions from the mating pool, and exchanging or blending fragments of genetic material between them. Traditional crossover schemes used in binary-coded GAs, e.g. *n-point* or *uniform crossover* that swap chromosome fragments, are conservative in the sense that every gene carried by an offspring is a copy of a gene inherited from one of its parents. For non-binary strings, however, it is common to blend rather than swap parental genes to create new ones that are not carried by either parent [27]. Two types of bias are attributed to crossover: *distributional* and *positional bias* [11]. Distributional bias implies that the probability distribution of the number of genes transmitted to offspring is not uniform. Positional bias refers to the extent that the probability of transmitting a set of genes intact depends on their relative positions in the chromosome. The latter is important because it indicates which combinations of genes are more likely to be inherited by offspring from their parents. It is important to notice that these biases are characteristics of crossover operators that are independent of the fitness of the parent individuals.

2.4.10 Comparing EAs with Classical Search Methods

A number of general-purpose techniques have been proposed for use in connection with search and optimisation problems. They all assume that the problem is defined by a fitness function that must be optimised. Some of the traditional optimisation techniques are described below.

- **Enumerative Search:** The basis for enumerative techniques is simplicity itself. To find the optimum value in a problem space (which is finite) look at the function values at every point in the space. For very large problem spaces, an exhaustive search is computational infeasible.

- **Random Search:** Random search techniques simply perform random walks of the problem space, recording the best optimum values discovered so far. Similarly to enumerative search, random searches suffer from efficiency problems. For large problem spaces, they should perform no better than enumerative searches. Both enumerative and random search do not exploit any knowledge gained from previous results.

- **Gradient Methods:** There exist a number of optimisation techniques, generally referred to as hill-climbing, which rely on using information about the local gradient of the function to guide the direction of the search. Usually, hill-climbing techniques start with a randomly chosen point and then move in the direction that gives the greatest improvement in the value of the function. There are several drawbacks to hill climbing methods. Firstly, they assume that the problem space being searched is continuous in nature. In other words, derivatives of the function representing the problem space exist. This is not true of many real world problems where the problem space is noisy and discontinuous. Another major disadvantage of using hill climbing is that they can only find the local optimum in the neighbourhood of the current point. Hill-climbing methods can perform well on functions with only one peak (unimodal) but on functions with many peaks (multimodal), they suffer from the problem that the first peak found will be climbed and this may not be the

highest peak. Consequently hill-climbing techniques can easily get trapped into local optima as they have no way of looking at the global picture. Hybrid methods combining random and hill-climbing searches are possible. For instance, once a peak has been found, the hill-climbing is re-started, but with another randomly chosen point. This hybrid method performs well if the function does not have many optima. However, since each hill-climbing is carried out in isolation, no overall picture of the shape of the search space is obtained.

- **Simulated Annealing:** Simulated Annealing (SA) is essentially a modified version of hill-climbing. Starting from a random point in the search space, a random move is made. If the move leads to an improvement in the function then it is accepted. Otherwise, the move is accepted only with probability p(t), where t denotes time. The function p(t) begins returning a value close to 1, but gradually declines towards zero in a way analogous to a metal cooling and freezing into a minimum energy crystalline structure (annealing process). Initially therefore, any moves are accepted, but as the "temperature" p(t) reduces, the probability of accepting a negative move that degrades the value of the function decreases. SA's major advantage over other methods is an ability to avoid becoming trapped at local minima by allowing negatives moves. However, too many negative moves will simply lead away from the global optimum. Like the random search, however, SA only deals with a single solution at a time and so does not build up an overall picture of the search space.

- **Evolutionary Algorithms:** The working principle of EAs is very different from most of the traditional optimisation algorithms. A comprehensive discussion regarding these differences can be found in [27].

  One of the fundamental differences is that EAs maintain a population of potential solutions instead of a single solution. Since many solutions are processed at the same time, EAs perform a global search in the search space, in contrast with the local search achieved using traditional optimisation methods.

  Another important difference is the kind of transition rules that are used to

move one solution to another solution. In a typical EA, the genetic material of individuals is altered in a stochastic way through the application of various non-deterministic genetic operators (recombination, mutation etc). In contrast, traditional optimisation techniques, i.e. hill-climbing methods, use pre-determined transition rules, which may cause the algorithm to get trapped into local optima. The probabilistic character of EA search along with the random initialisation of individuals allows EAs to avoid local optima.

From an implementation point of view, EAs are highly parallel procedures and can be easily and conventionally used in a parallel system. For instance, since EAs are made up from several tasks (selection, reproduction, recombination, mutation) involving a group of individuals rather than the entire population, several processing units can work on the same task. Additionally, since in many real-world applications the evaluation of individuals is the most computationally expensive part of EAs, various distributed or parallel architectures can be employed to reduce the running time substantially.

The trade-off between exploration and exploitation is well-known in optimisation problems. In particular, searching a large and complex search space involves a trade-off between exploring all possible solutions and exploiting the information obtained up until the current generation. EAs achieve exploration through the recombination and mutation operators, while the selection-reproduction strategy is responsible for exploiting the obtained information. An important issue is the balance between the extent of exploitation and exploration. As Deb [27] pointed out "*...if the solutions obtained are exploited too much premature convergence is expected. On the other hand if too much stress is given on a search, the information obtained thus far has not been used properly and the search exhibits a similar behaviour to that of random search...*". The term premature convergence means that the EA converges into some local optima rather than the global one. EAs allow explicit control over the ratio of exploitation to exploration by varying the parameters of the corresponding genetic operators. In contrast, traditional optimisation techniques have

fixed degrees of exploitation and exploration and consequently are unable of performing a flexible search.

2.4.11 Advantages and Disadvantages of EAs

This section outlines some of the advantages and disadvantages of EAs [11, 12, 27, 47, 71].

Advantages

- Intrinsic search parallelism because of the population-based approach.
- Stochastic transition rules to move within the search space that allow relatively easy avoidance of local optima, and fast recovery from poor initialisations.
- Potential to generate several alternative solutions to the target problem.
- Fast recovery from poor decisions using the reservoir of knowledge acquired through the collective learning of a population of individuals.
- No presumptions about the problem space, e.g. gradient information.
- No requirements for auxiliary information, except the fitness function. Problem specific information can be utilised to speed up the EA.
- Flexibility in balancing exploration and exploitation.
- Straightforward parallel implementation, and flexible hybridisation.
- Low development and application cost in a wide variety of problems.
- Interpretable solutions, unlike for instance neural networks.

Disadvantages

- No guarantee of finding the optimal solution within finite time.
- Computationally expensive due to the population-based search.

- Fine-tuning several EA-related parameters may not be always trivial.
- Avoiding premature convergence requires a good choice for the selection pressure. This in turn, may require preliminary experiments to set the parameters before solving the problem.
- The conventional genetic operators are "blind", producing variations at random without any attempt to optimise the fitness of the new individual(s).

2.5 Clustering with Evolutionary Algorithms

There have been many attempts to use EAs in clustering applications such as [34, 38, 46, 49, 66, 88, 87, 90, 95]. Three recently compiled textbooks [22, 38, 43] provide a brief introduction to EA-based clustering. Since in most clustering problems some criterion function is to be optimised, EAs can be used to avoid local optima and poor initialisations. This section surveys in detail the most prominent methods to apply EAs for clustering problems.

2.5.1 Centroid-Based Representation Schemes

In principle, the centroid-based EAs encode and evolve the coordinates of the cluster centers. This means that the length of the individuals is proportional to the dimensionality of the problem and not to the number of records, as opposed to partitioning-based encoding schemes (see section 2.5.3).

Optimal Centroid-Seed Selection

In an early hybrid approach proposed in [9], a GA encoding the set of centroids optimised the selection of the initial centroid seeds and then a standard k-means algorithm was applied to find the final partitions.

Genetically Guided Algorithm - GGA

In GGA [49] the fixed-length individuals represent the coordinates of the centers of the k desired clusters. Both real-valued and binary representation was used, but no clear advantage to either approach is observed. The minimisation of the total within cluster variation (square-error criterion) serves as fitness function for both Hard K-Means (or HKM) and Fuzzy C-Means (or FCM). The fitness function includes terms to penalise degenerate solutions - individuals with empty clusters. The results indicate that GGA provides almost identical data partition that standard fuzzy or hard k-means algorithms will generate when the latter are given the best possible initialisation. Additionally, GGA avoids local extrema and has minimal sensitivity to initialisation. However, the execution time of GGA can take up to two orders of magnitude more than standard FCM/HKM.

Genetic C-Means Clustering Algorithm - GCMA

GCMA [90] is a hybrid approach that combines a GA with the classical HKM to solve colour image quantisation problems. Each individual represents the coordinates of the cluster centers using integer or real-valued coding while the total within-square-error criterion serves as fitness function. Task specific variation operators, e.g. mutation and recombination, create offspring as in conventional EAs. However, before the evaluation stage all individuals are forced into local optima by applying the standard k-means algorithm in the partition encoded by each individual for a small number of iterations. To improve scalability the k-means algorithm runs against a small randomly selected subset of the dataset. As expected the hybrid EA is less sensitive to random initialisation than the HKM.

HYBRID-GA

HYBRID-GA [34] is one of the most comprehensive studies regarding the hybridisation of GAs with other hill-climbing techniques for clustering. Initially the author presents in a unifying framework a careful analysis of the time requirements of several subtasks of traditional relocation techniques, e.g. HKM, FCM. HYBRID-GA is a tightly coupled approach rather than a serialisation of iterative methods with GAs. HYBRID-GA employs a modified version of the GGA algorithm described earlier in this section for a small number of generations, i.e. 50. Instead of using GGA's mutation operator, HYBRID-GA applies with probability 1/3 to each new individual two standard iterations of FCM/HKM. Finally, the best individual is used to initialise a standard FCM/HKM and obtain the final partition. Experimental results show that HYBRID-GA performs better than the standard FCM/HKM avoiding easily local optima. Additionally, HYBRID-GA outperforms GGA in execution time because it is hybridised with iterative methods which are relatively fast in finding local optima.

Real Coded Genetic Algorithm - RCGA

RCGA [19] represents the cluster centroids as floating-point numbers, thereby enabling the exploration of large domains without sacrificing precision. RCGA

attempts to optimise the FCM criterion function using genetic operators that are appropriate for non-binary strings. The real-valued solution vector is mutated using Michalewicz's [71] non-uniform mutation operator, which makes a uniform search of the space initially, and a very local search at the later stages.

Self-Adaptive Genetic Algorithm - SAGA

Fine tuning the EA-related parameters may be a time consuming task. Since little in the way of universal settings exist except in well-prescribed sub-domains, many EA studies incorporate self-adaptation of EA-related parameters. In the SAGA system [65], in addition to cluster centroids, strategy parameters such as the crossover method [39] and mutation probability are also encoded. The evolutionary optimisation is extended to the vector of strategy parameters to achieve good quality results with minimal effort for parameter tuning.

2.5.2 Medoid-Based Representation Schemes

The search space for the k medoids partitioning problem is significantly smaller than its counterpart k-means problem because only existing data points are valid candidates to represent clusters. The use of medoid encoding is more robust against outliers compared to centroid-based schemes in the same way that medians are more robust against outliers than the arithmetic mean value [43].

In [35] the set of k medoids is encoded as a string of integer-valued genes, where each gene is a distinct number in the range 1,...,N, where N denotes the number of objects in the dataset. To prevent the generation of infeasible solutions - individuals containing genes with repeated values, the algorithm employs a special type of recombination that tends to preserve medoids numbers occurring in both parents in the resultant offspring. Each gene is mutated with a very small probability into a new randomly generated value drawn from 1,...,N, with the constraint than no other gene in the same chromosome has the same value. Several interesting crossover and mutation operators to assure the generation of non-lethal individuals in the context of the k-medoids problem have been also proposed in [21]. Finally, successful hybridisations between EAs and hill-climbing techniques for k-medoids clustering have been reported in the literature, e.g. [36].

2.5.3 Partitioning-Based Representation Schemes

Partitioning-based variants use the so-called *string-of-group-numbers* encoding, where each gene in the string corresponds to a data point while its integer value represents the cluster to which the data point belongs. This approach suffers from severe scalability problems because the length of the individuals is equal to the number of records in the dataset. Some typical examples are [66, 70, 73].

Genetic K-means Algorithm - GKA

GKA [66], a hybrid approach, combines the robust nature of the genetic algorithm with the high performance of the k-means algorithm. Each gene corresponds to a data point while its integer value represents the cluster to which the data point belongs. The minimisation of the total within cluster variation guides the search. The novelty in this study is to replace the crossover operator used in conventional EAs with one step of a standard k-means algorithm. Additionally, a task-specific mutation operator acts as a generator of biased random perturbations by altering the value of a gene in a way that the probability of changing an allele value to a cluster number is more if the corresponding cluster center is closer to the data point. The use of elaborate crossover and mutation operators improves the speed of convergence significantly, while helping avoid local optima. However, encoding the set of points instead of the cluster centroids is infeasible for large datasets.

2.5.4 Geometric-Based Representation Schemes

Another approach to clustering with EAs is to search for cluster descriptors, e.g. hyper- ellipses or boxes rather than representative points. Approximating data distributions with simple geometric primitives provides undoubtedly more comprehensible descriptions compared to point-based cluster representations.

Hyper-Ellipses

In an early attempt described in [95], each variable-length chromosome contains a set of ellipsoid-shaped cluster descriptors that may overlap. An ellipsoid is completely defined by its geometric parameters, namely, its origin, length of its

axes, and its orientation with respect to the axes of reference. These parameters are encoded into a fixed-length binary sub-string that is treated as a complete sub-solution to the clustering problem. Task-specific operators stochastically manipulate the genetic material in a way that the resultant solutions contain always a feasible set of ellipsoids whose number is dynamically determined on the fly. The latter property eliminates the restriction of specifying the number of clusters *a priori*. The fitness function is rather complex reflecting in essence the extent to which the generated ellipsoids correctly classify the given training samples. A major limitation of the proposed method is that it requires a training sample with known cluster labelling to compute the fitness of the individuals.

Hyper-Boxes

In [46] an EP clustering algorithm evolves a variable number of two-dimensional hyper-boxes in light of a minimum coding criterion. Each hyper-box is represented by five parameters, (x,y) coordinates of the center, width, height and the rotation angle. Each solution also incorporates five self-adapted strategy parameters that control the variance of mutation for the hyper-box parameters. Self-adapted parameters specifying the number of hyper-boxes and how often a box is added or deleted are also included. Each parent produces a single offspring by adding a Gaussian random variable to the parent's hyper-box parameters. The fitness function uses a minimum description length (MDL) principle such that a minimum coding for a given dataset would be obtained. Despite the encouraging results in two-dimensional datasets this method has not been evaluated on large scale clustering problems.

2.5.5 Graph-Based Representation Schemes

The authors in [78] proposed an alternative representation scheme where the clustering problem is cast as a graph-partitioning problem. In particular, each vertex (or node) in the graph represents a data point, and there exist an edge between two vertices if the data points corresponding to these vertices belong to the same cluster. The objective is to use a GA to find connected subgraphs that

represent clusters. Each candidate solution encodes a set of N (N: number of data points) integer values where each data point has a corresponding position in the chromosome. The alleles (possible values of a gene) are not the cluster labels, but the indices of other data points. If the ith position in the chromosome has value j, then there is a link in the graph connecting the vertices that correspond to ith and jth data points. To reduce search combinations, the values for each position are limited to the k-nearest neighbours of each data point, where k is an input parameter. An advantage is that the number of clusters does not have to be specified in advance. The algorithm is not scalable as the length of each candidate solution is equal to the number of data points in the dataset, while one must determine the nearest neighbours of all the data points. Computing the nearest neighbours for massive and high dimensional datasets is a prohibitively expensive task. Perhaps the most serious drawback of the algorithm is that its output, i.e. graph, is of very limited usefulness as it is difficult to be interpreted.

2.5.6 Advantages and Disadvantages of EA-Clustering

In short, the advantages and disadvantages of EAs for clustering are:

- **Advantages**
  - Clustering algorithms attempting to optimise a local criterion function suffer from entrapments into local optima, e.g. k-means. It has been shown, e.g. [49, 66], that the approach to optimising local criterion functions with EAs offers avoidance of local optima and minimal sensitivity to the initialisation phase. This is attributed to the global nature of EA search.
- **Disadvantages**
  - Conventional clustering techniques operating on a single solution at a time are relatively fast in finding local optima. In contrast, EA-based approaches to optimising local heuristics for clustering are significantly slower than their conventional counterparts because of the population

based search. Existing EA-based clustering approaches have been evaluated only on datasets with a limited number of data points of very low dimensionality. [34, 49].

- Although hybridising EAs with local iterative methods may improve scalability, it is not clear how to allocate the computing time: should we use many generations of the EA and a few iterations of the local methods, or run the EAs for a few generations and use the local methods to improve the solutions considerably?

2.6 Summary

This chapter gave the computational background to the thesis - a detailed survey of DM clustering techniques, and a brief discussion regarding the foundations of EAs. In particular, the chapter has motivated the use of intelligent DM techniques to semi-automate the discovery of hidden nuggets of knowledge in large databases. The challenges for DM clustering in high dimensional spaces were outlined. This chapter has surveyed several approaches towards clustering for data mining (DM) applications. The severe impact of the curse of dimensionality phenomenon on both the efficiency and effectiveness of clustering techniques was also discussed.

The literature survey led the author to believe that current clustering techniques do not address adequately all requirements for DM clustering, although considerable work has been done in addressing each requirement separately. It has also established the aims of the research in this thesis: *to develop a generic and robust EA clustering methodology that meets the key criteria for DM clustering*. The work described in the following chapters aims to address this challenge.

Part II

System Design and Implementation

Chapter 3

Analysing Univariate Histograms

Capsule

In this Chapter a novel statistical methodology, called *$\mathcal{U}$niform-region $\mathcal{D}$iscovery $\mathcal{A}$lgorithm* (or $\mathcal{UDA}$), is proposed to analyse smooth univariate density histograms. The goal of $\mathcal{UDA}$ is to identify axis-parallel cutting planes from univariate density histograms that produce cleanly separable regions of flat quasi-uniform (or $\mathcal{U}$-region) data distribution. A $\mathcal{U}$-region is defined as a set of contiguous bins whose histogram values exhibit only a marginal variation. $\mathcal{UDA}$ combines standard histogram smoothing techniques (i.e. Kernel Density Estimation) with new heuristics that perform a fine localised analysis of the data distribution. $\mathcal{UDA}$ is exceptionally resistant to noise and local data artifacts. The $\mathcal{U}$-regions identified by $\mathcal{UDA}$ are extensively used for both data *quantisation* (Chapter 4) and *clustering* (Chapter 5).

3.1 Introduction

Recent DM research suggests that to tackle the curse of dimensionality (section 2.3.2), clustering for high dimensional datasets should involve searching for "hidden" subspaces with lower dimensionalities, in which relatively tight cluster structures can be observed when the data are projected onto the subspaces [5, 7, 23, 72, 74]. Discovering such inter-attribute correlations and location of the corresponding clusters is known as the *projective clustering problem*. Flat

or uni-modal regions along density histograms help to generate cluster "signatures" ($\mathcal{U}$-regions) approximating densely populated regions in some subspaces. Hence, projected clusters and their corresponding subspaces can be recovered by analysing univariate density histograms.

This Chapter presents a novel statistical methodology - *$\mathcal{U}$niform-region $\mathcal{D}$iscovery $\mathcal{A}$lgorithm* (or $\mathcal{UDA}$) - to construct and analyse univariate density histograms for the purposes of data quantisation (Chapter 4) and projective clustering (Chapter 5). $\mathcal{UDA}$ has the ability to identify $\mathcal{U}$-regions of arbitrary density, while it is exceptionally resistant to noise and local data artifacts.

The remainder of this Chapter is structured as follows: Section 3.2 outlines the weaknesses of the classical density and frequency histograms as means to study the density distribution of univariate samples. Section 3.3 motivates the use of Kernel Density Estimation (KDE) techniques for automatic construction of reasonably smooth density histograms, which are subsequently analysed by $\mathcal{UDA}$. An efficient binned KDE technique is also presented in section 3.3. Section 3.4 presents the principles of $\mathcal{UDA}$ consisting of three novel statistical tests that are discussed in sections 3.4.1 - 3.4.3.

3.2 Classical Frequency and Density Histograms

The oldest and most widely used density estimators for a univariate sample are the classical *frequency* and *density histograms* [92, 94, 97]. These are usually formed by dividing the real line into equally sized *intervals* (or *bins*). The frequency histogram is a step function with height for a bin being the number of points contained in that bin. The density histogram is also a step function with height for each bin being the proportion of the sample contained in that bin divided by the width of the bin. Hence, the density histogram is a normalised frequency histogram, and thereby integrates to one.

3.2.1 Limitations of Classical Histograms

While simple, both the classical frequency and density histograms have weaknesses. In particular, both histograms are very sensitive to the choice of bin width and the placement of the bin edges [92, 94, 97]. However, it is the choice of bin width which, primarily, controls the amount of smoothing being applied to the data. Depending on the choice of bin width the histogram gives a different impression of the shape of the density of the data. A small bin width leads to a relatively jagged histogram that under-smooths the local distribution density. It is difficult to study the density distribution using a very jagged histogram due to the presence of many local artifacts. In contrast, large bin widths result in an over-smoothed histogram that may flatten important multi-modal density structures.

It is essential to construct histograms that allow both the detection of significant differences in density and that have smoothed out local data artifacts. The *Kernel Density Estimator* (*KDE*) - a nonparametric technique for density estimation - is insensitive to the placement of the bin edges and automatically creates a reasonably smoothed approximation of the real density. The thesis makes extensive use of univariate KDE techniques to construct smoothed histograms for the purposes of analysing the data distribution in the full-dimensional space.

3.3 Kernel Density Estimation - *KDE*

This section describes the *Kernel Density Estimation* (*KDE*) method that is used to automatically create a reasonably smooth approximation of the real density.

3.3.1 Kernel Smoothing

Kernel Density Estimation is a nonparametric technique (given a suitable assumption for the data distribution, e.g. normality assumption in page 65) for density estimation in which a known density function, the *kernel*, is averaged across the observed data points to create a smooth approximation of the real

density. Given n observations $X_1, ..., X_n$ the kernel density estimation at a point x can be thought of as being obtained by "*...spreading a probability mass of size 1/n associated with each data point about its neighbourhood...*" [97] by centring a scaled kernel function, usually termed as "bump", at each observation and then summing the n kernel ordinates at that point. Combining contributions from each data point means that in regions where there are many observations, and it is expected that the true density has a relatively large value, the kernel estimate should also assume a relatively large value [97]. The opposite should occur in regions where there are relatively few observations.

The shape of the bump is determined by a mathematical function called *kernel* K, which is usually chosen to be a unimodal probability density function (*pdf*) that is symmetric about zero and integrates to one. The spread of the kernel is determined by a *window-* or *band-width*, h, that is analogous to the bin width of a classical density histogram. A detailed discussion for KDE can be found elsewhere [92, 94, 97]. The kernel density estimate at point x is given by

$$\boxed{f(x,h) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right)} \tag{3.3}$$

Various types of kernels have been proposed in the literature [92, 94, 97]. Table 3.1 illustrates some commonly used symmetric kernel functions.

Normal $K(\upsilon) = \frac{1}{\sqrt{2\pi}} e^{\frac{-\upsilon^2}{2}}$, $\upsilon \in \mathbb{R}$	**Epanechnikov** $K(\upsilon) = \begin{cases} \frac{3}{4}(1-\upsilon^2), \text{ if } \lvert\upsilon\rvert < 1 \\ 0, \ otherwise \end{cases}$
Triangular $K(\upsilon) = \begin{cases} 1-\lvert\upsilon\rvert, \ if \lvert\upsilon\rvert < 1 \\ 0, \ otherwise \end{cases}$	**Biweight** $K(\upsilon) = \begin{cases} \frac{15}{16}(1-\upsilon^2)^2, \text{ if } \lvert\upsilon\rvert < 1 \\ 0, \ otherwise \end{cases}$
Rectangular or Uniform $K(\upsilon) = \begin{cases} \frac{1}{2}, \ if \lvert\upsilon\rvert < 1 \\ 0, \ otherwise \end{cases}$	**Triweight** $K(\upsilon) = \begin{cases} \frac{35}{32}(1-\upsilon^2)^3, \text{ if } \lvert\upsilon\rvert < 1 \\ 0, \ otherwise \end{cases}$

Table 3.1: Common Kernel Functions

It has been widely recognised that the shape of the kernel function is not particularly crucial to the quality of the density estimate. It is the choice of the bandwidth h that primarily determines its statistical performance since it controls the amount of smoothness in the estimate of the density function [92, 94, 97].

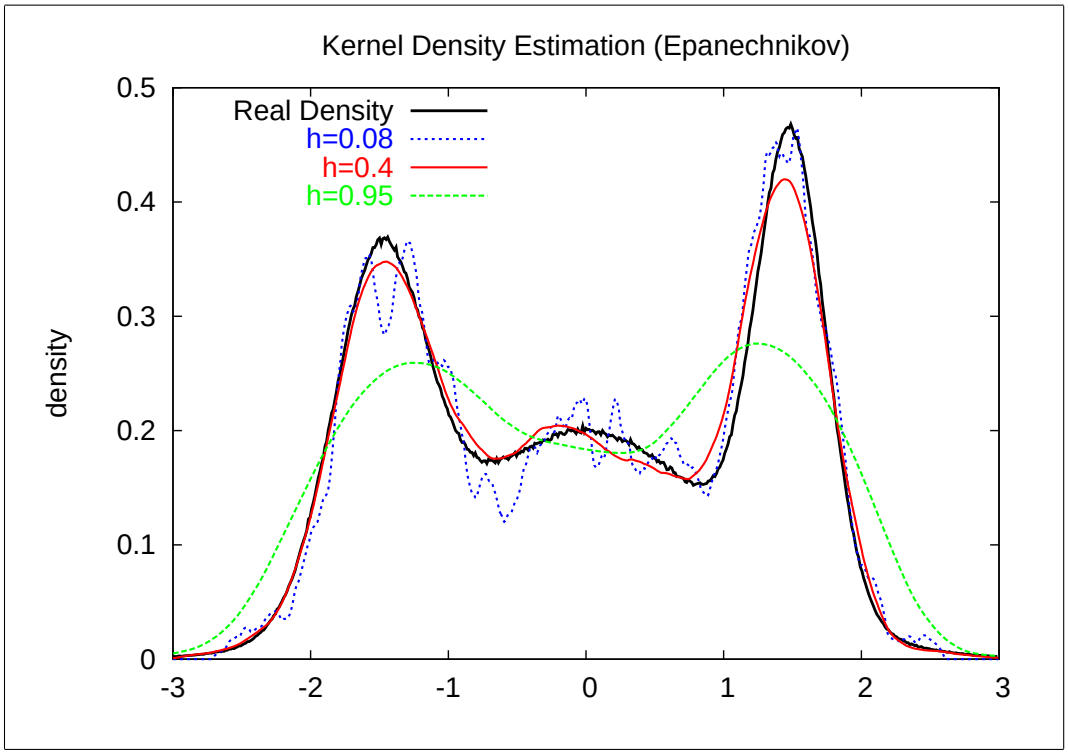


Figure 3.6: *Epanechnikov KDE* for Various Smoothing Parameters

Figure 3.6 shows three kernel density estimates based on a sample of size $n = 1000$ from a density that is a mixture of three Gaussian distributions $N(0, 1)$, $N(-\frac{3}{2}, (\frac{1}{3})^2)$ and $N(\frac{3}{2}, (\frac{1}{4})^2)$ with probabilities $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{4}$, respectively. The *Epanechnikov* kernel has been used to construct the estimates. If h is chosen too small (e.g. h=0.08) then spurious fine structures become visible, while if h is too large (e.g. h=0.95) then the trimodal nature of the distribution is obscured. Evidently, in this particular example the value h=0.4 provides a good compromise since both the essential structure of the distribution has been recovered while most of the local noise has been smoothed away.

3.3.2 Automatic Bandwidth Selection

The problem of choosing an appropriate smoothing level is of great importance in density estimation. A naive approach would entail considering several density estimates obtained over a range of bandwidths and selecting subjectively by eye the most satisfactory estimate. However, when density estimation is to be used routinely in large-scale problems then an automatic fast procedure is essential.

The Normal Scale Bandwidth Rule

A popular method for automatic selection of h, is the oversmoothing of the *normal scale bandwidth rule*, which computes the optimal bandwidth for a normal density with the same scale as the underlying density that is to be estimated. In particular, if σ denotes the standard deviation of the data then h is given by [92]:

$$\boxed{h = 1.144\ \sigma\ n^{-1/5}} \tag{3.4}$$

Formulae 3.4 suggests a nonparametric way of computing the optimal bandwidth to be used with the normal kernel (table 3.1), and is expected to produce reasonable smoothing when the data distribution is close to normal. As Scott noted, if h_1, h_2 are optimal bandwidths for kernels K_1 and K_2, respectively, then one can switch between different kernels by scaling according to the standard deviations σ_{K_1} and σ_{K_2} [92]:

$$\boxed{h_2 \approx \left(\frac{\sigma_{K_1}}{\sigma_{K_2}}\right) h_1} \tag{3.5}$$

Therefore, one can easily compute the optimal bandwidth for a family of kernels e.g. biweight, Epanechnikov, using the normal scale bandwidth rule as starting point and then rescaling the obtained bandwidth according to equation 3.5. Table 3.2 summarises the factors for equivalent smoothing among a family of popular kernels.

From - To	Normal	Uniform	Epanc.	Triangle	Biweight	Triweight
Normal	1	1.740	2.214	2.432	2.623	2.978
Uniform	0.575	1	1.272	1.398	1.507	1.711
Epanec.	0.425	0.786	1	1.099	1.185	1.345
Triangle	0.411	0.715	0.910	1	1.078	1.225
Biweight	0.381	0.663	0.844	0.927	1	1.136
Triweight	0.336	0.584	0.743	0.817	0.881	1

Table 3.2: Factors for Equivalent Smoothing Among Kernels [92]

For departures from normality, e.g. multimodal or heavily skewed density distributions, a global bandwidth approach such as the normal scale bandwidth rule, may result in undersmoothing in areas with only sparse observations while oversmoothing in others. To deal with cases where the optimal amount of smoothing

varies across the domain various extensions of the basic KDE have been proposed in the literature [92, 94, 97]. These methods either use a broader kernel over observations located in sparse regions (i.e. *variable kernel density estimator*) or employ a different bandwidth at each point where the density is estimated (i.e. *local kernel density estimator*). Both methods adapt the amount of smoothing to the sparseness of the data by varying the bandwidth inversely with the real density. Despite their increased flexibility, these extensions are prohibitively expensive for large and high-dimensional datasets, simply because one must precompute multiple bandwidths.

3.3.3 Binned Kernel Density Estimation

Motivation for Binned KDE

For moderate-to-large size samples or procedures involving a substantial number of density estimations, e.g. massive and high-dimensional datasets, the direct use of the basic KDE of section 3.3.1 is very inefficient [94, 97]. Consider, for example, the problem of obtaining a kernel density estimate over a mesh of M grid points, $g_1, ..., g_M$. Indeed, given a set of n observations, computing the KDE over the mesh of M points would require $O(nM)$ kernel evaluations [97]. This number can be much reduced if one uses kernels with compact (confined) support so that some data points fall outside the support of K. The support of a kernel is the interval where the kernel function is nonzero. But then one also needs to perform a test to see if this is the case. With binning, however, the computational order is reduced to only $O(M)$ resulting in an immense saving [97]. This is because there are only M distinct grid point differences, and therefore by the symmetry of K no more than M kernel evaluations are required. In practice, the approximations are usually reasonable for moderate values of M (e.g. $100<M<500$), while for larger values the approximations and the exact estimates are virtually indistinguishable [97].

Principles of Binned KDE

Let K be a symmetric kernel with finite support confined on [-t, t] (t>0). Additionally, let $[l, u]$ denote the real-valued interval of the problem domain that has been partitioned into m bins of uniform width w.

The goal is to compute a *smoothed* kernel density estimate for all (m) bins. But what resolution must one use for the binned KDE? Following the recommendation that the resolution for binned KDE must be relatively fine [97], the new binning algorithm proceeds by partitioning each one of the original bins (m) into (p+1) disjoint sub-intervals using p equi-spaced splitting sites (1≤p). These splitting sites along with the edges of the original bins define a regular grid consisting of a mesh of M=(m+1+mp) points, that is, l=g_1 < ... < g_M=u with spacing $\left(\frac{w}{p+1}\right)$. Following the recommendation that M should be set to moderate values [97] (e.g M=100) so as to obtain a reasonable approximation of the exact density estimate, p is automatically computed as follows:

$$\boxed{p = \max\left(1, \lceil (M - m - 1)/m \rceil\right)} \tag{3.6}$$

where $\lceil x \rceil$ is the smallest integer that is greater than or equal to x.

The basic idea of binning KDE methods relies on rounding each observation to the nearest point on a regular spaced grid according to a binning rule. In this thesis we use the so-called, *simple binning strategy*, where for each observation the nearest grid point is assigned a unit weight. When binning is completed, each grid point g_i (i=1,...,M) has an associated bin count c_i that is the sum total of all the weights that correspond to sample data points that have been assigned to g_i.

The binned kernel density estimator at the jth grid point is now given by [97]:

$$\boxed{f(j,h) = \frac{1}{nh} \sum_{i=1}^{M} K\left(\frac{g_j - g_i}{h}\right) c_i, \ \text{j=1,...,M}} \tag{3.7}$$

Given that c_i is zero outside $[1, M]$ it follows that equation 3.7 can be rewritten as:

$$\boxed{f(j,h) = \sum_{z=1-M}^{M-1} c_{j-z} k_z, \ j = 1, ..., M} \tag{3.8}$$

where the kernel weight k_z is defined as:

$$\boxed{k_z = \frac{1}{nh} K\left(\left(\frac{1}{h}\right) \frac{(u-l)z}{M-1}\right), \ z \in \mathbb{Z}} \tag{3.9}$$

The advantage of binning stems essentially from the fact that K is symmetric with finite support that is confined to [-t, t]. Therefore, k_z need to be evaluated *only once*, and only for those values of z where k_z is nonzero, that is $z = 0, .., L$ where

$$\boxed{L = \min(\lfloor th(M-1)/(u-l) \rfloor, M-1)} \tag{3.10}$$

where $\lfloor x \rfloor$ is the largest integer that is smaller than or equal to x.

The final step is to perform the direct convolution of c_z and k_z in $O(M^2)$ time using:

$$\boxed{f(j) = \sum_{z=-L}^{L} c_{j-z} k_z, \;\; j = 1, ..., M} \tag{3.11}$$

Finally, the density estimate for each of the original bins (m) can now be determined by averaging the density estimates of the grid points lying inside that bin including the bin edges.

3.3.4 Binned KDE for Bounded Domains

Often the domain of definition of a density is an interval of the real line. Since the KDE has no knowledge of the boundary, when the unknown density does not vanish in the boundary regions some probability mass associated with data points belonging to these regions may spill outside the boundaries [97]. Therefore, the density estimate obtained will no longer integrate to unity. Various modifications of the basic kernel method have been proposed to ensure that the density estimator performs well both near the boundaries and in the main part of the distribution [92, 94, 97].

In this thesis, the boundary problem is tackled by employing special *boundary kernels* that are a linear multiple of the basic kernel K (formulae 3.3) [97]. Without loss of generality, suppose that the lower (l) and upper (u) bounds of the interval of interest are located at zero and wm (m, w: the number and width of bins, respectively). Additionally, let K be a kernel with support confined to [-1, 1], e.g. biweight, Epanechnikov.

When estimating the density for grid points lying inside the main part of the distribution $(h, wm - h)$ the ordinary binned kernel of section 3.3.3 can be safely

used because, centred on these grid points it does not overspill the boundary. The problem of "losing" a substantial amount of probability mass arises when the KDE is trying to estimate the density for grid points that are located within one bandwidth of the boundaries [0, h) (left) and (wm-h,wm] (right), provided of course that the real density does not vanish in these regions. Suppose that our aim is to estimate the density at grid point g_j=αh (for a definition of g_j see section 3.3.3) such that ($0 \leq \alpha < 1$).

The standard technique for preventing KDE from assigning probability mass outside the kernel support at the left boundary region is to use the following linear multiple of the kernel K [97]:

$$K^L(\upsilon,\alpha) = \begin{cases} \frac{\nu_{2,\alpha}(K) - \nu_{1,\alpha}(K)\upsilon}{\nu_{0,\alpha}(K)\nu_{2,\alpha}(K) - \nu_{1,\alpha}(K)^2} K(\upsilon) \text{ if } (-1 < \upsilon < \alpha) \\ 0 \; otherwise \end{cases} \tag{3.12}$$

where, $\upsilon = \frac{g_j - g_i}{h}$ and $\nu_{r,\alpha}(K) = \int_{-1}^{\alpha} x^r K(x) dx$.

For instance, one can determine the value of the integral $\nu_{r,\alpha}(K)$ for the Epanechnikov kernel as a function of α as follows:

$\nu_{0,\alpha}(K)$	$\nu_{1,\alpha}(K)$	$\nu_{2,\alpha}(K)$
$\frac{3}{4}\alpha + \frac{1}{2} - \frac{1}{4}\alpha^3$	$-\frac{3}{16}\alpha^4 - \frac{3}{16} + \frac{3}{8}\alpha^2$	$-\frac{3}{20}\alpha^5 + \frac{1}{10} + \frac{1}{4}\alpha^5$

(3.13)

After the boundary correction the binned kernel density estimator at g_j=αh ($0 \leq \alpha < 1$) point is [97]:

$$f(j,h,\alpha) = \frac{1}{nh} \sum_{i=1}^{M} K^L(\upsilon,\alpha) c_i, \text{ j=1,...,M} \tag{3.14}$$

where, $\upsilon = \frac{g_j - g_i}{h}$.

The derivation of the kernel density estimate for the right boundary is the dual of the procedure described above.

3.3.5 KDE for Frequency Histogram Smoothing

The binned KDE with boundary correction described in section 3.3.4 yields a smoothed density histogram. One can easily obtain a smoothed frequency histogram by multiplying the kernel density estimate by the factor (nh), where h and n denote the smoothing bandwidth and sample size respectively.

3.4 $\mathcal{U}$niform-region $\mathcal{D}$iscovery $\mathcal{A}$lgorithm - $\mathcal{UDA}$

This section presents the *$\mathcal{U}$niform-region $\mathcal{D}$iscovery $\mathcal{A}$lgorithm* (or $\mathcal{UDA}$), a novel statistical methodology that is used to analyse the density of data points in smoothed univariate histograms. $\mathcal{UDA}$ identifies $\mathcal{U}$-regions in uni-dimensional frequency histograms. A $\mathcal{U}$-region is defined as a set of contiguous bins with small histogram value variation. $\mathcal{UDA}$ consists of three elaborate statistical tests that are discussed in detail in sections 3.4.1 - 3.4.3. $\mathcal{UDA}$ is extensively used for both data *quantisation* (Chapter 4) and *clustering* (Chapter 5). Henceforth, the terms density and frequency histogram are used interchangeably to refer to a smoothed frequency histogram obtained using the KDE method of section 3.3.5, unless otherwise stated. Additionally, in the remainder of this Chapter a candidate rule $\mathcal{R}$ can be viewed as an axis-parallel hyper-rectangular region being a proper subset of the feature space $\mathcal{F}$ (section 2.3.1).

3.4.1 The First Homogeneity Test - $\mathcal{HT}_1$

Motivation Behind $\mathcal{HT}_1$

The *first homogeneity test* ($\mathcal{HT}_1$) attempts to discriminate *well-separated clusters*, and to distinguish clusters from the surrounding noise. Two neighbouring clusters are well-separated if their densities differ significantly or there is a noise region between them whose density is considerably lower compared to both cluster regions. In other words, the goal of $\mathcal{HT}_1$ is to identify areas inside the feature space $\mathcal{F}$ where the density of points does not change considerably.

$\mathcal{HT}_1$ identifies axis-parallel cutting planes, i.e. splitting sites, from univariate density histograms that produce cleanly separable regions of quasi-uniform data distribution. Given a smoothed density histogram, a quasi-uniform region, or $\mathcal{U}$-region, is defined as a set of contiguous bins whose histogram values exhibit only a marginal variation.

For instance, consider the density histograms of the candidate rule $\mathcal{R}$, as shown in figure 3.7. Undoubtedly, $\mathcal{R}$ is non-homogeneous as it encloses four distinct clusters, in addition to noise. Examination of the horizontal-axis histogram

reveals that there are six well-separated $\mathcal{U}$-regions in that dimension. Furthermore, it can be easily observed that high quality cutting planes, plotted in figure 3.7 as bold dashed lines, pass through the boundaries of $\mathcal{U}$-regions.

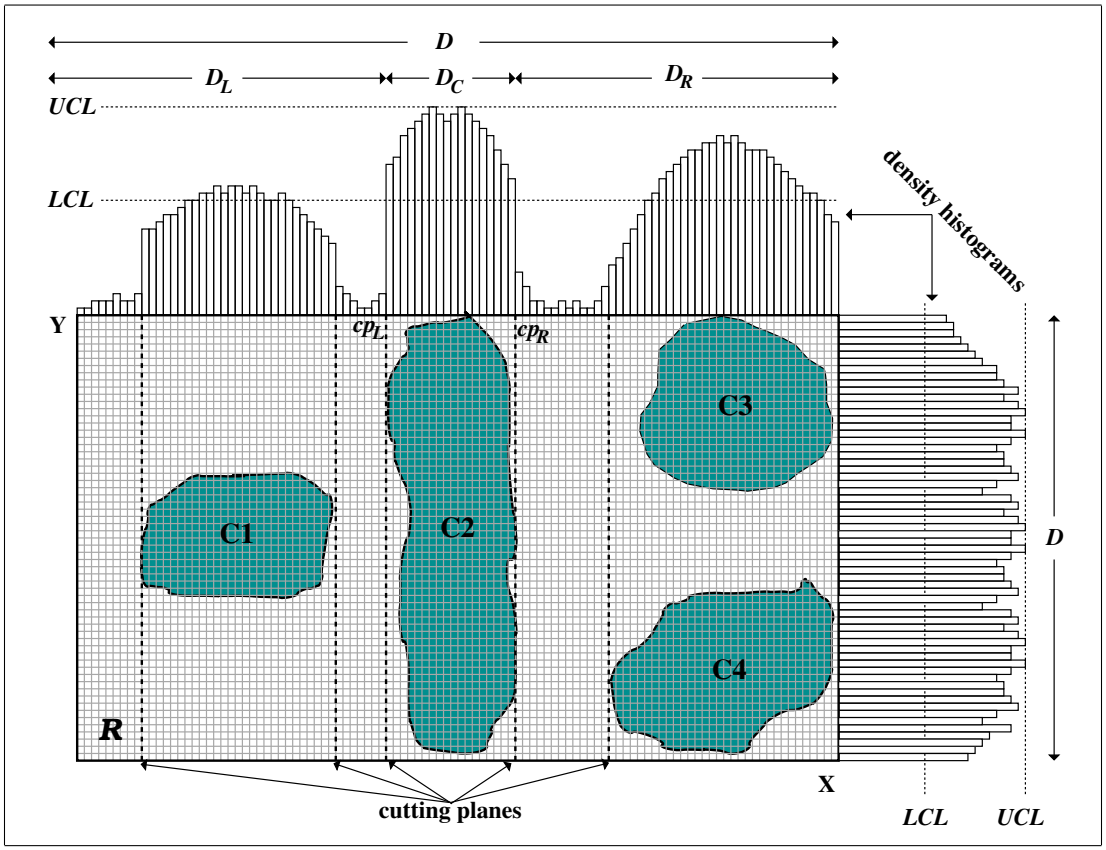


Figure 3.7: Motivation Behind the First Homogeneity Test - $\mathcal{HT}_1$

Principles of $\mathcal{HT}_1$ Algorithm

As mentioned above, $\mathcal{HT}_1$ seeks axis-aligned cutting planes partitioning the original histogram into disjoint $\mathcal{U}$-regions. In this thesis, a *high quality* cutting plane passes through a bin (*valley*) whose density is significantly lower compared to the histogram value of the most densely-populated bin (*peak*) and the valley bin is located at the borders of the region containing the highest peak. A valid cutting plane need not necessarily be surrounded on both sides by bins of significantly-higher density. This is simply because the goal of $\mathcal{HT}_1$ is to identify all $\mathcal{U}$-regions regardless of their density. Notice that the valley with the lowest histogram value may not always be the best splitting point since such an approach is prone to

over-splitting low density $\mathcal{U}$-regions that adjoin $\mathcal{U}$-regions of higher density.

Let us assume that the ith feature-gene (see section 5.3) $[l_{ij}, u_{ij}]$ of jth rule covers k bins (u_{ij}-l_{ij}+1=k). Additionally, let $f_{l_{ij}}, ..., f_{u_{ij}}$ be the k smoothed histogram values, i.e. number of points contained in each bin, that correspond to the ith dimension of jth rule. The partitioning of the histogram into $\mathcal{U}$-regions is performed recursively as follows:

Initially, there is a single region D=$[l_{ij}, u_{ij}]$ comprising all bins. $\mathcal{HT}_1$ proceeds by finding the most densely populated bin inside D, and sets a so-called *Upper Control Limit* (UCL) to the histogram value of the highest peak, UCL=max($f_{l_{ij}}$,..., $f_{u_{ij}}$). The baseline *Lower Control Limit* (LCL), which specifies the splitting density level is then computed as:

$$\boxed{LCL = T_h * UCL} \tag{3.15}$$

where the user-specified *homogeneity* or *uniformity* threshold $T_h \in (0, 1]$ controls the desired degree of uniformity.

If all histogram values exceed LCL then D is temporarily deemed a $\mathcal{U}$-region and kept intact. For instance, $\mathcal{HT}_1$ detects only one $\mathcal{U}$-region along the vertical-axis histogram in figure 3.7. In general, a $\mathcal{U}$-region detected in a uni-dimensional histogram may be the result of the joint projection of multiple clusters. Note that D needs to pass two additional homogeneity tests described in sections 3.4.2 and 3.4.3, to be securely considered as a $\mathcal{U}$-region.

If $\mathcal{HT}_1$ found at least one bin whose histogram value falls bellow LCL, then D is split into three contiguous regions, namely, D_L, D_C, and D_R. This case can be seen along the density histogram that corresponds to the horizontal-axis of rule $\mathcal{R}$ in figure 3.7. Initially, the central segment D_C comprises only one bin with the highest density. D_C is then grown as much as possible in both directions until finding the closest bin (if any) with density less than LCL. Valid cutting planes would pass through the boundaries of such bins. If cp_L and cp_R denote the left and right cutting planes, respectively, then the newly formed regions are D_L=$[l_{ij}, cp_L]$, D_C=$[cp_L$+1, cp_R-1] and D_R=$[cp_R, u_{ij}]$. Depending on the data distribution the boundary regions D_L and D_R may be empty. D_C is

then temporarily deemed a $\mathcal{U}$-region and is not analysed further by $\mathcal{HT}_1$. The above procedure is recursively applied to each newly formed boundary region independently until no more splitting sites occur. In essence, as $\mathcal{HT}_1$ proceeds, $\mathcal{U}$-regions are gradually detected at decreasingly low levels of density.

To suppress the subsequent formation of sparse rules (section 5.3) and to reduce computation, $\mathcal{HT}_1$ instantly discards all sparse regions including those that are $\mathcal{U}$-regions. A region is sparse if the number of points inside that region is less than NT_s, where N: total number of points, T_s: sparsity threshold.

3.4.2 The Second Homogeneity Test - $\mathcal{HT}_2$

Motivation Behind $\mathcal{HT}_2$

Experiments in section 6.5 showed that when the level of noise is relatively low, $\mathcal{HT}_1$ suffices for separating neighbouring clusters of similar density. In particular, it is highly likely that the histogram values of the noise regions between the actual clusters, do not exceed the baseline LCL. In contrast, high levels of noise may significantly hinder the ability of $\mathcal{HT}_1$ to effectively discriminate adjacent clusters of similar density, because the necessary separating valleys between the clusters could potentially be obscured by the increased amount of noise.

To tackle this problem a *second homogeneity test* ($\mathcal{HT}_2$) is independently applied to all $\mathcal{U}$-regions obtained earlier by $\mathcal{HT}_1$. In short, $\mathcal{UT}_2$ consists of a series of chi-square (χ^2) tests at decreasing levels of density to reveal (if any) significant multi-modal histogram structures that potentially indicate the existence of multiple clusters in higher dimensionality spaces.

Principles of $\mathcal{HT}_2$ Algorithm

Let l and u denote the lower and upper bound of a $\mathcal{U}$-region ($[l, u]$), respectively, covering k bins (u-l+1=k). Additionally, let $f_l, f_{l+1}, ..., f_{u-1}, f_u$ be the k smoothed histogram values that correspond to the given $\mathcal{U}$-region, while UCL=max(f_l, f_{l+1},..., f_{u-1}, f_u).

Let $T_{\chi^2} \in (0, UCL)$ denote the current level of density where the χ^2 test is to be applied. Starting from the most densely populated bin inside $[l, u]$, $\mathcal{HT}_2$ finds

the left (l') and right (u') most bins whose histogram values exceed a baseline threshold T_{χ^2}. It then computes the trimmed frequencies $f_{l'}$, $f_{l'+1}$,...,$f_{u'-1}$, $f_{u'}$ of the k'= (u'-l'+1) bins inside [l', u'] ([l', u'] $\subseteq$ [l, u]). More specifically, if the histogram value f_i of the ith bin, $i \in [l', u']$, is greater than T_{χ^2} ($T_{\chi^2} < f_i$) then $f_{i'} = T_{\chi^2}$, otherwise, $f_{i'} = f_i$. The *Goodness of Fit* for a uniform distribution assumes all k' trimmed bins inside $[l', u']$ are expected to have an equal frequency $E_i = \frac{1}{k'} \sum_{i=l'}^{u'} f_{i'}$, $i \in [l', u']$. Formally, $\mathcal{HT}_2$ assesses the uniformity of the trimmed data using a standard χ^2 test [50]:

Null hypothesis H_o	:	The trimmed data in $[l', u']$ follow uniform distribution
Alternative hypothesis H_a	:	The trimmed data are not uniformly distributed
Test Statistic value	:	$\chi^2 = \sum_{i=l'}^{u'} \frac{(O_i - E_i)^2}{E_i}$
Rejection region	:	$\chi^2_{a,k'-1} \leq \chi^2$
where $O_i = f_{i'}$ and E_i are the observed and expected frequency for bin i, respectively		
Significance level $a = 0.05$		

Figure 3.8: The Standard χ^2 Test for Uniformity

Therefore, the null hypothesis (H_o) that the trimmed data is from a quasi-uniform distribution is rejected if $\chi^2 \geq \chi^2_{a,k'-1}$, where $\chi^2_{a,k'-1}$ is the chi-square percent point function with k'-1 degrees of freedom and a significance level of a.

Each $\mathcal{U}$-region discovered by $\mathcal{HT}_1$ undergoes a series of χ^2 tests where T_{χ^2} gradually decreases from $(0.95 * UCL)$ to (LCL) by a fixed factor $(0.05 * UCL)$. If the null hypothesis (H_o) is rejected then $\mathcal{HT}_2$ splits the original $\mathcal{U}$-region [l, u] along its lowest density bin, thereby creating two new partitions. The original $\mathcal{U}$-region is then discarded. Similar to $\mathcal{HT}_1$, $\mathcal{HT}_2$ instantly discards any newly formed region that is sparse, to suppress the subsequent formation of spurious rules and to reduce computation. The χ^2 test is then independently reapplied to each one of the newly formed rules.

If a $\mathcal{U}$-region passes all χ^2 tests then it is deemed a $\mathcal{U}$-region for this stage

and is not further processed by $\mathcal{HT}_2$. Notice that if the χ^2 test at a given density level T_{χ^2} is accepted, and additionally, $l' = l$, and $u' = u$, then the original region is automatically deemed a $\mathcal{U}$-region, without performing the remaining (if any) tests.

An $\mathcal{HT}_2$ Example

Figure 3.9 demonstrates why $\mathcal{HT}_2$ is vital to discriminate neighbouring clusters of relatively similar density under noise conditions. In this example, the candidate rule $\mathcal{R}$ encloses two distinct clusters surrounded by highly noisy regions. Notice that the relative darkness inside $\mathcal{R}$ indicates the density of points. Obviously $\mathcal{HT}_1$ fails to detect the two clusters because in both histograms all density values exceed the corresponding LCL. In contrast, $\mathcal{HT}_2$ is able to detect one cutting plane (bold dashed line) along the horizontal axis.

In particular, when $\mathcal{HT}_2$ applies the χ^2 test at relatively high density levels, e.g. $T_{\chi^2} = 0.8 * UCL$, the corresponding trimmed histogram (grey-shadowed area in figure 3.9(b)), exhibits only a marginal variation, thereby the null uniformity hypothesis (H_o) is accepted. However, at a lower density level, i.e. $T_{\chi^2} = 0.5 * UCL$ (figure 3.9(c)), where the trimmed histogram contains bins belonging to both clusters, the χ^2 test rejects the null hypothesis, thereby, $\mathcal{HT}_2$ detects the cutting plane shown as dashed line that eventually splits $\mathcal{R}$ into two regions.

3.4.3 The Third Homogeneity Test - $\mathcal{HT}_3$

Motivation Behind $\mathcal{HT}_3$

The combination of $\mathcal{HT}_1$ and $\mathcal{HT}_2$ partition the original histogram into regions of either quasi-uniform or uni-modal distribution. Although the first two tests effectively discriminate the clusters from one another, they may fail alone to delineate the boundaries of clusters with accuracy when the level of surrounding noise is high. This is because noise points located near the boundary of a cluster may make the density in the tails of the uni-dimensional histograms exceed the splitting level LCL. As a result, fully repaired rules would be extended far beyond the cluster edges towards the noisy regions.

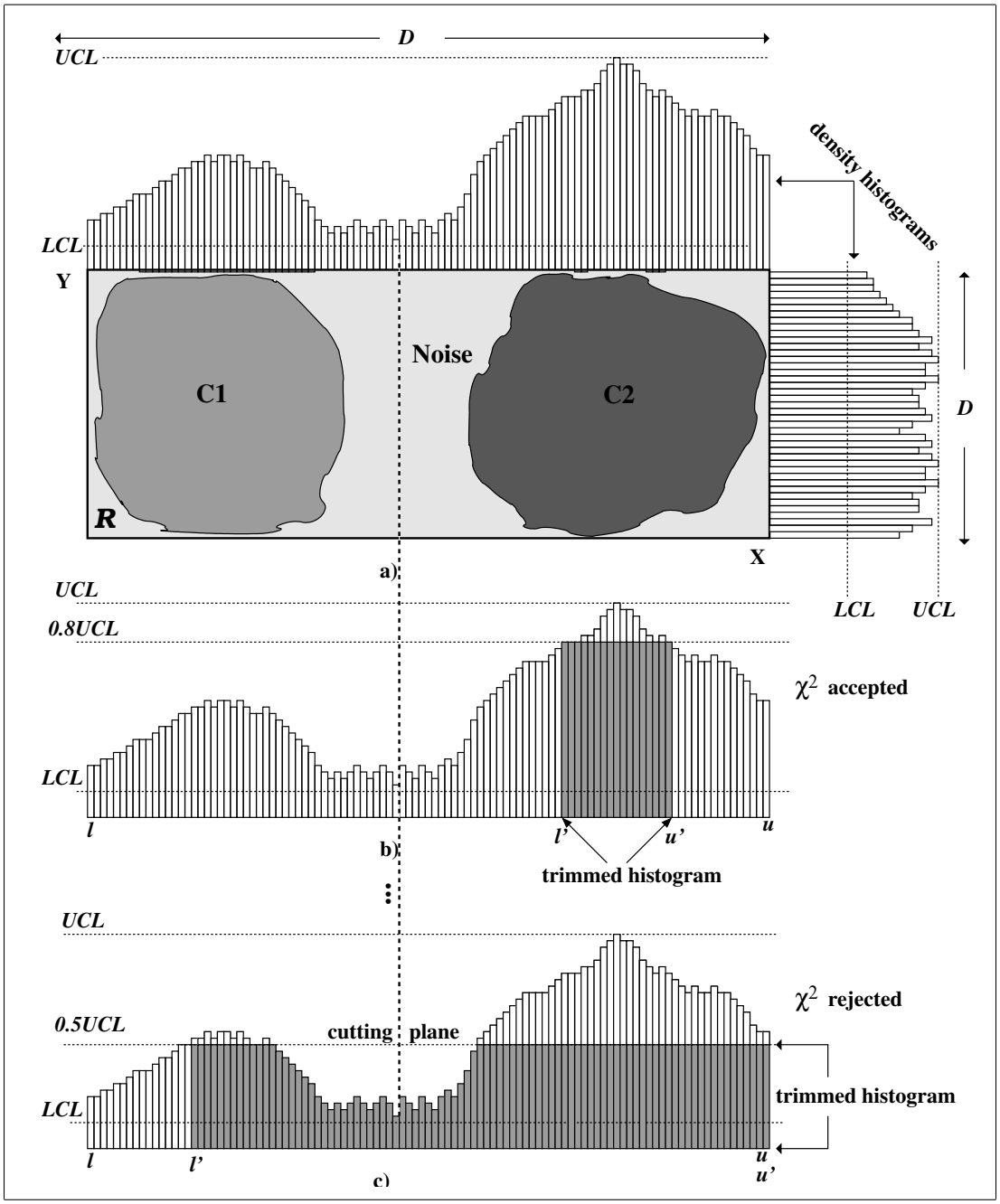


Figure 3.9: The Principles of $\mathcal{HT}_2$ Homogeneity Test

In such a case, there are many bins with relatively low density corresponding to noise, and few higher density bins corresponding to the central part of a cluster. In other words, the distribution of the histogram values is expected to be positively skewed. $\mathcal{UDA}$ tackles this problem using the *third homogeneity test* ($\mathcal{HT}_3$), an elaborate statistical test that examines whether the distribution of the histogram values is significantly skewed in the right tail.

Principles of $\mathcal{HT}_3$ Algorithm

Statistically, right extreme outliers are observations lying $3.0 * IQR$ above the third quartile (Q_3), where IQR=Q_3-Q_1 (Q_1: first quartile) denote the inter-quartile range of the sample [50]. Let l and u denote the lower and upper bound of a $\mathcal{U}$-region ($[l,\ u]$), respectively, covering k bins (u-l+1=k). Additionally, let $f_l, f_{l+1}, ..., f_{u-1}, f_u$ be the k smoothed histogram values that correspond to the given $\mathcal{U}$-region. To decide whether the distribution of the histogram values is significantly skewed in the right tail, $\mathcal{HT}_3$ computes the right outlier fence (Q_3+3.0$*IQR$) for the observations $f_l, f_{l+1}, ..., f_{u-1}, f_u$. If there are no histogram values above the outlier fence the corresponding region is finally deemed a $\mathcal{U}$-region. Otherwise, similar to $\mathcal{HT}_1$, the histogram is split into three regions as follows: Initially, the central region contains only the bin with the highest peak. $\mathcal{HT}_3$ grows the central region as much as possible in both directions until finding a bin whose histogram value falls below the outlier fence. Valid cutting planes would go through the boundaries of the central region. All the newly formed regions undergo $\mathcal{HT}_3$ independently until no more splitting sites occur. Similar to $\mathcal{HT}_1$ and $\mathcal{HT}_2$, $\mathcal{HT}_3$ instantly discards all newly formed sparse regions. Finally, $\mathcal{HT}_3$ is not applied to very low density histograms, e.g. a histogram where the bin with the highest peak contains less than 35 data points [23, 29], because such low density bins reflect local artifacts rather than a statistically significant trend.

An $\mathcal{HT}_3$ Example

Figure 3.10 demonstrates the beneficial effect of $\mathcal{HT}_3$ on the quality of the clustering results. In short, the candidate rule $\mathcal{R}$ remains non-homogeneous even after having successfully passed the first two homogeneity tests, $\mathcal{HT}_1$ and $\mathcal{HT}_2$, in both dimensions. Obviously, although the density histogram for the horizontal axis is characterised by a sharp uni-modal structure, the increased amount of noise makes the density in the surrounding regions exceed the splitting baseline LCL. $\mathcal{HT}_3$, in contrast, is able to separate the tails from the central part of the distribution by exploiting the presence of a few high density bins lying beyond the outlier fence. Therefore, the original rule $\mathcal{R}$ is split along the cutting planes

shown as dashed lines in figure 3.10.

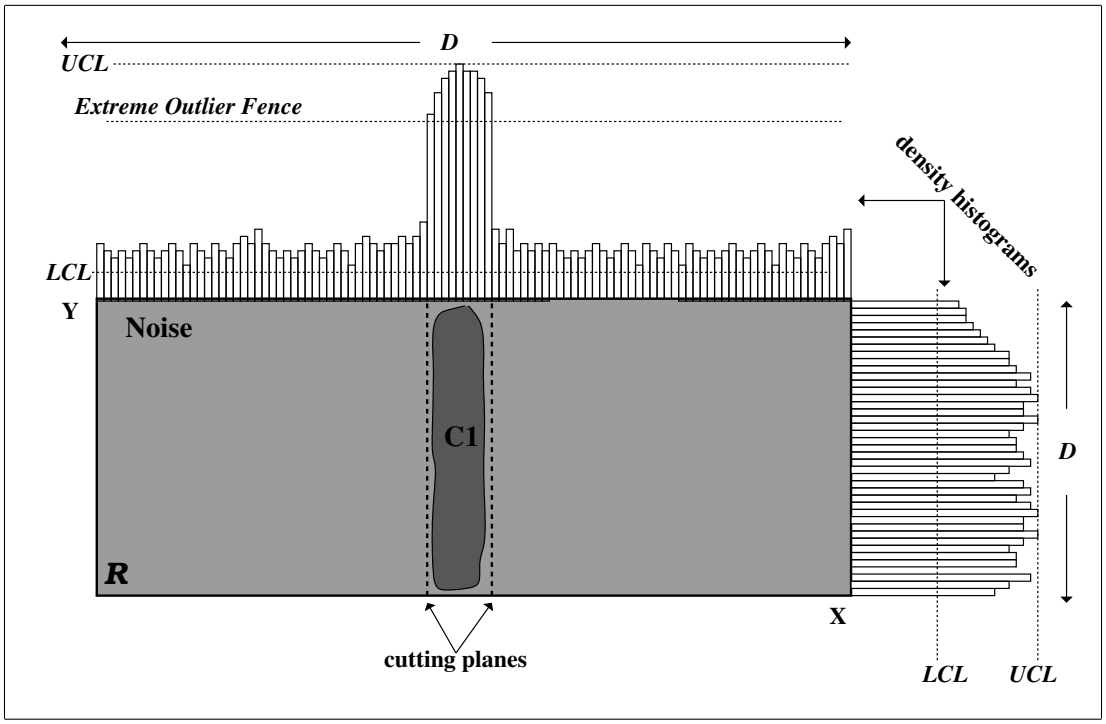


Figure 3.10: The Principles of $\mathcal{HT}_3$ Homogeneity Test

3.5 Summary

In this chapter we have described a novel univariate analysis methodology ($\mathcal{UDA}$) identifying flat density regions ($\mathcal{U}$-regions) in smooth histograms. Neighbouring $\mathcal{U}$-regions co-existing at different density levels are of great importance as they indicate the existence of distinct cluster structures in higher dimensional spaces. Often, univariate $\mathcal{U}$-regions help to generate accurate cluster "signatures", as their boundaries coincide with the edges of the actual clusters. We have shown how $\mathcal{UDA}$ detects such regions using three elaborate statistical tests. Advanced kernel smoothing techniques to construct histograms that allow both the detection of significant differences in density and that have smoothed out local data artifacts, were also discussed. It has been demonstrated that $\mathcal{UDA}$ can discover $\mathcal{U}$-regions of arbitrary density even under very noisy conditions.

$\mathcal{UDA}$ plays a vital role in both data *quantisation* and *clustering* as subsequently explained in chapters 4 and 5, respectively.

Chapter 4

Quantisation of the Data Space

Capsule

In this Chapter a new statistical quantisation algorithm, the $\mathcal{TSQ}$ ($\mathcal{T}$wo $\mathcal{S}$tage $\mathcal{Q}$uantisation), is proposed to support the clustering of large and high dimensional numerical datasets. The quantised data are subsequently analysed by $\mathcal{NOCEA}$ - a novel evolutionary-based clustering algorithm that is described in detail in Chapter 5. In particular, $\mathcal{TSQ}$ imposes a multi-dimensional grid structure onto the data space to reduce the search combinations for $\mathcal{NOCEA}$. The classical trade-off is between computational complexity and resolution, where the latter greatly determines the quality of the clustering results.

$\mathcal{TSQ}$ quantises the dataspace using a novel statistical analysis of unidimensional density histograms. It determines an appropriate grid resolution that enables the discrimination of clusters, while preserving accuracy and acceptable computational cost. It combines standard statistical techniques like Kernel Density Estimation, with new heuristics that reflect the local distribution. Unlike other grid-based techniques, $\mathcal{TSQ}$ has no specific bias toward coarse resolutions, because $\mathcal{NOCEA}$ can operate on relatively fine grids as it attempts to produce highly homogeneous rather than highly dense clusters.

4.1 Introduction

In spite of recent advances in the field of unsupervised learning, high dimensionality continues to pose challenges to clustering algorithms because of the inherent sparsity of the data space and the fact that different types of inter-attribute

correlations may occur in different subsets of dimensions in different localities (neighbourhoods) of the data. Grid-based clustering techniques, e.g. CLIQUE, MAFIA, ENCLUS, WaveCluster, include a pre-processing step, hereafter referred to as *quantisation*, that imposes a multi-dimensional grid structure onto the data space on which all the clustering operations are performed. Summarised information about the data points in each cell is stored, and subsequently exploited by the clustering algorithm.

Quantisation reduces the search combinations by aggregating together points in the same cell, and the classical trade-off is between computational complexity and resolution, where the latter greatly determines the quality of the clustering results. Since multi-dimensional analysis is prohibitively expensive due to the exponential growth in combinatorial optimisation as dimensionality increases, the dataspace is quantised by analysing each dimension *independently*. Unfortunately *any* type of uni-dimensional analysis can only produce a rough approximation of the optimal resolution as it disregards all inter-attribute correlations occurring in higher dimensional spaces. Most grid-based clustering techniques regard clusters as unions of connected high-density cells. To reduce computational complexity while locating adjacent dense cells in high dimensional spaces, *grid-based techniques adopt coarse resolutions at the expense of the accuracy of the clustering results* [7, 23, 74].

Chapter Contribution

In this Chapter a novel $\mathcal{T}$wo $\mathcal{S}$tage $\mathcal{Q}$uantisation ($\mathcal{TSQ}$) algorithm is proposed and implemented, to support the clustering of large and high dimensional datasets. $\mathcal{TSQ}$ prepares the raw data for the new evolutionary-based $\mathcal{NOCEA}$ clustering algorithm described in Chapter 5. $\mathcal{NOCEA}$ performs a grid rather than a continuous-space search. Fine grid resolutions lead to enormous amount of computation, whereas the quality of the clustering results may be substantially degraded using coarse grids. $\mathcal{TSQ}$ strives to compute the maximal required resolution that enables $\mathcal{NOCEA}$ to produce good quality results with an acceptable computational cost. To achieve this the data distribution in each dimension is

thoroughly analysed by sophisticated statistical procedures. The basic idea of $\mathcal{TSQ}$ is to locate uni-dimensional histogram regions with only a marginal variation in density and then apply standard quantisation techniques to these regions so as to derive a locally optimal resolution. A weighted-sum method is then used to scalarise these local resolutions into a uniform global bin width.

Unlike other grid-based techniques that operate on coarse grids so as to locate dense regions, $\mathcal{TSQ}$ is unbiased towards coarse resolutions, because in the quantised space $\mathcal{NOCEA}$ seeks homogeneous regions of arbitrary density.

The remainder of this Chapter is structured as follows: Section 4.2 provides an overview of $\mathcal{TSQ}$. Section 4.3 gives a formal definition of the multi-dimensional grid. Section 4.4 discusses the challenges of quantising high dimensional datasets, and recalls the limitations of previously proposed quantisation methods. Section 4.5 describes in detail the two stages of $\mathcal{TSQ}$. In particular, sections 4.5.1 and 4.5.2 present the gross and detailed statistical analysis of $\mathcal{TSQ}$, respectively.

4.2 An Overview of $\mathcal{TSQ}$

Figure 4.11 shows the two stages of the $\mathcal{TSQ}$ quantisation algorithm. Initially, a *gross statistical analysis* is applied to each dimension independently aiming to determine an appropriate bin width that allows capturing large-scale differences in density at different localities of the dimension. The initial bin width is computed using two important distributional metrics, namely, *dispersion* and *entropy*. Other techniques either ignore the distribution, e.g. CLIQUE, ENCLUS, WaveCluster, or use limited statistical information, e.g. MAFIA. The *detailed statistical analysis* firstly identifies regions in the histogram where the distribution of points is approximately uniform. Each quasi-uniform region undergoes then a localised analysis, which determines an optimal width for this region based on a standard formulae. In contrast, other quantisation techniques, e.g. MAFIA, create a single bin for each quasi-uniform region, which may yield poor quality results if multiple clusters with very different characteristics overlap in this flat-density region.

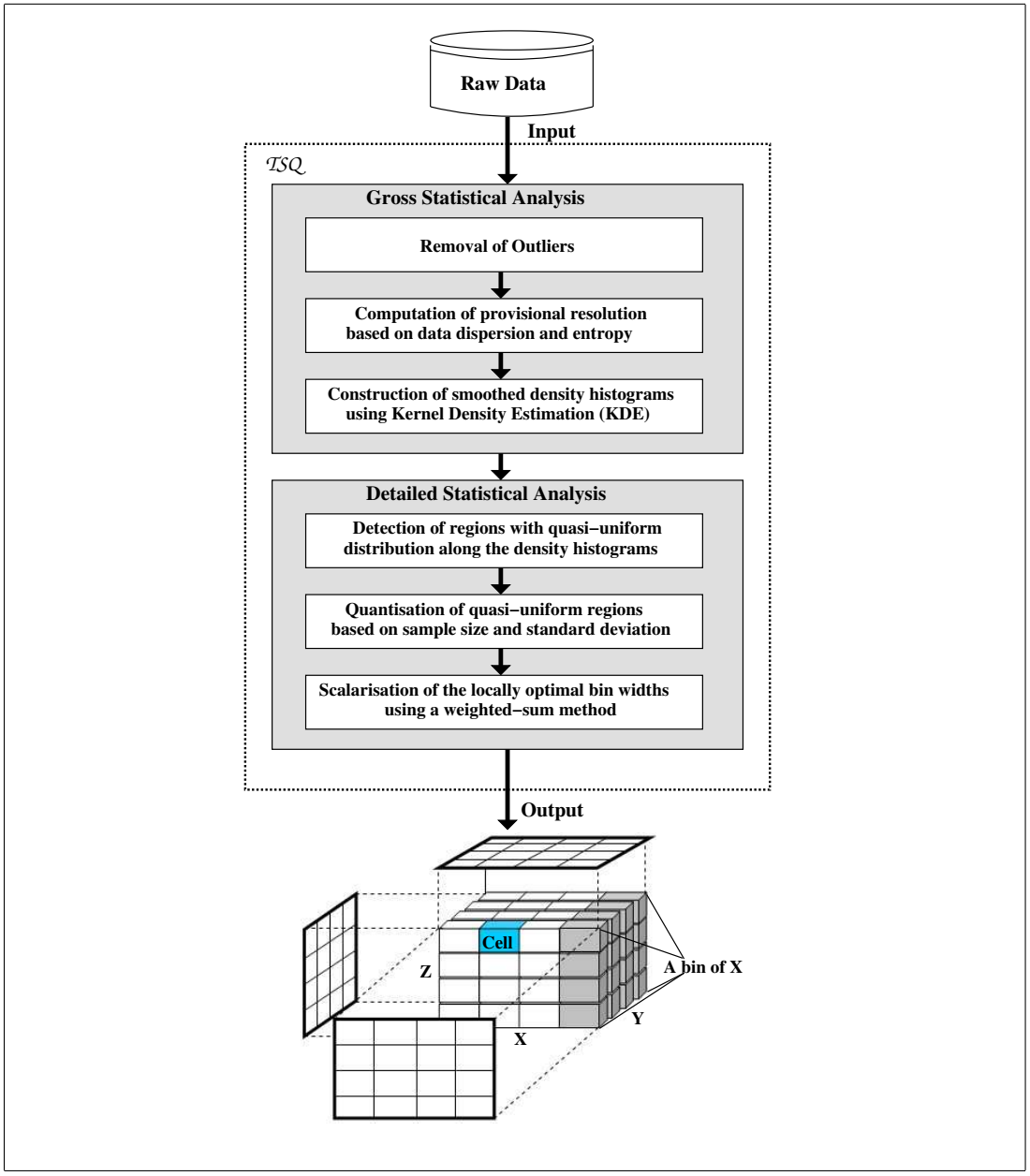


Figure 4.11: $\mathcal{T}$wo $\mathcal{S}$tage $\mathcal{Q}$uantisation Algorithm ($\mathcal{TSQ}$)

Ideally, each such region would keep its own bin width leading to non-uniform grids delineating cluster boundaries more accurately. However, a non-uniform grid adds a substantial amount of extra work when evaluating candidate solutions. Therefore, a weighted-sum method is used to scalarise the set of different widths into a single value by pre-multiplying each width with a specific weight that is proportional to the percentage of points lying inside that region.

4.3 The Multi-Dimensional Grid

Let $\mathcal{F} = \mathcal{A}_1 \times ... \times \mathcal{A}_d$ be the d-dimensional *feature space* as defined formally in section 2.3.1. A multi-dimensional grid structure is imposed into the continuous space $\mathcal{F}$ by partitioning the domain $[a_i, b_i]$, $i = 1, ..., d$ of each dimension $\mathcal{A}_i$, $i = 1, ..., d$ into non-overlapping *intervals* or *bins* of uniform or variable size [51, 52]. Hence, $\mathcal{F}$ is decomposed into a finite set of disjoint rectangular *units* (or *cells*) on which all the clustering operations are performed. An individual cell C is formed by the direct Cartesian product of one interval from each dimension. Formally, a cell C is a subset of the space $\mathcal{F}$ ($C \subseteq F$) and can be geometrically interpreted as an axis-parallel hyper-rectangle, that is, $C = [l_1, r_1) \times ... \times [l_d, r_d)$, where l_i and r_i denote the left and right bounds of C in ith dimension. By definition the jth point $p_j = [p_{j1}, ..., p_{jd}]$, $j = 1, ..., N$, is contained in C if and only if $l_i \leq p_{ji} < r_i$, $\forall\ i = 1, ..., d$. A cell is *dense* if the fraction of the total number of data points contained in the cell exceeds a user-defined sparsity threshold $T_s \in (0, 1]$.

4.4 Quantising High-Dimensional Data

During quantisation there are two important problems that must be carefully addressed, namely *scaling* and *aggregation*. Scaling refers to the selection of an appropriate bin *width* (w) in each dimension while aggregation is the problem of summarising the distribution properties of the points contained in each cell.

4.4.1 Aggregation

The amount and kind of aggregated information heavily depends on the type of clustering algorithm that is used. Usually, each cell is assigned a list of feature-related statistical parameters, which are used for efficient query processing operations [51]. For instance, in CLIQUE and WaveCluster each cell is assigned a label (dense or not dense) based on the number of points lying inside this cell [7, 93]. In STING each cell maintains a list of statistical metrics such as number of points within it, minimum, maximum, mean, standard deviation and

type of distribution of points in the cell [98].

Despite the appealing efficiency of grid-based techniques in low dimensional spaces because of aggregation, moderate-to-high dimensionality outlives its usefulness even for coarse resolutions. In particular, due to the sparsely filled space the vast majority of points map into different cells and there exist many empty cells [55]; aggregation in moderate-to-high dimensional spaces is meaningless. Hence, aggregation in the light of summarising and storing properties of data is not further considered in the thesis.

4.4.2 Scaling

It seems very unlikely that there will ever be either a purely statistical or mathematical solution for the scaling problem [76, 93]. To approximate the optimal grid resolution (determine bin width in each dimension) application domain knowledge can be incorporated but this is rarely complete and consistent.

Often, real-world datasets contain different types of inter-attribute correlations occurring in different subsets of dimensions in different data localities [5, 7]. Ideally, one would first identify all different correlations and then assign appropriate grid resolutions to each region based on the local characteristics of the distribution. Unfortunately, this kind of multi-dimensional analysis is prohibitively expensive due to the exponential growth in computational complexity as dimensionality increases. Consequently, the feature space is quantised by considering each dimension independently. However, uni-dimensional projections *flatten* all local relationships between data points, e.g. inter-attribute correlations, in higher dimensional spaces. Thus, there is no guarantee for optimality in resolution using uni-dimensional statistical analysis.

The choice of the width of the bins (w) in each dimension has an enormous impact on both complexity and quality of the results. For very coarse grids, more points reside in the same cell and thus the probability of assigning to the same cell points belonging to different clusters increases. This problem is called *under-quantisation* and may cause the merging of distinct clusters, which, in turn,

decreases the quality of the clustering output. On the other hand, for very fine resolutions, the data tend to be separated in different cells which may cause the discovery of many unnecessary and very small clusters. This case is known as *over-quantisation*. Over-quantisation decreases the quality of clustering output because it is likely to split relatively compact clusters into a multitude of small clusters. Some of these clusters may be very small and thus can be discarded as noise.

The main goal of uni-dimensional projective analysis is to identify large-scale variations in the density of points along the uni-dimensional orthogonal density histograms [5, 7, 23, 74]. When the point concentration (or density) varies significantly across the real line, i.e. domain, this may indicate the existence of multiple clusters in higher dimensional spaces. However, quantitative information regarding the location, density or type of correlation for each cluster can not be inferred from a univariate histogram because the latter hides inter-attribute correlations.

From a density estimation viewpoint, the choice of the bin width has an enormous effect on the appearance of the resulting histogram. A very fine bin width results in a jagged histogram where most observations are located in distinct bins, whereas a very coarse bin width results in a histogram that does not capture the differences in density in different localities of the dimension. Ideally, the bin width should be chosen so that the histogram displays the essential structure of the distribution, but at the same time maintains significant localities of the distribution. A detailed discussion in bin width selection can be found in [92].

4.4.3 Terrell's Quantisation Rule

Terrell [96] suggested a practical data-based rule for setting the upper bound on bin width for univariate density histograms. In particular, the bin width w should be directly proportional to the standard deviation σ of the univariate sample and inversely proportional to $N^{1/3}$, where N is the size of the sample [92, 96]:

$$\boxed{w \leq 3.729\sigma N^{-1/3}} \tag{4.16}$$

Hence, a naive solution to the scaling problem would be to directly apply equation 4.16 in each dimension independently to derive an over-smoothed estimator of the bin width. While simple, this approach has weaknesses. A way of identifying the limitations of equation 4.16 is to investigate the effect of various factors that are involved in it.

Although w changes at a rate inversely proportional to $N^{1/3}$, this rate is much faster for relatively small datasets, e.g. $N<50000$. However, for moderate-to-large datasets the factor $N^{-1/3}$ exhibits only a marginal variation. Bearing in mind that massive datasets are common in real world DM applications, and w is more sensitive to changes in σ compared to N, it becomes clear why the effect of σ on w deserves careful examination.

A major limitation of Terrell's quantisation rule is the strong dependency of w on σ, which in turn is sensitive to extreme values (or *outliers*). For univariate samples, outliers are observations lying far from the central part of the distribution and can greatly influence the standard deviation of the sample. Formally, the limits (or fences) of outliers lie $1.5IQR$ below the first (Q_1) and above the third (Q_3) quartile, where IQR=Q_3-Q_1 denotes the inter-quartile range of the sample [50]. Very coarse resolutions attributed to the harmful effect of outliers in formulae 4.16, can make it difficult or even impossible to discriminate closely adjacent clusters or to produce accurate cluster descriptors.

Terrell's quantisation rule has another serious drawback as it does not take into consideration the essential shape of the distribution. For instance, significant multi-modal structures indicating large scale data discontinuities are not utilised by formulae 4.16. Furthermore, the local density of points is also ignored. In essence, the more isolated the clusters are, the larger the σ, and thereby, the coarser the grid resolution. In other words, significant data discontinuities, e.g. noise regions surrounding clusters, may have a substantial impact on σ, hence, increasing the probability of under-quantisation.

Intuitively a robust quantisation algorithm must guard against outliers and at the same time must utilise information regarding the local density of points. These issues are addressed in the subsequent sections of this Chapter.

4.5 $\mathcal{TSQ}$: A $\mathcal{T}$wo $\mathcal{S}$tage $\mathcal{Q}$uantisation Algorithm

In this section we propose a novel $\mathcal{T}$wo $\mathcal{S}$tage $\mathcal{Q}$uantisation Algorithm ($\mathcal{TSQ}$) to address the challenges of quantising high dimensional datasets. The main stages of $\mathcal{TSQ}$ are depicted in figure 4.11. The ultimate goal of $\mathcal{TSQ}$ is to reduce the search combinations for clustering, but at the same time to determine an adequate level of grid resolution that allows both discriminating clusters and producing accurate cluster descriptors.

Initially, a *gross statistical analysis* is applied to each dimension independently to remove outliers in the tails of the distribution and to detect large-scale data discontinuities in the main part of the distribution. A provisional fine bin width is computed using two important distribution metrics, namely, *dispersion* and *entropy*. Other techniques either ignore the distribution, e.g. *CLIQUE*, *WaveCluster*, or use limited statistical information, e.g. *MAFIA*. The final step of the gross analysis involves constructing a smooth approximation of the density distribution using the popular *Kernel Density Estimation* (KDE), as described in detail in section 3.3.

Next, a *detailed statistical analysis* locates the $\mathcal{U}$-regions with only a marginal variance in point density along the smoothed histograms using the three sophisticated statistical tests ($\mathcal{HT}_1$, $\mathcal{HT}_2$, and $\mathcal{HT}_3$) of the $\mathcal{UDA}$ algorithm, as described in detail in Chapter 3 (see sections 3.4.1-3.4.3). Each $\mathcal{U}$-region is then assigned an appropriate bin width by applying Terrell's quantisation rule (equation 4.16) on the data contained in that region. Other quantisation techniques, e.g. *MAFIA*, create a single bin for each quasi-uniform region, which may yield poor quality results if multiple clusters with very different characteristics, i.e. density or geometry, overlap in this $\mathcal{U}$-region. Finally, since the clustering algorithm $\mathcal{NOCEA}$ (see Chapter 5) has been designed to operate on uniform grids a weighted-sum method scalarises the set of different widths into one global value.

4.5.1 Gross Statistical Analysis

Step 1. Removal of Outliers

To guard against the harmful effects of outliers in highly skewed distributions, $\mathcal{TSQ}$ ignores them. This simply entails limiting the statistical analysis in the central part of the distribution whose boundaries are delineated by the outliers fences as described in section 4.4.3. Hence, $\mathcal{TSQ}$ focuses on data lying inside the outlier-free interval E [50] rather than the entire domain $[a, b]$:

$$\boxed{E = [max(a, (Q_1 - 1.5IQR)), min(b, (Q_3 + 1.5IQR))]} \tag{4.17}$$

where Q_1, Q_3, and $IQR = Q_3 - Q_1$ denote the first quartile, third quartile, and inter-quartile range of data along the given dimension, respectively.

Step 2. Computation of Provisional Resolution

For departures from normality or uniformity such as multi-modality or heavily skewed distributions descriptive statistics such as central tendency (e.g. arithmetic mean) or dispersion (e.g. standard deviation) are not sufficient to describe the essential structure of the distribution. In other words, it is difficult to detect and quantify very low density valleys, e.g. noise regions, located in the main part of the distribution using descriptive statistics. Similar to outliers, significant data discontinuities easily cause under-quantisation if using equation 4.16 due to the impact on σ. Hence, it is vital to guard against significant data discontinuities. $\mathcal{TSQ}$ relies on the *entropy* of the data sample E to implicitly quantify the scale of such data discontinuities.

Entropy is a widely used concept to quantify information and in principal measures the amount of uncertainty of a random discrete variable X. Let $x_1, ..., x_k$ be the set of all possible outcomes of X and $p(x)$ be the probability mass function of X. The entropy $H(X)$ is then defined by the following expression [24]:

$$\boxed{H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i)} \tag{4.18}$$

Let $b_1, ..., b_k$ be the set of all bins in a particular dimension and d_i denotes their density, i.e. percentage of total points N lying inside each bin. In analogy to the

entropy of a random discrete variable, the entropy along the given dimension is:

$$\boxed{H = -\sum_{i=1}^{k} d_i \log_2 d_i} \tag{4.19}$$

When the probability of X is uniformly distributed, we are most uncertain about the outcome and thus the entropy is the highest. On the other hand, when the probability mass function is highly concentrated around the modes the result of a random sampling is likely to fall within a small set of outcomes around these modes, so the uncertainty and thus entropy are low. Intuitively, when univariate data points are uniformly distributed we are most uncertain in which bin a data point would lie and therefore the entropy is the highest. In contrast, the more densely populated and closely located the univariate clusters are, the smaller the uncertainty and thus entropy, as a given point is highly likely to fall within bins belonging to a cluster. This fundamental property of entropy is utilised by $\mathcal{TSQ}$ to quantify significant data discontinuities in univariate samples.

If the data in E from formulae 4.17 is uniformly distributed, then a small fraction δ (i.e. $\delta = 0.5\%$ of the total points N) of them is expected to be found inside an interval whose length (ϵ) will approximately be:

$$\boxed{\epsilon = \delta \left(\frac{N}{N_E}\right) l_E} \tag{4.20}$$

where $l_E = min(b, (Q_3 + 1.5IQR)) - max(a, (Q_1 - 1.5IQR))$ and N_E denote the length and number of points in E, respectively.

Using ϵ as initial resolution, $\mathcal{TSQ}$ constructs two conventional density histograms (as described in section 3.2), one for the target dimension and one for a uniform distribution both defined in E. It then computes the entropy for both histograms using equation 4.19. To obtain the entropy ratio $r_H \in (0, 1]$ the entropy of the actual points in E is divided by the entropy of the corresponding uniform distribution.

The value of r_H is a quantitative measure of the difference between the actual data distribution in E and a uniform distribution with the same number of points and range of values. Indeed, *densely populated regions separated from one another by widespread low density regions are implicitly detected through small values of* r_H *and vice-versa.*

Clearly, any packing of points into tight clusters requires a smaller bin width than the uniform distribution. Intuitively, $\mathcal{TSQ}$ incorporates quantitative information related to both data discontinuities and concentration by modifying ϵ by a factor r_H:

$$\boxed{\epsilon' = r_H \epsilon} \tag{4.21}$$

where ϵ' denotes the modified value of ϵ.

The provisional bin width ϵ' is a particularly robust estimator of the lower bound of bin width w because a) it is resistant to outliers, b) it is relatively cognisant of the essential shape of the distribution, and c) it provides fine resolution since it reflects the spreading of a very small percentage (δ) of the total data points (N).

Step 3. Smoothing via Kernel Density Estimation - KDE

The next step is to construct a smooth frequency histogram for the data falling inside the interval of interest E using the binned KDE with the boundary correction as discussed in section 3.3.5.

The practical implementation of the KDE method requires the specification of the bandwidth h, which controls the smoothness of the frequency histogram. A simple solution would be to directly use the automatic normal scale bandwidth selection rule (formulae 3.4) as described in section 3.3.2. However, for non-normal data distributions, e.g. multi-modal or heavily skewed distributions, the statistical performance of formulae 3.4 is poor [92, 94, 97].

$\mathcal{TSQ}$ reaches a compromise between highlighting important features in the data and good scalability using a local adaptation of the normal bandwidth rule. The new methodology relies on dividing the interval of interest E into a finite set of disjoint intervals containing a relatively small percentage, e.g. 5%, of the total (N) data points. Then the normal reference rule is applied to each interval independently.

The division of the domain into intervals isolates local characteristics of the data distribution and guards against outliers or data discontinuities. To retain to some extent important features of the distribution at different data localities

while having a global bandwidth over the entire domain, a weighted sum method is used.

In particular, let us assume that the interval E of j-th dimension is partitioned into k sub-intervals of approximately equal data coverage, while h_{ij} denotes the local bandwidth computed by the normal reference rule (see equation 3.4) for the i-th interval in j-th dimension. The set of locally obtained bandwidths h_{ij} is scalarised into a single bandwidth (h_j) by pre-multiplying each local bandwidth with a specific weight and then forming their sum. The weight of an interval is simply the percentage of total points of E (N_E) lying inside that interval. Hence, the $\mathcal{TSQ}$ bandwidth scalarisation is:

$$\boxed{h_j = \sum_{i=1}^{k} \left(\frac{N_{ij}}{N_E}\right) h_{kj}} \tag{4.22}$$

where k is the number of intervals in the j-th dimension, while N_{ij} denotes the number of points in the i-th interval. It can be easily observed that the weights are normalised, that is, $\sum_{i=1}^{k} \left(\frac{N_{ij}}{N_E}\right) = 1$.

4.5.2 Detailed Statistical Analysis

Step 1. Detection of Quasi-Uniform Regions

Let us assume that during the gross statistical analysis stage the outlier-free interval of interest E is partitioned into m uniform bins of size ϵ' determined by equation 4.21. Additionally, let $d_0, ..., d_{m-1}$ be the histogram values of the smooth frequency histogram as computed by the binned KDE method with boundary correction (see section 3.3.5). $\mathcal{TSQ}$ then employs the $\mathcal{UDA}$ (section 3.4) statistical analysis to obtain all *non-sparse* $\mathcal{U}$-regions along the smooth frequency histogram.

Step 2. Quantisation of Quasi-Uniform Regions

The rationale of partitioning the original smooth histogram with $\mathcal{UDA}$ is to enable a more detailed analysis within the quasi-uniform regions identified. In particular, Terrell's quantisation rule (equation 4.16) can now be safely applied to each $\mathcal{U}$-region independently, because both outliers and significant data discontinuities affecting σ have been removed.

As Scott elaborated "*...in principle, there is no lower bound on bin width (w) because the unknown density can be arbitrarily rough...*" [92]. However, an extremely fine resolution computed by equation 4.16, even if it is valid from a statistical point of view, incurs high computational costs for clustering, especially for high dimensional datasets [7, 74]. Therefore, it is necessary to set a lower bound on w that yields a reasonable compromise between efficiency and effectiveness.

Recall from section 4.5.1 that the provisional bin width ϵ' given by equation 4.21 is a particularly robust estimator of the lower bound of w. Hence, $\mathcal{TSQ}$ balances computation and quality of the clustering results by setting the bin width w_{ij} for the ith $\mathcal{U}$-region of jth dimension as follows:

$$\boxed{w_{ij} = \max\left(\epsilon', (3.729\ \sigma_{ij} N_{ij}^{-1/3})\right)} \tag{4.23}$$

where N_{ij} and σ_{ij} denote the number and standard deviation of points in the ith $\mathcal{U}$-region of the jth dimension, respectively.

In contrast, other quantisation techniques, e.g. *MAFIA*, create a single bin for each $\mathcal{U}$-region, which may yield poor quality results if the projection of multiple clusters with very different densities overlap in that region. Finally, for discrete or even continuous attributes of finite precision, it is inappropriate to select a bin width that is smaller than the step of the natural precision of the data.

Step 3. Scalarisation of Local Resolutions

Ideally, each $\mathcal{U}$-region would keep its own bin width leading to a non-uniform grid, which may delineate cluster boundaries more accurately since it reflects the local distribution. However, this idea adds a substantial amount of extra work when evaluating the quality of candidate solutions (see section 5.7).

Therefore, $\mathcal{TSQ}$ uses a simple *weighted-sum* method to scalarise the set of locally optimal bin widths into a single global value. This can be simply done by pre-multiplying each width with a specific weight and then forming their sum. Usually, the weights are chosen in a way so as to make the sum of all weights equal to one. One of the ways to achieve this is to normalise each weight by dividing it by the sum of all the weights. Although the idea is simple, it introduces a non

trivial question: what values of the weights must one use? Of course, there is no unique answer to this question. The answer strongly depends on the importance of each $\mathcal{U}$-region in the context of quantisation and clustering. The work in this thesis solely focuses on the discovery of highly homogeneous rather than highly dense clusters. Therefore, a $\mathcal{U}$-region irrespective of its density is important as long as it covers a relatively large portion of the data.

$\mathcal{TSQ}$ assigns a specific weight to each $\mathcal{U}$-region that is proportional to the number of points in that region with respect to the total number of points covered by all $\mathcal{U}$-regions in the same dimension. Hence, the domain $[a_j, b_j]$ of jth dimension is partitioned into disjoint equi-sized intervals of length w_j that is determined by the following weighted-sum expression:

$$\boxed{w_j = \sum_{i=1}^{k} \left(\frac{N_{ij}}{total_{N_j}} \right) w_{kj}} \tag{4.24}$$

where k is the number of $\mathcal{U}$-regions in the j-th dimension, while $total_{N_j} = N_{1j} + ... + N_{kj}$ denotes the total number of points covered by these regions. It can be easily observed that the weights are normalised so as $\sum_{i=1}^{k} \left(\frac{N_{ij}}{total_{N_j}} \right) = 1$.

4.6 Summary

In this Chapter we have described the $\mathcal{TSQ}$ quantisation algorithm imposing a multi-dimensional grid structure onto the dataspace to reduce the search combinations for clustering large and high dimensional datasets.

Initially the Chapter has identified the limitations of other quantisation algorithms, and it has motivated the analysis of univariate density histograms as the only computationally feasible means to construct the multi-dimensional grid.

The Chapter has investigated the use of standard quantisation techniques along with new heuristics (e.g. $\mathcal{UDA}$ in Chapter 3) reflecting the local distribution, to determine an appropriate grid resolution that enables the discrimination of clusters, while preserving accuracy and acceptable computational cost.

The quantised dataspace is subsequently analysed by the novel evolutionary-based clustering algorithm $\mathcal{NOCEA}$ that is described in the following Chapter.

Chapter 5

Clustering with Evolutionary Algorithms

Capsule

This Chapter - the core part of the thesis - presents the novel evolutionary algorithm $\mathcal{NOCEA}$ that efficiently and effectively clusters massive and high dimensional numerical datasets. The discussion details key aspects of the proposed methodology, including an elaborate integer-valued representation scheme, a simple data coverage maximisation fitness function, several novel genetic operators, as well as advanced post-processing algorithms to simplify the discovered knowledge. Finally, task parallelism to improve scalability when the data to be mined is massive, is also explored. The salient properties of $\mathcal{NOCEA}$ are discussed and demonstrated on both artificial and real-world datasets in Chapters 6 and 7, respectively.

5.1 Introduction

There is a great deal of interest in developing robust clustering algorithms to extract hidden nuggets of knowledge from large databases related to business or scientific activities, to achieve competitive advantage [31, 40, 51]. The work described in this chapter contributes towards exploiting the powerful search mechanism of evolutionary algorithms to mine high quality clustering rules from massive and high dimensional databases.

Over the years several approaches, both evolutionary-based and conventional,

for clustering have been proposed. Chapter 2 gives a detailed survey of these approaches. Most of these approaches do not address all of the requirements (see section 2.3.2) for data mining clustering adequately, although considerable work has been done in addressing each requirement separately.

The clustering approach advocated in this thesis is to explore the enormous and sparsely-filled dataspace with a parallel, semi-stochastic evolutionary search. In particular, the core idea is to evolve a population of individuals, where each candidate solution comprises a *variable* number of *disjoint* and *axis-aligned hyper-rectangular* rules with *homogeneous* data distribution. To conform with all the important requirements for data mining clustering (see section 2.3.2), task-specific genetic operators were devised.

The remainder of this Chapter is structured as follows: Section 5.2 provides an overview of $\mathcal{NOCEA}$. Section 5.3 gives a formal definition of the fundamental piece of knowledge in $\mathcal{NOCEA}$, the clustering rules. Section 5.4 describes the individual representation scheme and motivates the use of integer-valued encoding rather than binary or floating-point. Section 5.5 presents a novel fitness function for clustering and briefly discusses its salient features. Section 5.6 thoroughly discusses the inductive bias that $\mathcal{NOCEA}$ uses to constrain the search space, i.e. representational bias, and to favour the selection of particular solutions, i.e. preference bias. Sections 5.7-5.10 cover the novel genetic operators that were developed to discover high quality clustering rules. In particular, section 5.7 describes the *homogeneity* or *repair* operator, which ensures that the space enclosed by candidate rules has quasi-uniform data distribution. Section 5.8 describes $\mathcal{NOCEA}$'s advanced recombination operator. Section 5.9 describes a novel *generalisation* operator that strives to minimise the length of individuals and to make rules as generic as possible. Section 5.10 describes two novel mutation operators that provide the main exploratory force in $\mathcal{NOCEA}$. Section 5.11 explains how $\mathcal{NOCEA}$ tackles the problem of subspace clustering. Section 5.12 describes a post-processing algorithm that groups adjacent rules into clusters. Section 5.13 describes a preliminary parallelisation of $\mathcal{NOCEA}$ to improve scalability. Finally, section 5.14 discusses the default parameter settings in $\mathcal{NOCEA}$.

5.2 $\mathcal{NOCEA}$ Overview

$\mathcal{NOCEA}$[1] utilises the powerful search mechanism of EAs to efficiently and effectively mine highly-homogeneous clustering rules from large and high dimensional numerical databases. The abstract architecture of $\mathcal{NOCEA}$ is shown in figure 5.12. $\mathcal{NOCEA}$ includes several pre- and post-processing stages to prepare the raw data and simplify the discovered knowledge, respectively.

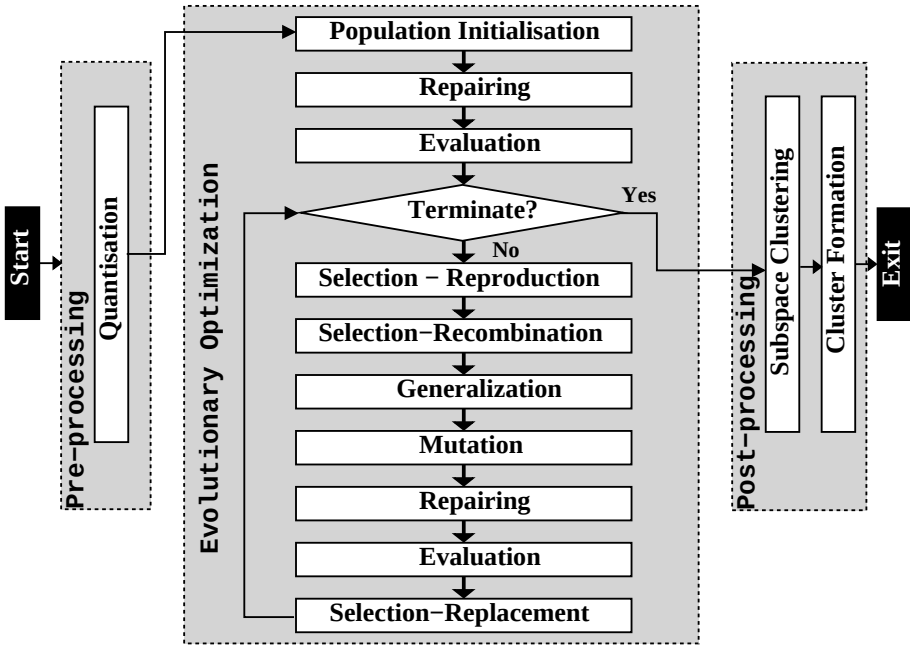


Figure 5.12: Architecture of $\mathcal{NOCEA}$

$\mathcal{NOCEA}$ evolves individuals of variable length comprising disjoint and axis-aligned hyper-rectangular rules with homogeneous data distribution. The antecedent part of the rules includes an interval-like condition for each dimension. Initially, a statistical-based quantisation algorithm imposes a regular multi-dimensional grid structure onto the data space to reduce the search combinations, as described in Chapter 4. The boundaries of the intervals are encoded as integer values reflecting the automatic discretisation of the dataspace. Like most EAs, $\mathcal{NOCEA}$ begins with an initial population of individuals whose chromosomes are independently initialised with a single randomly generated rule.

[1] $\mathcal{N}$on-$\mathcal{O}$verlapping $\mathcal{C}$lustering with $\mathcal{E}$volutionary $\mathcal{A}$lgorithms

Next, a task-specific genetic operator, the *repair operator* (section 5.7) shrinks the boundaries of rules or splits candidate rules, if necessary, to ensure the space enclosed by each feasible rule is uniformly filled with data points.

The evolutionary search is guided by a simple *fitness function* (maximisation of total point coverage), unlike the commonly used distance-based functions.

Next, some of the repaired individuals are selected according to the fitness to form a new generation, e.g. the higher the fitness value, the more chance an individual has to be selected for *reproduction*.

Variation in the population is introduced by conducting genetic operations on the selected individuals including: *crossover* (section 5.8), *generalisation* (section 5.9), and *mutation* (section 5.10). Various constraints are imposed during these semi-stochastic operations to ensure that the resultant individuals always comprise rules that are syntactically valid, disjoint, and axis aligned. During *crossover*, two individuals are selected from the mating pool at random and carefully selected part(s) of rules are exchanged between them to create two new solutions. The individuals of this new population are then subject to a parsimony operator, called *generalisation*, that attempts to minimise the size of the rule set, reducing thus computational complexity and improving comprehensibility. The *mutation operator*, in turn, grows existing rules at random and creates new candidate rules, according to a certain small probability.

Next, the newly generated offspring are repaired and evaluated. After the new offspring have been created via the genetic operators the two populations of parents and children are merged to create a new population. To maintain a fixed-sized population only the appropriate number of individuals survive based on some *replacement* strategy. The individuals of this new generation are, in their turn, subjected to the same evolutionary process for a certain number of generations or until a solution with the desired performance has been found.

After convergence, a post-processing routine performs *subspace clustering* (section 5.11) removing redundant conditions from the antecedent part of the rules. Finally, adjacent rules with similar densities are grouped together to *assemble* (section 5.12) clusters, and report them in Disjunctive Normal Form (DNF).

5.3 Clustering Rules

IF-THEN clustering rules are intuitively comprehensible for most humans since they represent knowledge at a high level of abstraction involving logical conditions rather than point-based cluster representations. In this thesis a *clustering rule* $\mathcal{R}$ defined in the continuous space $\mathcal{F}$ (sections 2.3.1 and 4.3) is knowledge representation in the form:

$$\boxed{\mathcal{R}: \ \textbf{IF } cond_1 \wedge ... \wedge cond_d \textbf{ THEN } cluster\_label}$$

The *premise* or *antecedent* part of the rule (*IF*-part) consists of a logical conjunction of d conditions, one for each feature, whereas the *conclusion* or *consequent* (*THEN*-part) contains the cluster label. The semantics of this kind of clustering rule is as follows: if all the conditions specified in the antecedent part are satisfied by the corresponding feature values of a given data point, then this point is assigned to (or covered by) the cluster, identified by the consequent. Each condition is in the form of a right-open feature-interval pair, e.g. $(10000 \leq \texttt{Income} < 25000)$. Formally, a clustering rule $\mathcal{R}$ is a subset of the feature space $\mathcal{F}$ ($\mathcal{R} \subseteq \mathcal{F}$) and can be geometrically interpreted as an axis-parallel hyper-box $\mathcal{R}$=$[l_1, u_1) \times ... \times [l_d, u_d)$, $i = 1, ..., d$, where $l_i \in \mathbb{R}$ and $u_i \in \mathbb{R}$ denote the lower and upper bounds of $\mathcal{R}$ in the ith dimension, respectively.

Recall from Chapter 4 that the quantisation of the continuous space $\mathcal{F}$ yields a multi-dimensional grid, thereby reducing the search space for the clustering algorithm. Therefore, it is necessary to specialise the above definition of clustering rules to accommodate the fact that rule boundaries are not placed arbitrarily, but rather they coincide with the grid bin edges.

During quantisation, the domain $[a_i, b_i]$, $i = 1, ..., d$, of the ith dimension is partitioned into $m_i \in \mathbb{Z}^{*}$[2] disjoint intervals $\mathbb{B}_i$[3]$=0, ..., (m_i - 1)$ of uniform length w_i. In such a space an axis-aligned hyper-rectangular rule $\mathcal{R}$ is: $\mathcal{R}$=$[l_1, u_1] \times ... \times [l_d, u_d]$, where $l_i, u_i \in [0...m_i - 1]$ and $l_i \leq u_i$, $\forall i \in [1, d]$. The simple decoding function 5.25 maps the integer-encoded rule boundaries l_i and u_i into the interval

[2]The ordered set of nonnegative integers, $Z^* = \{0\} \cup Z^+$, where Z^+ are the positive integers.
[3]The ordered set of the m_i disjoint intervals in ith dimension.

$[a_i, b_i]$.

$$\boxed{dl_i = a_i + l_i w_i, \quad du_i = a_i + (u_i + 1)w_i, \quad \forall i \in [1, d]} \tag{5.25}$$

where dl_i and du_i denote the decoded values of l_i and u_i, respectively.

By definition the ith point $p_i = [p_{i1}, ..., p_{id}]$ (section 4.3) is contained in $\mathcal{R}$ if and only if $dl_j \leq p_{ij} < du_j$, $\forall\ j = 1, ..., d$. The *coverage* $cov(\mathcal{R}) \in [0, 1]$ of a rule $\mathcal{R}$ is defined to be the fraction of total points covered by $\mathcal{R}$, $cov(\mathcal{R}) = \frac{N_{\mathcal{R}}}{N}$, where $N_{\mathcal{R}}$ is the number of points inside $\mathcal{R}$. $\mathcal{R}$ is deemed *non-sparse* if its coverage exceeds an input *sparsity threshold* $T_s \in (0, 1]$.

5.4 Individual Representation

The choice of encoding for the candidate solutions is critical for the performance of any search algorithm. Usually, in EA-based optimisations the individual representation is inherent to the nature of the problem. For instance, in the context of the k-means clustering each individual represents the coordinates of the cluster centroids. The choice of an efficient representation scheme depends not only on the target problem itself, but also on the search method used to solve the problem. As Deb insightfully observed "*...the efficiency and complexity of a search algorithm largely depends on how the solutions have been represented and how suitable the representation is in the context of the underlying search operators...*"[11].

5.4.1 *What* Do Candidate Solutions Represent?

Although there are no well-founded measures of knowledge comprehensibility, small, coherent and informative structures, e.g. rule-sets, are widely considered as highly comprehensible within the DM community. Since clustering is all about summarising data distributions, the thesis adopts *clustering rules* as a readily interpretable structure to describe the discovered knowledge. $\mathcal{NOCEA}$ evolves individuals of *variable-length* comprising *disjoint* and *axis-aligned hyper-rectangular* rules. Two d-dimensional rules $\mathcal{R}_1$ and $\mathcal{R}_2$ are *disjoint* if there is at least one

dimension, say c, such that the upper bound of the first rule is less than the lower bound of the second or the opposite, i.e. $(u_{c2} + 1) \leq l_{c1}$ or $(u_{c1} + 1) \leq l_{c2}$.

A single rule constitutes a *completely specified sub-solution* to the clustering problem. Each *fixed-length* rule, in turn, is composed of d genes, henceforth termed *feature-genes*, that encode an interval-like condition in one dimension. The ith feature-gene ($i = 1, ..., d$) of the jth rule ($j = 1, ..., k$) is subdivided into two discrete fields: the *lower* (l_{ij}) and *upper* (u_{ij}) bounds, where ($l_{ij} \leq u_{ij}$) and l_{ij}, $u_{ij} \in \mathbb{B}_j$ (section 5.3).

Two d-dimensional rules $\mathcal{R}_1$ and $\mathcal{R}_2$ are *connected* if they have a *common face*, or if there exists another rule $\mathcal{R}_3$ such that both $\mathcal{R}_1$ and $\mathcal{R}_2$ are connected to $\mathcal{R}_3$. Rules $\mathcal{R}_1$ and $\mathcal{R}_2$ have a common face if there is an intersection between them in (d-1) dimensions, and there is one dimension, say c, such that the rules are touching, i.e. $l_{c1} = (u_{c2} + 1)$ or $l_{c2} = (u_{c1} + 1)$.

A set of connected rules with similar densities and homogeneous data distributions define the skeleton of a cluster. In most real clustering problems, except in some specific sub-domains, the optimal number of clusters is not known *a priori*. Thus, a data driven system, where the number of rules/clusters is automatically self-adapted during the course of evolution, is very desirable. The obvious advantage of the variable-length genotype is the transfer of control over the optimal number of rules/clusters from humans to the genetic search mechanisms of $\mathcal{NOCEA}$.

Finally, a positive implication of evolving only disjoint partitions is that there is no need to encode the consequent part of clustering rules in the genotype. This is because cluster identifiers neither change the spatial distribution of data nor influence the transition rules used by $\mathcal{NOCEA}$ to move from one candidate solution to another. An advanced post-processing algorithm described in section 5.12, fills the consequent part of rules with the appropriate cluster identifier.

5.4.2 *What* is the Search Space for Clustering?

$\mathcal{NOCEA}$ performs a semi-stochastic search throughout the space $\mathcal{S}$ of *all possible feasible solutions* to determine those that maximise the total point coverage. Specialised operators have been devised in this thesis to enforce the feasibility of individuals in the population. But what are the properties of a feasible solution? How does a feasible solution differ from a candidate solution as defined in section 5.4.1?

In the clustering context of this thesis a *feasible solution* always complies with all the following requirements:

1. **Semantic Correctness:** The upper bound of all rules must be at least equal to its associated lower bound for all dimensions.

2. **Axis-Alignment**: All hyper-rectangular rules are by definition axis-aligned.

3. **Disjointness:** No overlapping among rules is allowed in the chromosome.

4. **Homogeneity:** The d-dimensional region enclosed by a feasible rule must have a relatively homogeneous distribution of points. For example, rule $\mathcal{R}_2$ in figure 5.13(a) is not homogeneous, even though it is semantically valid and axis-parallel.

5. **Sparsity:** The point coverage of a feasible rule must be statistically significant to minimise the danger of over-fitting (i.e. to cover very few instances) the data. *Sparse* rules (section 5.3) are eliminated, because they reflect spurious relationships that are unlikely to occur in unseen data.

$\mathcal{NOCEA}$ employs variation operators, i.e. recombination, mutation and generalisation, with semi-stochastic constrained functionalities to comply with requirements 1-3, and a specialised repair operator to enforce the formation of homogeneous rules.

5.4.3 Knowledge Abstraction in the Chromosome

Concerning interdependencies in the chromosome, one can distinguish four levels of knowledge abstraction. Starting from the bottom, the elementary level of *bound-gene* represents either the lower or the upper bound of a rule in a particular dimension. The second level, or *feature-gene*, expresses the constraint that the upper bound in a particular dimension of a rule must be always greater or equal to the lower bound in the same dimension. In the third level of abstraction, or *rule-gene*, all the feature-genes associated with a particular rule are grouped together into one entity by forming their Cartesian product. Finally, in the top level, *rule-genes* are concatenated together to form the entire chromosome. Notice that in the top of the hierarchy there are no interdependencies because rule-genes do not overlap. Clearly, specialised operators are required to preserve these constraints, since traditional genetic operators disregard semantic linkages among genes in the chromosome.

5.4.4 *How* Are Candidate Solutions Encoded?

The chromosome of an individual comprising k rules can be viewed as a one-dimensional array of $2dk$ integer-valued slots. Each rule is encoded in a $2d$-length substring that is formed by concatenating together the lower and upper bounds for each dimension. Unlike typical EAs, the relative position of a rule in the chromosome is unimportant. The reasons for using an integer-valued representation rather than floating-point or binary are explained in detail in section 5.4.5.

Figure 5.13(a) depicts a hypothetical distribution in a two dimensional space defined by the continuous features `Income` and `Expenditure` that are bounded in the range [500, 1300] and [0, 340], respectively. Additionally, let w_{Income}=25 and $w_{Expend.}$=20 be the bin width for `Income` and `Expenditure`, respectively. Figure 5.13(b) shows the structure of the genotype corresponding to the candidate solution of figure 5.13(a), that has three rules $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$. Figure 5.13(c) depicts the conventional binary representation of these rules using five bit precision for both dimensions.

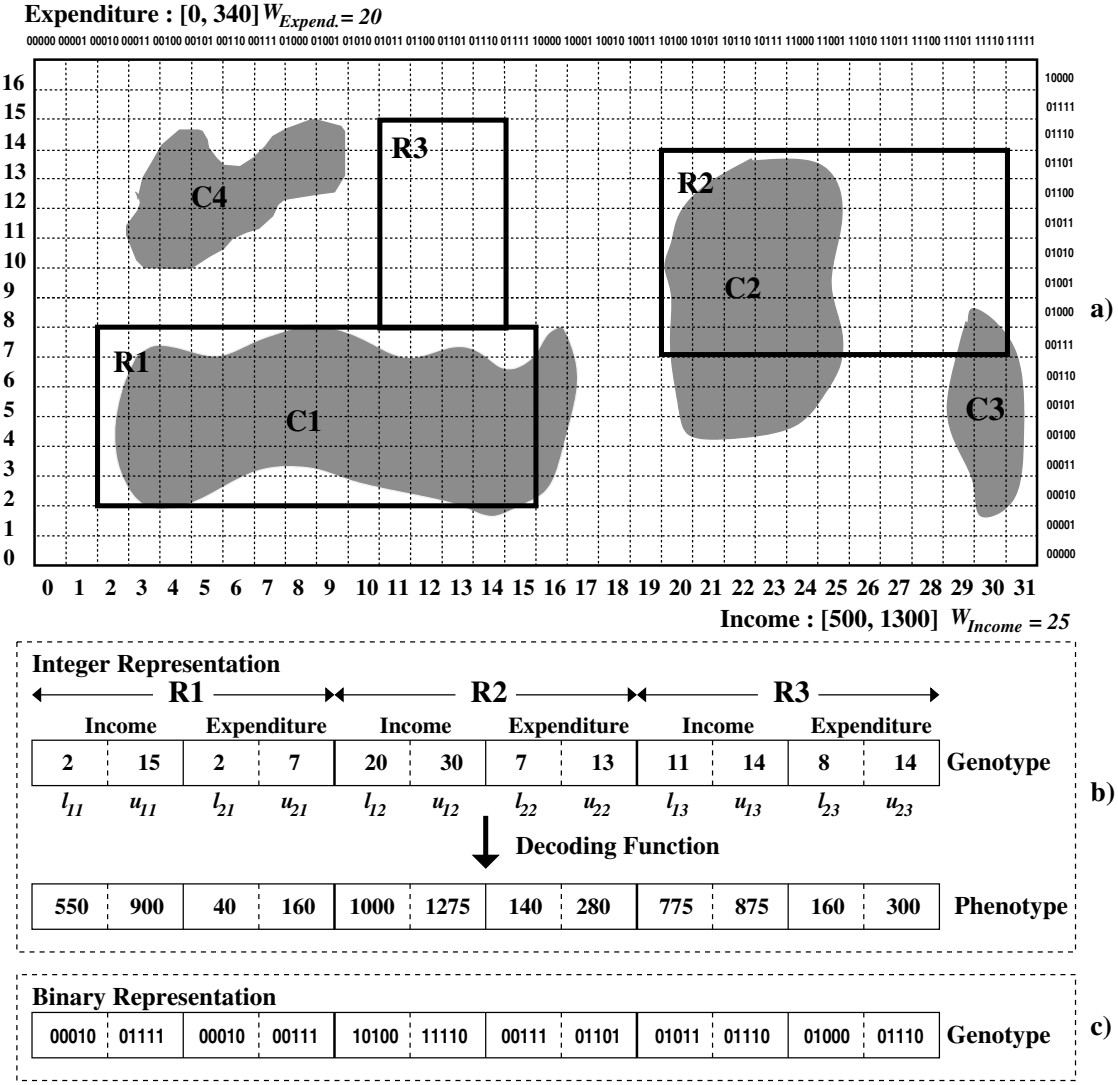


Figure 5.13: Integer-Valued Representation of Candidate Solutions

5.4.5 *Why* Use Integer-Valued Encoding?

It has been shown that for some real-valued numerical optimisation problems floating-point representations outperform binary encodings in terms of precision and execution time [11, 71]. As D. Fogel argues *"...there is no empirical evidence to indicate that binary coding allows for greater effectiveness or efficiency in solving real-valued optimisation problems..."* [11]. Although a real-valued representation enables arbitrary-precision solutions to be found, it generates prohibitively large search spaces for problems of high dimensionality. Despite the

fact that the optimal solutions in the continuous space $\mathcal{F}$ (section 4.3) and the derived discrete space, e.g. multi-dimensional grid, may be marginally different, the obtained solutions are usually acceptable in most practical search problems.

There are also several reasons to abandon binary encoding in $\mathcal{NOCEA}$:

1. **Hamming cliffs:** One difficulty with conventional binary encoding is the presence of *Hamming cliffs* associated with certain strings, e.g. 01111 and 10000, where the transition to a neighbouring solution in the grid space requires the alteration of many bits. Hamming cliffs hinder fast local fine-tuning. For instance, in figure 5.13(a), an evidently simple one-bin extension of the upper `Income` bound of $\mathcal{R}_1$ requires the simultaneous alteration of all five bits, while with integer encoding this is achieved in one step.

2. **Redundancy:** An l-bit substring encoding a particular variable has a total of 2^l different states. If a discrete variable can only take on an exact finite set of values whose size is different from some power of 2, then there is redundancy in the representation. For instance, given that the `Expenditure` domain is partitioned into 17 bins, as shown in figure 5.13(a), $\mathcal{NOCEA}$ needs at least 5 bits to cover this range. However, a 5-bit length binary encoding yields 32 possible states in total, from which 15 are redundant - correspond to non-existing bins. Clearly, extra computational effort is required to prevent the formation of individuals with erroneous bit combinations. In contrast, integer-valued representations do not suffer from such problems because the possible states that a rule bound can take are from a minimum length integer-valued sequence $0, ..., (m-1)$, where m denotes the number of bins in each dimension.

3. **Excessive String Length:** A candidate solution comprising k rules can be viewed as a $2dk$ slots vector. A binary representation scheme when applied to real-world multidimensional problems of high-precision would produce individuals having considerably longer strings compared to integer representations. For instance, a binary-coded solution comprising 50 rules in a 100-dimensional space with 64 bit precision for all dimensions requires

a $((2 * 100 * 50) * 64)$-slot vector, while an integer-representation yields a $(2*100*50)$-slot vector. Representations with minimal-slot solution vectors must be adopted to reduce computational complexity.

4. **Feasibility:** Specialised genetic operators are required to satisfy two important feasibility constraints, i.e. rule disjointness and syntactical validity, as defined in section 5.4.2. Usually, the larger the string length the more expensive the application of the genetic operators. The burden of producing non-overlapping rules that are syntactically valid is significantly higher for binary representations because a rule boundary is mapped by a combination of bits, each of which is treated as an individual entity. In contrast, with integer encoding there is a one-to-one relationship between a given position in the chromosome and the corresponding rule boundary. Hence the manipulation of a given rule bound by the genetic operators can be completed in one step rather than many as in binary representations.

5.5 Fitness Function

5.5.1 Design of the Fitness Function

In a typical EA, each individual in the population is assigned, by means of a *fitness function*, a "figure of merit" that reflects the performance of the given individual in solving the target problem [47, 71]. This value is the quantitative information the EA uses to guide the search. The fitness function (f) takes as argument a single individual ($\mathcal{I}$) and returns a scalar numerical *fitness* that indicates the utility or ability of the individual in the context of the underlying optimisation problem.

$$\boxed{f : \mathcal{I}_{\mathcal{S}} \rightarrow \mathbb{R}} \tag{5.26}$$

where $\mathcal{I}_{\mathcal{S}}$ denotes the phenotype search space.

The fitness of an individual determines the probability that the given individual will survive into and be selected for reproduction in succeeding generation(s). In order for an EA-based system to effectively search for the optimal solution, an

appropriate fitness function must be carefully devised. In general, the choice of the fitness function is closely related to the problem at hand. Although the fitness function is very much application dependent, there are a few general design principles [11, 47, 71]:

- The fitness landscape must be as smooth and regular as possible, so that chromosomes with reasonable fitness are close (in the phenotype space) to chromosomes with slightly better fitness.
- The fitness landscape should not have too many local maxima, or a very isolated global maximum.
- The fitness function must reflect the relevant measures to be optimised.
- The fitness function should provide enough information to drive the selective pressure of the EA.
- The values of the fitness function must be graded in a fine-grained manner providing enough quality information to drive the selective pressure of the EA.

5.5.2 A Robust Clustering Fitness Function

In this thesis a simple and robust fitness function is proposed to guide the evolutionary search. In our clustering context, the fitness function $f : \mathcal{S}^4 \rightarrow [0, 1]$ simply maps to the data coverage, i.e. the proportion of the dataset covered by the disjoint rules of the individual. In particular, the fitness of a *feasible* individual ($\mathcal{I}$) is the fraction of *total points* N that are covered by the rules of $\mathcal{I}$:

$$f(\mathcal{I}) = \left(\frac{1}{N} \sum_{i=1}^{k} N_i \right) \tag{5.27}$$

where, k denotes the number of rules in $\mathcal{I}$, and N_i is the number of points covered by the ith rule.

Clearly, f is greedy with respect to the data coverage of feasible solutions, and is, by definition, always bounded in $[0, 1]$.

[4]The set of all possible feasible solutions. For a definition of feasibility see section 5.4.2

5.5.3 Fitness Function Salient Features

Aiming to maximise data coverage with a set of disjoint feasible rules, is a particularly robust fitness function, well suited for high dimensional datasets where the concepts of density and proximity are vague. The salient features of the proposed fitness function are:

1. **Not a Distance-based Clustering Criterion:** Unlike distance-based clustering techniques that relocate points among clusters deterministically - each instance is assigned to the cluster with the closest representative point - $\mathcal{NOCEA}$ traverses the search space stochastically. Since individuals are assessed on the basis of point coverage with no distance bias, f neither favours the formation of hyper-spherical clusters of similar sizes nor is affected by outliers and noise, as opposed to distance-based techniques.

2. **Not a Density-based Clustering Criterion:** Density-based clustering techniques require that the point density of a cluster must exceed some user-defined threshold. Depending on the choice of the threshold, it is likely to miss low density clusters. In $\mathcal{NOCEA}$, by contrast, rules can "grow" to arbitrarily large sizes and in as many dimensions as required. This is simply because the utility of a rule is not assessed on the basis of density.

3. **Resistance to Curse of Dimensionality:** The curse of dimensionality phenomenon (section 2.3.2) has no impact on f because the concepts of sparsity and point proximity are not encapsulated in the fitness function.

4. **Bounded Range:** Due to the disjointness of rules in the chromosome, the range of the fitness function, $Range(f) = f(\mathcal{S}) = \{f(\mathcal{I}) : I \in \mathcal{S}\}$, is always bounded in the interval $[0, 1]$. In contrast, the extreme values for other clustering criterion functions, e.g. square-error, are not known *a priori* and more importantly, are very much data dependent. Knowing the range of f helps monitor the progress of the evolutionary search and tuning various clustering related parameters, e.g. rule sparsity threshold T_s.

5. **No Preference Bias with Respect to the Structural Characteristics of a Clustering Solution:** In general, *preference* or *search bias* refers to any criterion that is used to determine how the learning system traverses the search space [67]. The most relevant aspect of preference bias in the context of the fitness function is *how* candidate solutions are evaluated. Clearly, formulae 5.27 does not incorporate any preference bias being associated with the structural characteristics of an individual, e.g. number, size, geometry, and density of rules. This is a deliberate choice mainly due to the difficulty of balancing such incommensurable concepts. Rather such concepts are taken into account by the genetic operators. $\mathcal{NOCEA}$'s fitness function focuses only on maximising the point coverage, allowing thus the exploration of more search combinations. The maximisation of point coverage with feasible rule-sets can be considered as a preference bias favouring more complete clustering solutions, rather than solutions where many data points belonging to clusters that are not covered by clustering rules.

   Additionally, evaluating solutions solely on the basis of data coverage provides the ground for a fair and straightforward comparison between clustering solutions with differences in the number, size, density and point coverage of candidate rules. Other techniques, e.g. k-means, require that each individual contain the same number of rules. Since the only driving force is the coverage, individuals may have exactly the same performance with radically different genetic material, which helps preserve the desired diversity between the population members. However, this also tends to make the search space less smoothed and less regular.

In summary, the proposed fitness function is simple, robust, well-suited for high-dimensional clustering problems, and conforms with most of the requirements listed in section 5.5.1.

5.6 Inductive Bias

Learning is a fundamental characteristic of all living organisms (*cognitive systems*), which helps them to develop adaptive behaviour to cope with their constantly changing environment. These coping strategies can be viewed as the ability of organisms to improve their performance on a problem by utilising acquired experience from the past. *Inductive learning* is the model that allows learning generalisations from a set of examples [67]. During the learning phase the cognitive system observes its environment and recognises similarities among objects and events.

Inductive learning systems are classified as *supervised* or *unsupervised* learning. Supervised learning (or classification) as its name implies, is equivalent to learning with an external teacher, who supplies the cognitive system with a finite set, called *training set*, of predefined classes and examples for each class as well. The task of the supervised learning is to discover common properties among examples of the same class and to induce correct concept descriptors for each class. In contrast, in unsupervised learning (or clustering) the cognitive system is also provided with examples from the environment but no predefined classes are available. The system must discover by itself which concepts exist and induce their descriptions.

An important issue for all inductive learning algorithms is the *inductive bias*, which refers to any criterion, except consistency with the data, that either explicitly or implicitly, a cognitive system (or learner) uses to constrain the concept space or to favour the selection of particular concepts within that space [67]. In general, inductive learning can be viewed as a search in a hypotheses space. A hypothesis is a concept descriptor being expressed in some knowledge representation form, i.e. classification rules, clustering rules, decision trees. In all but the most trivial domains, a potentially infinite number of such hypotheses may be formed and the problem of exploring and evaluating all of them is practically impossible. Various pruning techniques can be introduced to address this problem [67]. The factors that either explicitly or implicitly influence the definition and selection of hypotheses are widely known as inductive bias. There are two main

ways of introducing inductive bias, namely, *representational* and *preference bias*.

5.6.1 Representational Bias

Representational bias refers to syntactical constraints of the knowledge representation language [67]. Inductive learning systems construct candidate solutions within the limits of the fixed "vocabulary" supported by the representation language. The main purpose of introducing representational bias is to reduce the size of the solution space, but an excessive use of constraints may leave the learner incapable of representing the concepts that it is trying to learn. $\mathcal{NOCEA}$ has certain constraints determining its representational biases:

- **Discovery of Axis-aligned Hyper-rectangular Rules:** *Propositional-like* or *0*-th order representations use a logic formulae consisting of attribute-value pair conditions, e.g. (10000$\leq$`Income`$<$50000). In contrast, *first-order logic* (FOL) conditions are not restricted to attribute-interval pairs, but may contain arbitrarily complex predicates, involving different features with compatible domains, e.g. ((5 $*$ `Expenditure`) $<$ `Income`). Undoubtedly, FOL conditions have more expressive power than propositional like conditions because the former can encapsulate inter-attribute relationships. However, when using a more expressive formalism, the search combinations increase enormously, especially for high-dimensional datasets. For instance, there are several ways of approximating a trapezoid (grey-shadowed) cluster, as depicted in figure 5.14.

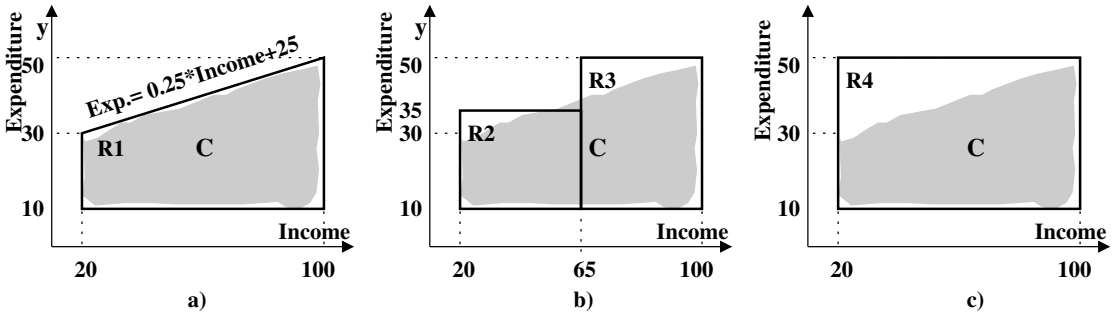


Figure 5.14: The Expressive Power of First-Order Logic Conditions

Obviously, the FOL expression $\mathcal{R}_1$: (20≤`Income` <100) ∧(10≤`Expenditure` ≤(0.25`Income`+25)), sketched in figure 5.14(a), clearly outperforms propositional-like expressions such as $\mathcal{R}_2 \vee \mathcal{R}_3$: ((20≤`Income`<65) ∧(10≤`Expenditure`≤35)) ∨ ((65≤`Income` <100)∧(10≤`Expenditure`≤50)), $\mathcal{R}_4$: (20≤`Income`<100)∧(10≤ `Expenditure`≤50) that are depicted in figures 5.14(b-c), respectively. This is because $\mathcal{R}_1$ provides the most accurate and homogeneous cluster description with minimal number of rules, and it is straightforward to induce the trapezoid pattern from the geometry of $\mathcal{R}_1$. Despite the appealing flexibility of FOL conditions the thesis adopts propositional rules primarily due to their comprehensibility, and secondarily because of the enormous search space associated with FOL conditions when clustering datasets of high dimensionality.

- **Evolution of Disjoint Rules:** There are several reasons to restrict the evolutionary search to the space of disjoint rules.

  1. The most prominent reason is the tremendous increase in the number of search combinations when rule overlapping is allowed.
  2. The fitness function must be properly modified to accommodate disjoint rules. But, how should two individuals with overlapping rules be compared? How should individuals with and without overlapping rules be compared? Addressing these sort of questions is not a trivial task.
  3. There is a direct relationship between the degree of overlapping among rules and the redundancy of knowledge in the chromosome - the same region in the feature space is likely to be captured by multiple rules.
  4. Assessing and possibly enforcing the homogeneity of rules, is a time consuming task, especially for massive high dimensional datasets. Bearing in mind that rules are treated as individual entities, extra computational overhead is introduced by the fact that the homogeneity of a region that is covered by multiple rules, is unnecessarily reassessed.
  5. Disjoint rules could be more easily interpreted and accepted by users in some applications. However, in other applications where points may

belong to different trends users might find overlapping rules natural and more informative.

However, there are some penalties to be paid for the anticipated gains from evolving disjoint rule-sets.

1. Computationally expensive genetic operators with semi-stochastic constrained functionalities are required to preserve the disjointness of rules in the chromosome.
2. Arbitrary-shaped clusters may be captured using fewer and more generic rules when rule overlapping is allowed, as depicted in figure 5.15.

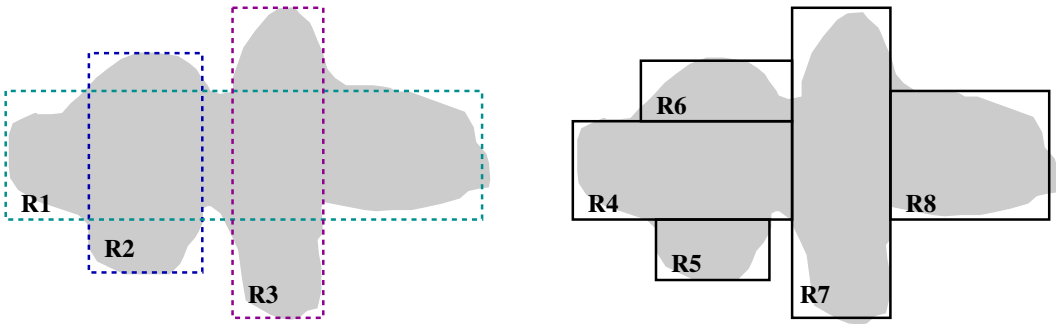


Figure 5.15: Rule Overlapping may Yield Short and Generic Approximations for Arbitrary-shaped Clusters

3. Sparse parts (if any) of arbitrary-shaped clusters sticking out from the backbone of the clusters are lost as a result of eliminating sparse candidate rules.

The thesis investigates only the discovery of disjoint partitions, but future research might explore the use of individuals with overlapping rules.

5.6.2 Preference Bias

Preference or *search bias* refers to any criterion (except consistency with the data) that is used to determine how the system traverses the search space [67]. Often, this kind of bias takes the form of heuristics for *a*) assessing the quality of candidate solutions, *b*) choosing the best ones, and *c*) propagating the knowledge

encapsulated in current solutions into subsequent iterations by generating new candidate solutions based on the current best solutions.

- **Maximisation of Point Coverage:** Like typical EAs, the most prominent kind of preference bias used in $\mathcal{NOCEA}$ is based on the fundamental principle of *survival of the fittest* as expressed by the fitness function. The performance of candidate solutions is measured in the context of the underlying fitness function (section 5.5). The fitness of an individual heavily determines the probability that the individual will survive into and be selected for reproduction in succeeding generation(s). In general, the EA search paradigm offers two opportunities for biasing selection of candidate solutions: *a*) selection for mating (reproduction), and *b*) selection of individuals from the parent and child populations to produce the new population (replacement) [11]. Although a number of different biased selection methods have been proposed in the literature, in essence, all these heuristics rely on the assumption that the fittest individuals, i.e. high data coverage, must receive preference as candidate solutions.

- **Elimination of Sparse Rules:** Sparse rules represent spurious relationships with minor statistical significance. Additionally the computational burden to store and manipulate individuals comprising many clustering rules is very high for high dimensional datasets. Therefore, $\mathcal{NOCEA}$ instantly eliminates sparse rules.

- **Enforcement of Rule Homogeneity:** Not all axis-aligned non-sparse rules are necessarily good sub-solutions to the clustering problem. Perhaps the most important requirement, from a clustering point of view, is to ensure that the space enclosed by a feasible rule is as uniformly filled with points as possible. The natural interpretation of a homogeneous rule is the *absence of any strong inter-attribute correlation* for the data inside the rule. Under such circumstances, the boundaries of the rule along with its data coverage and density can adequately describe the data distribution. $\mathcal{NOCEA}$ employs the task-specific repair operator to form rules with homogeneous data distribution.

5.7 Homogeneity Operator

This section describes a task-specific genetic operator, the *repair* or *homogeneity* operator. The repair operator manipulates, when necessary, candidate rules so that the space enclosed by the resultant variations, i.e. rules, have quasi-homogeneous data distribution. The terms repair and homogeneity are used interchangeably throughout the thesis.

The repair operator relies on the $\mathcal{UDA}$ algorithm (section 3.4) to identify $\mathcal{U}$-regions along the orthogonal uni-dimensional projections of candidate rules. Recall that a $\mathcal{U}$-region is defined as a set of contiguous bins with small histogram value variation. The repair operator exploits the observation that cleanly separable univariate $\mathcal{U}$-regions are "signatures" of clusters in higher dimensional spaces [5, 7, 23, 72, 74].

Finally, the repair operator considers only *non-sparse* $\mathcal{U}$-regions to suppress the subsequent formation of spurious rules over-fitting the data and to reduce computation. A univariate $\mathcal{U}$-region is deemed as *non-sparse* if its data coverage (i.e. percentage of total points falling onto that region) exceeds the standard input *sparsity threshold* $T_s \in (0, 1]$ (see section 5.3).

5.7.1 Motivation for Homogeneity

The fitness function proposed in section 5.5 is totally blind to the quality of the clustering results, *solely* seeking to maximise data coverage. In particular, the fitness function 5.27 lacks any bias that would yield:

- effective discrimination of clusters
- separation of the genuine clusters from the noise regions
- precise approximation of clusters
- homogeneous data distribution in the space enclosed by candidate rules

In essence, since there is no constraint to prevent rules from growing arbitrarily, $\mathcal{NOCEA}$ would easily produce *super-solutions*, e.g. highly-fit individuals covering

substantial parts of the feature space $\mathcal{F}$, or in the worst case all of $\mathcal{F}$. Under such circumstances the meaningfulness of clustering may be easily called into question.

5.7.2 Natural Interpretation of Homogeneous Rules

Unlike other operators, e.g. mutation, recombination and generalisation that are used to traverse the search space, the repair operator concentrates on yielding high quality clustering rules with homogeneous data distributions. The natural interpretation of an homogeneous rule is *the absence of any strong inter-attribute correlation* for the points covered by the rule. As a result, the boundaries of such a rule, along with its data point coverage and density, accurately describe the data distribution. In contrast, the descriptor of a non-homogeneous rule must be accompanied with the types and localities of correlations occurring within the given rule.

From a statistical viewpoint, a d-dimensional rule $\mathcal{R}$ is homogeneous if each cell that is enclosed by $\mathcal{R}$ contains approximately the same number of points. However, creating a histogram that counts the points contained in each cell is infeasible in high dimensional spaces because the number of cells is exponential with the dimensionality. As a result of the sparsely filled space it is impossible to determine the type of distribution with sufficient statistical significance [55]. Notice that the number of available points cannot grow exponentially with the dimensionality, which, in turn, means that the vast majority of points map into different cells and there are many empty cells. The only thing that can be easily verified is that any axis-parallel projection of a set of uniformly generated points follows a quasi-uniform distribution. This observation, along with the fact that clusters become separated because of the different extent of point concentration (density) motivated the design of the repair operator. The repair operator applies several statistical tests to each candidate rule independently as described in sections 3.4.1 - 3.4.3.

5.7.3 Principles of Homogeneity Operator

This section describes in depth how the repair operator combines the three homogeneity tests ($\mathcal{HT}_1$, $\mathcal{HT}_2$, and $\mathcal{HT}_3$) of the $\mathcal{UDA}$ algorithm (section 3.4) to ensure that the space enclosed by all feasible rules has a quasi-homogeneous data distribution. In particular, all dimensions of a candidate rule $\mathcal{R}$ undergo the following processing stages with a random order:

1. Construction of Smoothed Frequency Histogram

Initially, $\mathcal{NOCEA}$ computes the smoothed frequency histogram along the orthogonal univariate projection of the current dimension using the binned KDE (Kernel Density Estimation) with the boundary correction as described in section 3.3.5. It is essential to construct histograms that allow both the detection of significant differences in density and that have smoothed out local data artifacts.

Unlike the classical frequency and density histograms (section 3.2), the KDE method is insensitive to the placement of the bin edges and creates a reasonably smooth approximation of the real density. The latter property is essential for low-to-moderate density rules where the traditional frequency histogram tends to be very jagged making thus difficult to locate non-sparse $\mathcal{U}$-regions. To improve scalability when constructing the smoothed frequency histograms, $\mathcal{NOCEA}$ employs a binned version of the KDE method as explained in section 3.3.3. Henceforth, the term density or frequency histogram will refer to a binned KDE histogram as defined in section 3.3.5, unless otherwise stated.

The practical implementation of the KDE during the repairing stage requires the specification of the bandwidth h, which controls the smoothness of the frequency histogram. A simple solution would be to directly use the automatic normal scale bandwidth selection rule (formulae 3.4) as described in section 3.3.2. However, for non-normal data distributions, e.g. multi-modal or heavily skewed distributions, the statistical performance of formulae 3.4 is poor [92, 94, 97].

We propose here a modification of the automatic bandwidth selection algorithm of section 3.3.2 to adapt h to the local characteristics of the data distribution. The algorithm for a dimension is:

1. Split dimension into k (e.g. k=4) equi data coverage segments.

2. Apply the oversmoothing Normal Reference Rule (formulae 3.4) for each segment independently to obtain the local bandwidths, $h_{(i)} = 1.144\,\sigma_{(i)}\,N_{(i)}^{-1/5}$, where $N_{(i)}$ and $\sigma_{(i)}$ denote the number of points and the standard deviation of data in the ith segment, respectively.

3. Compute a provisional bandwidth h by scalarasing the local bandwidths using the weighted-sum method $h = \sum_{i=1}^{k} \left(\frac{N_{(i)}}{N_{(1)}+\ldots+N_{(k)}} \right) h_{(k)}$

4. Using the bandwidth found in step 3, construct a smooth frequency histogram based on the binned KDE with boundary correction as explained in section 3.3.5.

5. Apply the $\mathcal{UDA}$ (section 3.4) to the smooth frequency histogram obtained in step 4 to locate non-sparse $\mathcal{U}$-regions.

6. If no non-sparse $\mathcal{U}$-regions can be found in step 5, set the bandwidth (h) to the value found in step 3 and exit. Otherwise, repeat steps 3-4 to the newly formed $\mathcal{U}$-regions to compute the final bandwidth.

Having determined the smoothing bandwidth (h), $\mathcal{NOCEA}$ builds the final smooth frequency histogram (section 3.3.5) for the current dimension.

2. Detection of Cutting Planes and Histogram Splitting

Then, the repair operator reapplies $\mathcal{UDA}$ to the smooth frequency histogram to detect valid cutting planes. If no splitting points were found the repair operator proceeds with the next, randomly selected, dimension. Otherwise, the original rule $\mathcal{R}$ is split along the cutting planes of the current dimension, and is discarded. Each newly formed rule undergoes stages 1-2 recursively in all dimensions. If there is a dimension with no non-sparse $\mathcal{U}$-regions the original rule $\mathcal{R}$ is simply discarded without creating new ones. Finally, if no splitting sites are detectable along any dimension the original rule $\mathcal{R}$ is finally deemed *homogeneous* and is not processed further.

Repairing Example An example of repairing is shown in figure 5.16, where the candidate non-homogeneous rule $\mathcal{R}$ of figure 5.16(a) is hierarchically decomposed into a set of disjoint-feasible rules (figure 5.16(b)) using axis-aligned cutting planes that are denoted by dashed lines. Evidently, as the repair operator progresses the refined rules become increasingly more homogeneous.

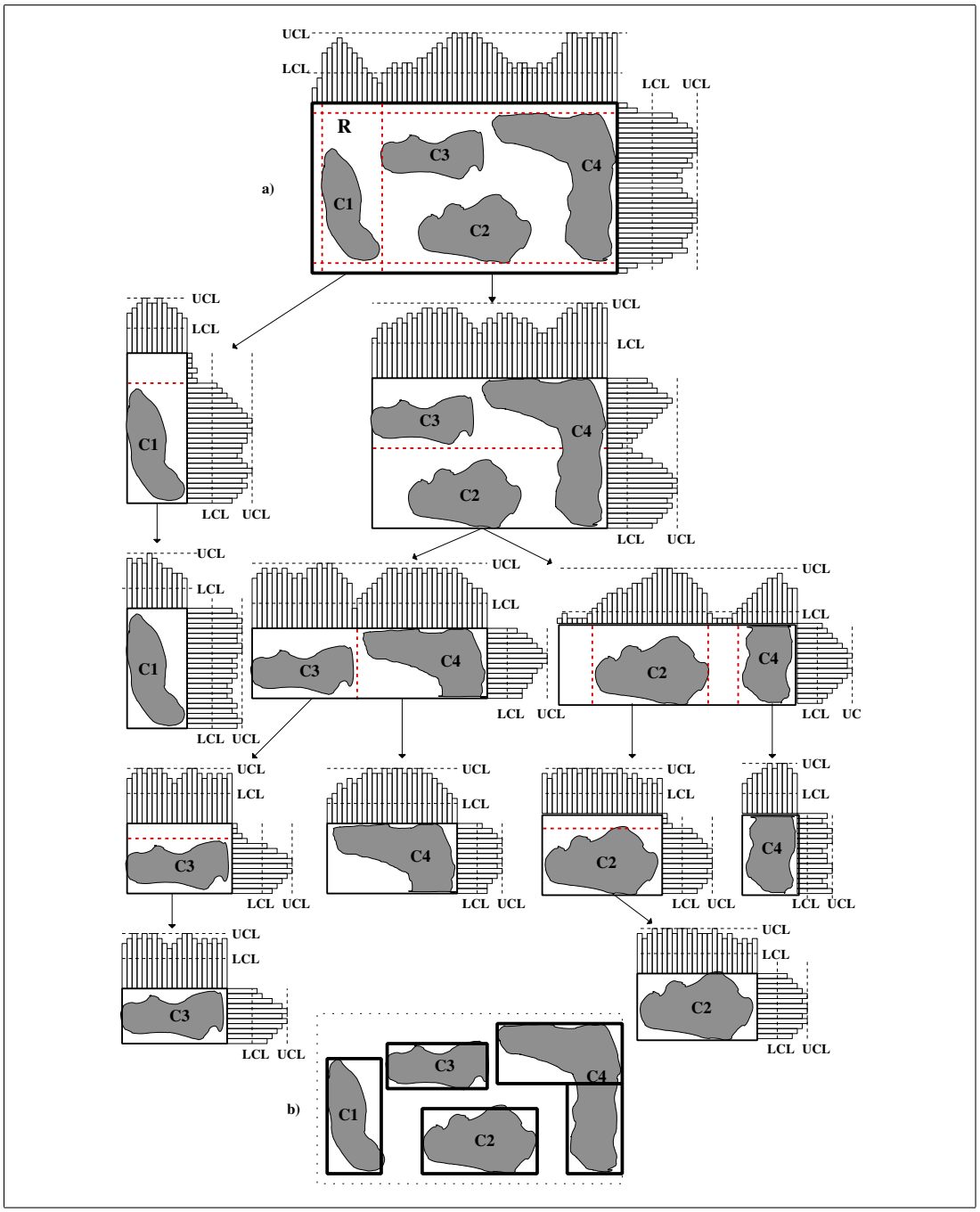


Figure 5.16: The Repairing of a Non-homogeneous Rule

5.7.4 Localised Homogeneity Analysis

Motivation Behind Localised Homogeneity Analysis

Often, real-world databases contain subsets of correlated dimensions forming arbitrary-shaped clusters, e.g. linear or higher order inter-attribute dependencies, with flat univariate orthogonal projections. Additionally, in the uni-dimensional projections some clusters may overlap, and thereby not be distinguishable. In other words, since uni-dimensional orthogonal projections flatten all inter-attribute correlations existing in higher dimensional spaces, it may not always be feasible to detect the boundaries of arbitrary-shaped clusters using a non fully-dimensional axis-parallel partitioning scheme. A representative example is shown in figure 5.17(a) where both projections fail to reveal the strong inter-attribute correlations existing inside the candidate rule $\mathcal{R}$. One possible solution is to use general *contracting*, e.g. non-axis aligned projections [55], but this approach is expensive since the number of potentially interesting projections is very large.

Localised Homogeneity Analysis Algorithm

$\mathcal{NOCEA}$ tackles the problem of detecting strongly correlated dimensions using the same principal of examining uni-dimensional orthogonal projections, but the analysis is now more *localised*. In particular, rather than considering the entire rule $\mathcal{R}$, the ordinary repair operator is applied in appropriately selected sub-regions of $\mathcal{R}$. The algorithm is as follows:

1. **Split Original Rule:** Initially, the original rule $\mathcal{R}$ is tessellated into equal data coverage disjoint sub-regions, each containing approximately $T_s N$ (sparsity level) points. More specifically, $\mathcal{R}$ is recursively split by applying a single cutting plane at a time, which passes along the centre of gravity of the dimension with the longest interval. The rational behind the splitting of $\mathcal{R}$ is to perform a localised homogeneity analysis in the hope of reducing the harmful effect of the joint projection of multiple clusters that make the histograms appear quasi-uniform.

2. **Repairing of New Rules:** All the newly formed rules from the previous stage undergo repairing, as described in section 5.7.3.

3. **Generalisation of Repaired Rules:** Since the splitting of rules in stage 1 may result in cutting homogeneous clusters, the generalisation operator is applied to the rule-set to recover from any wrongly done splitting.

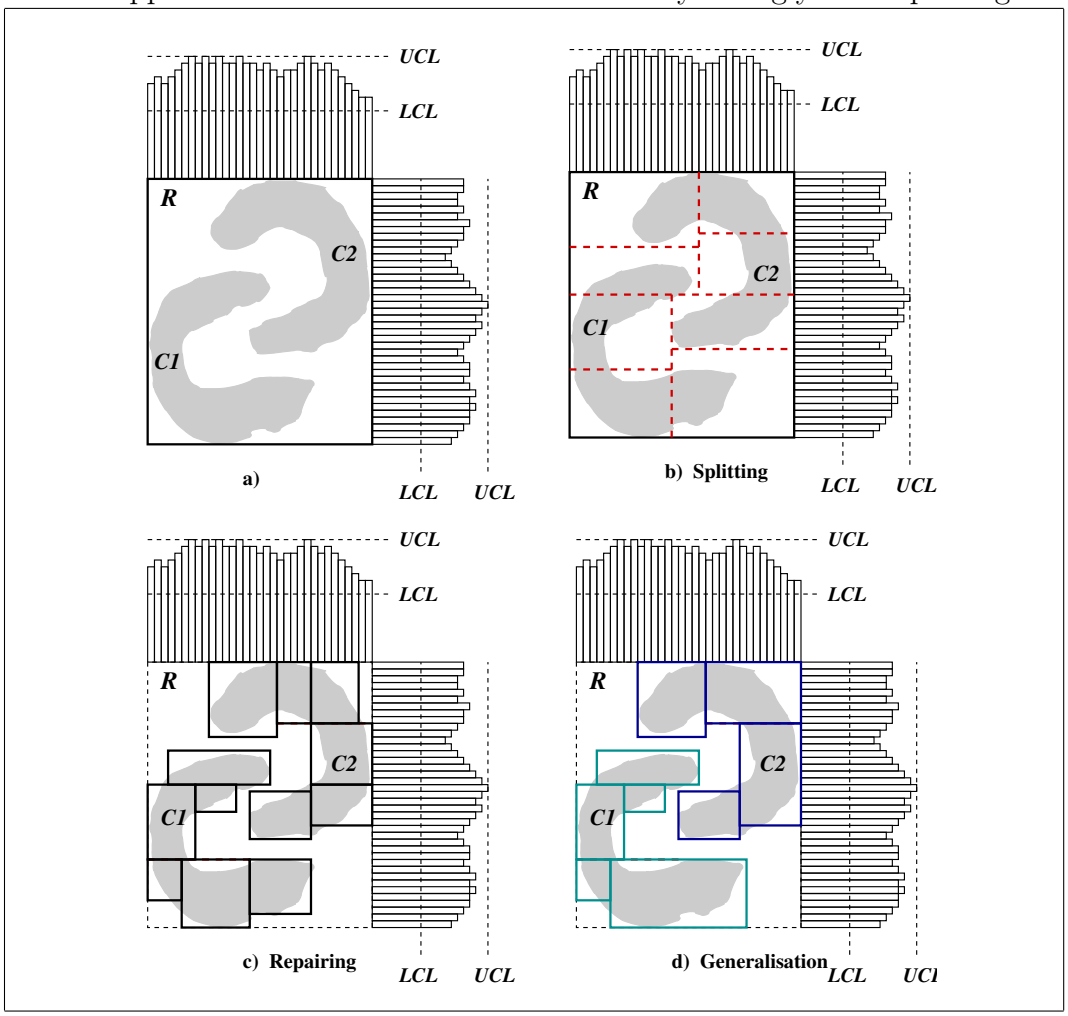


Figure 5.17: Localised Homogeneity Analysis to Repair Strong Correlations

The probability that a candidate rule undergoes localised rather than ordinal repairing is set to a very small value, e.g. 0.05.

Localised Homogeneity Analysis Example

The merit of localised homogeneity analysis is shown in figure 5.17(a), where the uni-dimensional orthogonal projections of the original rule $\mathcal{R}$ reveal no correlation. In contrast, after the balanced splitting of $\mathcal{R}$ (figure 5.17(b)) the ordinal repair operator can easily detect and fix the discontinuities in data distribution in the resultant rules as shown in figure 5.17(c). Finally, generalisation is used to recover from wrongly done splitting actions in stage 1, as shown in figure 5.17(d).

5.8 Recombination Operator

This section presents $\mathcal{NOCEA}$'s novel recombination scheme, the $\mathcal{O}$verlaid $\mathcal{R}$ule $\mathcal{C}$rossover (or $\mathcal{ORC}$) operator. $\mathcal{ORC}$ semi-stochastically recombines the genetic material of individuals to preserve and propagate intact rules from parents to offspring, and at the same time to blend them in the hope of creating even better individuals. The constrained functionality of $\mathcal{ORC}$ ensures the generation of offspring with disjoint rules.

5.8.1 Motivation for Recombination

The aim of *recombination* or *crossover* in EAs is to combine the best characteristics of highly fit individuals in the hope of creating even better solutions [11, 12, 47, 71]. As the EA is unaware of what characteristics account for the good performance, the best it can do is to recombine characteristics at random. Stochastic crossover may lead to deleterious, neutral, or beneficial changes in the behaviour (performance) of individuals. However, due to the selective pressure of EAs, poorly performing offspring will not survive for long. During recombination, parent-solutions are selected from the mating pool at random and chromosome fragments are exchanged between them to create the offspring.

$\mathcal{NOCEA}$'s fitness function (section 5.5) has no bias towards rules of specific type, e.g. generic or highly-dense, and consequently *each* rule can be viewed as an important *building block* or *good schema*. This is because each homogeneous rule contributes to the fitness, regardless of its size, geometry, and data coverage. Thus, crossover must not simply preserve and propagate intact rules from parents to offspring, but at the same time must blend them with rules present in other parents in the hope of producing even fitter solutions. The obvious caveat is that the manipulation of the genetic material by crossover must always yield *non-lethal* individuals. A non-lethal solution comprises non-sparse, semantically valid and disjoint rules. In analogy to binary EAs where disruption means the breaking up of critical schemata (bit combinations) conveying high fitness, in $\mathcal{ORC}$ disruption after crossover is the splitting of parental rules to create non-lethal offspring.

5.8.2 Principles of Recombination

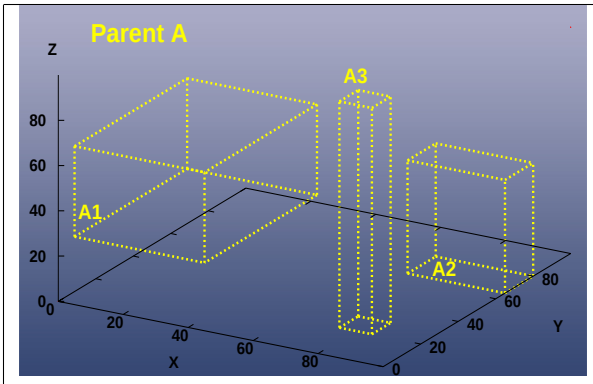
$\mathcal{NOCEA}$ employs a specialised $\mathcal{ORC}$ recombination scheme whose functionality is geared toward: *a*) *minimisation of disruption of the genetic material*, *b*) *elimination of positional bias*, *c*) *minimisation of distributional bias*, and *d*) *generation of non-lethal offspring*.

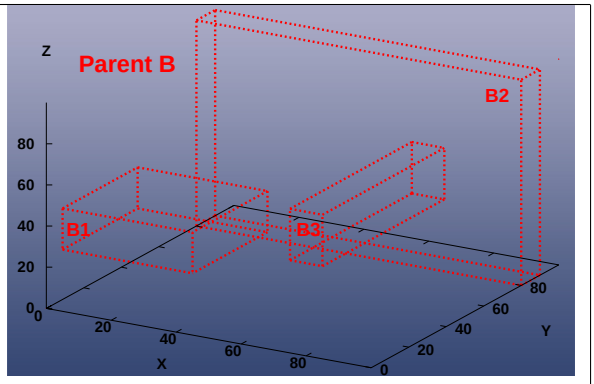
Instead of stochastically exchanging chromosome fragments, i.e. rules, between the parents, $\mathcal{ORC}$ initially creates a clone of each parent and then overlays it with the rules from the other parent. Those parts of rules from the second parent that do not intersect with the rules from the first parent are directly copied in the offspring while the rest are discarded. The principles of $\mathcal{ORC}$ are explained with the help of the example depicted in figure 5.18, where the colour of a rule indicates its parental origin. $\mathcal{ORC}$ operates on two parent solutions at time and creates two non-lethal offspring in a way that rule disruption is minimised.

The main processing stages in $\mathcal{ORC}$ are as follows:

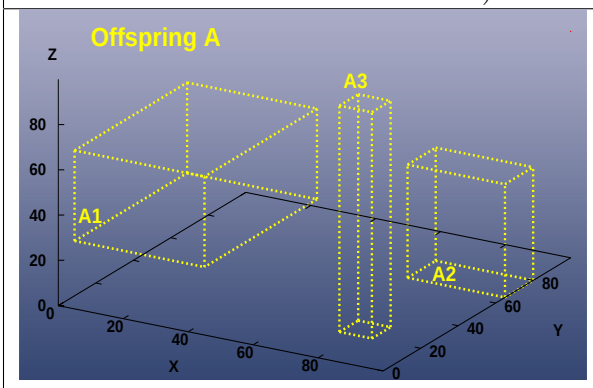
1. **Cloning Parents:** Initially, each parent transmits intact its rules to one of the generated offspring. Henceforth, the parent that is initially cloned to create an offspring is termed as the *primary parent* of that offspring, while the other parent is termed *secondary parent*. By firstly cloning the parents, $\mathcal{ORC}$ achieves propagating each rule present in the parental chromosomes in at least one offspring. It can easily be observed from figure 5.18(b) that each offspring inherits, at first glance, all rules from its primary parent.

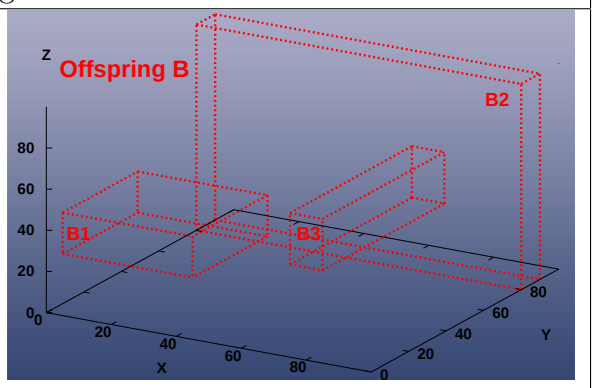
2. **Exchanging Disjoint Rules:** In the next stage, the genetic material of each offspring is enhanced by directly copying all rules from its secondary parent that do not intersect with the rules of the offspring. For instance, offspring A, in figure 5.18(c), receives unaltered rule $\mathcal{B}_2$ from its secondary parent B, since such an operation yields a non-lethal solution. $\mathcal{ORC}$ proceeds then by identifying, for each offspring, those rules in its secondary parent that are fully covered by rule(s) in the chromosome of the offspring, e.g. rule $\mathcal{B}_1$ in respect of offspring A. These rules (if any) are effectively omitted from further processing because the d-dimensional regions that are



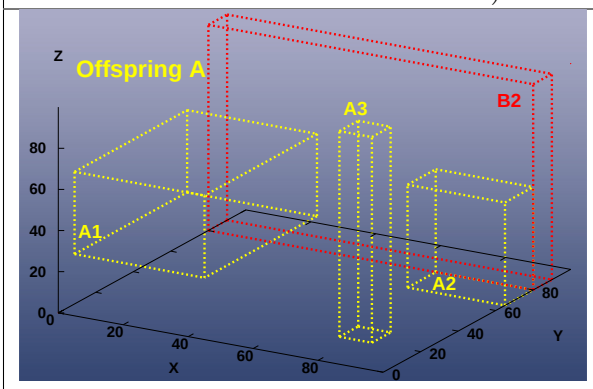


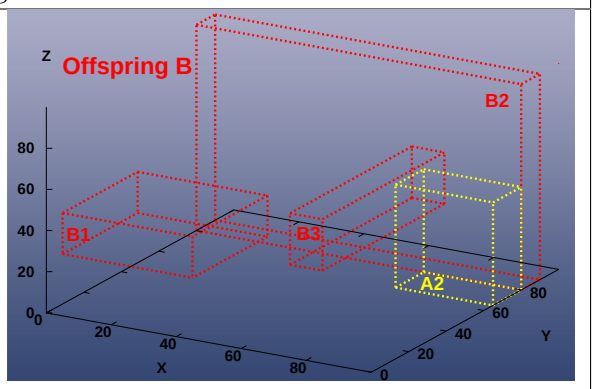
a) Selecting Parents



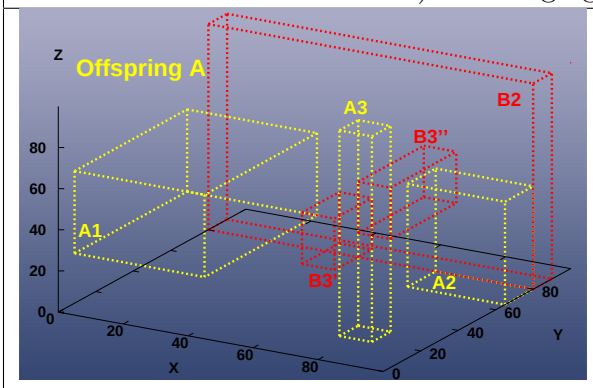


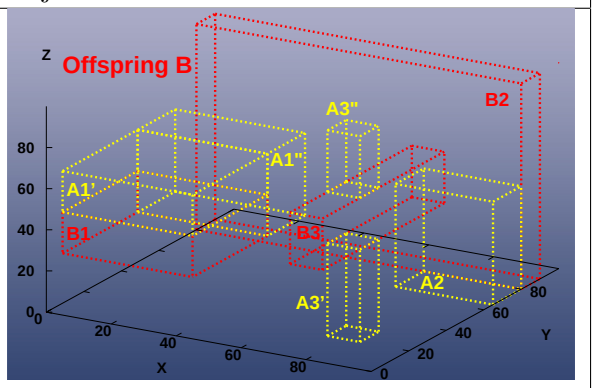
b) Cloning Parents





c) Exchanging Disjoint Rules





d) Resolving Rule Overlapping, and Exchanging Newly Formed Disjoint Rules

Figure 5.18: $\mathcal{ORC}$ Recombination Principles

enclosed by them, are entirely known to the target offspring. So far, no disruptive effect is evident, meaning that the resultant offspring at this stage are *at least* fit as their primary parents. Notice that up until now $\mathcal{ORC}$ is a purely deterministic operation.

3. **Resolving Rule Overlapping:** What follows next is the splitting algorithm of figure 5.19 that stochastically resolves all instances of overlapping between an offspring and the remaining rules of its secondary parent. Let $\mathcal{V}$ be a vector containing the rules of the secondary parent that partially intersect with the offspring. In essence, during a single iteration, a randomly selected rule $\mathcal{R}$ from $\mathcal{V}$ is split along a randomly selected cutting plane passing through a proper bound of an offspring rule that intersects with $\mathcal{R}$. A single splitting operation yields a set of new rules where one of them is disjoint with the offspring rule. After the completion of the splitting algorithm all the newly formed non-sparse rules (if any) are copied into the offspring, enriching its genetic material and consequently improving its performance.

1. Randomly select the jth rule ($\mathcal{R}_j$) from $\mathcal{V}$

2. Randomly select the ith rule ($\mathcal{R}_i$) of the offspring intersecting with $\mathcal{R}_j$

3. Randomly select the splitting dimension s such that (l_{sj}<l_{si}) or (u_{si}<u_{sj})

4. If (l_{sj}<l_{si}) $\wedge$ (u_{si}<u_{sj}) randomly select whether the cutting plane passes through the lower or upper bound of $\mathcal{R}_i$. Otherwise, if (l_{sj}<l_{si}), choose the lower bound of $\mathcal{R}_i$ while if (u_{si}<u_{sj}) choose the upper bound of $\mathcal{R}_i$

5. Split $\mathcal{R}_j$ along the selected bound and discard $\mathcal{R}_j$

6. Discard any newly formed sparse rules

7. Copy the new rules having no intersection with the current offspring rules to the offspring and insert the remaining new rules into $\mathcal{V}$

8. If $\mathcal{V}$ is empty exit, otherwise go to step 1

Figure 5.19: Algorithm for Resolving Rule Overlapping in $\mathcal{ORC}$

5.8.3 Properties of the Recombination Operator

From an exploratory viewpoint, $\mathcal{ORC}$ is of limited power in the sense that the resultant variations are proper subsets of the union of the parental rules. In other words, although crossover can introduce new rules that are not present in the current population, the new genetic material represents regions in the feature space $\mathcal{F}$ that were previously identified by at least one parent. However, provided that the union of two parental chromosomes assembles the optimal solution, $\mathcal{NOCEA}$ has the exceptional ability to reach this optimal point of the search space in a *single* $\mathcal{ORC}$ operation.

In short, the salient features of $\mathcal{ORC}$ are:

- **Non-lethal Variations:** Despite its semi-stochastic functionality, $\mathcal{ORC}$ always guarantees the generation of non-lethal offspring.
- **Beneficial Variations:** $\mathcal{ORC}$ improves the mean performance of the population because the offspring are always at least as fit as their primary parents.
- **No Positional Bias:** $\mathcal{ORC}$ has no positional bias (section 2.4.9) because the transmission of rule-genes to offspring is *absolutely* independent of their relative positions on the parental chromosomes.
- **Distributional Bias:** $\mathcal{ORC}$ has a distributional bias (section 2.4.9) in the sense that the expected number of rules that are transmitted to an offspring is bounded minimally by the number of rule-genes of its primary parent. Concerning the secondary parent, clearly the variation associated with the number of transmitted rules is expected to be relatively large during the early stages of the search, provided of course that individuals were initialised randomly. However, the variation reduces as the search progresses because the individuals become increasingly more similar. No other distributional bias is present.

5.9 Generalisation Operator

This section presents *Generalisation*, a novel genetic operator that delivers end-user comprehensibility and simplification of the clustering results. Generalisation has a parsimony pressure in the sense that it strives to minimise the length of individuals and to make rules as generic as possible. This is achieved by replacing adjacent (section 5.9.3) rules satisfying several conditions with a single and hopefully more generic rule. Additionally, generalisation improves scalability because the overall computational complexity of $\mathcal{NOCEA}$ heavily depends on the total number of rules that are processed each generation.

5.9.1 Motivation for Generalisation

There are several incommensurable factors affecting the comprehensibility of the discovered knowledge, e.g. format of the knowledge representation language, familiarity with the application domain, syntactical complexity, level of knowledge abstraction [43, 81]. However, to avoid difficult subjective issues, it is common in DM literature to assess knowledge comprehensibility by considering just two objectives: *a*) the *length of rules*, that is, the number of conditions involved in the antecedent part, and *b*) the *size of the rule set*, that is, the number of discovered rules. In general, *the smaller the rule-set and the shorter the rules the more comprehensible the knowledge is* [43].

The simplification of the antecedent part of the clustering rules along with the minimisation of the size of the rule-set are precisely the goals of generalisation. The generalisation operator strives: *a*) *to replace pairs of adjacent rules with a single and hopefully more generic rule*, and *b*) *to encourage the discovery of generic rather than specific rules because a relatively generic rule is more likely to detect irrelevant features* (see section 5.11), *which permit "dropping" the corresponding conditions in the antecedent part.*

The motivation for generalisation is clearly demonstrated in figure 5.20, where similar performance, i.e. data coverage, is achieved with radically different genetic material. Undoubtedly, the solution depicted in figure 5.20(b) is the more

comprehensible since it comprises less rules than the individual shown in figure 5.20(a). Additionally, the capturing of the vertical elongated cluster in figure 5.20(b) with a single-generic rule allows dropping the corresponding condition from the antecedent part of that rule because it is extended to the entire domain.

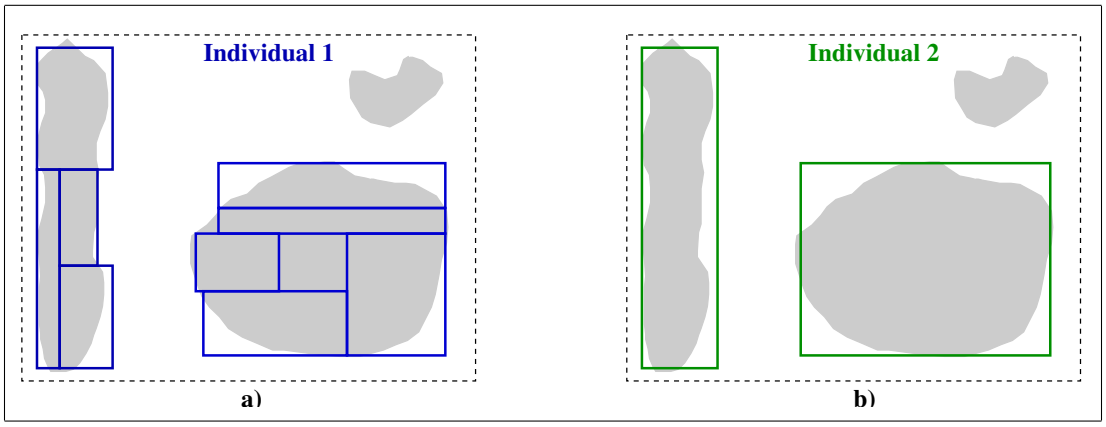


Figure 5.20: Motivation for Generalisation

5.9.2 Preference Bias of Generalisation

Since minimising the size of the rule set improves both comprehensibility and efficiency, one might wonder: why not include some parsimony factor to the fitness function to introduce explicit bias toward small rule-sets? The answer to this question is straightforward: *because it is not clear how to weight the effects of point coverage and size of the rule set.* For instance, should an individual become fitter when discovering a new rule with relatively small coverage given the increase in the size of the rule set? Alternatively, should only the discovery of rules with moderate-to-high point coverage outweigh the effect of increasing their number? An obvious drawback associated with the latter case is that rules with relatively small coverage either will be missed completely, or their discovery and inclusion in the individuals will be postponed until all moderate-to-high coverage rules have been recovered. This in turn may be a reason for running $\mathcal{NOCEA}$ for more generations, especially in cases where there are isolated clusters. To avoid these difficulties $\mathcal{NOCEA}$'s fitness function simply measures data coverage, but there is a stochastic refinement of the discovered knowledge through generalisation, which eventually delivers the desired minimisation in the number of rules.

5.9.3 Principles of Generalisation

Generalisation is always applied to a *pair* of adjacent rules at a time satisfying some conditions, and produces a *single* and hopefully more *generic* rule. Two d-dimensional rules are *adjacent* if they have a common face, i.e. there are d-1 dimensions where there is an intersection between the rules and additionally, there is one dimension where the rules are contiguous.

Let us assume that the pair of ith and jth rules undergo generalisation along the gth dimension, which is the dimension where the rules have a common face, i.e. they are "touching" (see page 100 for a formal definition) . The original rules are put together in an incomplete generalisation G, that must not overlap with neighbouring rules. To achieve this, the generalisation operator firstly determines the backbone of the rule G that will eventually replace the two rules. In particular, the lower (l_k) and upper (u_k) bounds of G are:

$$[l_k, u_k] = \begin{cases} [\min(l_{ki}, l_{kj}),\ \max(u_{ki}, u_{kj})]\,, & \text{if } k = g \\ [\max(l_{ki}, l_{kj}),\ \min(u_{ki}, u_{kj})]\,, & otherwise \end{cases} \tag{5.28}$$

Having determined the incomplete generalisation G, the operator proceeds dimension by dimension in a random order. More precisely, G is gradually expanded along every dimension, apart from g, so that no overlapping with neighbouring rules occurs. The growing of G to the left and to the right in the kth ($k \neq g$) dimension is bounded by $\min(l_{ki},\, l_{kj})$ and $\max(u_{ki},\, u_{kj})$, respectively. However, it is likely that the expansion will be limited if there are rules that may overlap with a fully expandable G. For instance, the generalisation of adjacent rules $\mathcal{R}_1$ and $\mathcal{R}_2$ shown in figure 5.21(a) yields initially the incomplete generalisation G shown as grey-shadowed region in figure 5.21(b). Concerning the vertical axis, G is clearly expandable up to the left vertical bound of $\mathcal{R}_3$ rather than to the right-vertical bound of $\mathcal{R}_2$, because such a growing operation will cause overlapping between G and $\mathcal{R}_3$. After generalisation completes the resultant solution in figure 5.21(c) it comprises fewer and more generic rules compared to the solution that is depicted in figure 5.21(a).

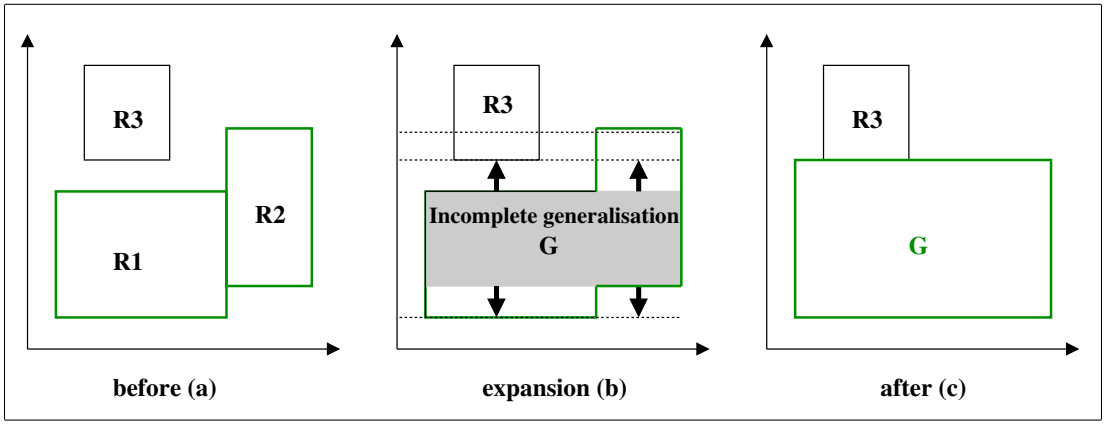


Figure 5.21: Generalisation Principle

5.9.4 Constrained Generalisation

Generalisation is subject to some constraints that help to prevent the formation of rules that are non-homogeneous and/or have significantly lower data coverage compared to the aggregated coverage of the two original rules. To explain the nature of these constraints consider the generalisation examples that are depicted in figures 5.22(a-c), where the relative darkness indicates the density of the clusters.

In the first case - figure 5.22(a), the density of rules $\mathcal{R}_1$ and $\mathcal{R}_2$ differs significantly and therefore the resulting generalisation, which inherits this difference, is non-homogeneous. In the second case - figure 5.22(b), although the rules $\mathcal{R}_3$ and $\mathcal{R}_4$ are of similar density and geometry, their centers are not properly aligned, and consequently the large regions at the top-right and bottom-left corners with unknown density that are added in the generalisation produce a non-homogeneous rule. In both cases (figures 5.22(a-b)), the generalisation is an unsuccessful operation as it generates non-homogeneous rules requiring repairing.

Perhaps the most severe drawback of unconstrained generalisation occurs when there are large differences between the sizes of rules, e.g. $\mathcal{R}_7$, $\mathcal{R}_8$ under generalisation and additionally there exist other rules, e.g. $\mathcal{R}_5$, $\mathcal{R}_6$ in close proximity, as shown in figure 5.22(c). In such cases, it is likely to lose substantial parts of the rules under generalisation to avoid overlapping with rules nearby. As a result, an unconstrained generalisation can substantially degrade the performance of the individual, which in turn, poses a strong obstacle for $\mathcal{NOCEA}$ to converge into an optimal solution.

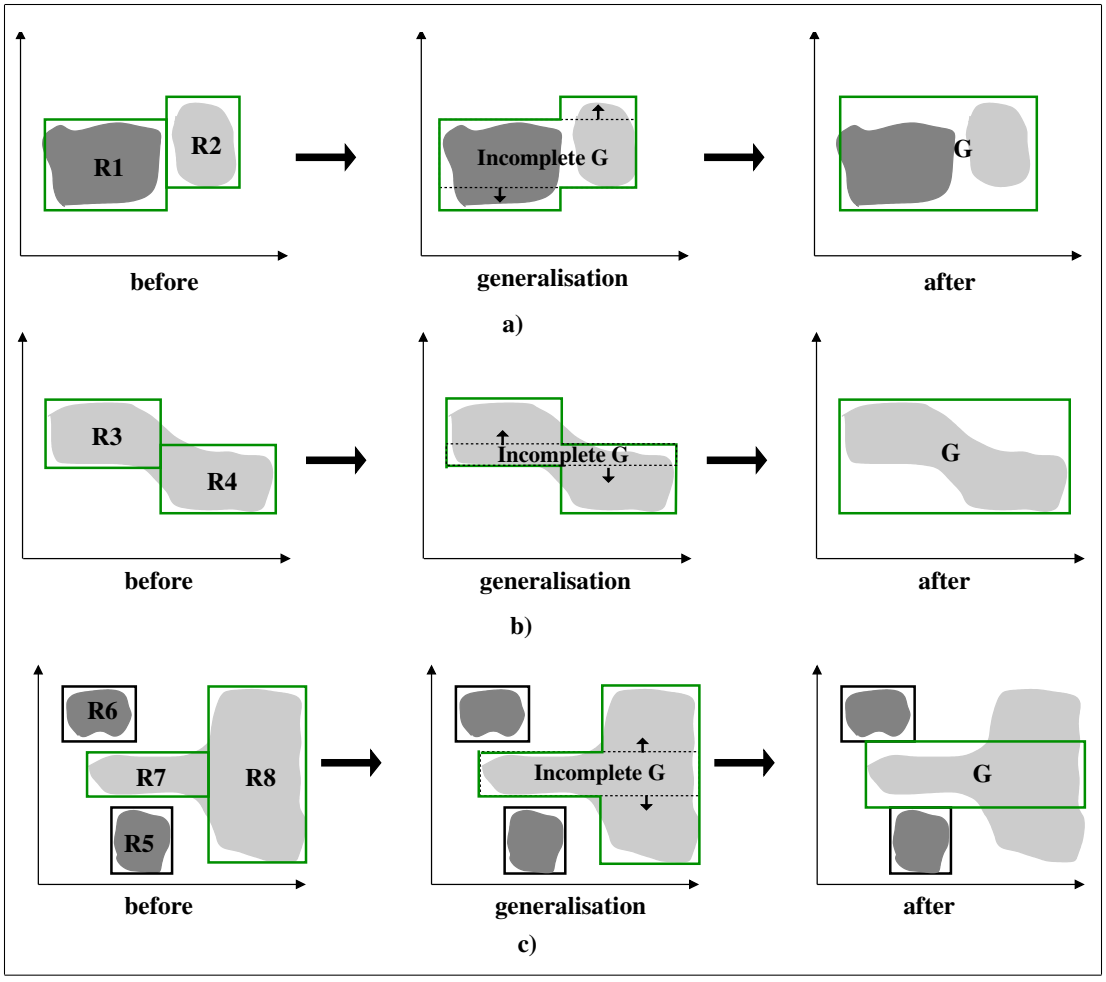


Figure 5.22: Pitfalls of Unconstrained Generalisation

To reduce the severity of the side-effects associated with generalisation, $\mathcal{NOCEA}$ allows the operation to proceed only if the rules under generalisation have similar densities, sizes, and proper alignment. In particular, the pair of adjacent rules $\mathcal{R}_i$ and $\mathcal{R}_j$, are generalised only if the following conditions are true for every dimension $l = 1, ..., d$, excluding g (g: touching dimension for $\mathcal{R}_i$ and $\mathcal{R}_j$):

$$\boxed{\frac{\min(D_i,D_j)}{\max(D_i,D_j)} \geq T_h \quad \text{and} \quad \frac{R_{li} \cap R_{lj}}{u_{lm} - l_{lm}} \geq T_g, \; m \in \{i, j\}} \tag{5.29}$$

where, D_i, D_j denote the density of ith and jth rule, respectively, while $R_{li} \cap R_{lj}$ is the length of the intersection between the two rules in the lth dimension. l_{lm} and u_{lm} are the decoded values (section 5.3) of the lower and upper bound of mth rule in the lth dimension, respectively. The *homogeneity* $T_h \in (0,1]$ and *generalisation* $T_g \in (0,1]$ thresholds are discussed in section 5.14. The first condition prevents generalising rules with very different densities, while the second reflects the requirement of generalising rules with proper alignment and similar sizes.

5.10 Mutation Operators

This section presents two novel, semi-stochastic mutation operators, namely, *Grow-* and *Seed-Mutation*, that provide the main exploratory force in $\mathcal{NOCEA}$. The goal of mutation is threefold: *a) to perform local fine-tuning by randomly increasing the size of existing rules*, *b) to discover previously unknown and potentially promising regions within the enormous feature space* $\mathcal{F}$, and *c) to ensure that every uncovered region in* $\mathcal{F}$ *is accessible to* $\mathcal{NOCEA}$. *Grow-* and *Seed-Mutation* operate under constraints to ensure the formation of individuals with semantically valid and disjoint rules.

5.10.1 Motivation for Mutation

Mutation serves to prevent premature loss of population diversity by randomly sampling new points in the search space. The probability of mutation must be kept small, otherwise the optimisation process degenerates into a random search (section 2.4.10). Typically, an EA mutation operator acts on a single individual at a time and replaces the value of a gene with another, randomly generated value, leading to deleterious, neutral, or beneficial changes in the performance of the individual [11, 12, 47, 71]. From an exploration viewpoint, mutation is particularly useful as it can introduce into an individual a gene value that is not present in the current population. In most GA studies, mutation is treated as a background operator, supporting the recombination operator, by ensuring that all possible combinations of gene values of the search space are accessible to EA. In ES and EP, in contrast, mutation plays the central role in exploring the search space.

Traditionally, mutation disregards semantic linkages among genes in the chromosome in the sense that the positions in the string to undergo mutation and the new values for the mutated genes are determined at random regardless of what happens at other positions in the string. In our case, however, since rules must always be semantically valid and disjoint, the mutation of a particular rule-bound gene is likely to influence or even prevent subsequent mutations in other genes.

5.10.2 Principles of *Grow-Mutation*

Functionality of *Grow-Mutation*

The *Grow-Mutation*, as implied by its name, is primarily used to grow existing rules in an attempt to increase their data coverage, and thereby to make the individuals fitter [87, 88]. Bearing in mind that comprehensibility is a desired property for the discovered knowledge, it seems reasonable to focus on the discovery of as few and as generic rules as possible. Due to its nature, *Grow-Mutation* has a parsimony pressure for small and generic rule-sets, thereby improving comprehensibility and reducing computational complexity. The general form of *Grow-Mutation* can be written as:

$$\boxed{\mu = \mu' + U} \tag{5.30}$$

where μ' and μ denote the integer value of a gene, i.e. lower or upper bound, before and after *Grow-Mutation*, respectively. U is a uniform discrete random variable in [0, μ_{max}] for the upper bound, and [-μ_{max}, 0] for the lower bound. μ_{max}[5] represents the maximum possible modification for a valid expansion that does not produce overlapping rules. Figure 5.23 shows the algorithm for determining μ_{max} if the upper bound u_{ij} of the jth rule ($\mathcal{R}_j$) is mutated along the ith dimension. The derivation of μ_{max} for the lower bound is the dual procedure (see figure 5.24).

1. Find all rules $\mathcal{R}_l$, $l = 1, ..., k$, $l \neq j$, where $u_{ij} < l_{il}$

2. Sort rules in ascending order of l_{il}

3. If the sorted list is empty, set μ_{max} = (m_i - u_{ij} - 1) and exit. Otherwise proceed to step 4

4. Pick the next rule $\mathcal{R}_l$ from the sorted list. If $\mathcal{R}_l$ intersects with $\mathcal{R}_j$ in every dimension excluding ith, set μ_{max} = (l_{il} - u_{ij} - 1) and exit. Otherwise, repeat step 4

5. If no rule in the sorted list satisfies the condition in step 4, set μ_{max} = (m_i - u_{ij} - 1)

m_i:total number of bins in ith dimension, k:number of rules

Figure 5.23: Algorithm for Computing μ_{max} for an Upper *Grow-Mutation*.

[5] $\mu_{max} \in \{0\} \cup Z^+$ and $\mu_{max} < m$, where Z^+ denotes the positive integers while m is the total number of bins in the given dimension.

1. Find all rules $\mathcal{R}_l$, $l = 1, ..., k$, $l \neq j$, where $u_{il} < l_{ij}$

2. Sort rules in descending order of u_{il}

3. If the sorted list is empty, set μ_{max} = l_{ij} and exit. Otherwise proceed to step 4

4. Pick the next rule $\mathcal{R}_l$ from the sorted list. If $\mathcal{R}_l$ intersects with $\mathcal{R}_j$ in every dimension excluding ith, set μ_{max} = (l_{ij} - u_{il} - 1) and exit. Otherwise, repeat step 4

5. If no rule in the sorted list satisfies the condition in step 4, set μ_{max} = l_{ij}

m_i:total number of bins in ith dimension, k: number of rules

Figure 5.24: Algorithm for Computing μ_{max} for a Lower *Grow-Mutation*.

Grow-Mutation is applied with a very small fixed probability, e.g. 0.005, to a single bound of a rule at a time.

Local Fine-Tuning

Figure 5.25 clearly demonstrates the effectiveness of *Grow-Mutation* as a mechanism to perform local fine-tuning. Let us assume that the upper bound of rule $\mathcal{R}_1$ undergoes *Grow-Mutation* along the horizontal axis. After having determined μ_{max} with the algorithm in figure 5.23, $\mathcal{R}_1$ is randomly expanded to the right within the rectangle (*abcd*) that is demarcated by the dashed lines, as shown in figure 5.25(b). Notice that, although the rule $\mathcal{R}_5$ is located in the same hyperplane where the *Grow-Mutation* is taking place, it does not constrain this operation because there is no intersection with $\mathcal{R}_1$ in the vertical axis. Since the new values for the rule boundaries are *randomly* chosen from a window of predefined size (μ_{max}), *Grow-Mutation* may create non-homogeneous rules, e.g. $\mathcal{R}_1$ in figure 5.25(b). In such cases, the repair operator (section 5.7) enforces feasibility on the candidate solutions, as depicted in figure 5.25(c). Clearly the data coverage of $\mathcal{R}_1$ has been increased by the *Grow-Mutation*.

Exploration

Apart from performing local fine tuning, *Grow-Mutation* in association with the repair operator, plays a central role in the exploration of previously unknown

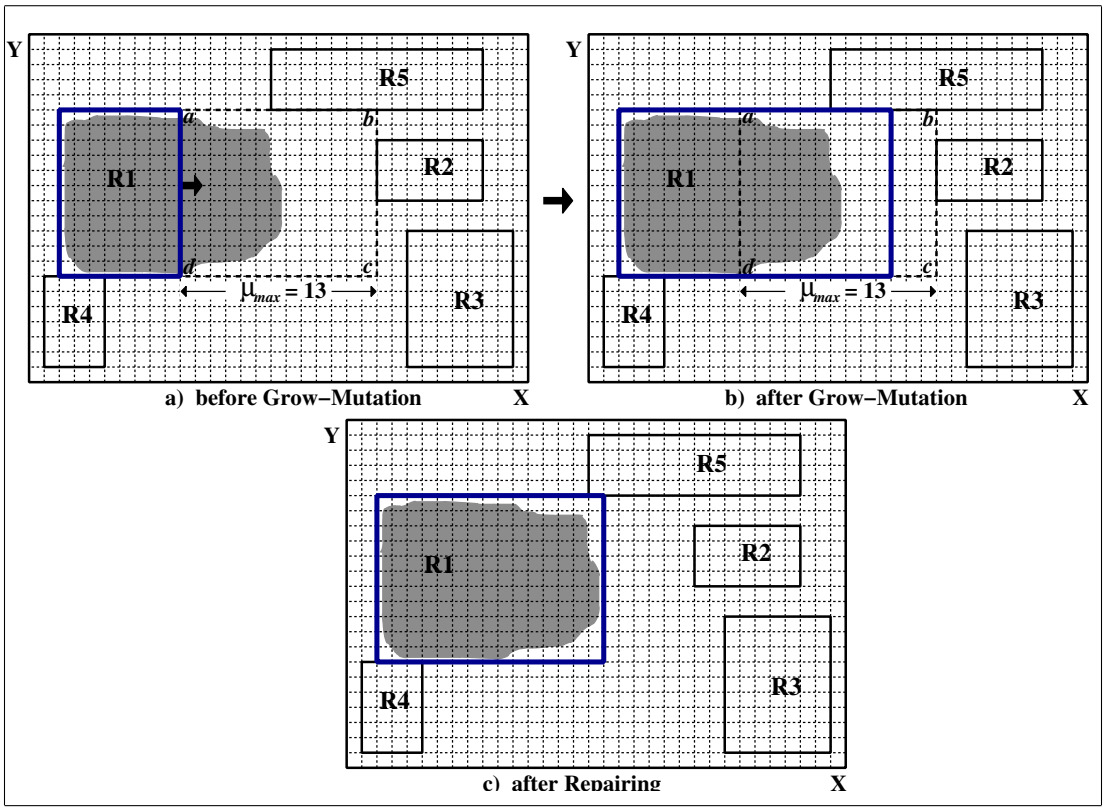


Figure 5.25: Performing Local Fine-Tuning with *Grow-Mutation* and *Repairing*

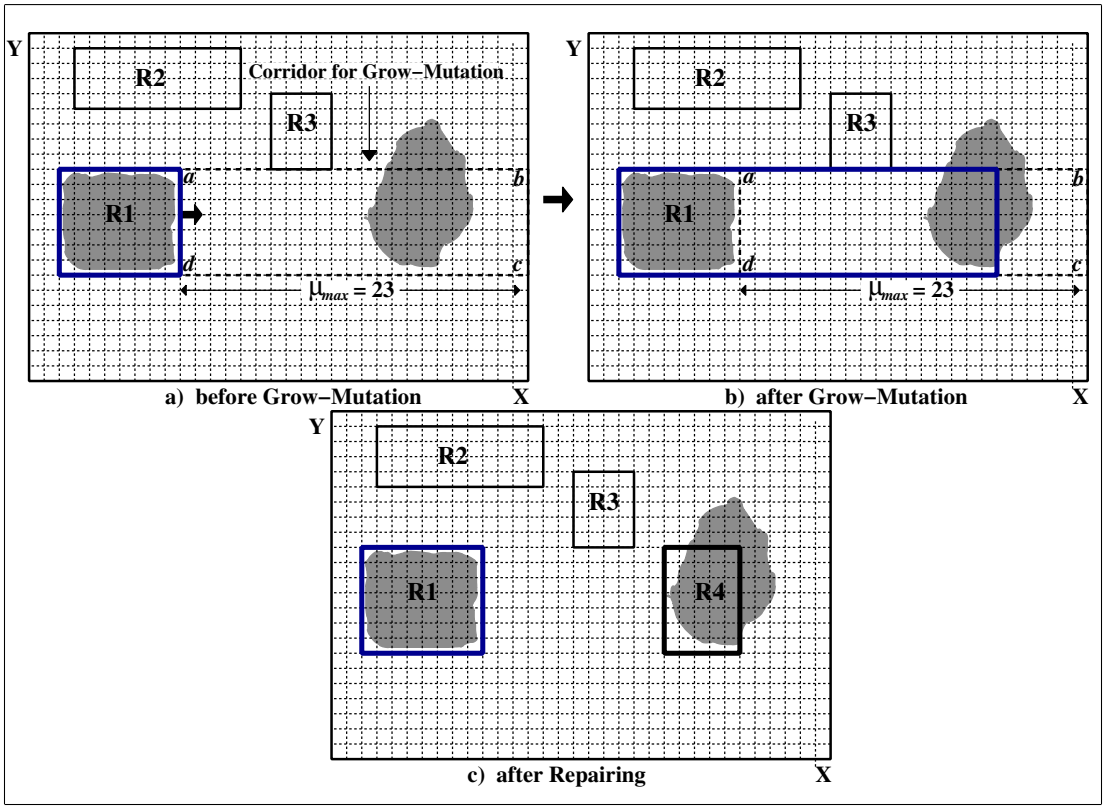


Figure 5.26: Discovering Unknown Clusters with *Grow-Mutation* and *Repairing*

and potentially promising regions. Unlike other *neighbourhood-move* mutation operators, e.g. *non-uniform* [71] or *zero-mean Gaussian* [11], every cluster that intersects with the *d*-dimensional *corridor* - e.g. rectangle (*abcd*) in figure 5.25(a) - along which a *Grow-Mutation* is performed, can be potentially recovered, *regardless* of its distance from the rule under mutation. For instance, the *Grow-Mutation* of the upper bound of $\mathcal{R}_1$ along the horizontal axis in figure 5.26(a), produces an intermediate non-homogeneous rule. However, now the mutated rule $\mathcal{R}_1$ in figure 5.26(b) encloses a significant part of a previously unknown cluster. The subsequent repairing of $\mathcal{R}_1$ in figure 5.26(c) yields two homogeneous rules, where one ($\mathcal{R}_4$) partially covers a newly found cluster.

5.10.3 *Grow-Mutation* as Source of Variation

What Constitutes a Successful *Grow-Mutation*?

Before investigating the effects of *Grow-Mutation* as source of variation in $\mathcal{NOCEA}$, it is necessary to establish an objective definition of what constitutes a successful *Grow-Mutation*. Bearing in mind that the repaired version of an individual replaces the original before the evaluation stage, it is evident that not every alteration of the genetic code made by the *Grow-Mutation* operator is entirely accepted. In fact, given that the repair operator fixes every violation of the rule-homogeneity constraint, only those parts of the alterations that lead to an increase in the data coverage of existing rules or the discovery of new non-sparse rules, are kept, while the rest are discarded. Therefore, a *Grow-Mutation* is regarded as successful when it yields feasible expansions of existing rules or in association with the repair operators, helps discovering new clusters.

Candidates Schemes for *Grow-Mutation*

Concerning real-valued representations, various types of mutations have been proposed in the literature [11]. The simplest mutation scheme would be to select the replacement value for a gene randomly from the entire domain [71]. In general, this type of mutation is independent of the parent solution and thus it may cause losing most of inheritance from parent to offspring that is a fundamental

principle in every EA-based search algorithm. Alternatively, to preserve to some extent intact the ties between parent and offspring, the new value can be created in the vicinity of the parent solution (*creep mutation*), that is, the parent value is randomly mutated within a small window of predefined size [25]. However, if the parent solution resides in a local optimum and the distance from other optima is greater than the step size, creep mutation leads to entrapment [11]. Another step-size control mechanism for mutating real-valued genes is the *non-uniform mutation* [71]. The non-uniform mutation permits large-scale modifications in the early stages of the evolutionary search, thus acting like a random mutation operator, while the probability of creating a solution in the vicinity of parent solution rather than away from it increases over the generations, thus allowing a more focused search. The most popular mutation scheme for real-valued representations is the *zero-mean Gaussian* mutation, where the value of a gene is mutated by adding to it a random number that is drawn from the normal distribution $N(0, \sigma)$. The zero-mean Gaussian mutation operator attempts to create offspring that are *"... on average no different from their parents and increasingly less likely to be increasingly different from their parents..."* [11].

Why Use Random Mutation?

Unlike most EA-based optimisation techniques where a mutation event may have a deleterious impact on the performance of an individual, *Grow-Mutation* has the unusual property of producing *only beneficial or in worst case near neutral changes in the genetic material of individuals*. This is because, regardless of the mutation rate and the amount of modification, the parent solution is always a proper subset of the offspring solution before of course the repairing stage. Therefore, large-scale *Grow-Mutation*s not only do not destroy the inheritance from parent to offspring, but rather allow a fairly robust and fast search, since they accomplish both local fine-tuning and vigorous exploration of new regions simultaneously.

There are several reasons to avoid *neighbourhood-move* mutation operators, e.g. creep, non-uniform or zero-mean Gaussian, in $\mathcal{NOCEA}$. Firstly, determining

appropriate step sizes for every dimension poses a significant challenge, even though various methods have been proposed to tune these strategic parameters on the fly [11, 12]. These methods are either *deterministic* where the step sizes are altered based on some time-varying schedule without incorporating feedback from the search, *adaptive* where the direction and magnitude of change are determined using feedback from the search, or *self-adaptive* where the strategic parameters themselves are subject to evolution.

Secondly, using a uniform random variable that is bounded within the maximal allowable range rather than within a window of small predefined size, permits capturing large size clusters rapidly. Although the non-uniform mutation can also support fast approximation of large size clusters, unfortunately it loses this property as search progresses, which simply means that at the later stages of the search, a non-uniform mutation scheme may require many iterations to entirely capture large clusters. A Gaussian-like grow-mutation suffers from the same problem because although large moves are possible during the entire course of evolution, yet they are not so common. Finally, neighbourhood-move mutation operators are of limited exploratory power for regions that are far away from the rule under mutation. In contrast, *Grow-Mutation* has the capability of reaching isolated regions easily, throughout the evolutionary search.

In an alternative implementation of grow mutation, one could incrementally grow a rule as long as it yields a feasible (i.e. homogeneous) expansion, but this approach is computationally expensive for high dimensional datasets.

5.10.4 Principles of *Seed-Mutation*

Despite its appealing exploratory power, *Grow-Mutation* is incapable of assuring that every uncovered region of the feature space $\mathcal{F}$ is accessible to $\mathcal{NOCEA}$. More specifically, due to the constraint of evolving disjoint rule-sets, it may not always be feasible to accomplish local fine-tuning or to locate previously unknown clusters using *Grow-Mutation*. These limitations are evident in figure 5.27(a)

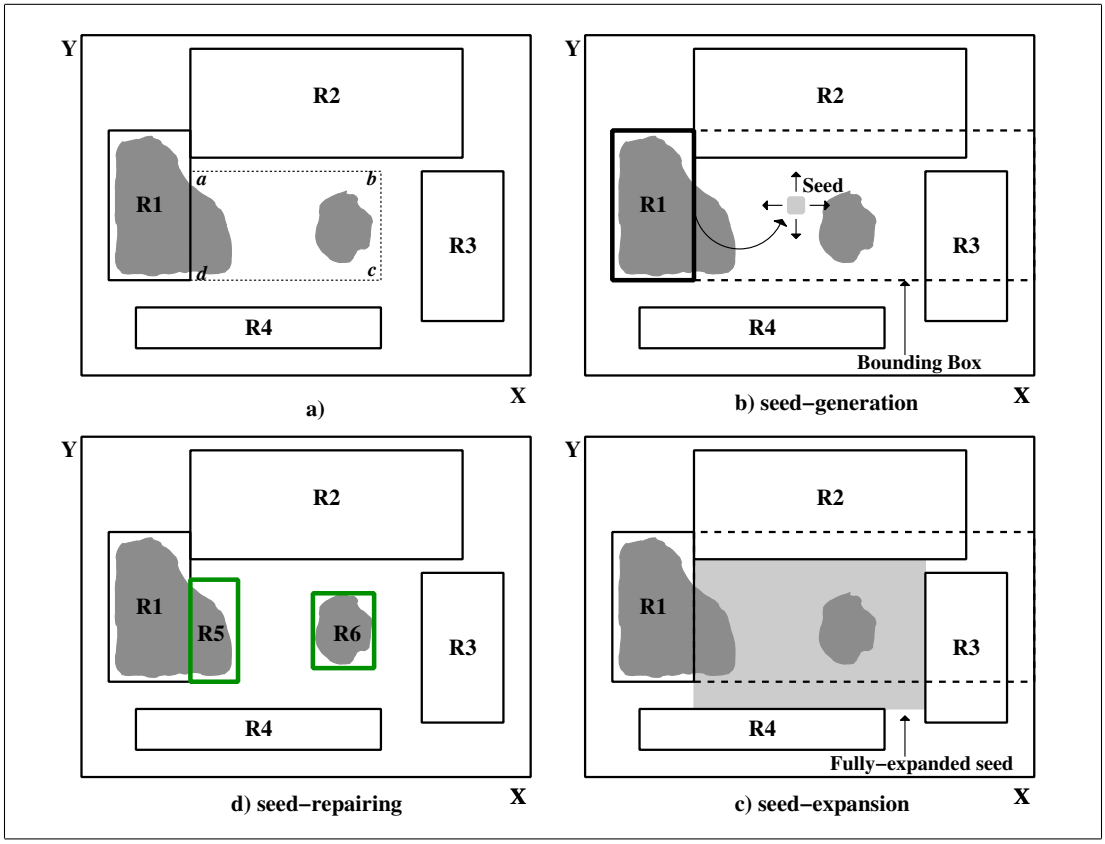


Figure 5.27: *Seed-Mutation* (b-d) overcomes limitations of *Grow-Mutation* (a)

where $\mathcal{NOCEA}$ has reached a deadlock in increasing the coverage of the individual. Clearly, $\mathcal{NOCEA}$ has been entrapped into a local optimum from which it is impossible to escape using only *Grow-Mutation*. This is because, no *Grow-Mutation* can explore the rectangular region (*abcd*) that is enclosed by the four rules, $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$.

This limitation of *Grow-Mutation* motivated the design of a complementary type of mutation, called *Seed-Mutation*. In short, *Seed-Mutation* is applied with a very small fixed probability, e.g. 0.005, to a single bound of a rule at a time, and generates, when it is possible, a *new* rule within a specific region, hereafter called *bounding box*, that is fully determined from the parent rule. Similarly to *Grow-*, the *Seed-Mutation* operator produces variations at random, yet the resulting offspring contain no overlapping rules.

Assuming that the upper bound u_{ij} of the jth rule ($\mathcal{R}_j$) undergoes *Seed-Mutation* in the ith dimension the operation proceeds as follows: Initially, the algorithm determines the lower (lb) and upper (ub) boundaries of the axis-aligned

hyper-rectangular bounding box that corresponds to u_{ij}:

$$[lb_k, ub_k] = \begin{cases} [l_{kj},\ u_{kj}], & \forall\ k = 1, ..., d, k \neq i \\ [(u_{ij} + 1),\ (m_k - 1)], & \text{if } k = i \end{cases} \tag{5.31}$$

where, m_k is the total number of bins in the kth dimension. The derivation of the bounding box for the lower bound l_{ij} of $\mathcal{R}_j$ is the dual procedure:

$$[lb_k, ub_k] = \begin{cases} [l_{kj},\ u_{kj}], & \forall\ k = 1, ..., d, k \neq i \\ [(0,\ (l_{kj} - 1)], & \text{if } k = i \end{cases} \tag{5.32}$$

If the bounding box contains at least one uncovered cell the algorithm selects semi-randomly (section 5.10.5) one, and creates a *new* rule, the *seed*. In the case that no empty space exists or the bounding box itself is empty, the operation is aborted. The next step is to grow the seed in every dimension, both to the left and to the right, as much as possible without causing overlapping with other rules. The expansion is performed dimension-by-dimension in a random order. The boundaries in a specific dimension are also processed in a random order.

The rationale behind the large-scale expansion of the seed is: *a*) to increase the probability of producing a non-sparse rule, and *b*) to accelerate the exploration of irrelevant features inside the given rule. Figures 5.27(b-d) show how $\mathcal{NOCEA}$ breaks the deadlock by employing *Seed-Mutation* in the right bound of rule $\mathcal{R}_1$ along the horizontal axis. In this case *Seed-Mutation* creates a new rule (light-grey rectangle in figure 5.27(c)) inside a previously unreachable region. The subsequent repairing of the fully-expanded seed yields two *new* homogeneous rules $\mathcal{R}_5$ and $\mathcal{R}_6$ as shown in figure 5.27(d). This example demonstrates the ability of *Seed-Mutation* to perform both local fine-tuning and discovery of new clusters.

The selection of the seed inside the bounding box is unbiased with respect to the parent rule, yet the size and location of the bounding box itself are dependent on the parent rule. It is important to clarify the difference between random initialisation and *Seed-Mutation*. In particular, in the former type of rule generation any uncovered cell of the feature space $\mathcal{F}$ is a candidate seed, while in *Seed-Mutation*, only a specific sub-region of $\mathcal{F}$ is examined. The location of this

region is deliberately chosen in a way that enables a localised search in the vicinity of the parent rule, where neighbouring rules may not allow accomplishing local fine tuning using *Grow-Mutation*. Additionally, since both the bounding box and the seed are maximally constructed in the space that is available, the ability of *Seed-Mutation* to discover isolated clusters should not be underestimated.

5.10.5 Seed Discovery Algorithm - $\mathcal{SDA}$

From a computational point of view, sampling randomly for a seed inside the bounding box during *Seed-Mutation*, becomes increasingly inefficient as the intersection between the bounding box and rules increases. For instance, if there is only a single seed within a 50-dimensional bounding box covering just two bins per dimension, the probability of *sampling* randomly that cell is only $(1/2^{50}) \rightarrow 0$. $\mathcal{NOCEA}$ relies on the novel $\mathcal{S}$eed $\mathcal{D}$iscovery $\mathcal{A}$lgorithm (or $\mathcal{SDA}$) of figure 5.28 to accelerate the discovery of a proper seed.

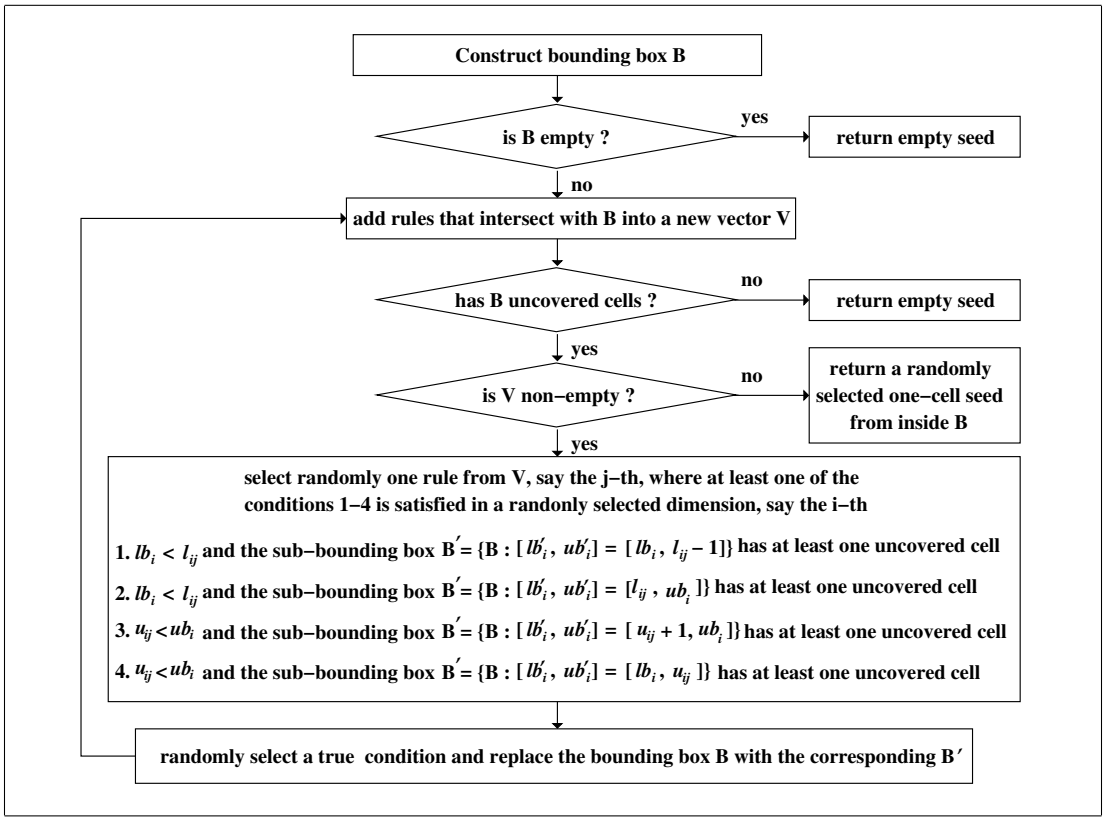


Figure 5.28: $\mathcal{S}$eed $\mathcal{D}$iscovery $\mathcal{A}$lgorithm - $\mathcal{SDA}$

$\mathcal{SDA}$ is a *divide-and-conquer* algorithm that recursively splits the bounding box into two disjoint sub-regions using axis-aligned cutting planes. A valid cutting plane passes through the borders of a rule that intersects with the bounding box. Some additional constraints, as described in figure 5.28, are introduced to ensure that the reduced bounding box will contain available space for seed generation. In essence, as $\mathcal{SDA}$ progresses less valid cutting planes are detectable. The procedure continues until obtaining a bounding box that does not overlap with rules. Finally, $\mathcal{SDA}$ randomly samples a cell inside the final bounding box to play the role of the seed.

But how does $\mathcal{SDA}$ determine whether a bounding-box has uncovered space? A naive solution that can replace completely $\mathcal{SDA}$ would be to examine every single cell enclosed by the bounding box, but such an exhaustive search is prohibitively expensive for high dimensional datasets because the number of cells increases exponentially with dimensionality. A simpler and more efficient method is to compute the difference between the volume of the bounding-box and the aggregated volume of the parts of rules covered by the former. If this difference is greater than zero then there is available space to generate a new seed.

5.10.6 Scheduling *Grow-* and *Seed-Mutation*

During the mutation stage, an individual consisting of k rules can be viewed as a vector of $2dk$ integer values, where each element corresponds to a rule bound in a particular dimension. A mutation event is regarded as a four part entity `MEvent`=[`Rule`, `Feature`, `Bound`, `Type`], where `Rule` ∈ `[1, k]`, `Feature` ∈ `[1, d]`, `Bound` ∈ `[lower, upper]`, denote the rule, feature and bound, respectively undergoing mutation whose type is specified in the field `Type` ∈ `[grow, seed]`. The list of mutation events is shuffled to assure randomness in the order by which bounds, features and rules are processed.

In $\mathcal{NOCEA}$, mutations are scheduled and executed in the following manner:

In a typical EA the mutation operator disregards any linkage among genes in the chromosome, that is, a gene is mutated to a new value independently of what

1. Determine the positions for mutation using a uniform random choice. Each bound has the same small probability p_m of undergoing mutation.

2. Select either *grow* or *seed* mutation with an equal probability for the selected position.

3. Perform mutations in random order.

Figure 5.29: Scheduling and Executing Mutations in $\mathcal{NOCEA}$

happens at other positions in the string. In our case, in contrast, a scheduled mutation event may be heavily affected or even cancelled by preceding mutation(s). This is because, any form of mutation must yield non-lethal variations, i.e. solutions with disjoint and syntactically valid rules.

5.11 Subspace Clustering

5.11.1 Motivation for Subspace Clustering

High dimensionality continues to pose a significant challenge to clustering algorithms because of the inherent sparsity of the feature space. In fact, recent studies argued that for moderate-to-high dimensional spaces all pairs of points are almost equidistant from one another, for a wide variety of data distributions and proximity functions [4, 18]. Under such circumstances, there is very poor discrimination between points belonging to different clusters in the full dimensional space.

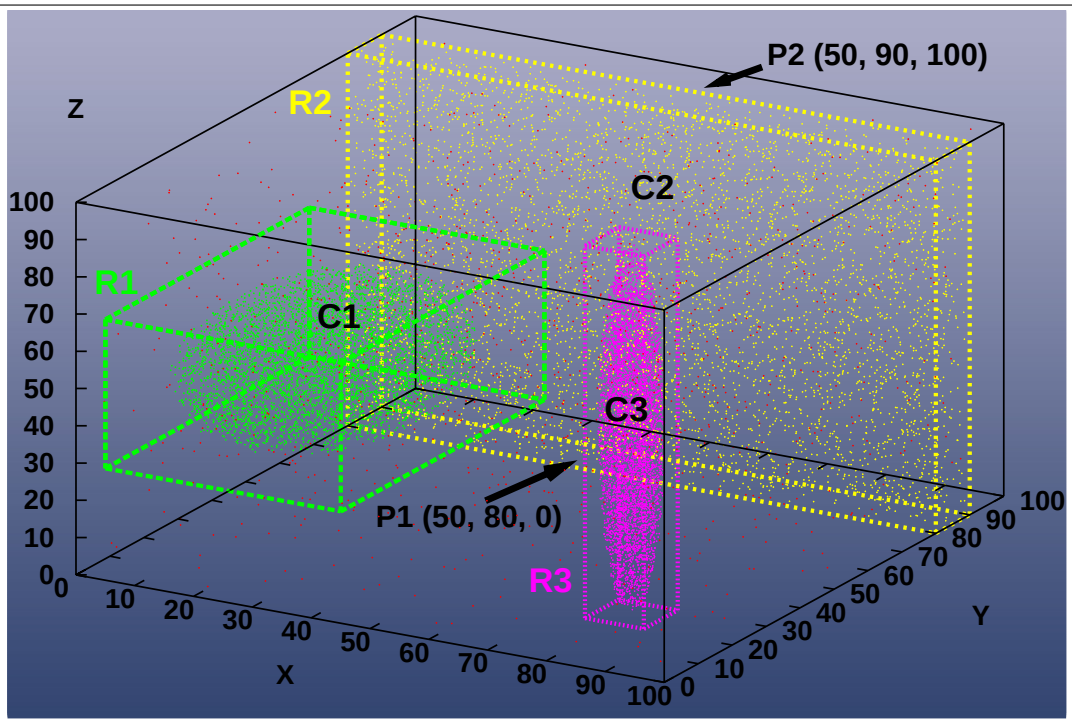
A possible way of dealing with the sparsity of the feature space $\mathcal{F}$ is to identify and retain only those features that are *relevant* to the clustering while ignoring the rest. The term relevant refers to dimensions forming subspaces where the points of clusters are closely located. Consider the example 3-dimensional dataset of figure 5.30(a), which contains two ellipsoids $\mathcal{C}_1$ and $\mathcal{C}_3$, and one orthogonal cluster $\mathcal{C}_2$. Clearly, $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_1$ are bounded in one, two, and three dimensions, respectively. Considering the pair of points $\mathcal{P}_1$(50, 80, 0) and $\mathcal{P}_2$(50, 90, 100), it can be easily observed from figures 5.30(c-d) that, although these points belong to the same cluster $\mathcal{C}_2$, they are far apart from one another in every subspace involving the dimension Z. However, $\mathcal{P}_1$ and $\mathcal{P}_2$ are very close in the subspace X×Y as shown in figure 5.30(b).

Various dimensionality reduction techniques, e.g *Principal Components Analysis (PCA)* [45] can be used to detect irrelevant features. However, since different subsets of points may be correlated in different subspaces, any attempt to reduce the high dimensionality by heuristically pruning away some dimensions is susceptible to a substantial loss of information.

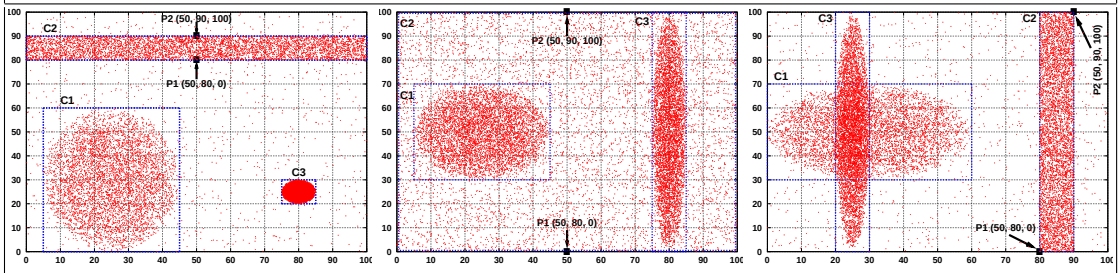
5.11.2 Principles for Subspace Clustering in $\mathcal{NOCEA}$

$\mathcal{NOCEA}$ is absolutely insensitive to the presence of irrelevant features in high dimensional spaces, as opposed to traditional clustering techniques [55]. This is because $\mathcal{NOCEA}$ attempts to maximise both the homogeneity and data coverage

of rules rather than to optimise some distance or density based criterion function. Hence, $\mathcal{NOCEA}$ is unusual in operating in the full-dimensional space, thereby avoiding artifacts produced by the joint projection of clusters in subspaces.



a) $X \times Y \times Z$ Full Dimensional Space



b) $X \times Y$ Subspace c) $X \times Z$ Subspace d) $Y \times Z$ Subspace

Figure 5.30: Example Clusters Embedded in Different Subspaces

In practice, $\mathcal{NOCEA}$ simply ignores the problem of detecting irrelevant features during the evolutionary search, and after convergence simplifies the discovered rules by pruning away irrelevant features. For example, let us assume that $\mathcal{NOCEA}$ discovered the following rule-set for the dataset shown in figure 5.30(a).

$\mathcal{R}_1$:	**IF**	(5≤X≤45)	$\wedge$	(0≤Y≤60)	$\wedge$	(30≤Z≤70)	**THEN** C_1
$\mathcal{R}_2$:	**IF**	(0≤X≤100)	$\wedge$	(80≤Y≤90)	$\wedge$	(0≤Z≤100)	**THEN** C_2
$\mathcal{R}_3$:	**IF**	(75≤X≤85)	$\wedge$	(20≤Y≤30)	$\wedge$	(0≤Z≤100)	**THEN** C_3

Examination of the rules reveals that the information encapsulated within some specific conditions, e.g. (0≤X≤100) in $\mathcal{R}_2$, is redundant in the sense that the length of such a feature-gene is approximately equal to the size of the entire domain for that dimension. Bearing in mind that the rules are always aligned to the coordinate axes and relatively homogeneous, e.g. features are either independent of one another or weakly correlated, reporting a rule in the full-dimensional space gives us no more knowledge than looking at the subspace formed by the bounded dimensions.

To decide whether a particular dimension is relevant to the clustering of points inside a rule, $\mathcal{NOCEA}$ compares the length of the rule in that dimension with the spreading of points along the entire dimension. Recall from Chapter 4 (section 4.5.1) than an outlier-resistant estimator of the spreading of points in the ith dimension is the length l_E of the interval $E = [max(a_i, (Q_{1i}-1.5IQR_i)), min(b_i, (Q_{3i}+1.5IQR_i))]$, where Q_{1i}, Q_{3i} and IQR_i denote the first quartile, third quartile and the interquartile-range of points in ith dimension, respectively, while its domain is represented by $[a_i, b_i]$. In our clustering context, the ith condition of the jth rule is redundant if the following condition is true:

$$\boxed{T_r \leq \left(\frac{u_{ij}-l_{ij}}{l_E}\right)} \tag{5.33}$$

where here l_{ij} and u_{ij} denote the decoded values (see linear decoding function 5.25 in section 5.3) for the lower and upper bounds of the jth rule in the ith dimension. The default setting for the input threshold $T_r \in (0,1]$ is discussed in section 5.14.

Although the antecedent part of rules in the genotype has fixed-length (d), irrelevant features are interpreted so that the phenotype of individuals, i.e. rule-set that is reported to end-users, has variable length in the rule-level, since conditions corresponding to irrelevant features are simply ignored without a substantial loss

of information. After applying the simplification analysis, the rules in our example reduce to a more informative knowledge:

$\mathcal{R}_1$:	**IF**	(5≤X≤45)	∧	(0≤Y≤60)	∧	(30≤Z≤70)	**THEN** $\mathcal{C}_1$
$\mathcal{R}_2$:	**IF**			(80≤Y≤90)			**THEN** $\mathcal{C}_2$
$\mathcal{R}_3$:	**IF**	(75≤X≤85)	∧	(20≤Y≤30)			**THEN** $\mathcal{C}_3$

Retaining only the relevant features helps in developing a better understanding of the inter-attribute correlations that can greatly facilitate KDD phases, e.g. the decision making process [31, 40, 51]. Examples of irrelevant features in real-world seismic data along with their interpretation can be found in Chapter 7 (section 7.11.1).

5.11.3 Subspace Clustering Under Noise Conditions

The neighbourhoods of noise in the full dimensional space are generally much sparser compared to the cluster regions [48]. Due to the high difference in density the clusters automatically stand out and clear the noise regions around them. However, there may exist clusters whose point density in some subspaces formed by irrelevant dimensions is similar to the density of the surrounding noise regions, especially when the level of background noise is relatively high. This means that a *feasible* rule that partially covers a cluster in the subspace of its irrelevant dimensions would easily be extended far beyond the boundaries of the cluster along the relevant dimensions. A representative example is illustrated in figure 5.31, where due to the increased background noise the rule $\mathcal{R}_1$ thinly cuts the cluster $\mathcal{C}_1$ along the only irrelevant dimension (Z) of the latter.

Although $\mathcal{R}_1$ is a perfectly feasible rule, it incorrectly covers both noise and cluster points. More severely, the excessive fragmentation of the body of clusters like $\mathcal{C}_1$, by rules like $\mathcal{R}_1$, may not allow placing non-sparse rules within the backbone of these clusters, while subspace clustering might prove problematic or even impossible. For instance, none of the rules ($\mathcal{R}_1$, $\mathcal{R}_4$, and $\mathcal{R}_5$) that intersect with $\mathcal{C}_1$ has a large enough interval along the Z-axis to detect that irrelevant dimension

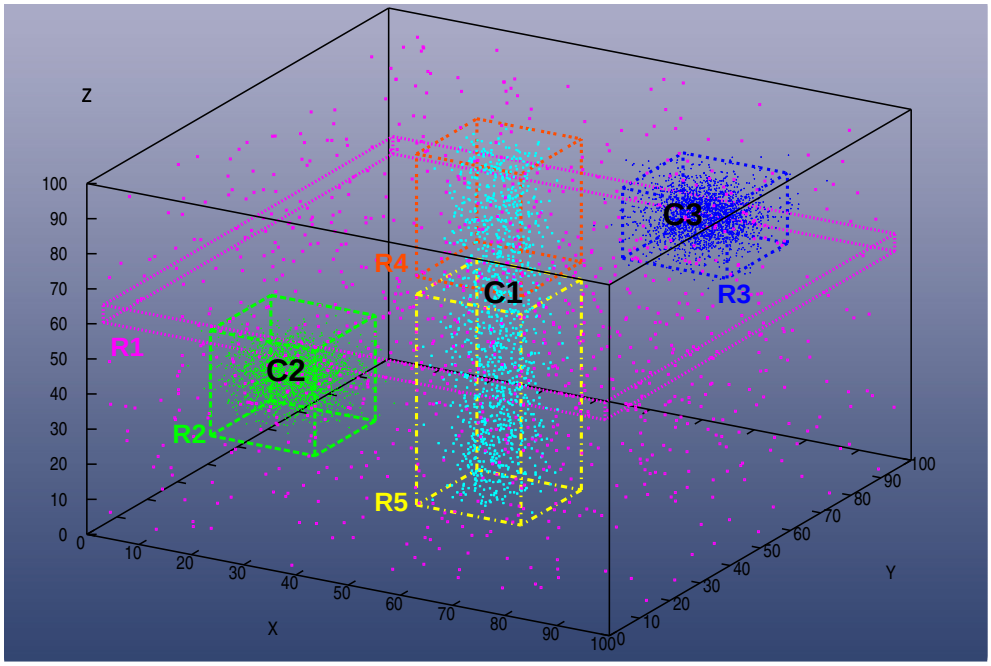


Figure 5.31: Challenges for Subspace Clustering Under Noise Conditions

in $\mathcal{C}_1$. Compounding this problem, generalisation, that would potentially solve this problem, is not feasible because $\mathcal{R}_1$ has considerably different density and geometry than $\mathcal{R}_4$ and $\mathcal{R}_5$.

$\mathcal{NOCEA}$ tackles this problem by eliminating all low density rules that potentially cover noise and cluster points, even if they are feasible, during the early stages of the search. However, this density bias is gradually relaxed and eventually discarded to allow discovering homogeneous rules of any density. The main idea is to bias the evolutionary search to discover first as dense rules as possible, thereby reducing the probability of accepting a feasible rule that covers both noise and cluster points.

Formally, the density bias requires the density of all feasible rules to exceed the *global density level* (GDL) by a time-variable factor (c). The *global density level* is defined as the average density that would have been observed if the data points were uniformly distributed throughout the feature space $\mathcal{F}$. An outlier-resistant estimator for the global density level can be obtained by dividing the number of points lying inside the non-outlier region of the feature space by the

volume of that region. The non-outlier region of $\mathcal{F}$ is a hyperbox whose interval (E) (see Chapter 4 at section 4.5.1) in the ith dimension is $E = [max(a_i, (Q_{1i} - 1.5IQR_i)), min(b_i, (Q_{3i} + 1.5IQR_i))]$, where Q_{1i}, Q_{3i} and IQR_i denote the first quartile, third quartile and the interquartile-range of points in the ith dimension, respectively, while its domain is represented by $[a_i, b_i]$.

In this thesis the density factor $c \in [0, 2]$ is linearly decreasing with time as:

$$c(t) = \begin{cases} 2(1 - t/150), & \text{if t<150 generations} \\ 0, & \text{otherwise} \end{cases} \tag{5.34}$$

where t denotes the current generation.

Thorough investigation related to the elimination of very low density rules is reported in Chapter 6 (section 6.5.4).

5.12 Assembling Clusters

This section describes a bottom-up post-processing algorithm that assembles the genuine clusters from the discovered rules.

5.12.1 Motivation

Often real world databases contain correlated subsets of dimensions that lead to points getting aligned along arbitrary shapes in lower dimensional spaces. Clearly, clusters with non-convex geometry require multiple rules to obtain an accurate and homogeneous descriptor.

In this thesis, a *cluster is a data pathway defined by a set of adjacent rules with a marginal variation in point density.* This is not to suggest that all rules constituting a cluster are of similar density in all possible subspaces, but only that these rules must exhibit only a marginal variation in density in the full dimensional space $\mathcal{F}$. Hence, a cluster descriptor is in the form of a *DNF* (*D*isjunctive *N*ormal *F*orm) expression, where each disjunct represents an axis-parallel rule. Once $\mathcal{NOCEA}$ converges, the chromosome of the best individual undergoes the bottom-up grouping algorithm of section 5.12.2, to fill the consequent part of the rules with the appropriate cluster identifier.

5.12.2 Principles of Cluster Formation Algorithm

Initially each rule belongs to a distinct cluster. Each step of the grouping algorithm involves merging two clusters that are the most similar. The similarity between two clusters is measured by the density ratio between the sparser rule from the two clusters and the denser rule belonging to the other cluster. Formally, two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ are merged if the following three conditions are satisfied:

1. $\mathcal{C}_1$ and $\mathcal{C}_2$ are directly connected through at least two adjacent rules $\mathcal{R}_{C1}$ and $\mathcal{R}_{C2}$ belonging to $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

2. The similarity of $\mathcal{C}_1$ and $\mathcal{C}_2$ exceeds the homogeneity threshold T_h.

3. The ratio of the length of intersection between $\mathcal{R}_{C1}$ and $\mathcal{R}_{C2}$ in every dimension -excluding of course the dimension where the rules are contiguous

- to the length of the corresponding feature-gene of at least one rule exceeds an input threshold $T_c \in (0, 1]$. T_c is discussed in section 5.14.

In short, the first condition reflects the requirement that rules must be adjacent to be considered as members of the same cluster. The second condition imposes the constraint that an arbitrary-shaped cluster can only be assembled by rules of similar density. The third condition requires that two adjacent clusters must have a large enough touch to be members of the same cluster.

5.12.3 An Example of Cluster Formation

Figure 5.32 shows an example dataset containing both convex and arbitrary-shaped clusters, where the relative darkness indicates the density of the clusters. Observe that the arbitrary-shaped cluster $\mathcal{C}_4$ has been captured using a set of rules ($\mathcal{R}_4$, $\mathcal{R}_5$ and $\mathcal{R}_6$), while, in contrast, the non-convex orthogonal clusters, $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ require a single rule. Although the rule $\mathcal{R}_2$ adjoins rule $\mathcal{R}_3$, they are not considered as members of the same cluster, as they have very different densities. Finally, the rules $\mathcal{R}_1$ and $\mathcal{R}_2$ despite being adjacent and of similar density, have a very limited touch, thus they do not belong to the same cluster.

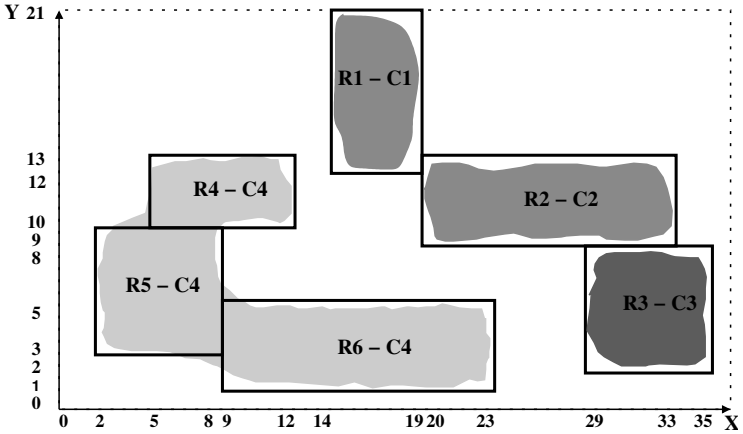


Figure 5.32: Capturing Non-Convex Clusters with Disjoint Rule-Sets

Hence, the discovered knowledge is reported in the following DNF expression:

IF (14≤X≤19) ∧ (12≤Y≤21) **THEN** cluster $\mathcal{C}_1$
IF (20≤X≤33) ∧ (9≤Y≤13) **THEN** cluster $\mathcal{C}_2$
IF (29≤X≤35) ∧ (2≤Y≤8) **THEN** cluster $\mathcal{C}_3$
IF [(9≤X≤23)∧(1≤Y≤5)]∨[(2≤X≤8)∧(3≤Y≤9)]∨[(5≤X≤12)∧(10≤Y≤13)] **THEN** cluster $\mathcal{C}_4$

5.13 Task Parallelism

This section explores the use of task parallelism to speed up $\mathcal{NOCEA}$ when the data to be mined is large and high dimensional.

The core idea behind $p\mathcal{NOCEA}$, a *p*arallel version of $\mathcal{NOCEA}$, is to maintain a single population of individuals in a central coordinator machine, and to distribute the execution of expensive genetic operations to remote machines. Figure 5.33 depicts the abstract architecture of $p\mathcal{NOCEA}$, where several processor-memory-disk units are attached on a communication network, and coordinated by a central master machine.

Due to their population-based nature, EAs are generally considered as slow compared to more conventional optimisation techniques that operate on a single solution at a time. Therefore, to establish the practicality of an EA-based clustering algorithm for large-scale data mining applications, it is necessary to introduce parallelism. Insightful discussions of both data and task parallel DM can be found in [43, 44].

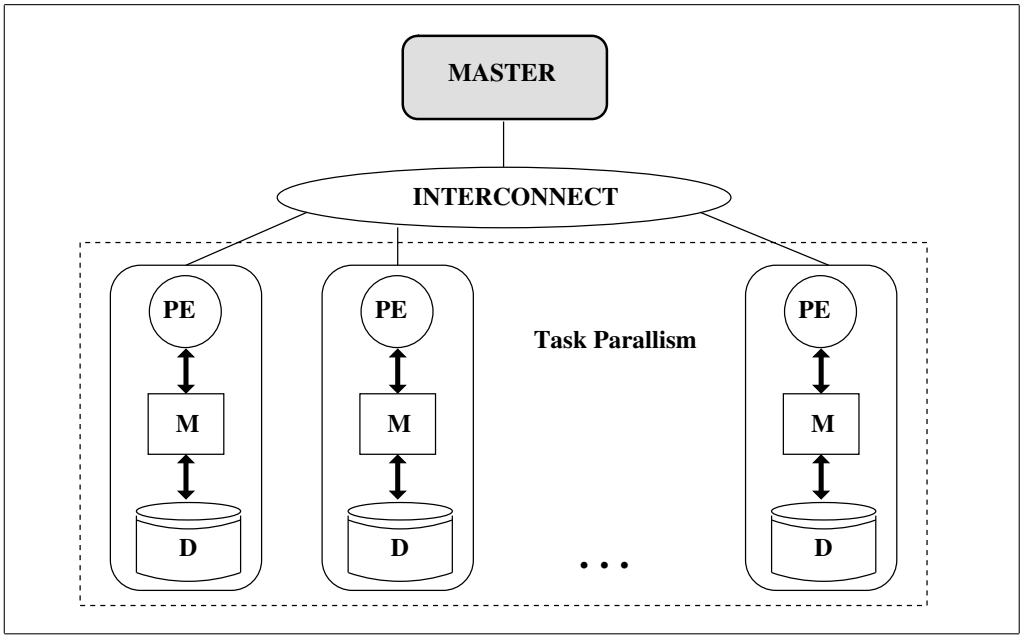


Figure 5.33: Parallel $p\mathcal{NOCEA}$ Architecture

5.13.1 Why is Task Parallelism Feasible in EAs?

In essence, parallel processing involves the simultaneous execution of tasks by several processors. From an implementation point of view, EAs are highly parallel procedures and can be easily and conventionally used in parallel systems. This is because EAs are made up from several cleanly separated *stages*, i.e. selection, reproduction, recombination, mutation, evaluation, and replacement. Furthermore, each stage consists of a number of individual *tasks*, e.g. a single recombination operation, involving a group of solutions rather than the entire population. Since the execution of an individual task is independent of other tasks, several processors can work simultaneously on the same stage or even on the same task.

5.13.2 Data Placement

$p\mathcal{NOCEA}$ implements a *share-nothing* architecture where each remote processor (*PE*) has direct access only to its local memory (*M*), as shown in figure 5.33. In the current implementation each local memory contains a replica of the entire dataset (D). Under this assumption the need to migrate incomplete tasks between processors is eliminated because all tasks involving access to the data, i.e. generalisation, recombination, and repairing, can be completed on a single PE. The thesis explores only task parallelism assuming that the entire dataset fits in the main memory of each PE, but data parallelism with data distributed among different PEs is an interesting topic for future work.

5.13.3 Granularity

In the context of this thesis, a *thread* is a sequential unit of computation that is entirely executed in a single processor (PE) without interruption. *Granularity*, a key aspect of parallel processing, is defined as *the average computation cost of a thread*, or in other words, *the average size of tasks assigned to the processors*. By this definition of granularity, a parallel program is called *fine-grained*, if it consists of threads with only small pieces of computation compared to the total amount of computation. The remainder of this section tackles the following question:

How is computation partitioned for parallel processing in $p\mathcal{NOCEA}$?

In our clustering context, the type of tasks, i.e. genetic operations, during a generation varies, and more importantly the relative computation cost of tasks heavily depends on the characteristics of the dataset itself, such as the number and dimensionality of clusters, and both database size and dimensionality. Therefore, without adequate prior knowledge it is impossible to determine an appropriate level of granularity beforehand. Thereby, $p\mathcal{NOCEA}$ adopts a relatively *coarse-grained* approach by inheriting the natural partitioning of computation generated by an EA-based system into individual genetic operations. In other words, each individual genetic operation, e.g. the complete mutation of a candidate solution, constitutes a sequential thread of computation that is entirely executed in a single remote processor (PE).

5.13.4 Communication Model

This section answers the following question:

How is information exchanged between processors?

One of the main sources of overhead in a parallel system is communication. In most fine-grained parallel architectures communication is much more expensive than computation and it is very important to minimise communication. In contrast, the coarse-grained granularity of $p\mathcal{NOCEA}$ results in the average computation cost of threads being significantly higher compared to inter-processor communication cost. Furthermore, since each thread is entirely executed in one PE without interruption, the coordinator machine has to forward each thread only once. After a thread finishes its execution in a remote machine that PE returns the result to the coordinator machine with one transmission. Finally, no inter-PE communication occurs because tasks are independent of each other. The

actual communication is modelled via message passing, i.e. using *Remote Method Invocation* (RMI), between different processors, and it has been implemented in $Java^{TM}2$ Standard Edition 1.4.2_05.

Packing: A *packing* scheme prescribes how much information to encapsulate into one packet when transferring information between processors. $pNOCEA$ uses a bulk packing scheme with variable size packets. In particular, one packet encapsulates all necessary information to conduct one genetic operation, such as the genomes of all individuals involved in that operation, and various statistics, e.g. data coverage of rules. Obviously, the size of a packet depends on the type of genetic operation that is encapsulated.

Latency: The *latency* is defined as the time required to send one packet of information between two processors. In practice, latency often varies between pairs of processors and also depends on the network traffic. Due to the coarse-grained approach and the fact that no actual data are moved, the impact of latency on the scalability of $pNOCEA$ is negligible, and is not further addressed.

5.13.5 Load Balancing

This section answers the following question:

How is work distributed and balanced between processors?

The main challenge for the *load balancing* model is to efficiently and effectively distribute the available work, i.e. threads, to ensure that all processors are utilised, without imposing additional load on the system. $pNOCEA$ uses a *centralised passive* load balancing policy where idle processors have to explicitly ask for work.

During the various stages of a single generation, the coordinator machine maintains a pool of instructions, i.e. threads, that are being queued for execution. In the beginning of each stage, the coordinator generates the entire workload for

that stage, and adds the corresponding threads into the pool. When a remote machine becomes idle it asks for work, and then the coordinator selects randomly a thread from the pool and forwards an appropriate execution message to that *PE*, which is immediately marked as busy. Each message encapsulates the group of individuals being involved in that genetic operation while the response message includes the result, i.e. group of individuals, yielded by that operation. No load information is exchanged between processors. This mechanism tries to minimise the number of messages required for load balancing.

5.13.6 Limitations of $p\mathcal{NOCEA}$

Despite the fact that $p\mathcal{NOCEA}$ can achieve a satisfactory speed up, e.g. 13.8 on 16 processors (see section 7.15.5), a number of important limitations remain to be addressed:

- **Coarse-grained Task Parallelism:** When the number of available processors is relatively large, to achieve high utilisation of *all* processors, fine-grained partitioning of the entire workload is required. For instance, an obvious caveat of the coarse-grained approach used in $p\mathcal{NOCEA}$ is the fact that no speedup improvement is possible when the number of available processors exceeds the total number of threads in the pool. It will be interesting to explore finer granularity task parallelism in $p\mathcal{NOCEA}$, by allowing an individual task, e.g. one recombination operation, to be executed simultaneously on several processors. As usual, there is a trade-off between reducing the task parallelism overhead and maintaining a high level of task parallelism. Obviously, for a finer-grained task parallelism architecture more sophisticated mechanisms for generating threads, synchronising threads, communicating data between threads, and terminating threads have to be established.

- **Data Parallelism:** Clearly, $p\mathcal{NOCEA}$ exploits no data parallelism because each processor executes instructions, i.e. generalisation, recombination, and repairing, accessing only the local replica of the dataset. However, when

the data to be mined is massive, and consequently does not fit in the main memory of each PE, data distribution among different PEs is strongly recommended. Figure 5.34 depicts a potential parallel architecture that can explore both data and task parallelism. In this approach, the data is distributed across multiple PEs (data parallelism). Similar to $p\mathcal{NOCEA}$ there is an independent group of processors specially designated for conducting the genetic operations (task parallelism), but no raw data reside in these PEs. Obviously, a locally executed genetic operation may require access to multiple data processors. This approach requires an advanced communication model and load balancing; an interesting topic for future research.

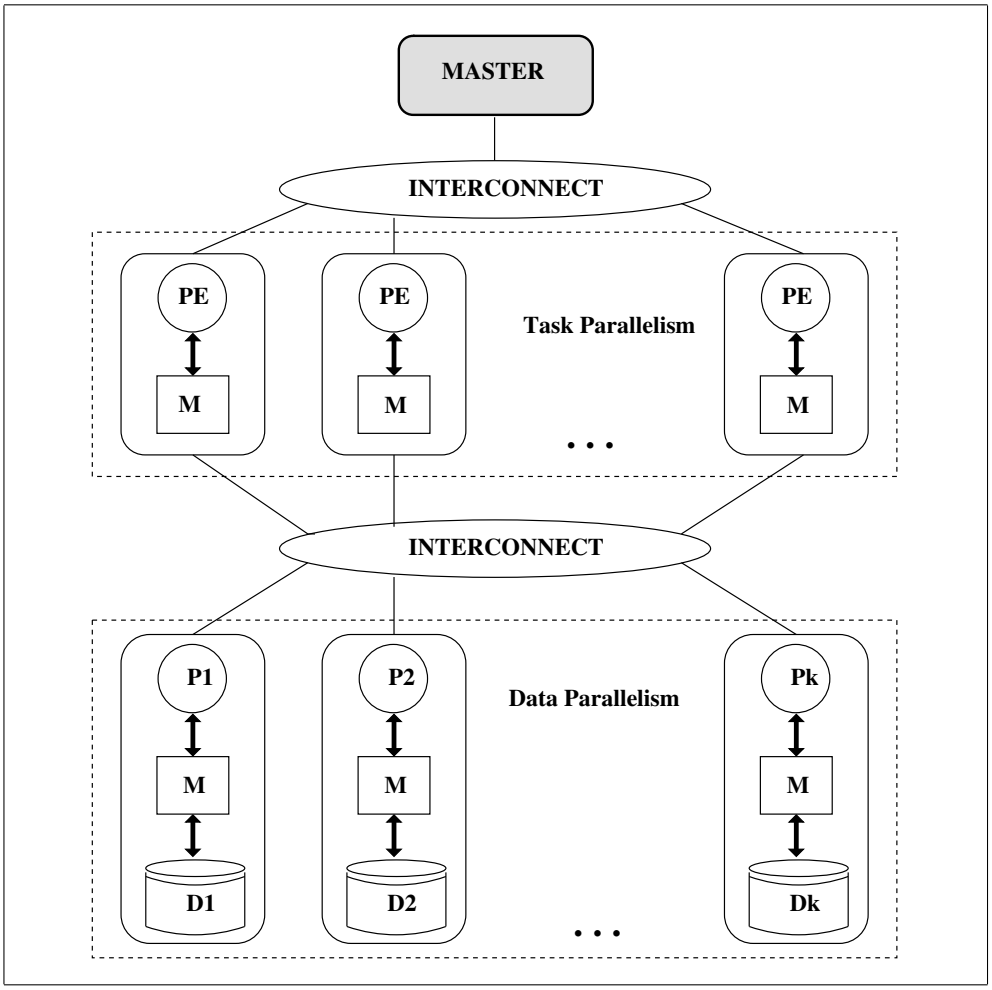


Figure 5.34: Task and Data Parallelism Architecture

5.14 Parameter Settings

This section discusses the default parameter settings in $\mathcal{NOCEA}$.

Population Size and Termination: The default population size is 50. $\mathcal{NOCEA}$ terminates if at least one of the following conditions is true: when the number of generations that have been executed exceeds a prespecified upper limit of 300 generations, or when the difference between the performance of the best individual and the average fitness of the population members, reaches a given level of stability (i.e. 1e-5) for a certain number of consecutive generations (i.e. 10 generation).

Initialisation: Each population member is *independently* initialised *at random* with a *single* hyper-rectangular rule, which covers the entire domain in (d-1) (d: data dimensionality) dimensions, while it is extended only in half of the domain in one, randomly selected dimension. The reason for initialising individuals with bulky rule-seeds is to increase the probability of locating non-sparse rule(s).

Reproduction: The primary objective of the *reproduction operator* is to make duplicates of good solutions and eliminate poorly performing individuals. $\mathcal{NOCEA}$ implements a typical k-fold (k=4) *tournament selection* scheme. In particular, each time an individual is requested for reproduction, k (the tournament size) distinct individuals are randomly drawn without replacement from the population, and the best one is selected. The selective pressure can be adjusted by changing the value of k.

Recombination: The *recombination rate* is the probability that recombination (instead of reproduction) is used to create new genomes. $\mathcal{NOCEA}$ applies the $\mathcal{O}$verlaid $\mathcal{R}$ule $\mathcal{C}$rossover ($\mathcal{ORC}$) operator (section 5.8) with probability 0.25, to two parents and creates two feasible offspring genomes. Similar to reproduction, in order to perform recombination parents are selected using k-fold *tournament selection*.

Generalisation: The *generalisation rate* indicating the probability that an individual undergoes *generalisation* (section 5.9), is set to 1.0. The *probability of generalising* a pair of adjacent rules that satisfy the generalisation requirements is set to 0.05. Recall from section 5.9.4 that the threshold $T_g \in (0, 1]$ is introduced to permit generalising only rules with proper alignment and similar size. Large values for T_g, e.g. 0.8, guard against the formation of non-homogeneous rules and degradation of the performance of individuals, but they do not facilitate effective subspace clustering, nor reduce the overall computational complexity. Small values for T_g, e.g. 0.2, have exactly the opposite effect. Fine-tuning T_g is a non-trivial task; as such, $\mathcal{NOCEA}$ adopts a middle-ground stochastic approach with variable T_g whose value for a given generalisation is drawn from a normal distribution $T_g = N(\mu, \sigma) : (0 \leq T_g \leq 1)$, where $\mu = 0.65$ and $\sigma = 0.1$. Thereby, extreme values are not completely avoided such that more search (generalisation) combinations can be explored, yet they are not so common. The second generalisation threshold, the *density* or *homogeneity threshold* T_h, is discussed in a subsequent paragraph entitled repairing.

Mutation: The *mutation rate*, that is, the probability that a newly created genome undergoes mutation, is set to 1.0. Each rule bound has the same small probability 0.01 of undergoing mutation. The type of mutation for the selected positions can be either *grow* (section 5.10.2) or *seed* (section 5.10.4) with an equal probability. Mutations are performed in a random order.

Repairing: The *repairing rate*, that is, the probability that a newly created genome undergoes repairing, is set to 1.0. Each candidate rule of an individual is fully repaired with probability 1.0. The homogeneity operator (section 5.7) requires two input parameters, the *sparsity* (T_s) and *homogeneity* (T_h) threshold. T_s controls the minimum percentage of total points that a feasible rule must cover to be considered as a statistically significant pattern. For very low dimensional datasets, e.g. $d < 5$, the default setting for T_s=0.5%, while for moderate-to-high dimensional datasets T_s=0.01%. The reason for selecting a lower T_s for the

higher dimensionality datasets is because, clusters tend to be less populated as dimensionality increases. Perhaps the most important parameter is T_h, which controls the level of homogeneity of the obtained rules. The experimental results have shown that for low dimensional datasets a value of T_h in the range [0.4-0.5] provides similar results of high quality. For higher dimensionality datasets, where clusters are expected to be considerably sparser and more isolated from one another, T_h should be set to [0.3-0.4] to reduce the loss of points in the boundaries of the clusters (section 6.10.3). We selected a higher value of T_h for the low dimensional datasets because as the dimensionality decreases clusters becomes less isolated, therefore it is necessary to have a higher T_h to effectively discriminate clusters.

Replacement: The *replacement strategy* prescribes how the current population and the newly created offspring are combined to create a new population of fixed size. $\mathcal{NOCEA}$ implements a simple *elite-preserving* replacement strategy, where the best performing individual of the current population is directly copied to the new population. $\mathcal{NOCEA}$ then finds the best performing offspring to fill the remaining slots of the new population. Elitism ensures that the statistics of the population-best solutions do not degrade with generations.

Subspace Clustering: The *subspace clustering* threshold T_r (section 5.11) determines when the length of a feature-gene is large enough, compared to the spread of points along the corresponding dimension, to be deemed as *irrelevant* to clustering. Notice that the value of T_r has no impact on the evolutionary search itself, but it does influences the quality of the clustering results returned to the user. This is because subspace clustering, a post-processing simplification stage, simply interprets the discovered knowledge without influencing its formation. The default value of T_r is 0.9.

Cluster Formation: The algorithm that groups adjacent rules into clusters (section 5.12) requires two input parameters: the standard *density* threshold T_h (see paragraph entitled "Repairing" above) and T_c. From a cluster formation

point of view, T_h controls the maximum allowable variance in the density of points along the pathway defined by the rules that constitute the body of an arbitrary-shaped cluster. T_c specifies when two adjacent rules have enough touch to be members of the same cluster. In all the experiments reported throughout the thesis, T_c was set to 0.2. Similar to T_r, T_c does not influence the evolutionary search. Finally, determining an appropriate setting for T_c is an application dependent task.

Table 5.3 summarises the default settings for both EA- and clustering-related parameters used by $\mathcal{NOCEA}$.

Parameter Name	**Value**
Population Size	50
Generations	300
Termination Condition	Maximum Number of Generations
Mutation Rate	1.0
Mutation Probability	0.01
Grow/Seed Mutation Ratio	0.5
Recombination Rate	0.25
Number of Offspring	2
Generalisation Rate	1.0
Generalisation Period	1
Generalisation Probability	0.05
Repairing Rate	1.0
Repairing Period	1
Selection Strategy	Tournament Selection (size=4)
Initialisation	Randomly Generated Singular-Rule Individuals
Replacement Strategy	Elitist (elite size =1)
Sparsity Threshold (T_s)	0.5% for low dimensional datasets, i.e $d < 5$ 0.1% for higher dimensional datasets
Homogeneity Threshold (T_h)	0.3
Subspace Clustering Threshold (T_r)	0.9
Generalisation Threshold (T_g)	$N(0.65, 0.1)$
Clustering Threshold (T_c)	0.2

Table 5.3: Default Parameter Settings in $\mathcal{NOCEA}$

5.15 Summary

This chapter has described in detail the key design aspects of the novel EA-based clustering algorithm $\mathcal{NOCEA}$. We have motivated the use of disjoint axis-aligned hyper-rectangular rules with homogeneous data distributions as a good alternative for representing clustering knowledge; IF-THEN rules are intuitively comprehensible for most end-users. The chapter has explained why an integer-valued representation scheme - reflecting the quantisation of the feature space into a multi-dimensional grid - is more suitable than binary or floating point encodings for high dimensionality clustering problems. It has been shown how the task-specific repair operator discovers homogeneous clusters by identifying flat density regions in univariate histograms. The chapter has also shown that a simple fitness function (maximisation of the data coverage of clustering rules) supported by the homogeneity operator, suffices to guide the evolutionary search into high quality disjoint partitions. The chapter has described novel recombination and mutation operators to efficiently and effectively traverse the enormous feature space while evolving disjoint rule-sets. A novel generalisation operator improving knowledge comprehensibility and reducing computation was also presented. Advanced post processing algorithms for performing subspace clustering and assembling the clusters were also described. Finally, the chapter has explored task parallelism to improve scalability when the data to be mined is large and high dimensional.

A thorough investigation into $\mathcal{NOCEA}$'s performance on both artificial and real-world datasets is reported in the following two chapters (6-7).

Part III

Evaluation

Chapter 6

Evaluation on Artificial Datasets

Capsule

This Chapter presents an experimental evaluation of $\mathcal{NOCEA}$ on a wide range of benchmark artificial datasets. We show that $\mathcal{NOCEA}$ has the following properties: It yields interpretable output being minimised for ease of comprehension; It discovers highly-homogeneous clusters of arbitrary shape, density, and size; It treats high dimensionality effectively, i.e. it can easily discriminate clusters in very high dimensional spaces; It performs effective subspace clustering, i.e. automatic detection of clusters being embedded in arbitrary subspaces of high dimensional data; It has exceptional resistance to the presence of increased background noise; It produces similar quality results irrespective initialisation and the order of the input data; It scales linearly with the database size, data, and cluster dimensionality; It has minimal requirements for domain knowledge, e.g. it automatically determines the optimal number of clusters and the subspace where each cluster is embedded, on the fly; Finally, it traverses the search space stochastically avoiding easily local optima.

6.1 Introduction

This Chapter reports a thorough investigation into $\mathcal{NOCEA}$'s performance on artificial datasets, many of which have been used to evaluate other well-established clustering algorithms. Real world data mining applications place specific requirements on clustering techniques (see section 2.3.2 in Chapter 2). Current clustering algorithms do not address all these requirements adequately, although considerable work has been done in addressing each requirement separately [17, 51, 52].

This Chapter reports a series of experiments that were conducted to validate all the salient properties of $\mathcal{NOCEA}$, as listed in section 1.2 (Chapter 1). The experimental results suggest that $\mathcal{NOCEA}$ is a remarkably robust and generic clustering algorithm, addressing well all the important requirements for data mining clustering. Most of the properties illustrated here prove important for the real-world case study mining of seismic data in Chapter 7.

The remainder of this Chapter is structured as follows. Section 6.2 presents the experimental environment, i.e. hardware and software apparatus, and benchmark datasets, that are used to evaluate $\mathcal{NOCEA}$. Section 6.3 discusses the end-user comprehensibility of the clustering results produced by $\mathcal{NOCEA}$. Section 6.4 shows that $\mathcal{NOCEA}$ is able to discover arbitrary-shaped clusters. Section 6.5 demonstrates the insensitivity of $\mathcal{NOCEA}$ to background noise. Section 6.6 verifies our intuition that $\mathcal{NOCEA}$ can discover clusters of arbitrary density and geometry. Section 6.7 proves that $\mathcal{NOCEA}$ produces similar results irrespective of the order by which input data are processed. Section 6.8 demonstrates that the quality of the clustering results of $\mathcal{NOCEA}$ is independent of the initialisation phase. Section 6.9 briefly discusses why $\mathcal{NOCEA}$ has minimal requirement for domain knowledge. Finally, section 6.10 evaluates the effectiveness and efficiency of $\mathcal{NOCEA}$ on massive and high-dimensional datasets containing clusters that are embedded in different subsets of dimensions.

6.2 Experimental Environment

This section describes the hardware-software apparatus used to evaluate $\mathcal{NOCEA}$ for DM clustering. It consists of hardware platforms, public benchmark datasets, and software tools that provide an appropriate experimental environment in which questions about the performance issues of $\mathcal{NOCEA}$ can be answered. The default parameter settings of table 5.3 are used throughout this chapter.

6.2.1 Hardware Apparatus

All experiments reported in this Chapter have been performed on a Linux Fedora Core 2 workstation with an Intel(R) Xeon(TM) CPU 3.06GHz processor, 512 KB cache, 2GB of DRAM, and 9GB of IDE disk.

6.2.2 Software Apparatus

$\mathcal{NOCEA}$ has been entirely implemented in $Java^{TM}$ 2, $Sun-Microsystems$ Platform, Standard Edition (J2SE, 1.4.2_05). The *Eos* [20] software platform, developed by BT's (British Telecommunication) Future Technologies Group (URL: http://research.btexact.com/islab/FTGpublic/EosPlatform.html), provides the basic EA functionality to $\mathcal{NOCEA}$.

Eos supports the rapid development and prototyping of evolutionary algorithms, ecosystem simulations and hybrid models. Amongst others the toolkit supports Genetic Algorithms and Evolutionary Strategies. It defines basic classes and various implementations for: genomes, recombination operators, mutation operators, selection strategies, replacement strategies, interactions, and more. The *Eos* platform is built using the Object Oriented design paradigm so that it is customisable and extensible. The flexibility of the *Eos* platform makes it a powerful environment for developing new algorithms and architectures. Eos is entirely implemented in *Java* and runs on all major operating systems.

6.2.3 Artificial Datasets

We experimented with six different datasets containing points in two dimensions whose size, i.e. number of points, distribution, and origin are shown in figure 6.35. These datasets have been extensively used as benchmarks in the field of data mining clustering [48, 61, 93, 99]. Their wide popularity arises from the fact that they contain clusters with challenging characteristics. In addition to these datasets, a new data generator was developed to produce datasets for evaluating $\mathcal{NOCEA}$ under highly noisy environments, as well as, in the presence of irrelevant features (subspace clustering).

Dataset DS1: The first dataset, DS1 [48], has 100000 points forming five convex clusters that are of different size, shape (three spherical and two ellipsoids) and density, and contains randomly scattered noise as well as special artifacts, i.e. a chain of outliers connecting the two ellipsoids. The density of points within the

two small circles is roughly 7.5 and 1.5 times greater compared to the density of the big circle and the two ellipsoids, respectively. Due to the large differences in size and densities as well as the presence of outliers, traditional partitioning and hierarchical clustering algorithms fail in DS1 [48]. For instance, distance based clustering is prone to splitting large clusters, i.e. large circles, to minimise the distance criterion function [48].

Dataset DS2: The second dataset, DS2 [61], with 8000 points, contains six clusters of varying size, shape, and orientation, as well as random noise and special artifacts such as streaks running across clusters. A particularly challenging feature of DS2 is that some clusters have arbitrary shapes.

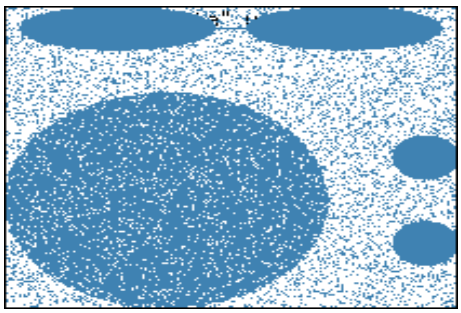
Dataset DS3: The third dataset, DS3 [61], with size 10000 points, is particularly popular within the DM research community to evaluate clustering algorithms. Its popularity arises from the fact that it contains clusters of varying shape, size and orientation, some of which are inside the space enclosed by other clusters. Additionally, DS3 is embedded with random noise and special artifacts, e.g. vertical streaks, running across clusters. It has been shown elsewhere [61] that many well established clustering algorithms including CURE and DBSCAN, fail to discover the genuine clusters. Similar to DS2, all clusters in DS3 have similar densities.

Dataset DS4: The fourth dataset, DS4 [93], contains 250000 points that are spread around two closely located parabolic clusters following uniform random distribution. The merit of including DS4 in our experiments is primarily to investigate the effectiveness of $\mathcal{NOCEA}$ under distributions which are characterised by strong inter-attribute correlations whose orthogonal univariate projections appear quasi-uniform. Due to the latter property clusters are not easily distinguishable in uni-dimensional projections.

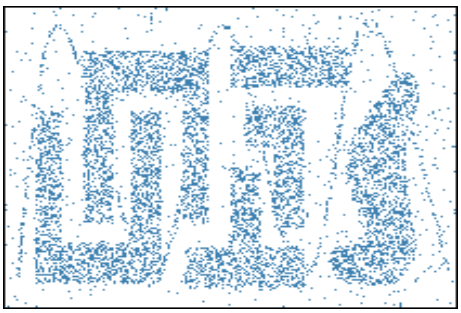
Dataset DS5: The fifth dataset, DS5 [99], consists of 100000 instances that are equi-distributed among 100 closely adjacent clusters with centres arranged in

a 10×10 grid pattern. The data points of each cluster are generated according to a 2-D independent normal distribution whose mean coincides with the centre and whose variance in both dimensions is fixed to 1.

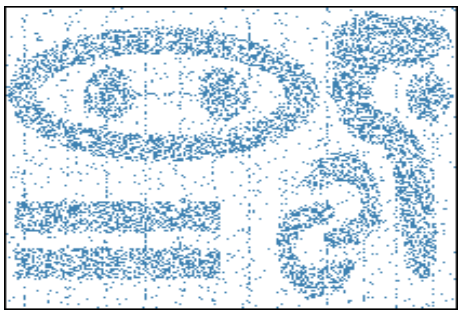
Dataset DS6: Finally, the sixth dataset DS6 [93], a moderately sized dataset with approximately 230000 instances, was generated by spreading points in a two dimensional space following uniform random distribution in the shapes of rectangles and an annular region.



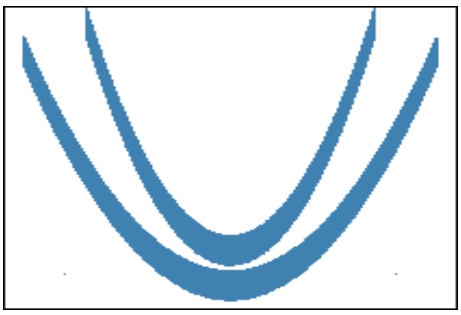
a) DS1 (100000 points, *source*: [48])



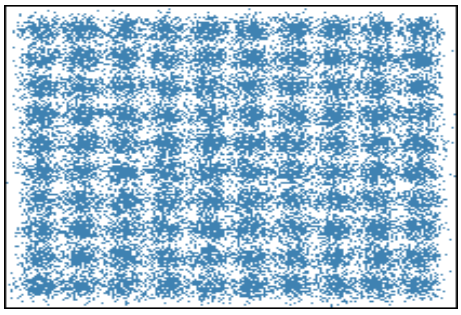
b) DS2 (8000 points, *source*: [61])



c) DS3 (10000 points, *source*: [61])



d) DS4 (250000 points, *source*: [93])



e) DS5 (100000 points, *source*: [99])

f) DS6 (228828 points, *source*: [93])

Figure 6.35: The Artificial Datasets Used to Evaluate $\mathcal{NOCEA}$

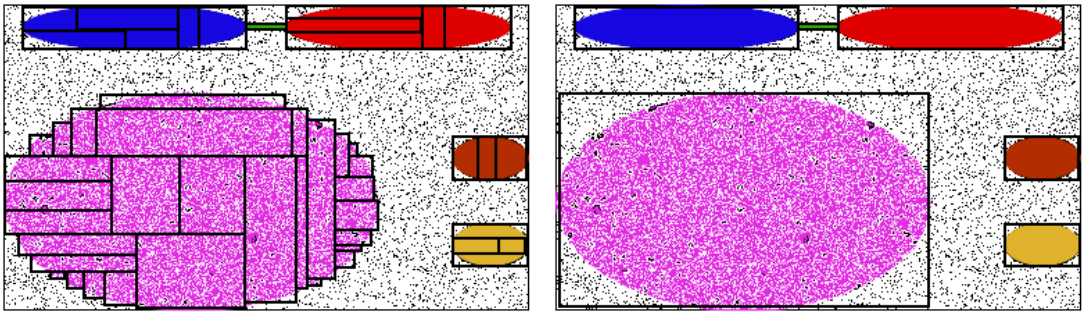
6.3 Comprehensible Clustering Output

This section provides evidence to support the claim that $\mathcal{NOCEA}$ produces highly comprehensible output that can be readily assimilated by humans.

Research in data mining has not paid enough attention to the factors that make learned models comprehensible [80, 81]. A comprehensible model would provide insight to an expert data analysts. This insight could be communicated to others and could support a variety of decision-making tasks. The intrinsic comprehensibility of the knowledge representation language in $\mathcal{NOCEA}$ (section 5.4) along with the parsimony pressure introduced by the generalisation operator (section 5.9) for as few and generic rules as possible, yield cluster descriptors of minimal syntactical complexity at a high level of abstraction.

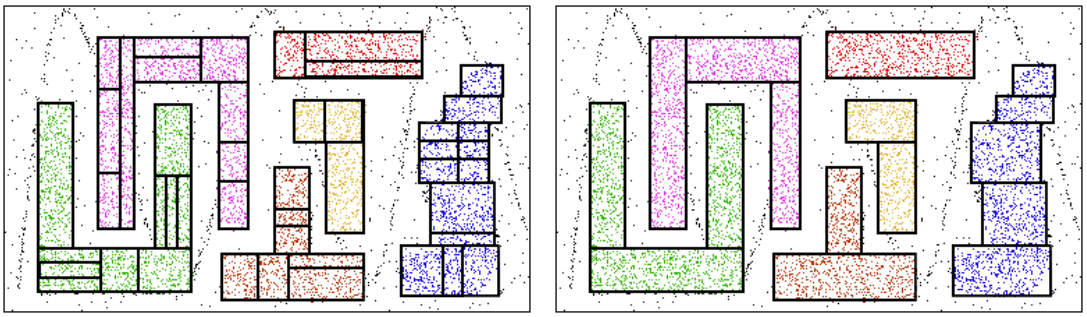
To assess the comprehensibility of the clustering results in $\mathcal{NOCEA}$, a set of experiments were conducted using two benchmark datasets, DS1 and DS2 (section 6.2.3). $\mathcal{NOCEA}$ was run twenty times against both datasets, varying the random rule seed used to generate the initial population of individuals. Figures 6.36(b)-6.37(b) and 6.36(a)-6.37(a) depict the clustering rules found by $\mathcal{NOCEA}$ for each one of the datasets with and without generalisation, respectively. Notice that points that belong to the same cluster have the same colour. As far as DS1 is concerned, $\mathcal{NOCEA}$ always converges to the clustering solution that is depicted in figure 6.36(b), which covers approximately 95% of the total data points in DS1. This is because DS1 contains convex clusters that are always captured by a single rule. The clustering rules discovered by $\mathcal{NOCEA}$ for DS2 are not always identical to those depicted in figure 6.37(b). This is due to the stochastic search of $\mathcal{NOCEA}$ and the fact that DS2 contains arbitrary-shaped clusters that can be approximated by different rule-sets (see section 6.8). The performance (data coverage) of the best individuals found over these twenty different random runs is confined in the interval i.e. 93-95%. The clustering solution of figure 6.37 was randonly selected. Clearly, in both cases $\mathcal{NOCEA}$ finds the genuine clusters.

Single-point cluster representation techniques, e.g. k-means or k-medoids, are of limited comprehensibility, since they can only describe adequately spherical



a) Generalisation Disabled b) Generalisation Enabled

Figure 6.36: The Rules and Clusters Discovered by $\mathcal{NOCEA}$ for DS1



a) Generalisation Disabled b) Generalisation Enabled

Figure 6.37: The Rules and Clusters Discovered by $\mathcal{NOCEA}$ for DS2

shape clusters of similar radius and density [48]. CURE utilises multiple representative points for each cluster that are well scattered throughout the body of each cluster. This enables CURE to adjust relatively well to the geometry of clusters having non-spherical shapes and wide variance in size, e.g. DS1. However, in both cases the output consists of a set of cluster representative points along with a data labelling table, where the latter stores information about the assignment of points to clusters. However, these methods provide no information about the shape or size of the clusters.

Rule-based cluster representation schemes, in contrast, shift the problem of clustering from data to space partitioning, providing cluster descriptors of a high level of abstraction. Data partitioning is induced by points' membership (inclusion) in segments resulting from space partitioning. $\mathcal{NOCEA}$ is capable of

adjusting automatically the size of the hyper-rules according to the data distribution without presuming a specific rule geometry. Thereby, one can easily induce the shape of both convex and arbitrary-shaped clusters from the geometry of the rules that jointly define the body of a given cluster.

Similar to the syntactical complexity and the level of abstraction of the knowledge representation language, generalisation heavily influences the comprehensibility of the discovered knowledge for a variety of reasons. First of all, generalising, when possible, many small rules leads to fewer and more generic rules that do not over-fit the data and consequently capture more general trends. For instance, the candidate solutions shown in figures 6.36(a) and 6.36(b) are of approximately the same performance, i.e. data coverage, yet the second rule-set makes more sense to most people. This is because the second solution is shorter (due to generalisation) and both the shape and size of the convex clusters can be easily induced by the geometry of the corresponding rules. In cases of arbitrary-shaped clusters, e.g. '⊓' shaped cluster in DS2, where inter-attribute correlations may be quite complex, generalisation improves comprehensibility by yielding as few and generic rules as possible that capture the essential backbone of the cluster. Finally, generalisation facilitates the detection of irrelevant features-genes whose removal from the antecedent part of the rules is necessary for the ease of comprehension (see real-world examples in section 7.11.1).

Data mining applications typically require descriptions that can be easily assimilated by humans as insight and explanations are of critical importance. It is particularly important to have simple representations, e.g. clustering rules, with minimal length (number of rules and number of conditions per rule) because most visualisation techniques do not work well in high dimensional spaces.

In the remainder of this seciton we study the performance of other clustering algorithms on DS1. Most of the analysis is based on previous work described in [48, 61]. Figure 6.38 shows the clusters found by BIRCH, MST, and CURE for the dataset DS1. As expected, since BIRCH uses a centroid-based hierarchical clustering algorithm for clustering the preclustered points, it cannot distinguish between the big and small clusters. BIRCH splits the larger cluster while merging

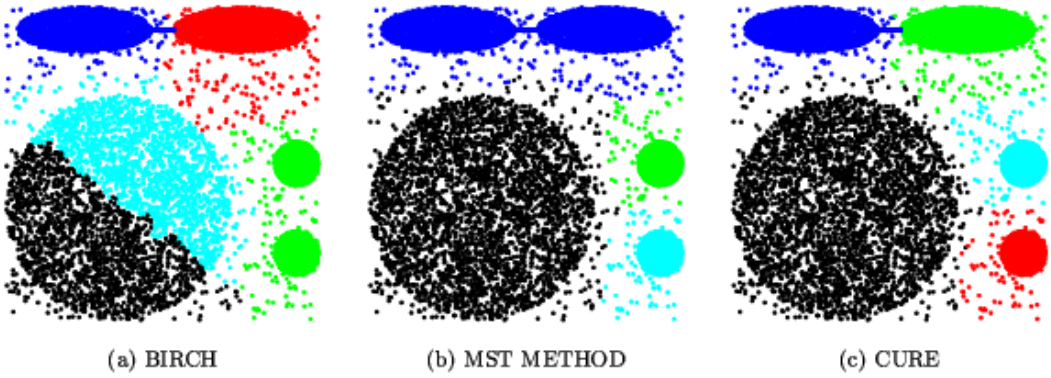


Figure 6.38: The clusters Discovered by BIRCH, MST, and CURE for DS1

the two smaller clusters that are adjacent to it. In contrast, the MST algorithm merges the two ellipsoids because it can not handle the chain of outliers connecting them. CURE successfully discovers the genuine clusters in DS1 with the default shrinking factor $a = 0.3$. The moderate shrinking towards the mean by a factor of 0.3 enables CURE to be less sensitive to outliers without splitting the large and elongated clusters.

Figure 6.39 shows the clusters found by CURE when the shrinking factor a is varied from 0.1 to 0.9. Evidently, 0.2-0.7 is a good range of values for the shrinking factor to identify non-spherical clusters while dampening the effects of outliers.

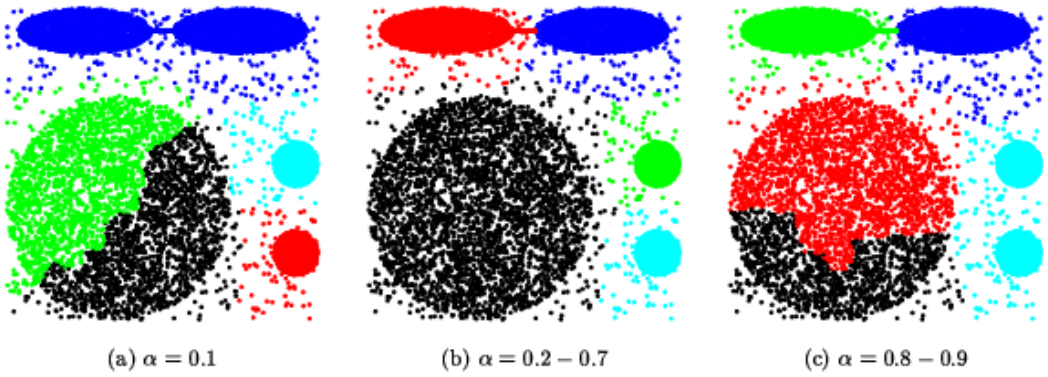


Figure 6.39: Sensitivity of CURE to the Shrinking Parameter

The DBSCAN clustering algorithm has been designed to find clusters of arbitrary-shapes. DBSCAN defines a cluster to be a maximum set of density-connected points. Every core point in a cluster must have at least a minimum

number of points (MinPts) within a given radius (Eps). DBSCAN can find arbitrary shapes of clusters if the right density of the clusters can be determined *a priori* and the density of clusters is uniform. Figure 6.40 shows the clusters found by DBSCAN for DS1 and DS2 for different values of the Eps parameter. Following the recommendation of [33], the MinPts was fixed to 4 and Eps was changed in these experiments. The clusters produced for DS1 illustrate that DBSCAN cannot effectively find clusters of different density. In the first clustering solution (Figure 6.40(a)), when Eps = 0.5, DBSCAN puts the two ellipses into the same cluster, because the outlier points connecting them satisfy the density requirements as dictated by the Eps and MinPts parameters. These clusters can be separated by decreasing the value of Eps as in the clustering solution shown in Figure 6.40(b), for which Eps=0.4. Here, DBSCAN keeps the ellipses together, but now it has fragmented the lower density cluster into a large number of small sub-clusters.

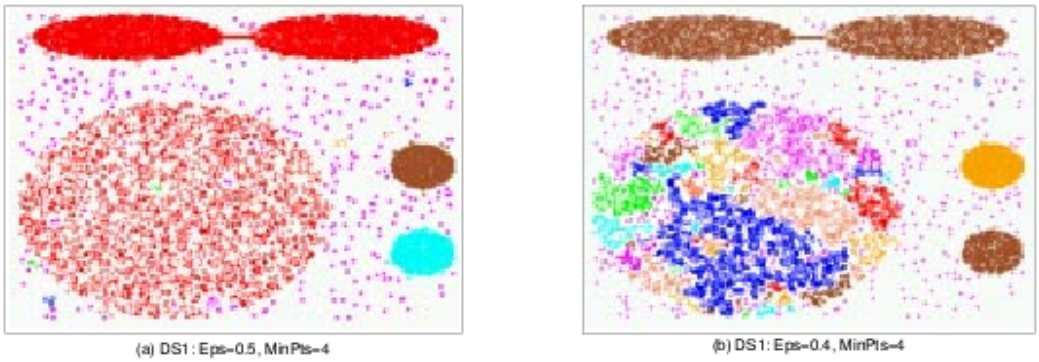


Figure 6.40: The Clusters Discovered by DBSCAN for the Dataset DS1

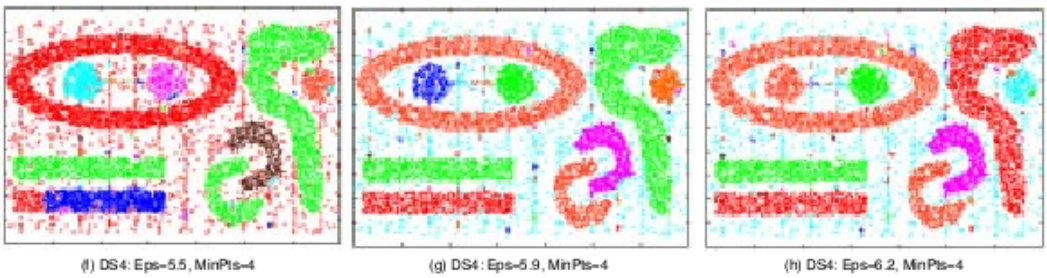


Figure 6.41: The Clusters Discovered by DBSCAN for the Dataset DS3

CHAMELEON is very effective in finding clusters of arbitrary shape, density

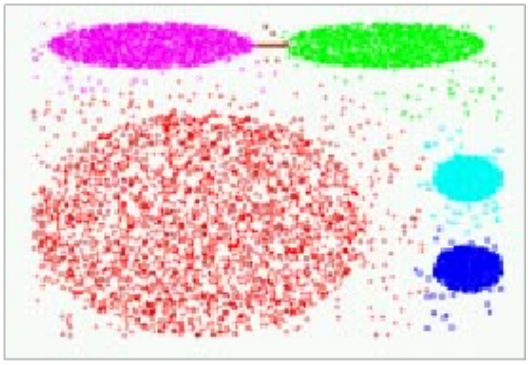


Figure 6.42: The Clusters Discovered by CHAMELEON for the Dataset DS1

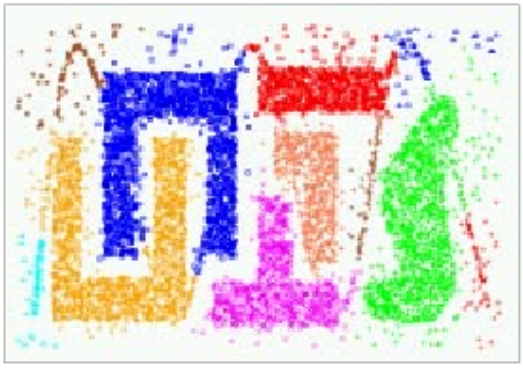


Figure 6.43: The Clusters Discovered by CHAMELEON for the Dataset DS2

and orientation, and is tolerant to outlier points, as well as artifacts. Figure 6.42 shows that CHAMELEON is able to correctly identify the genuine clusters in DS1.

Looking at Figure 6.43, we can see that CHAMELEON is able to correctly identify the genuine clusters in DS2. In particular, in the case of DS2, CHAMELEON finds eleven clusters, out of which six of them correspond to the genuine clusters in the data set, and the rest contain outliers. As these experiment illustrate CHAMELEON is very effective at finding clusters of arbitrary shape, density, and orientation, and is tolerant to outlier points, as well as artifacts such as streaks running across clusters.

CURE failed to find the correct clusters on the dataset DS2,as shown in figure 6.44 (see [48, 61] for more details). Since CURE is a hierarchical clustering algorithm, it also produces a dendrogram of possible clustering solutions at different levels of granularity. For each one of the data sets, Figure 6.44 shows two

different clustering solutions containing a different number of clusters. The first clustering solution (first column of Figure 6.44) corresponds to the earliest point in the agglomerative process in which CURE merges together sub-clusters that belong to two different genuine clusters. As we can see from figure 6.44, in the case of DS2, CURE selects the wrong pair of clusters to merge together when going from 18 down to 17 clusters, resulting in the red-colored sub-cluster which contains portions of the two π-shaped clusters. Similarly, in the case of DS3 (figure 6.45), CURE makes a mistake when going from 26 down to 25 clusters, as it chooses to merge together one of the circles inside the ellipse with a portion of the ellipse. The second clustering solution corresponds to solutions that contain as many clusters as those discovered by CHAMELEON. These solutions are considerably worse than the first set of solutions indicating that the merging scheme used by CURE performs multiple mistakes.

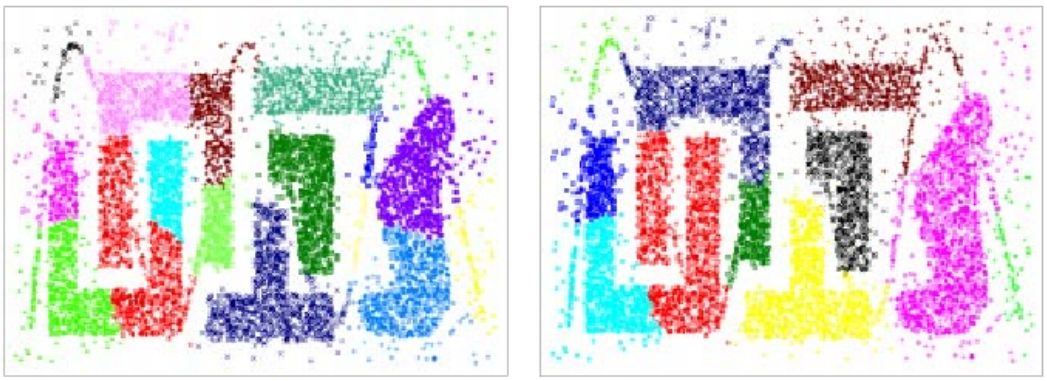


Figure 6.44: The Clusters Discovered by CURE for the Dataset DS2 (Varying Number of Clusters)

The above experiments have shown that although some clustering algorithm may be able to find the genuine clusters IN DS1, they cannot outperform $\mathcal{NOCEA}$ for two reasons: *a*) $\mathcal{NOCEA}$ produces more comprehensible, easy to assimilate, cluster descriptors, and *b*) $\mathcal{NOCEA}$ does not label outlier and noisy points as members of nearby clusters, as opposed to other clustering techniques.

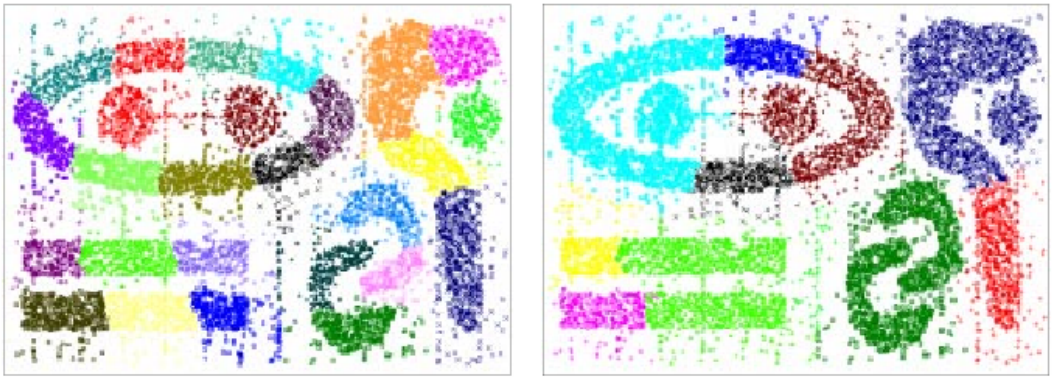


Figure 6.45: The Clusters Discovered by CURE for the Dataset DS3 (Varying Number of Clusters)

6.4 Discovery of Arbitrary-Shaped Clusters

This section evaluates $\mathcal{NOCEA}$ on publicly available benchmark datasets containing clusters with linear and higher order dependencies between subsets of dimensions.

Most real world datasets have subsets of correlated dimensions that lead to points getting aligned along arbitrary-shaped clusters in lower dimensional spaces [6]. Furthermore, these subsets of correlated dimensions may be different in different localities (neighbourhoods) of the data. Recall from section 5.4.2 that a feasible rule is characterised by the absence of strong inter-attribute correlations. Thereby, arbitrary-shaped clusters are decomposed into disjoint regions of homogeneous data distribution. To improve understanding of the complex correlations that may occur within an arbitrary-shaped cluster, one must consider jointly all the rules constituting the given cluster. There is a clear trade-off between the number of rules required to approximate an arbitrary-shape cluster and their homogeneity. $\mathcal{NOCEA}$ is strongly biased towards producing highly homogeneous rules rather than minimising the size of the rule-set.

$\mathcal{NOCEA}$ was tested twenty times against two challenging datasets containing arbitrary geometry clusters, DS3 and DS4. In both cases, as expected, different runs yielded different descriptors for arbitrary-shaped clusters, but convex clusters were always captured by a single rule. The arithmetic average of the results

(fitness of the best performing individual) for DS3 and DS4 over these twenty different random seeds was 96% and 93%, respectively. The solutions depicted in figures 6.46 and 6.47 were randomly selected from among the solutions of the twenty different runs.

Figure 6.46 shows both the rules and clusters found by $\mathcal{NOCEA}$ for DS3. It can be easily observed that $\mathcal{NOCEA}$ successfully recovers the genuine clusters in the dataset. Convex clusters, e.g. spherical or rectangular are approximated by one rule while arbitrary-shaped clusters require multiple rules. Obviously, the more complex the geometry of a cluster the more rules are required to obtain an accurate and homogeneous approximation of the cluster.

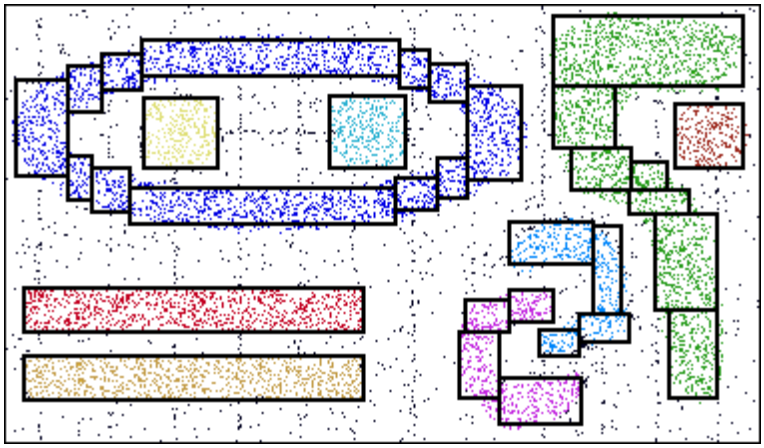


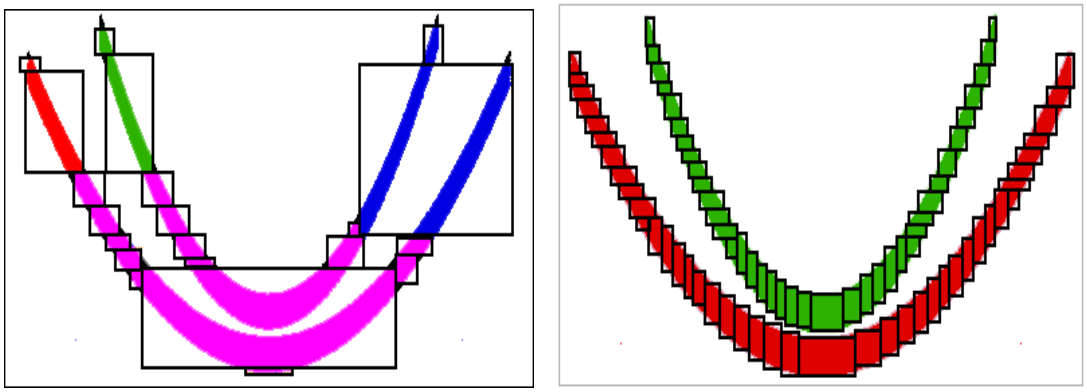
Figure 6.46: The Rules and Clusters Discovered by $\mathcal{NOCEA}$ for DS3

Axis-parallel uni-dimensional projections may not always suffice for detecting the separating valleys between the clusters [6, 55]. In particular, when there are linear or higher order inter-attribute dependencies, orthogonal univariate projections may appear quasi-uniform since they do not preserve this information, which, in turn, is necessary for clustering. The same problem of poor cluster discrimination may also be encountered with multiple clusters overlap along a uni-dimensional orthogonal projection. For instance, in DS4 the joint 2-D distribution is clearly non-uniform, but both uni-dimensional orthogonal projections are quasi-uniform.

Figure 6.47(a) depicts a typical output of $\mathcal{NOCEA}$ on DS4 without employing the localised homogeneity procedure, as described in section 5.7.4. Clearly, the

ordinal repair operator fails to detect the genuine clusters. Additionally many rules do not satisfy the homogeneity requirement. These problems arise from the limitation of uni-dimensional projections to preserve information about inter-attribute correlations that is necessary for clustering. Notice that quasi-uniform projections may be the result of a single cluster with linear or higher order dependencies between features, multiple clusters that overlap or both.

Figure 6.47(b) clearly demonstrates the beneficial effect of finer localised repairing in the quality of the clustering results. $\mathcal{NOCEA}$ detects now two parabolic clusters that are approximated with accuracy using highly homogeneous rules. Obviously, all instances of inter-attribute correlations have been successfully resolved.



a) Localised Repairing Disabled b) Localised Repairing Enabled

Figure 6.47: The Rules and Clusters Discovered by $\mathcal{NOCEA}$ for DS4

More complex examples of arbitrary-shaped clusters discovered by $\mathcal{NOCEA}$ in a real-world seismic dataset can be found in Chapter 7 (sections 7.7.4 and 7.12).

6.5 Insensitivity to Noise

This section evaluates the effectiveness of $\mathcal{NOCEA}$ on artificial datasets with varying levels of background noise. The experimental results clearly show that $\mathcal{NOCEA}$ is exceptionally resistant to uniform noise.

6.5.1 Apparatus

To study the sensitivity of $\mathcal{NOCEA}$ to varying levels of noise a new data generator was developed using some salient features of a very popular generator [5]. Let k be the number of clusters, while N and d denote the size and dimensionality of the dataset, respectively. The points have coordinates in the range [0, 100] and are either cluster points or noise. The percentage of noise is controlled by an input parameter $F_{noise} \in (0, 1]$. Noisy points are distributed uniformly at random throughout the entire feature space $\mathcal{F}$. The number of bounded dimensions associated with a cluster is given by a realisation of a Poisson random variable with mean μ, with the additional constraint that this number must be at least 2 and at most d. Similar to [5, 83], when generating cluster i+1, about 50% of its bounded dimensions are randomly chosen from among the bounded dimensions of cluster i. This is intended to model the fact that different clusters often share subsets of bounded dimensions. Assuming that the ith cluster is to be embedded in l dimensions the generator selects l variance values independently of each other. In particular, given a spread parameter r and a scale factor s_{ij} that is uniformly distributed in [1, s] the variance of points along the jth bounded dimension of the ith cluster is then computed as $(r \cdot s_{ij})^2$, where r=s=2 [5].

In the original generator the cluster centres or *anchor points* for the bounded dimensions are obtained by generating k uniformly distributed points in the respective subspaces. However, since the anchor points are chosen completely independently of each other and, additionally, clusters may be embedded in different subspaces, cluster overlapping is possible. To avoid overlapping, which, in turn, eases the validation of the results, all clusters are embedded in the first two dimensions while the anchor points are determined inductively as follows: The

coordinates of the anchor point for the first cluster are chosen at random. The coordinates of the anchor point for the ith (i=1,...,k) cluster are then chosen at random with the additional constraint that the ith cluster does not intersect with previously generated clusters. Let c_m^i and c_m^j be the m-coordinates of centres c^i and c^j that correspond to the ith and jth clusters, respectively. Two clusters, e.g. ith and jth, are disjoint if they share at least one bounded dimension e.g. mth, and the distance of the anchor points in that dimension is at least $3\sigma_m^i + 3\sigma_m^j$, where σ_m^i and σ_m^j is the standard deviation of the points in the ith and jth cluster along the mth dimension, respectively. The cluster points are equi-distributed among the k clusters. Finally, the points for a given cluster i are generated as follows: The coordinates of the cluster points on the non-relevant dimensions are generated independently at random from a uniform distribution over the range [0, 100]. For the mth bounded dimension of the jth cluster, the coordinates of the points projected onto dimension m follow either a uniform distribution defined in the interval $[c_m^j - 3\sigma_m^j,\ c_m^j + 3\sigma_m^j]$, or a normal distribution with mean at the respective coordinate of the anchor point and variance determined as explained earlier. The type of distribution for a given cluster is selected at random.

6.5.2 The *Recall* and *Precision*

In general, there are three potential sources of clustering imprecision: *a*) $\mathcal{NOCEA}$ may fail to create partitions for all the original clusters, and/or *b*) $\mathcal{NOCEA}$ may create spurious partitions that do not correspond to any of the original clusters. These effects are often measured separately using two metrics borrowed from the information retrieval domain: *recall*, *precision*, and *coverage ratio* [31, 72].

- **Recall** is defined as the percentage of the original clusters that were found and assigned to partitions. In essence, recall answers the question: *Have all the input clusters been correctly retrieved*?
- **Precision** is defined as the percentage of the found partitions that contain at least one original cluster. Precision answers: *Are all output clusters correspond to genuine input clusters or spurious clusters have been also identified*?

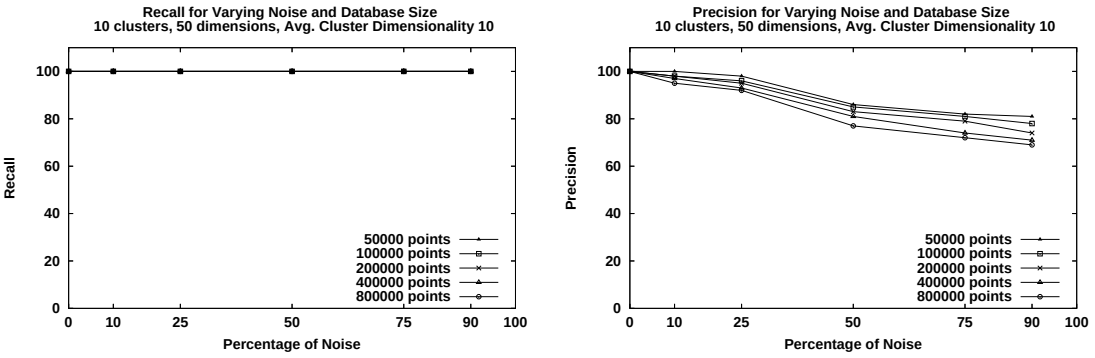
6.5.3 Noise Experiment Results

Figures 6.48, 6.49 and 6.50 illustrate $\mathcal{NOCEA}$'s performance, i.e. *Recall* and *Precision*, under noisy conditions for varying database size, data and cluster dimensionality, respectively. The precision and recall that are reported below are based on an arithmetic average of the results over twenty different runs.

Often, especially in cases of low dimensional and highly noisy datasets, extra spurious clusters were detected, typically comprising connected rules of extremely low density compared to rules that correspond to the actual clusters. In essence, this artifact is a by-product of the data generation algorithm that introduces uniform noise and the fact that the decision of accepting a candidate rule as a feasible sub-solution to the clustering problem is taken on the basis of data homogeneity rather than density. As a result, some neighbourhoods of the feature space had enough noise points to become non-sparse with quasi-uniform data distribution. The curves clearly show that $\mathcal{NOCEA}$ has an exceptionally stable behaviour for a wide range of noise level, i.e. 10%, 25%, 50%, 75%, and 90%. In particular, $\mathcal{NOCEA}$ correctly recovers all the original clusters in the datasets, even for very large noise levels of up to 90%. When the level of noise exceeds 90% then the overall data distribution throughout the entire feature space $\mathcal{F}$ becomes practically uniform, thereby $\mathcal{NOCEA}$ discovers a single super rule covering all of $\mathcal{F}$.

The Effect of Noise and Database Size

Figures 6.48(a-b) suggest that $\mathcal{NOCEA}$ has a relatively stable behaviour with respect to the database size even under highly noisy conditions. $\mathcal{NOCEA}$ is tested in moderately sized feature spaces comprising 50 dimensions where 10 clusters are embedded in some 10 dimensional subspace. Although an increasing number of records makes a fixed-volume feature space overall less sparse, the density ratio between the actual clusters and the noise regions remains constant. Hence $\mathcal{NOCEA}$'s homogeneity operator can effectively discriminate the genuine clusters from the noise, yielding thus a 100% recall in all cases. The curves also show that the precision slightly decreases as the number of records increases because



a) b)

Figure 6.48: The *Recall* (a) and *Precision* (b) Curves for Varying Database Size and Level of Noise.

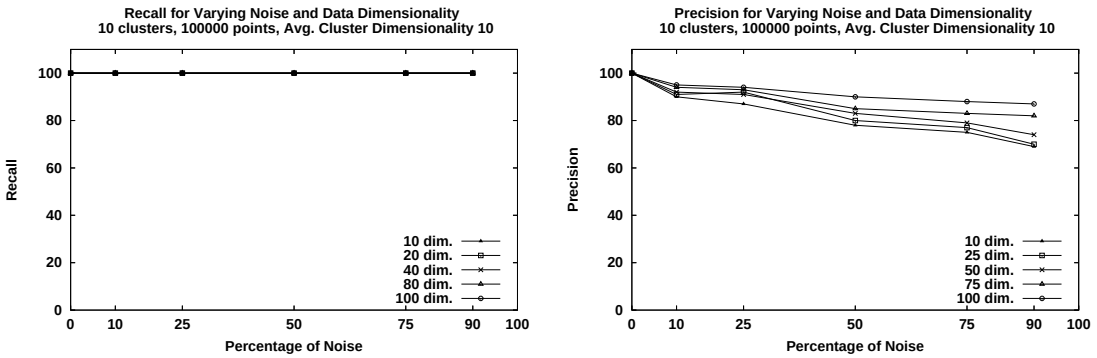
more neighbourhoods of the feature space $\mathcal{F}$ corresponding to noise are considered homogeneous. As a result of increasing database size, some neighbourhoods of the feature space had enough noise points to become non-sparse with quasi-uniform data distribution. Notice that due to the increasing database size the density histograms become increasingly more accurate since more points are used in the Kernel Density Estimator (section 3.3). In all experiments if one filters out all the very low density rules describing background noise the precision reaches the value of 100%.

The Effect of Noise and Data Dimensionality

Figures 6.49(a) and 6.49(b) show the *Recall* and *Precision* curves, respectively, as the background noise increases from 10% to 90% for varying data dimensionality ranging from 10 to 100 dimensions. Each dataset has 100000 points in total and there are 10 clusters embedded in some 10 dimensional subspaces on average.

Clearly, data dimensionality and uniform noise have no effect on the recall performance of $\mathcal{NOCEA}$. In fact, as the dimensionality increases noisy points become increasingly more isolated from one another making increasingly more difficult the formation of non-sparse and homogeneous rules from a collection of noisy points. Therefore, by definition higher dimensionality provides a slight advantage when handling noise, in the sense that it is easier to discriminate/prune

the genuine clusters from the spurious ones because the latter are extremely sparse in high dimensional spaces. This is obvious from the precision curve of figure 6.49(b). Filtering out low density rules describing noise brings the precision in all experiments to the value of 100%.

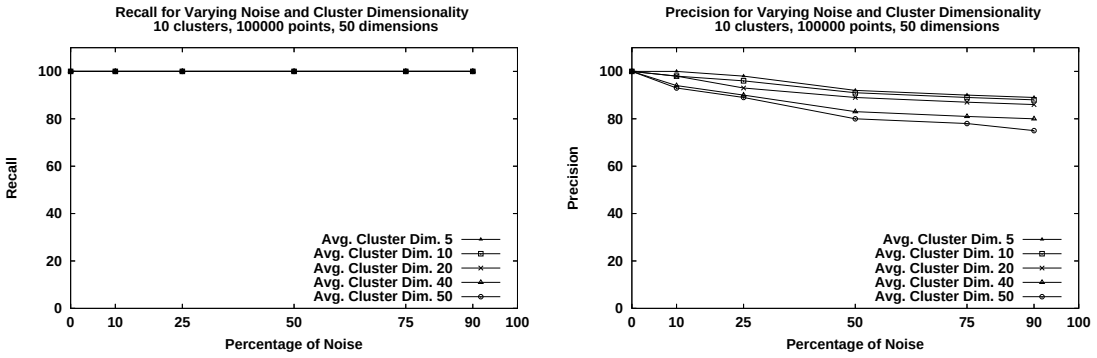


a) b)

Figure 6.49: The *Recall* (a) and *Precision* (b) Curves for Varying Data Dimensionality and Level of Noise.

The Effect of Noise and Cluster Dimensionality

Similar to the other two important characteristics of a database, i.e. size and data dimensionality, the compactness of clusters associated with their dimensionality does not affect the recall effectiveness of $\mathcal{NOCEA}$ even under very noisy conditions. Figures 6.50(a-b) illustrate the *Recall* and *Precision* as the percentage of the background noise increases from 10% to 90%. Each dataset has 100000 records in a 50-dimensional feature space while there exist 10 clusters with average dimensionality ranging from 5 to 50. The slight degradation of precision with increasing dimensionality is due to the fact that as the dimensionality of clusters increases more uncovered space within $\mathcal{F}$ becomes available (uncovered) for the evolutionary search to create spurious rules covering large sized noise regions. However, if one discards the low density noisy rules then the precision becomes 100%.



a) b)

Figure 6.50: The *Recall* (a) and *Precision* (b) Curves for Varying Cluster Dimensionality and Level of Noise.

6.5.4 Clustering Challenges Under Noise Conditions

Every clustering algorithm needs mechanisms to discount the background noise. Before explaining $\mathcal{NOCEA}$'s resistance to the presence of noise it is necessary to highlight the challenges that clustering algorithms face due to noise.

Given a candidate rule, the first homogeneity $\mathcal{HT}_1$, as described in section 3.4.1, attempts to discriminate well-separated clusters and to distinguish the clusters from the surrounding noise regions. To achieve these aims $\mathcal{HT}_1$ seeks relatively deep valleys along the uni-dimensional density histograms. When the level of noise is relatively low, $\mathcal{HT}_1$ suffices for separating the genuine clusters from the surrounding noise regions. In particular, due to the high difference in density along the orthogonal projections the clusters automatically stand out and clear the regions around them. From a uni-dimensional histogram analysis point of view, high levels of noise may significantly hinder the ability of $\mathcal{HT}_1$ to discriminate adjacent clusters of similar density, because the necessary separating valleys between the clusters could potentially be obscured by the increased amount of noise. $\mathcal{NOCEA}$ deals with this challenge by performing the second homogeneity test, $\mathcal{HT}_2$, as described in section 3.4.2. In short, $\mathcal{HT}_2$ consists of a series of chi-square tests at decreasing levels of density to reveal (if any) significant multi-modal structures that potentially indicate the existence of multiple clusters in

higher dimensionality spaces.

$\mathcal{HT}_1$ and $\mathcal{HT}_2$ yield a segmentation of the original histogram into regions of either quasi-uniform or in the worst case unimodal distribution. Although the first two tests can effectively discriminate the clusters from one another, they may fail alone to delineate the boundaries of clusters with accuracy when the level of noise is very high. This is because noise points located near the boundary of a cluster make the density in the tails of the histogram exceed the uniformity threshold T_h. As a result, fully repaired rules would be extended beyond the cluster edges towards the noise regions. $\mathcal{NOCEA}$ deals with this challenge by employing the third homogeneity test $\mathcal{HT}_3$ (section 3.4.3), an elaborate statistical test that examines the distribution of density histogram values for right outliers, i.e. bins corresponding to peaks in unimodal distributions.

For instance, figure 6.51 illustrates $\mathcal{NOCEA}$'s *Recall* curve under varying noise conditions having both enabled and disabled $\mathcal{HT}_2$ and $\mathcal{HT}_3$. Each dataset has 100000 points in 20 dimensions with 10 clusters embedded in 10 dimensions on average. In all the experiments, $\mathcal{NOCEA}$ was run twenty times, varying the random rule seed used to generate the initial population of individuals. The results reported below are based on an arithmetic average of the results over these twenty different random seeds. It can be easily observed that *Recall* degrades rapidly when the percentage of background noise exceeds 20-25%. Typically, when $\mathcal{HT}_2$ and $\mathcal{HT}_3$ are disabled for moderate-to-large levels of noise e.g. 50-70%$< F_{noise}$, $\mathcal{NOCEA}$ produces a single super-rule covering the entire feature space. Therefore, without these elaborate tests the meaningfulness of the obtained partitions may be easily called into question in highly noisy datasets.

The problem of poor cluster discrimination under noisy conditions is rapidly worsening as the dimensionality of the subspaces in which the clusters are embedded decreases. Recall from section 5.11 that the points of a given cluster are spread out along the non-relevant dimensions. In essence, for very large amounts of uniform noise the density of points in the subspace formed by the relevant dimensions of a given cluster and in the surrounding regions of noise is very

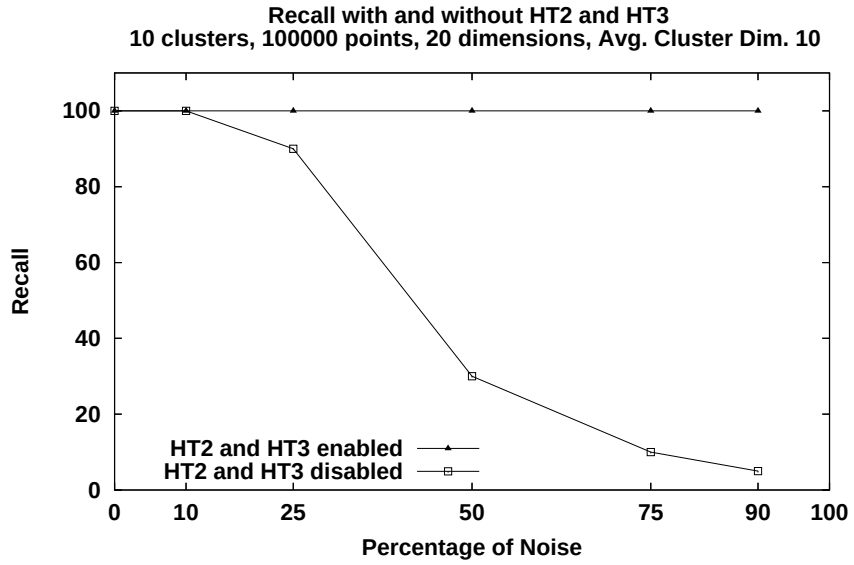


Figure 6.51: *Recall* degrades rapidly for Increasing Noise Without $\mathcal{HT}_2$ and $\mathcal{HT}_3$

similar. Therefore, a relatively thin slice within the subspace of relevant dimensions of a given cluster would easily be extended far beyond the boundaries of the cluster along the bounded dimensions. This could have grave consequences because some regions of the genuine clusters may be either incorrectly assigned to noise rules and/or more severely the fragmentation of the body of a cluster by noise rules may not allow placing non-sparse rules within the backbone of the cluster. Another side-effect is that when a cluster is fragmented by noise rules along its non-relevant dimensions subspace clustering becomes less effective. The *Precision* curves without pruning away the very low density rules, as described in section 5.11.3, during the early stages of the evolutionary search are quite similar to those illustrated in figure 6.51.

6.6 Discovery of Varying Density Clusters

This section verifies our intuition that $\mathcal{NOCEA}$ conforms with an essential requirement for data mining clustering applications, that is, *the discovery of varying density clusters*.

Since the survival of candidate rules depends on only two non-density or geometry-related criteria, i.e. data homogeneity and coverage, one would expect that $\mathcal{NOCEA}$ is by definition able to detect clusters of any density, provided of course that the survival conditions are satisfied. This section may be read in conjunction with section 7.8 (Chapter 7) where the latter gives examples of real-world clusters with large variances in point density.

6.6.1 Apparatus

To investigate the ability of $\mathcal{NOCEA}$ to discover clusters of varying density the particularly popular dataset DS5, shown in figure 6.35(e), is augmented in a higher dimensional space to artificially introduce large differences in the density among the clusters. Recall from section 6.2.3 that DS5 [99] consists of 100000 instances that are equi-distributed among 100 clusters with centres arranged in a 10×10 grid pattern.

To introduce large-scale differences in the density of clusters, DS5 is initially augmented from 2 to 102 dimensions and then each of the original 2-D clusters is embedded in a randomly selected subspace with specific dimensionality. The range of values is set to [0, 100] for all the new dimensions, i.e. 3 to 102. For all the newly added attributes that define the subspace in which the cluster is embedded, the cluster points are uniformly distributed within a small hyper-cube with fixed side length equal to 10. The centre of the hyper-cube is selected at random. For the remaining non-relevant dimensions, the coordinates for the cluster points are drawn independently at random from the uniform distribution defined over the entire range of the attribute. The dimensionality of the first cluster is set to 1, excluding of course the two original dimensions where all clusters are embedded. The difference between the number of bounded dimensions associated with any

two consecutive clusters, e.g. i and i+1 is always 1. As a result, the dimensionality of the first and last cluster is 3 and 103, respectively. To assure randomness in the data generation process the original clusters are processed with a random order that is irrespective of their position in the 2-D grid. Due to the varying cluster dimensionality the point density of the generated clusters spans an extremely wide range. For instance, the density ratio between the denser and sparser cluster is approximately $\left(\frac{1000}{6*6*10^{100}}\right) / \left(\frac{1000}{6*6*10*100^{99}}\right) = ... = 10^{99}$. Note that each cluster has roughly 1000 points while the spreading of points in the first two dimensions is approximately equal to six standard deviations ($\sigma = 1$).

6.6.2 Varying Density Experiment Results

$\mathcal{NOCEA}$ was run twenty times, varying the random rule seed used to generate the initial population of individuals. Due to the properties of the dataset (it contains only convex clusters) $\mathcal{NOCEA}$ always converges to the partitioning shown in figure 6.52 consisting of 100 single-rule clusters. $\mathcal{NOCEA}$ always identifies all the input clusters in the correct subspaces. In all experiments the value of both recall and precision metrics were 100%. These findings validate our intuition that $\mathcal{NOCEA}$ is a generic data mining clustering algorithm well suited even for cases where the density of clusters varies significantly.

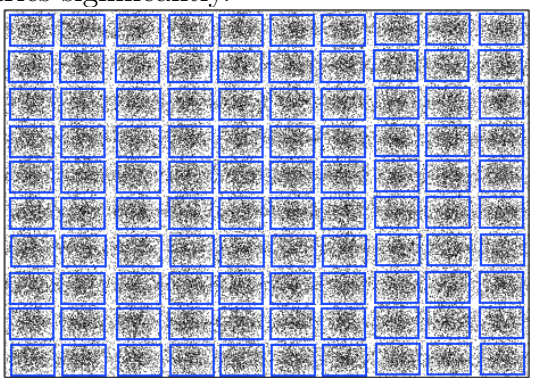


Figure 6.52: A Two-dimensional Projection of the 100 Single-rule Clusters Found by $\mathcal{NOCEA}$ for the Dataset (section 6.6.1) with Varying Density Clusters

Notice that due to discarding all the very low density rules during the early stages of the evolutionary search (section 5.11.3), clusters are recovered at a descending order of point density. However, this elimination procedure is eventually disabled during the evolutionary search, and thereby it does not hinder the discovery of low density clusters.

6.7 Insensitivity to the Order of Data Input

This section demonstrates that the quality of the clustering results by $\mathcal{NOCEA}$ is independent of the order in which the input data are presented. Some techniques, e.g. BIRCH [99], may produce dramatically different clusterings when the input data are processed with different order. In $\mathcal{NOCEA}$, data partitioning is induced by points' membership (inclusion) in clustering rules resulting from space partitioning, where the latter is based on stochastic evolutionary search.

In short, $\mathcal{NOCEA}$ is, by definition, a non data ordering sensitive clustering algorithm because:

- The clustering rules are manipulated by stochastic genetic operators. In other words, the transition rules to move from one solution to another solution is independent of data ordering.
- The fitness function that measures the overall goodness of a candidate solution encompasses no factor that is affected by the data ordering.
- The quality of candidate rules is assessed by analysing density histograms whose construction, in turn, is a non data ordering dependent task.

6.7.1 Apparatus

Two artificial datasets are used in this set of experiments to study $\mathcal{NOCEA}$'s sensitivity to the order of data input, DS1 [48] and DS5 [99], shown in figures 6.35(a) and 6.35(e), respectively. These datasets share a special characteristic that is well-suited for the purposes of this experiment: they both contain clusters that can be approximated by a single clustering rule, which, in turn, means that it is easy and straightforward to compare the results of different runs.

The original data generators [48, 99] were slightly modified to allow controls over the order of the input data. In the simplest version, the set of data points of all clusters and noise is permuted at random. In a less randomised version, the placement of data points in the dataset is controlled by an input chunk parameter specifying the percentage of data points belonging to a given cluster that will be

placed together in consecutive slots of the data table. $\mathcal{NOCEA}$ was tested with 10%, 25%, 50%, 75%, 90%, and 100% chunk size. Additionally, $\mathcal{NOCEA}$ was tested with both fixed and variable chunk size for each cluster. Finally, when the chunk size is less than 100%, there is also an option to process clusters with a random or sequential order.

6.7.2 Data Order Insensitivity Experiment Results

$\mathcal{NOCEA}$ detects the genuine clusters for both datasets approximating each cluster with a single rule. As expected, $\mathcal{NOCEA}$ produces nearly identical results for all possible configurations of data ordering. These results validate the intuition that $\mathcal{NOCEA}$'s output is independent of the order of data input.

6.8 Insensitivity to Initialisation

Some clustering algorithms, e.g. k-means, may generate completely different partitioning, possibly erroneous, with different initialisations. In $\mathcal{NOCEA}$, the stochastic nature of the evolutionary search along with the non distance-based fitness function ensure that the quality of the clustering results does not vary significantly over different initialisations. This is not to suggest that $\mathcal{NOCEA}$ always produces identical rule-sets from different initialisations, but only that the subset of feature space defined by the union of clustering rules, or in other words the clusters themselves, remain roughly the same irrespective of initialisation.

6.8.1 Apparatus

To study the effect of different initialisations to the quality of the clustering results three artificial datasets are used, DS1, DS2 and DS6, whose distributions are shown in figures 6.35(a), 6.35(b), and 6.35(f), respectively.

6.8.2 Insensitivity to Initialisation Experiment Results

Recall from section 5.14 that each individual in the population is initialised with a single hyper-box that is located and sized by a random process. Figures 6.53(a-j) show rule-sets that are typically found by $\mathcal{NOCEA}$ for the three datasets, over different initialisations. Connected rules belonging to the same cluster are assigned the same colour. Looking at these figures, one can easily observe that $\mathcal{NOCEA}$ is always able to correctly identify the genuine clusters irrespective of the selection of the initial rule-seed. Clearly, convex clusters, e.g. spherical, ellipsoid, or rectangular are always captured using a single rule, provided of course that generalisation is enabled. As far as arbitrary-shaped clusters that require multiple rules is concerned, e.g. 'π'-shaped, upside-down 'π'-shaped, or annular, there is no guarantee that $\mathcal{NOCEA}$ will always converge to identical rule-sets, simply because the evolutionary search itself is stochastic. However, the union of the set of disjoint rules that collectively define the body of an arbitrary-shaped cluster, yield an almost identical approximation of the shape of the given cluster.

6.9 Minimal Requirements for Domain Knowledge

$\mathcal{NOCEA}$ is a generic data mining clustering algorithm and has minimal requirements for domain knowledge. For instance, unlike other clustering techniques, e.g. k-means or PROCLUS, it does not require any auxiliary information regarding the number of clusters. In fact, in all the experiments reported throughout the thesis, the optimal number of rules and clusters was automatically determined on the fly by $\mathcal{NOCEA}$ based on the data distribution. This property is very important because domain knowledge is rarely complete and consistent. Additionally, $\mathcal{NOCEA}$ does not presume any canonical type of distribution for the input data. Unlike other clustering techniques, e.g. PROCLUS [5], ORCLUS [6], $\mathcal{NOCEA}$ does not require all clusters to be embedded in the same number of dimensions, neither does it rely on the user to specify the average cluster dimensionality.

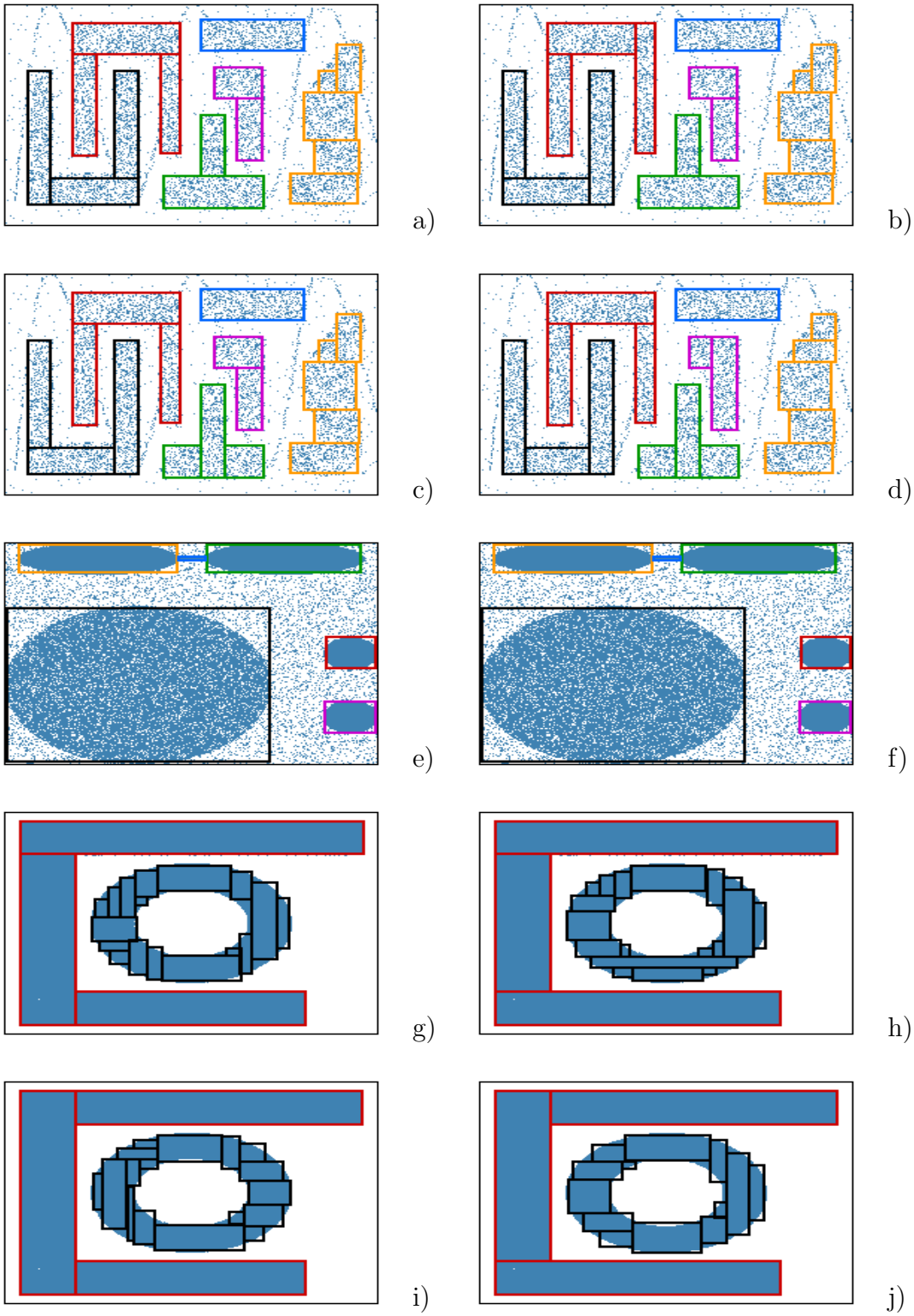


Figure 6.53: $\mathcal{NOCEA}$ Yields Similar Cluster Approximations over Different Initialisations

6.10 Subspace Clustering

This section evaluates the effectiveness and efficiency of $\mathcal{NOCEA}$ on massive and high dimensional datasets containing clusters that are embedded in different subsets of dimensions. In particular, the goals of the experiments are to assess:

- **Efficiency:** Determine how the execution time scales with:
  - Size of the database.
  - Dimensionality of the feature space.
  - Dimensionality of the clusters.
- **Accuracy:** Investigate whether $\mathcal{NOCEA}$ is able to determine the correct subspaces for each cluster, as well as to produce accurate cluster descriptors for the bounded dimensions of all clusters.

$\mathcal{NOCEA}$'s performance for subspace clustering is further demonstrated on seismic data with a multitude of clusters in section 7.15. Experimental results reported in sections 6.10.2, 6.10.3, and 7.15 suggest that $\mathcal{NOCEA}$ scales near linearly with database size and both data and cluster dimensionality, while it always identifies the hidden clusters in the correct subspaces.

6.10.1 Apparatus

The datasets are generated using the generator described in section 6.5.1 which provides control over the number of instances, the dimensionality of the data, and the dimensions for each of the input clusters. Unless otherwise specified, the cluster points are equally distributed among 5 clusters of varying dimensionality and density. Additionally, in all the experiments the level of uniform noise F_{noise} is fixed at 10% of the total number of points. Although $\mathcal{NOCEA}$ is set to run for 300 generations, in most experiments the evolutionary search converged after only 40-50 generations. Therefore the execution times reported in this section correspond to the convergence time rather than the time of completion of the maximum number of generations. In all the experiments reported in sections 6.10.2-6.10.3 twenty independent and randomly initialised runs were performed for all datasets. The reported measurements of execution time and recall-accuracy are based on an arithmetic average of the clustering results over the twenty different random runs.

6.10.2 Subspace Efficiency Results

Scalability With Database Size

Figure 6.54 shows the scalability curve of $\mathcal{NOCEA}$ as the number of records increases from 0.5 to 30 million. In this set of experiments each dataset has 20 dimensions while each one of the 5 hidden clusters is embedded, on average, in some 5-dimensional subspace. As expected, $\mathcal{NOCEA}$ scales near *linearly* with the number of instances. The linear behaviour is explained as: Given the relatively low dimensionality of the feature space $\mathcal{F}$ and the small number of clusters, the execution time is dominated by the construction of the frequency histograms, which, in turn, is a task of linear complexity with the database size. In other words, for low dimensional datasets containing a small number of clusters, $\mathcal{NOCEA}$ spends far more time repairing candidate rules rather than performing other genetic operations such as recombination, generalisation, and mutation. For low-dimensionality datasets, e.g. d<25, scalability could be further improved by replacing the linear data-scanning mechanism that is currently employed by $\mathcal{NOCEA}$ with a faster hyper-rectangular query mechanism, e.g. *range* or *k-d* tree [26].

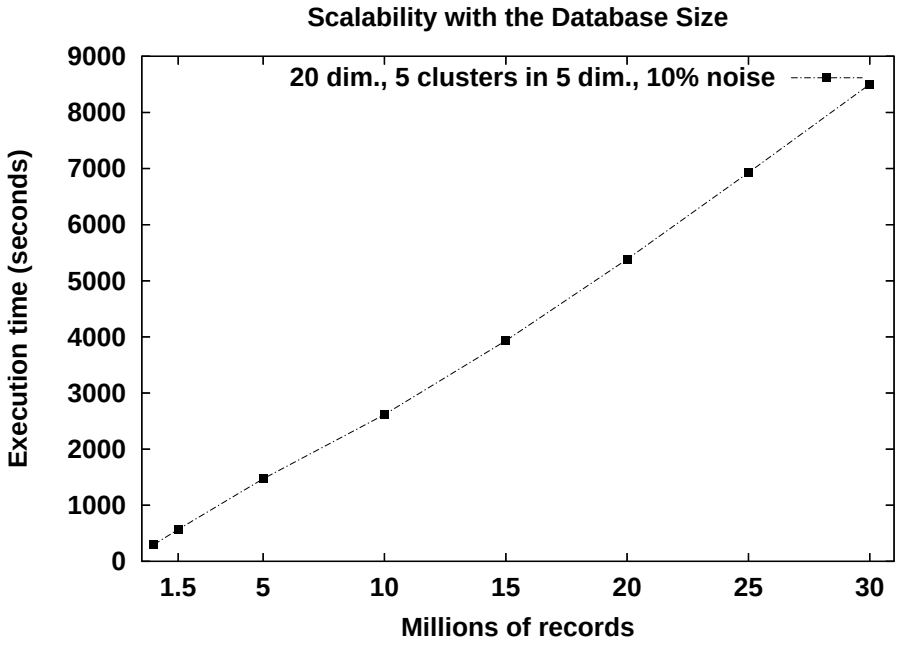


Figure 6.54: Execution Time *vs.* Number of Records

Scalability With Dimensionality of the Data Space

Figure 6.55 shows the scalability of $\mathcal{NOCEA}$ as the dimensionality of the data space increases from 20 to 200. In this series of experiments, each dataset has 100000 records and there are 5 clusters being embedded in some 5-dimensional subspace. The percentage of noise was set to 10% of the total number of points. The curve exhibits a small *super-linear* trend, as for a given rule-set, $\mathcal{NOCEA}$ must build at least one density histogram for each rule in every dimension. Additionally, as the data dimensionality increases the application of the genetic operators becomes increasingly more expensive mainly due to the increasing number of constraint-checking computations required to yield individuals without overlapping rules. Note that both tasks (construction of density histograms and constraint checking) are of linear complexity with data dimensionality.

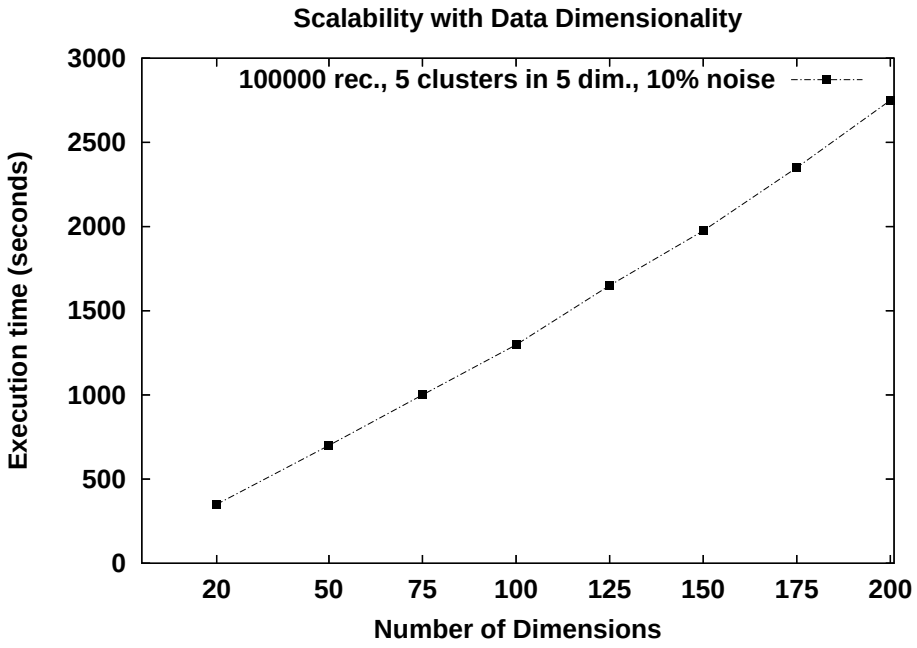


Figure 6.55: Execution Time *vs.* Dimensionality of the Data Space

Scalability With Average Cluster Dimensionality

Figure 6.56 shows the dependency of the execution time on the average cluster dimensionality where the latter increases from 5 to 50 in a 100-dimensional data space. In each case, the dataset has 100000 records distributed over 5 hidden clusters with a fixed 10% level of background noise. The *super-linear* scale up

in the execution time with the dimensionality of the hidden clusters is explained as follows: the higher the average cluster dimensionality the more likely it is for the evolutionary search operators to produce non-homogeneous candidate rules along the bounded dimensions of the clusters. Hence, extra repairing operations are required to obtain a feasible, i.e. homogeneous, rule-set. The computational overhead introduced by the additional repairing operations for a given cluster is generally proportional to the dimensionality of that cluster. Additionally, as the dimensionality of hidden clusters increases, more space becomes available, i.e. uncovered, for the mutation operator to grow existing rules or to produce new ones. Despite the fact that no access to the database is required when mutating a genome, the cost of applying the mutation operator may be substantial, especially for high dimensional spaces or datasets with a multitude of clusters (see section 7.15.4). Note that mutation in $\mathcal{NOCEA}$ is a linear complexity task with regard to the dimensionality of the dataset.

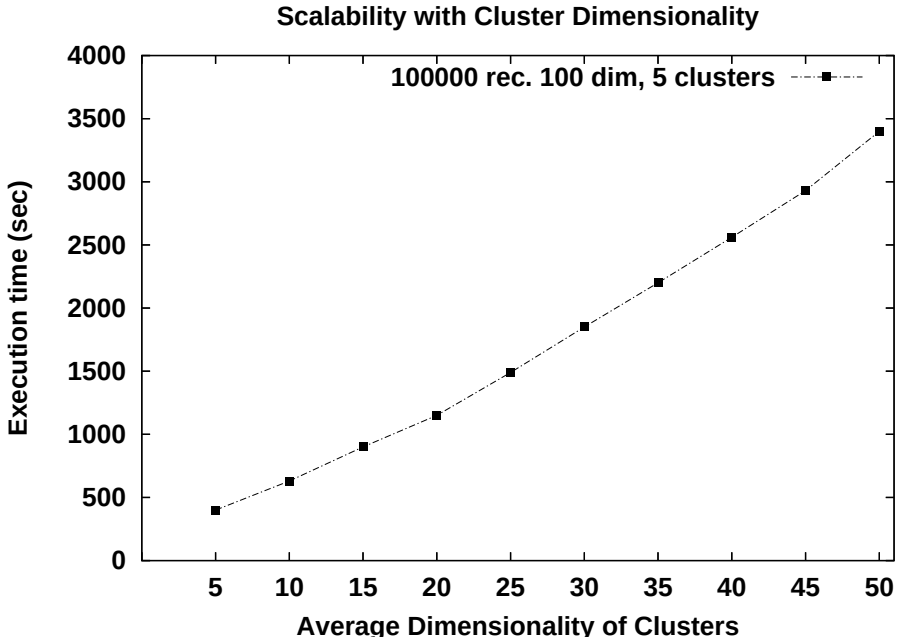


Figure 6.56: Execution Time *vs.* Average Cluster Dimensionality

6.10.3 Subspace Detection and Accuracy Results

This section investigates how accurately $\mathcal{NOCEA}$ is able to approximate the clusters and detect the corresponding subspaces.

Apparatus

To assess the quality of the results of $\mathcal{NOCEA}$ for subspace clustering, a synthetic dataset was generated using the generator of section 6.5.1. The results are presented as a mapping matrix that lists the relevant dimensions and the corresponding ranges of the input clusters as well as those output by $\mathcal{NOCEA}$. Table 6.4 depicts a typical case input and output cluster mapping on a 20-dimensional dataset of 100000 instances with F_{noise}=10%, where 10 equal data coverage clusters are embedded in some 5-dimensional subspaces on average. The type of a cluster can be either normal or uniform with an equal-probability. For each input and output cluster the mapping table lists: 1) `#points`: number of points, 2) `#Dim`: cluster dimensionality, 3) `Dim`: relevant dimensions, 4) `Range`: range of relevant dimensions. For the jth normal-type cluster the effective range of the distribution along the mth dimension is the interval $[c_m^j - 3\sigma_m^j,\ c_m^j + 3\sigma_m^j]$, where σ_m^j and c_m^j denote the standard deviation and centre of the points of the jth cluster along the mth bounded dimension, respectively. Additionally, each bounded dimension for the input clusters is accompanied by a column (`Std`) specifying the standard deviation of points along that dimension. Finally, the last column (`Missing`) shows the percentage of peripheral points in each input cluster that are missed by the corresponding output cluster.

Subspace Detection and Accuracy Experimental Results

Table 6.4 shows the substantially accurate mapping from the input data to subspace clusters identified. These results are an average of twenty independent runs which were randomly initialised. Remarkably, there is a perfect matching between the sets of relevant dimensions of the output clusters and their corresponding input clusters. This is important for applications that require not only a good partitioning of the data, but also additional information as to what dimensions are relevant for each partition. $\mathcal{NOCEA}$ always recovers all of the 10 input clusters in the correct subspaces using *only one* rule for each cluster.

For uniform-type clusters, the boundaries reported by $\mathcal{NOCEA}$ are a fairly accurate approximation of the input clusters. The marginal difference between the input and output cluster boundaries along the bounded dimensions is due to the discretisation error introduced by the quantisation of the feature space $\mathcal{F}$ that makes the values of the rule boundaries being drawn from discrete rather than continuous domains. The fact that for some uniform-type clusters, i.e. $\mathcal{C}_0$, $\mathcal{C}_1$, and $\mathcal{C}_2$, the corresponding output cluster has been assigned more points (indicated by the negative numbers in column `Missing` of the mapping table 6.4) is because the noise points were randomly distributed throughout the entire feature space, and therefore some of them have actually been placed inside the regions of clusters.

For normal-type clusters, although $\mathcal{NOCEA}$ produces less accurate descriptors compared to uniform-type clusters, it nevertheless detects all clusters in the correct subspaces. The less than perfect matching between the input and output cluster boundaries is because the repair operator has been designed to separate the central body of a normal distribution from its low density tails. This is not necessarily an error, because $\mathcal{NOCEA}$ seeks, by definition, regions with quasi-uniform data distribution. Not surprisingly, the number of points that are missed in the boundaries of a normal-type cluster is proportional to the dimensionality of that cluster. A small percentage of the total point loss is due to the properties of the normal distribution, where the maximum distance between a point in the cluster and the corresponding centre is unbounded. In other words, a data point may be arbitrarily far from its owning cluster. So a very limited percentage of points that are initially assigned to an input cluster, are placed far away from the corresponding centre, thereby they do not belong anymore to the given cluster. Finally, the greater the standard deviation of points, the smaller the kurtosis, i.e. concentration of data points around the centre, of the distribution, and consequently the fewer points are lost in the tails. $\mathcal{NOCEA}$ consistently delivers very similar qualitative results (i.e. correct detection of cluster subspaces and relatively accurate cluster boundaries) for all the experiments related to subspace clustering that are reported throughout this thesis. Further research is required to address the problem of missing peripheral cluster points.

Database Size=100000, Data Dimensionality=20, F_{noise}=10% Number of Clusters = 5, Average Cluster Dimensionality = 5										
	Input					Output				Missing
Cluster	#Points	#Dim.	Dim.	Range	Std	#Points	#Dim.	Dim.	Range	(%)
$\mathcal{C}_0$			0	73.106 - 96.213	3.851			0	72.946 - 96.469	
Type: Uniform	9000	2	1	64.982 - 85.133	3.359	9512	2	1	64.656 - 85.300	-5.68%
			0	46.888 - 69.094	3.701			0	46.483 - 69.418	
$\mathcal{C}_1$			1	51.805 - 70.558	3.125			1	51.618 - 71.175	
Type: Uniform	9000	3	13	11.188 - 23.495	2.051	9050	3	13	11.116 - 23.620	-0.5%
			0	25.503 - 42.929	2.904			0	25.312 - 42.954	
$\mathcal{C}_2$			1	5.203 - 22.168	2.827			1	4.896 - 22.281	
Type: Uniform	9000	4	13	39.483 - 62.581	3.85	9014	4	13	38.902 - 63.215	-0.1%
			18	55.356 - 76.222	3.478			18	55.000 - 76.999	
			0	75.035 - 96.521	3.581			0	74.710 - 97.057	
			1	28.413 - 46.607	3.032			1	28.257 - 46.728	
			8	60.534 - 83.662	3.855			8	60.000 - 83.999	
$\mathcal{C}_3$			10	34.682 - 53.629	3.158			10	34.000 - 54.000	
Type: Uniform	9000	8	11	10.275 - 22.941	2.111	9000	8	11	10.001- 24.000	0%
			12	77.158 - 96.654	3.249			12	76.235 - 97.026	
			13	37.469 - 56.874	3.234			13	36.818 - 56.963	
			19	53.763 - 72.894	3.189			19	53.000 - 73.000	
			0	73.053 - 86.327	2.212			0	72.946 - 86.472	
			1	33.74 - 48.859	2.52			1	33.69 - 49.444	
			4	6.515 - 24.593	3.013			4	6.001 - 25.000	
$\mathcal{C}_4$			5	69.213 - 88.092	3.146			5	68.749 - 88.391	
Type: Uniform	9000	9	7	84.252 - 97.811	2.26	9000	9	7	83.917 - 97.902	0%
			11	67.374 - 86.116	3.124			11	66.998 - 86.997	
			14	42.806 - 64.033	3.538			14	42.000 - 64.999	
			16	68.736 - 84.38	2.607			16	67.998 - 84.997	
			19	46.83 - 61.815	2.497			19	46.000 - 62.000	
$\mathcal{C}_5$			0	30.019 - 52.527	3.751			0	35.309 - 47.071	
Type: Normal	9000	3	1	73.078 - 85.827	2.125	6579	3	1	76.065 - 83.127	26.9%
			5	34.01 - 55.378	3.561			5	38.393 - 50.893	
			0	52.393 - 71.809	3.236			0	57.068 - 67.653	
$\mathcal{C}_6$			1	18.531 - 41.005	3.746			1	23.911 - 35.863	
Type: Normal	9000	4	8	6.883 - 30.404	3.92	6024	4	8	12.000 - 26.000	33%
			15	32.05 - 47.456	2.568			15	35.002 - 44.001	
			0	14.962 - 28.102	2.19			0	17.667 - 25.312	
			1	61.683 - 84.105	3.737			1	66.829 - 78.781	
$\mathcal{C}_7$			9	76.182 - 90.424	2.374			9	79.000 - 88.000	
Type: Normal	9000	6	11	28.783 - 46.442	2.943	5412	6	11	32.000 - 42.999	39.8%
			12	35.329 - 48.417	2.181			12	37.623 - 45.543	
			13	79.771 - 99.838	3.344			13	84.749 - 95.169	
			0	27.999 - 44.196	2.699			0	31.781 - 40.602	
			1	25.02 - 43.531	3.085			1	29.344 - 39.122	
			5	17.25 - 38.322	3.512			5	21.429 - 33.929	
$\mathcal{C}_8$			6	51.893 - 72.346	3.409			6	56.000 - 68.000	
Type: Normal	9000	7	7	47.66 - 71.095	3.906	4656	7	7	53.148 - 65.735	48.2%
			10	8.47 - 29.026	3.426			10	13.000 - 25.000	
			15	27.617 - 40.719	2.184			15	30.002 - 38.001	
			0	28.397 - 44.605	2.701			0	31.781 - 40.602	
			1	64.017 - 78.337	2.387			1	67.372 - 74.978	
			4	13.67 - 33.742	3.345			4	18.000 - 28.999	
			5	20.017 - 33.046	2.171			5	22.322 - 30.358	
$\mathcal{C}_9$			7	11.153 - 28.259	2.851			7	15.387 - 24.478	
Type: Normal	9000	9	12	65.085 - 80.606	2.587	3759	9	12	68.315 - 77.225	58.2%
			14	13.965 - 36.688	3.787			14	19.000 - 31.000	
			15	13.004 - 32.978	3.329			15	17.002 - 29.002	
			18	52.192 - 66.024	2.305			18	55.000 - 62.999	

Table 6.4: Typical Case Accuracy Results for Subspace Clustering by $\mathcal{NOCEA}$

6.11 Summary

This chapter has presented an experimental evaluation of $\mathcal{NOCEA}$ on a wide range of benchmark artificial datasets. The experimental results demonstrated that $\mathcal{NOCEA}$ meets the key DM clustering criteria. In particular we showed that: $\mathcal{NOCEA}$ produces interpretable output in the form of *disjoint* and *axis-aligned hyper-rectangular* clustering rules with *homogeneous* data distribution; the output is minimised for ease of comprehension. $\mathcal{NOCEA}$ has the ability to discover homogeneous clusters of arbitrary density, geometry, and data coverage. $\mathcal{NOCEA}$ effectively treats high dimensional data and it effectively identifies subspace clusters being embedded in arbitrary subsets of dimensions. $\mathcal{NOCEA}$ has near linear scalability with respect to the database size, and both data and cluster dimensionality. $\mathcal{NOCEA}$ produces similar quality results irrespective of initialisation and order of input data. $\mathcal{NOCEA}$ is exceptionally resistant to background noise. Finally, $\mathcal{NOCEA}$ has minimal requirements for *a priori* knowledge, and does not presume any canonical distribution of the input data.

Chapter 7

Earthquake Analysis Case Study

Capsule

This Chapter presents a real-world application of $\mathcal{NOCEA}$ in the earthquake domain. The analysis primarily focuses on clustering earthquakes associated with the highly-active crustal deformation along the African-Eurasian-Arabian collision boundary. Initially, a brief introduction regarding the geo-tectonics and seismicity associated with this region, is provided. The chapter continues with a detailed description of the dataset itself along with a preliminary human-eye clustering. Next, the discovered knowledge, e.g. rules and clusters, is listed, and accompanied by various statistics. The following sections verify the theoretical properties of $\mathcal{NOCEA}$ e.g. discovery of clusters with arbitrary data coverage, density, geometry, orientation, effective subspace clustering, in a challenging real-world case study, using representative pieces of knowledge discovered from the earthquake dataset. Next, the chapter explains how the discovered knowledge can be exploited to compile improved seismic hazard maps. A detailed study of evolution of high-magnitude seismicity whose social impact is severe, is also presented. Finally, the chapter concludes with an extensive efficiency and effectiveness performance evaluation on a combination of massive synthetic and real-world datasets. The scalability results show an impressive near-linear dependency on the database size, data and cluster dimensionality, as well as potential for high levels of task parallelism, reaching a speed up of 13.8 on 16 processors.

7.1 Introduction

The goal of earthquake prediction is to develop relatively reliable probabilistic estimates of potentially damaging earthquakes early enough to minimise loss of life and property. Scientists estimate earthquake probabilities in two ways: by studying the history of past earthquakes in a specific area and the rate at which strain accumulates in the rocks [1, 2]. Our goal is to mine a dataset containing seismic related parameters for possible correlations between earthquakes which occurred along the highly-active African-Eurasian-Arabian collision zone. In particular, $\mathcal{NOCEA}$ seeks clusters that represent regions with relatively homogeneous behaviour as far as the seismic activity is concerned. Our intention is *not* to interpret the seismicity of a region deeply, but rather to provide a comprehensive summary and visualisation of earthquake activity to aid seismologists in gaining a deeper insight into the phenomenon, and allow them to improve the reliability of their estimates.

The remainder of this Chapter is structured as follows: Section 7.2 briefly explains how are earthquakes generated. Section 7.3 recalls the well-known *Gutenberg-Richter* (*G-R*) power-law relation for the frequency of occurrence of earthquakes with a given magnitude. Section 7.4 gives a classification of earthquakes based on the Richter magnitude scale. Section 7.5 summarises the main seismogenic zones along the African-Eurasian-Arabian plate boundary. Section 7.6 provides a detailed description of the earthquake dataset used in this Chapter along with a preliminary human-eye clustering. Section 7.7 presents the knowledge discovered by $\mathcal{NOCEA}$ (i.e. rules, clusters, and various statistics) for the seismic dataset, the configuration settings for both EA- and clustering-related parameters, and a classification of clustering rules to facilitate the analysis of the results. Section 7.8 demonstrates $\mathcal{NOCEA}$'s ability to discover clusters of arbitrary density. Section 7.9 verifies $\mathcal{NOCEA}$'s ability to self-adjust well to the geometry and size of non-convex clusters. Section 7.10 proves that $\mathcal{NOCEA}$ can discover rules of arbitrary data coverage dictated by the underlying data distribution. Section 7.11 gives examples of subspace clustering in the seismic dataset,

explains the natural interpretation of irrelevant features from an earthquake prediction point of view, and finally highlights the vital role of generalisation and KDE histogram smoothing for effective subspace clustering. Section 7.12 validates the ability of $\mathcal{NOCEA}$ to discover arbitrary-shaped clusters of varying numbers of rules, point coverage, density, and geometry. Section 7.13 describes how the seismicity knowledge discovered by $\mathcal{NOCEA}$ can be exploited to compile improved seismic hazard maps. Section 7.14 discusses the historic evolution of moderate-to-high-magnitude seismicity. Section 7.15 evaluates the efficiency and effectiveness of $\mathcal{NOCEA}$ for subspace clustering by conducting a variety of experiments using a mixture of synthetic and real-world datasets. Section 7.15.5 reports experiments related to task parallelism.

7.2 How Are Earthquakes Generated?

One of the most frightening and destructive phenomena of nature is a severe earthquake and its terrible after-effects. The earth is formed of several layers that have very different physical and chemical properties [1, 2]. The outer layer, which averages about 70km in thickness, consists of about a dozen large, irregularly shaped tectonic plates that slide over, under and past each other on top of the partly molten inner layer. Most of the earth's seismic activity, e.g. volcanoes and earthquakes, occurs at the boundaries where the plates collide. The plates are made of rock and drift all over the globe, they move both horizontally and vertically. A fault is a fracture or zone of fractures in the earth's crust along which two blocks of the crust have slipped with respect to each other. Faults allow the blocks to move relative to each other. This movement may occur rapidly, in the form of an earthquake - or may occur slowly, in the form of creep. An earthquake is caused by the sudden slip of a fault. Stresses in the earth's outer layer push the sides of the fault together. Stress builds up and the rock slips suddenly, releasing energy in waves that travel through the earth's crust and cause the shaking that we feel during an earthquake.

7.3 Gutenberg-Richter (G-R) Power Law

The *Gutenberg-Richter* (*G-R*) power law relation for the frequency of occurrence of earthquakes is a well-known trait of the dynamics of seismicity, suggesting that most seismically active regions exhibit populations of small-to-moderate earthquakes that obey the *G-R* logarithmic relationship:

$$\log(n) = a - bM \tag{7.35}$$

where n is the number of earthquakes with seismic energy radiated greater than M (magnitude), while a and b are constants.

7.4 The Richter Magnitude Scale

The severity of an earthquake is usually expressed in the Richter magnitude scale that was developed in 1935 by Charles F. Richter of the California Institute of Technology as a mathematical device to compare the size of earthquakes. In short, magnitude is related to the amount of seismic energy released at the hypocenter of the earthquake. It is based on the amplitude of the earthquake waves recorded on instruments which have a common calibration. The magnitude of an earthquake is thus represented by a single, instrumentally determined value. To facilitate our analysis earthquakes are classified in six categories based on their magnitude (U.S. Geological Survey, http://www.usgs.gov/):

Descriptor	**Magnitude**
Great	8 and higher
Major	7 - 7.9
Strong	6 - 6.9
Moderate	5 - 5.9
Light	4 - 4.9
Minor	3 - 3.9

Table 7.5: Classification of Earthquakes Based on the Richter Magnitude Scale

7.5 African-Eurasian-Arabian Plate Boundary

The *Aegean* sea and the surrounding area, which extends from the Italian peninsula, in the west (7^oW), to the Karviola junction of the North (*NAF*) and East (*EAF*) Anatolian Faults in east Turkey (41^oE), experience a rapid and intense crustal deformation [64, 77]. Briefly, the deformation and the seismic excitation along this region is mainly attributed to the northward motion of the African and Arabian tectonic plates relative to the Eurasian. The solid bold line in figure 7.57 delineates the Africa-Eurasia, Eurasia-Arabia plate collision zone.

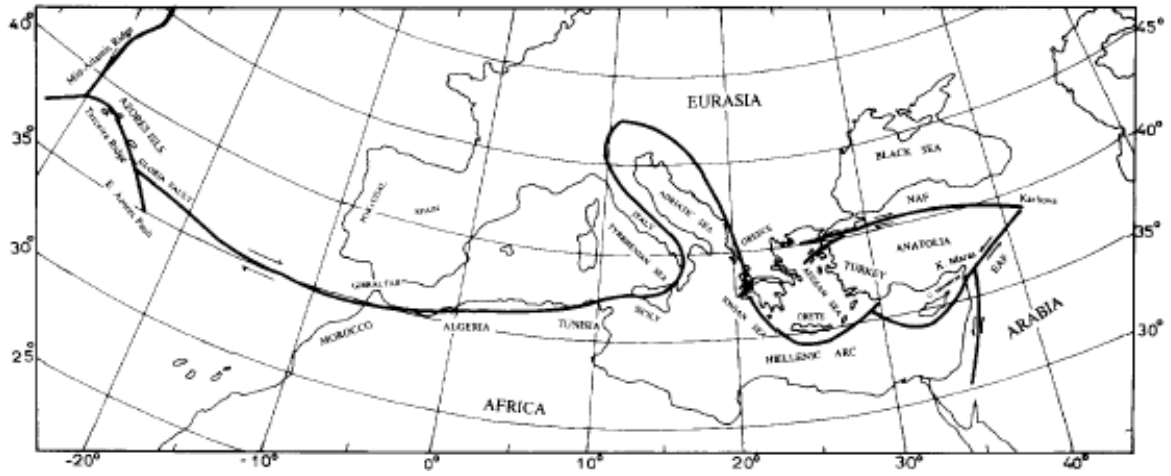


Figure 7.57: The Africa-Eurasia, Eurasia-Arabia Plate Boundary [64]

7.5.1 Seismogenic Zones in Italian Peninsula

The Africa-Eurasia plate boundary along the Italian peninsula is characterised by four seismogenic zones [77], as shown in figure 7.58: *a*) N. Italy, including the Alps and part of the northern Apennines, *b*) the area of central Italy, south of 43^oN along the central Apennines, *c*) the region of the southern Apennines and the Calabrian Arc, and *d*) Sicily. Further east, in the Adriatic Sea the collision boundary is not well defined.

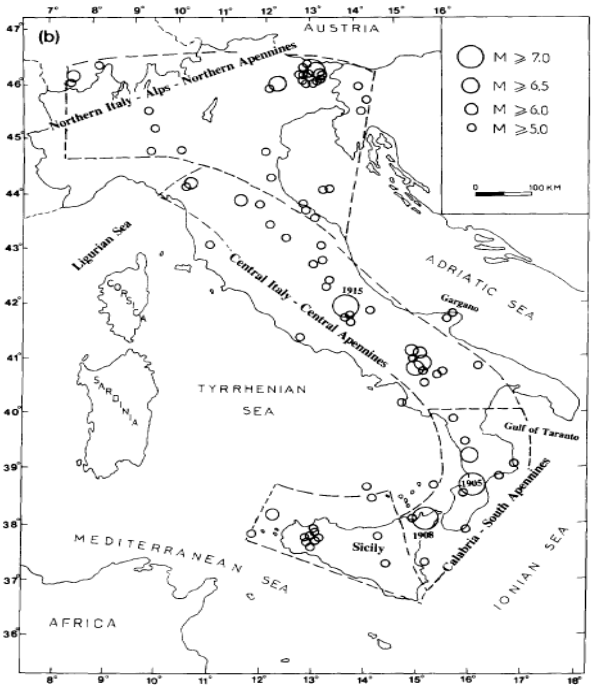


Figure 7.58: The Main Seismogenic Zones Along the Italian Peninsula [64]

7.5.2 Seismogenic Zones in Mainland Greece - Aegean Sea

The *Aegean* sea and the surrounding area including mainland Greece, Albania, FYROM, S. Bulgaria, W. Turkey and part of the Northern Eastern Mediterranean, lie on the most active part of the Africa-Eurasia collision zone. This region is mainly characterised by the westward motion of Turkey and the southwestward motion of the southern Aegean, relative to Eurasia. This motion causes the subduction of the African plate beneath the Eurasian plate along the *Hellenic Arc*. The arrows plotted in figure 7.59 provide a sense of how the crust in the southern Aegean is overriding the African plate from north to south.

The most prominent morphological features in the Aegean and surrounding area from south to north are: the *Mediterranean Ridge*, the *Hellenic Trench*, the *Hellenic Arc* and the *Northern Aegean Trough* [64]. Figures 7.59-7.60 illustrate the main seismogenic zones in the Aegean sea and surrounding lands based on two studies [64, 77]. The *Mediterranean ridge* is a submarine crustal swell that extends from the Ionian Sea to Cyprus and parallels the *Hellenic Trench*. The *Hellenic Trench* consists of a series of depressions with depth to about 5km. It

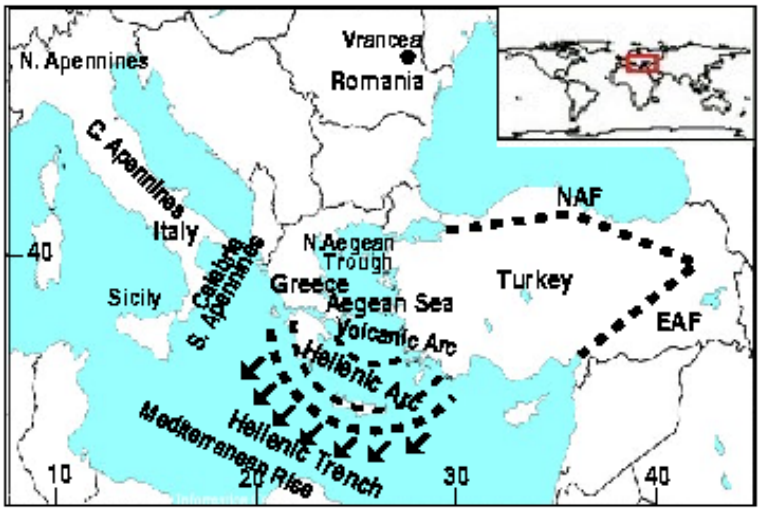


Figure 7.59: The Motion of Tectonic Plates in the Aegean Sea

parallels the *Hellenic Arc* and includes some linear trenches, such as the Pliny and Strabo southeast of Crete and the Ionian Trench. The *Hellenic Arc* is formed by the outer sedimentary arc, a link between the Dinaric Alps and the Turkish Taurides, and the inner volcanic arc, which parallels the sedimentary arc at a mean distance of about 120km. The *Volcanic Arc* consists of several volcanic islands and includes andesitic active volcanoes (Methana, Santorini, Nisyros) and solfatara fields. Between the sedimentary and the volcanic arc is the *Cretan Trough* with depth to about 2km. The most interesting feature of the Northern Aegean is the *Northern Aegean Trough* with depth to about 1.5km. Its extension to northeast is probably the small depressions of the Marmara Sea. The distribution of the epicenters of the large shallow shocks (depth<60km) form several seismic zones. The external seismic zones form a continuous large seismic belt along the external (convex) side of the *Hellenic Arc*, and its extension along the western coast of central Greece, Albania and former Yugoslavia. All other zones constitute the internal seismic zones, which have an almost east-west direction. The spatial distribution of the foci of the intermediate focal depth (60km<depth<180km) earthquakes is of much interest because it defines basic properties of the deep tectonics in this area and because the strongest earthquakes (with M≈8.0) in this region are of intermediate focal depth.

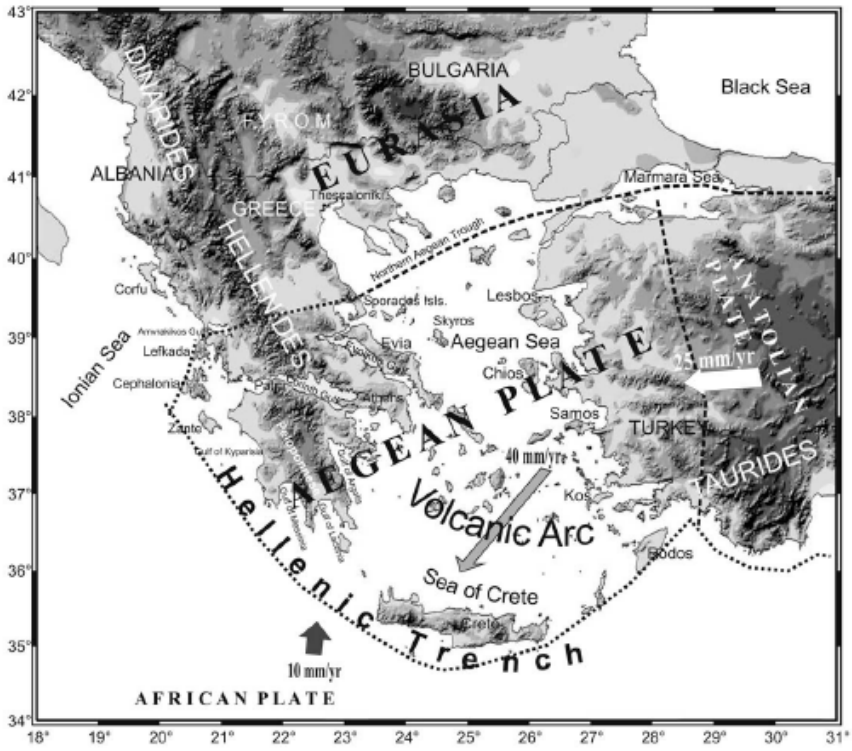


Figure 7.60: The Main Seismogenic Zones in Aegean Sea and Surrounding Lands

7.5.3 Seismogenic Zones in Turkey

Further east, as shown in figure 7.61, the motion of the Arabian plate towards the north, causes the westward movement of the Anatolian plate (Turkey) relative to Eurasia along the North Anatolian Fault (NAF) in the north and East Anatolian Fault (EAF) in the southeast.

7.5.4 Seismogenic Zones in Romania

Finally, the Vrancea region, as shown in figure 7.62, in south-eastern Romania, localised in the rectangle 45-46^{o}N and 26-27^{o}E, consists of both shallow (Depth<60km) and intermediate-depth (60km<Depth<300km) events with the largest magnitudes above 7.0. The Vrancea region is characterised by intense seismic activity in a remarkably confined volume and in a direction that is nearly perpendicular to the earth's surface.

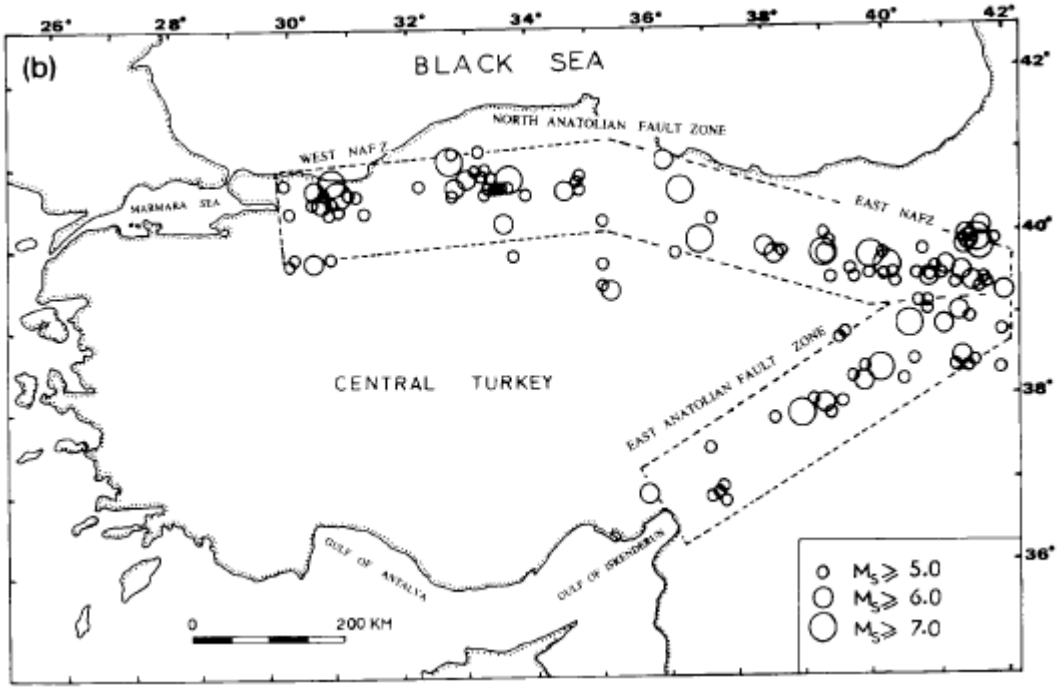


Figure 7.61: The Main Seismogenic Zones in Turkey [64]

Figure 7.62: The Vrancea Region in Romania and the Hellenic Arc in Greece

7.6 The *ANSS* Earthquake Catalogue

7.6.1 *ANSS* Dataset Description

The *American Advanced National Seismic System* (ANSS) (*http://www.anss.org/*) donated the earthquake dataset that is used in the thesis. It contains 34593 seismic events recorded from 07/01/1961 to 19/04/2004. The origin of an earthquake is specified by three spatial dimensions, that is, focal `Depth`, `Longitude-Latitude` epicentre coordinates, one temporal dimension (`Time`), and finally by the amount of energy (`Magnitude`) that was released. Table 7.6 summarises the characteristics of the dataset. $\mathcal{NOCEA}$ operates on a [`Time`×`Longitude`×`Latitude`×`Depth`×`Magnitude`] axis-aligned rectangular grid with resolution [$\texttt{125}^{days}\times\texttt{0.1}^{o}\times\texttt{0.1}^{o}\times\texttt{1.0}^{km}\times\texttt{0.1}^{Ric.}$]. The third column represents the bin width computed by the $\mathcal{TSQ}$ quantisation algorithm (Chapter 4) while the last column shows the precision of the recorded measurements.

Size=34593	**Domain**	**#Bins**	**Bin Width**	**Precision**
`Time (Days)`	07/01/1961 - 19/04/2004	126	125	1
`Longitude (Degrees E)`	7 - 45	381	0.1	0.1
`Latitude (Degrees N)`	32 - 47	150	0.1	0.1
`Depth (km)`	0 - 500	488	1.0	1.0
`Magnitude (Richter)`	3.0 - 7.7	47	0.1	0.1

Table 7.6: Various Characteristics of the *ANSS* Earthquake Dataset

7.6.2 Visual Clustering of the *ANSS* Dataset

This section provides a preliminary human-eye clustering of the *ANSS* dataset.

The *ANSS* earthquake dataset has distinct spatio-temporal-magnitude cluster structures mainly due to the geological heterogeneity of the spatial dataspace and the discontinuous nature of the faulting zones [64]. However, as shown in section 7.7, not all complex structures in a multi-dimensional space can be always extracted by a non fully-dimensional projective clustering method.

Figure 7.63 displays the spatial [`Longitude`×`Latitude`×`Depth`] distribution of earthquakes. Figures 7.64(a-e) depict the uni-dimensional frequency histograms

for all dimensions, while figures 7.65-7.66 illustrate some pairwise scatter diagrams. The visual inspection of these figures reveals some distinct trends of seismic activity.

- As expected, most earthquakes occur along the collision boundaries between the tectonic plates (figure 7.66(c)), particularly in the Aegean Sea, forming variable-length slices along the depth axis as clearly shown in figures 7.63 and 7.66(a-b). These events are located mainly close to the surface 0-50km. Another interesting observation that can be easily discerned from figures 7.64(e) and 7.66(a-b) is that the vast majority of seismic activity is confined along three shallow, remarkably thin (1km), horizontal slices located at focal depths 2-3km, 9-10km and 32-33km that are highlighted with purple, dark-blue and cyan colour in figure 7.63, respectively. These highly-active regions are separated from one another by significantly less active slices of varying thickness.

- As expected, the overall seismic activity is not evenly distributed along the magnitude axis (figure 7.64(d)). This finding obeys the G-R power-law (see section 7.3), stating that stresses built up in the earth's outer layer are usually relaxed via a few large-scale earthquakes that are accompanied by multitudinous low-to-moderate fore- and after-shock events.

- The highly-negative skew in the Time frequency histogram (figure 7.64(a)) is attributed to the incompleteness of the instrument measurements in the past, and does not reflect a chronologically increasing seismic excitation.

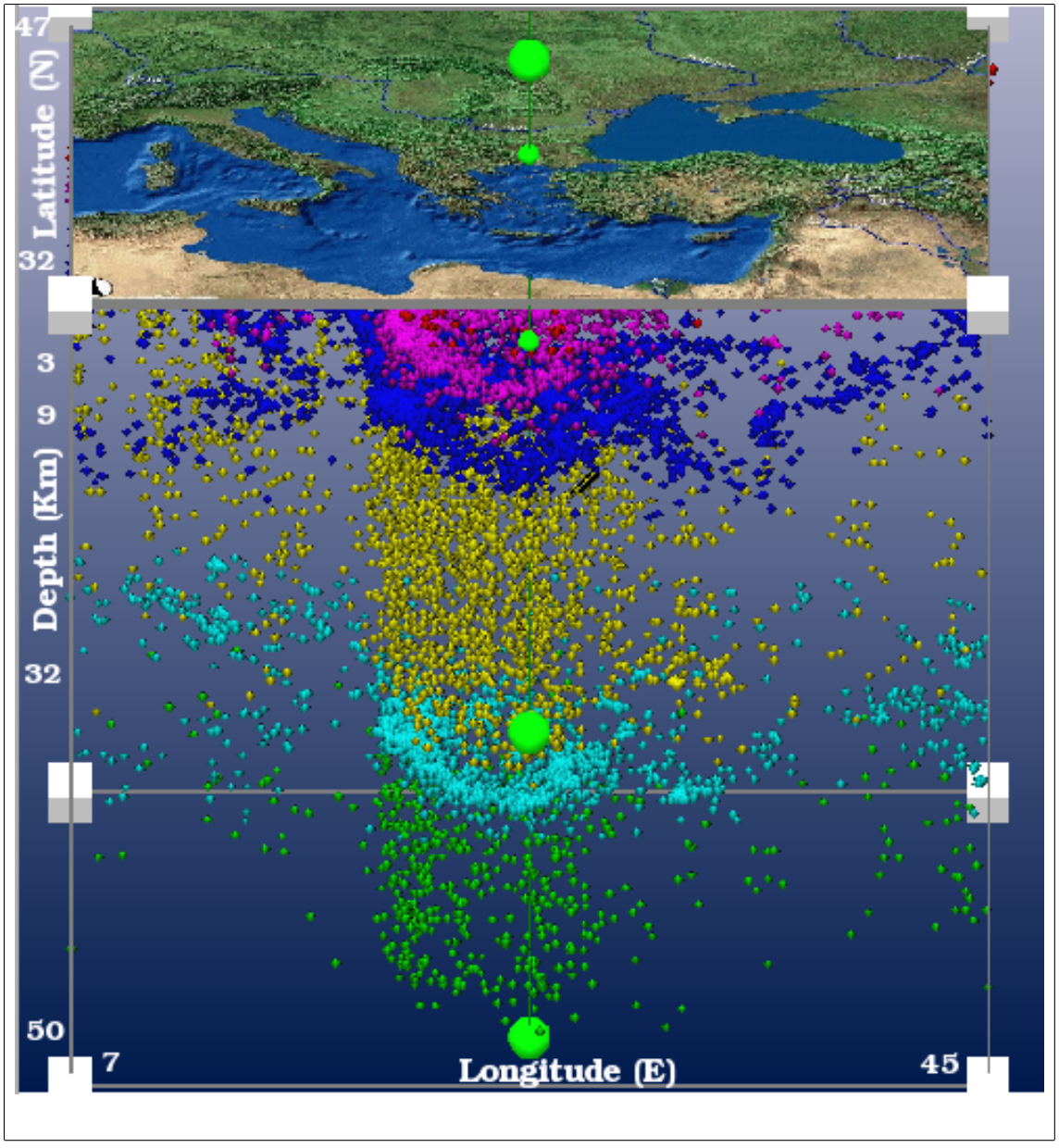
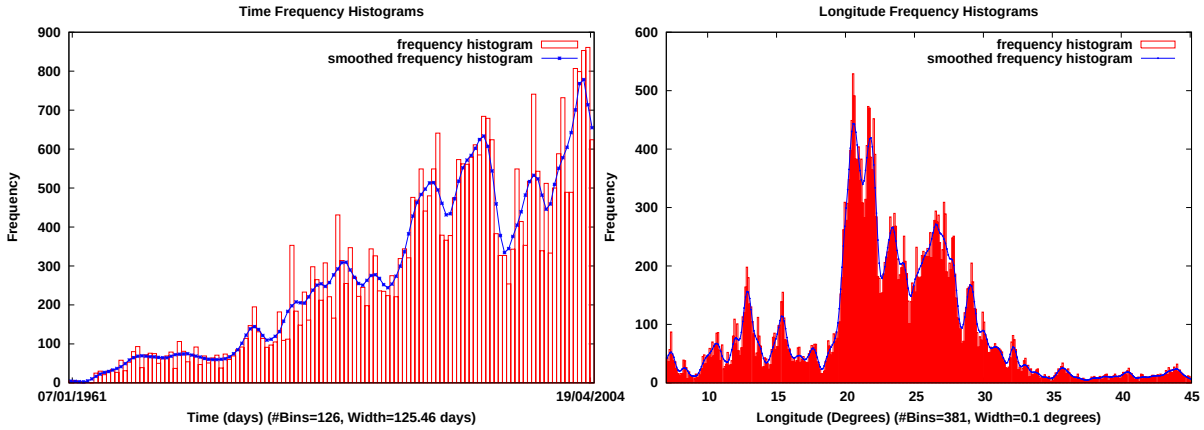
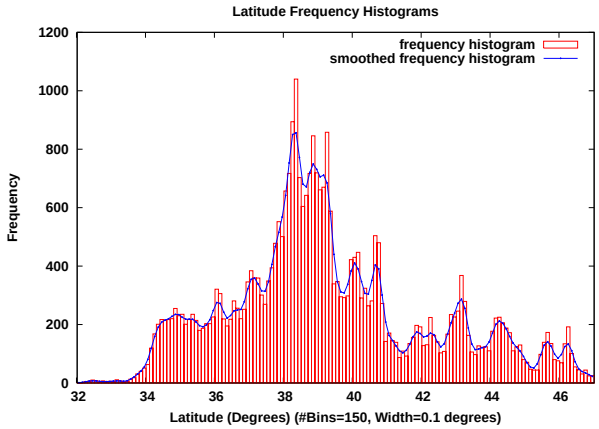


Figure 7.63: The Spatial [`Longitude×Latitude×Depth`] Distribution of Earthquakes

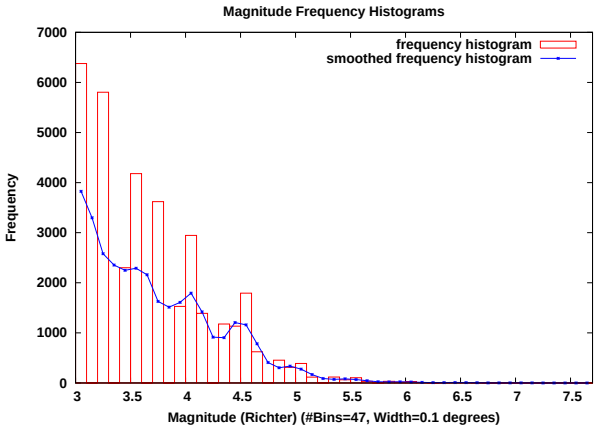


a)

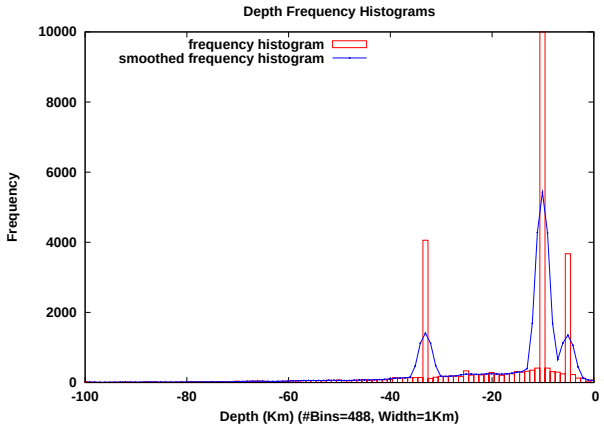
b)



c)

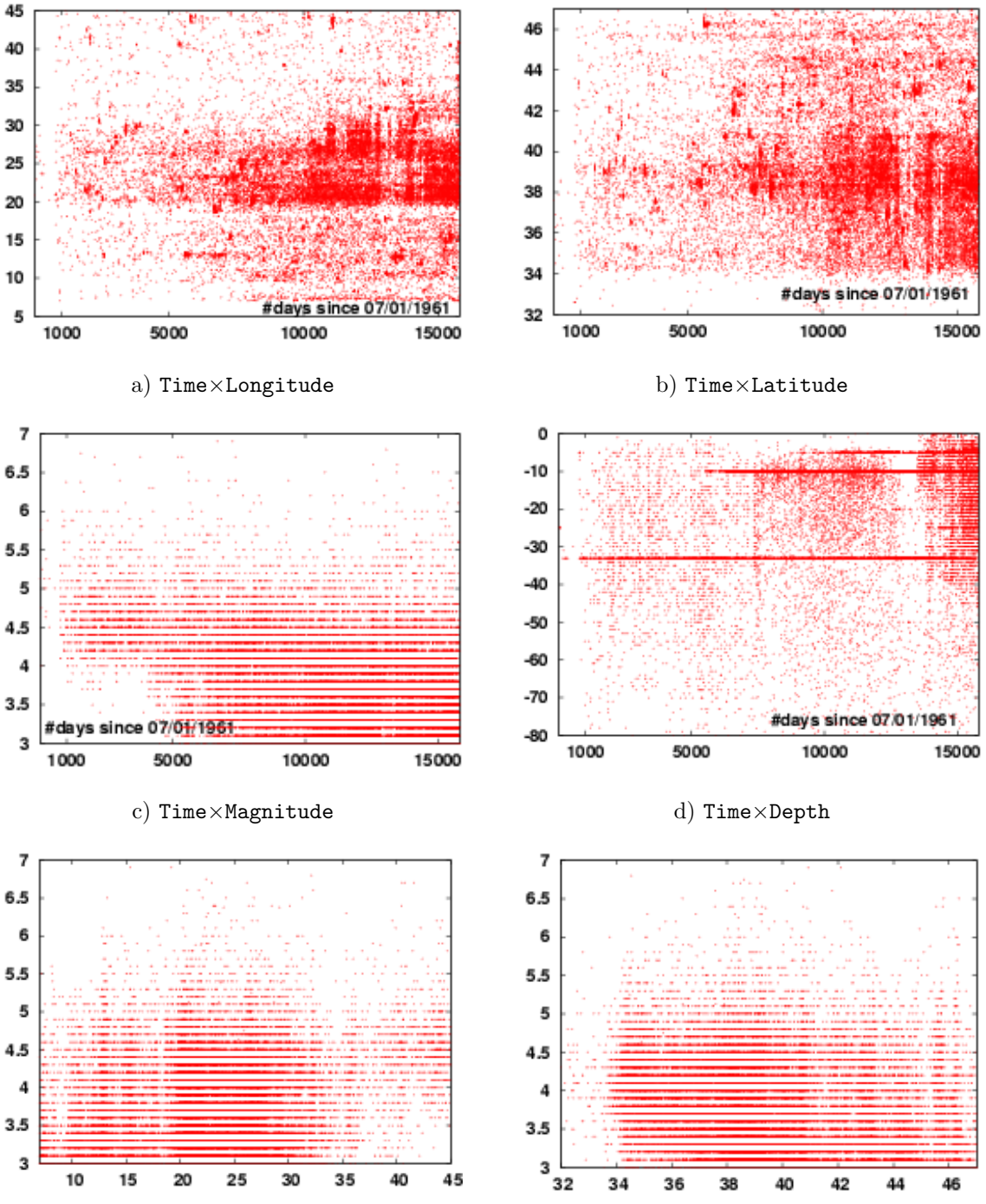


d)



e)

Figure 7.64: Uni-dimensional Frequency Histograms for the Earthquake Dataset

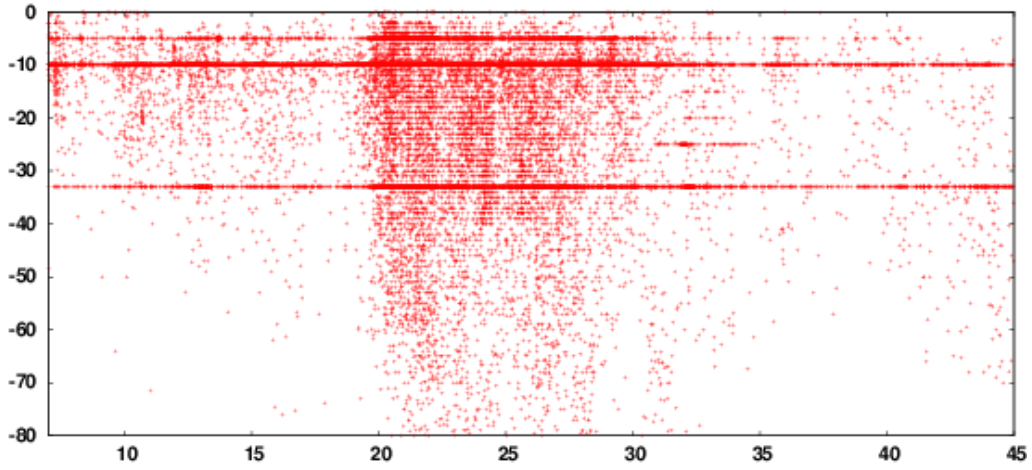


a) `Time`×`Longitude` b) `Time`×`Latitude`

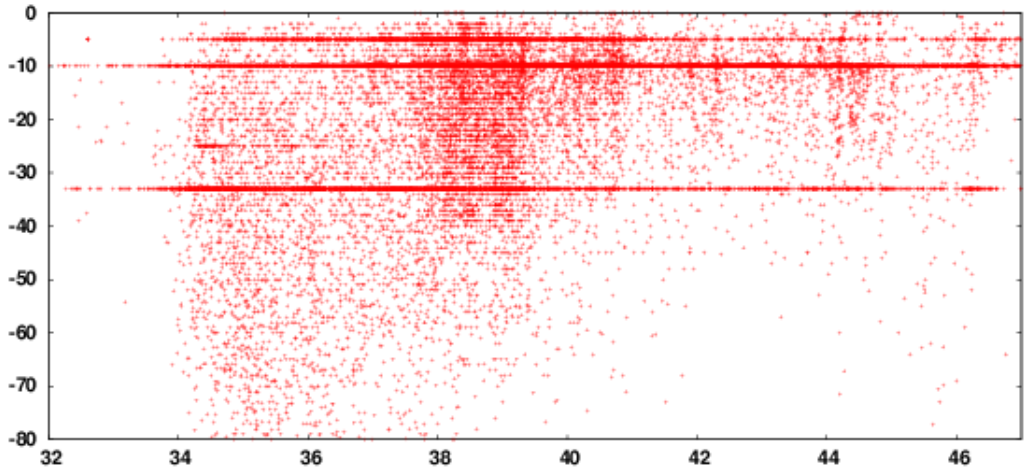
c) `Time`×`Magnitude` d) `Time`×`Depth`

e) `Longitude`×`Magnitude` f) `Latitude`×`Magnitude`

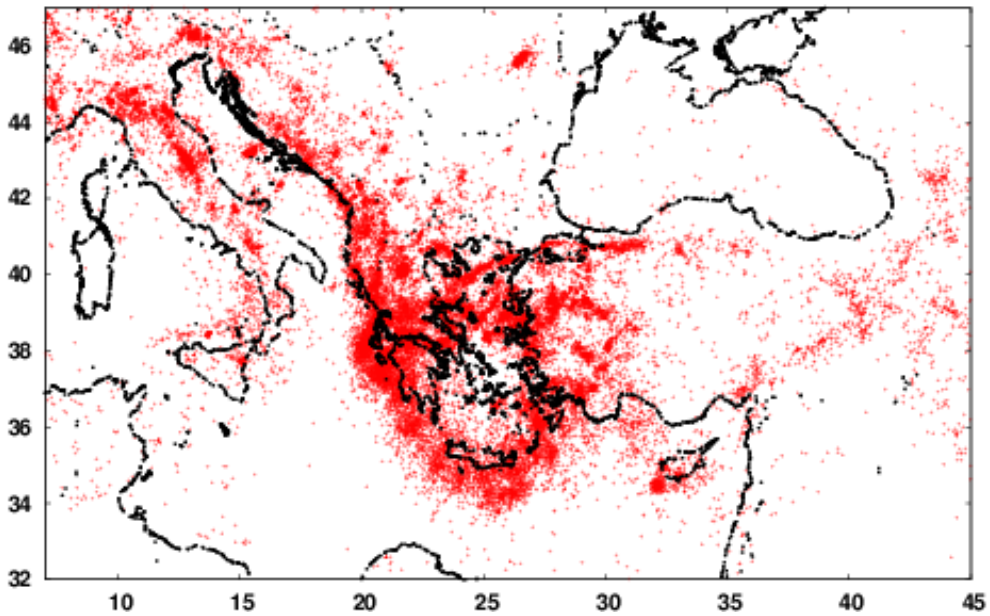
Figure 7.65: Pairwise Projections of Seismic Events (A)



a) `Longitude`×`Depth`



b) `Latitude`×`Depth`



c) `Longitude`×`Latitude`

Figure 7.66: Pairwise Projections of Seismic Events (B)

7.7 Evolutionary-based Clustering of *ANSS* Dataset

This section presents the knowledge discovered by $\mathcal{NOCEA}$ for the *ANSS* dataset, as well as the configuration settings for both EA- and clustering-related parameters. It also gives a classification of the rules to facilitate analysis of the results.

7.7.1 Analysing the *ANSS* Dataset

$\mathcal{NOCEA}$ discovered **237** highly-homogeneous hyper-rectangular rules forming **121** distinct clusters over the course of 300 generations. Some clusters have quite complex spatio-temporal structure probably due to the geological properties of the region and the dynamics of earthquakes. The rules of the fittest individual cover approximately **77%** of the total points (N_{total}=34593), while the remaining points (**23%**) were considered non-uniform background noise by $\mathcal{NOCEA}$.

Figure 7.67 depicts the performance of the best and worst individuals, as well as the mean fitness of the population members over the generations. Most rules were detected early in the evolutionary search, while in the rest of the time $\mathcal{NOCEA}$ was performing local fine-tuning. The complete set of cluster descriptors is given in section 7.7.4, while section 7.7.5 provides various statistics for the discovered clusters.

Not surprisingly, most of the discovered clusters, especially those with arbitrary shapes, are not distinguishable in most non fully-dimensional projections (see section 7.6.2), mainly due to the large degree of overlapping among the points of these clusters in lower dimensional projections. $\mathcal{NOCEA}$ is able to extract these complex structures as it always operates on the full-dimensional space, which guards against artifacts formed by the joint projection of multiple clusters in lower dimensional spaces.

7.7.2 Parameter Settings

Table 7.7 summarises the configuration settings for both the EA- and clustering-related parameters used by $\mathcal{NOCEA}$ to mine the *ANSS* dataset. All experiments

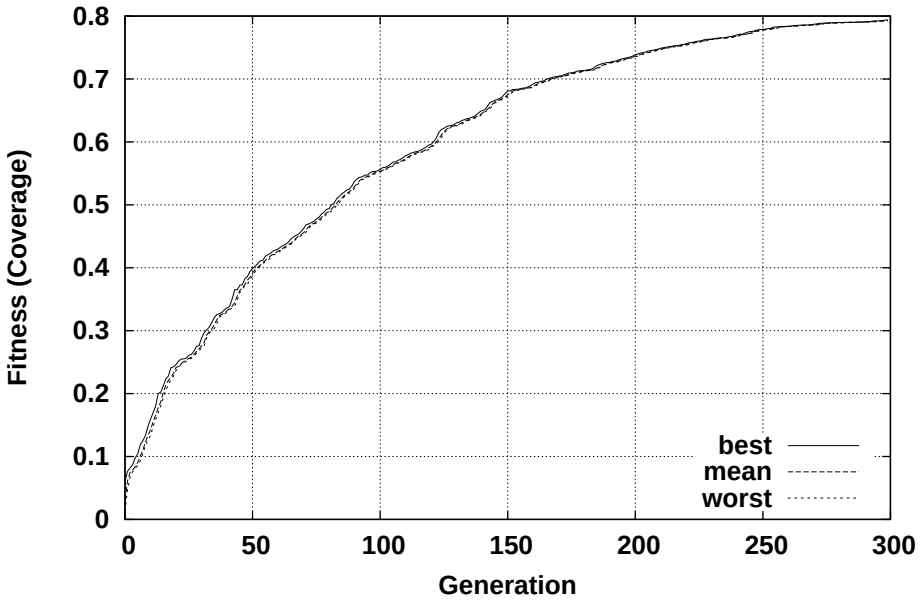


Figure 7.67: Fitness Diagram for the Best, Mean, and Worst Individual

reported in this Chapter have been performed on a Linux Fedora Core 2 workstation with an Intel(R) Xeon(TM) CPU 3.06GHz processor, 512 KB cache, 2GB of DRAM, and 9GB of IDE disk.

$\mathcal{NOCEA}$ uses typical EA parameter settings. In particular, the rates at which mutation and recombination are applied were set to 1.0 and 0.25, respectively. Each boundary of every rule undergoes either *grow* or *seed* mutation at random in every dimension with a small probability of 0.01. The size of tournament in the tournament selection procedure controlling how quickly the population is taken over by the dominant individuals, is set to 4. To ensure that the performance statistics of the population never degrades over generations, $\mathcal{NOCEA}$ adopts an elitist replacement strategy where the best individual of the current generation is directly copied into the next generation without undergoing any genetic operation. The population size was set to 50 individuals. $\mathcal{NOCEA}$ terminates after a prespecified number of 300 generations. Each population member is initialised with a single d-dimensional rule, which covers fully the domains in d-1 dimensions while extending to half of the domain in one, randomly chosen dimension. This is done to increase the possibility of generating non-sparse rules.

Parameter Name	**Value**
Population Size	50
Generations	300
Termination Condition	Maximum Number of Generations
Mutation Rate	1.0
Mutation Probability	0.01
Grow/Seed Mutation Ratio	0.5
Recombination Rate	0.25
Number of Offspring	2
Generalisation Rate	1.0
Generalisation Period	1
Generalisation Probability	0.05
Repairing Rate	1.0
Repairing Period	1
Selection Strategy	Tournament Selection (size=4)
Initialisation	Randomly Generated Singular-Rule Individuals
Replacement Strategy	Elitist (elite size =1)
Sparsity Threshold (T_s)	0.1% (0.025% for high-magnitude rules)
Homogeneity Threshold (T_h)	0.3
Subspace Clustering Threshold (T_r)	0.9
Generalisation Threshold (T_g)	$N(0.65, 0.1)$
Clustering Threshold (T_c)	0.2

Table 7.7: Parameter Settings for the *ANSS* Earthquake Dataset

As far as the clustering-related parameters of $\mathcal{NOCEA}$ are concerned, the following standard configuration was used in our experiments: T_s =0.1%, T_h =0.3, T_g =$N(0.65, 0.1)$, T_c =0.2 and T_r =0.9. In the case of the earthquake database and its extensions (see section 7.15.1) the sparse threshold T_s was deliberately set to a lower value 0.025% for high-magnitude rules, i.e. rules with magnitude exceeding 5.0 on the Richter scale, utilising a priori knowledge from the *Gutenberg-Richter* (*G-R*) power-law relation (section 7.3). In short, it is widely believed that populations of small-to-moderate earthquakes obey the *G-R* law defined as: $\log(n) = a - bM$, where n is the number of events with magnitude greater than M, while a and b are constants. Therefore, given the sparsity of the feature space in the high-magnitude neighbourhoods it is reasonable to introduce an adaptive sparsity threshold.

7.7.3 A Classification of the Clustering Rules

To facilitate analysis of the results we introduce the following classification for the discovered rules:

- `AS-Rule:` An *aftershock rule* represents a confined spatio-temporal region, which is characterised by a highly concentrated patch of low magnitude events, following a closely adjacent, in both time and space, strong earthquake.

- `R-rule:` An `R-rule` corresponds to a *regular* trend of seismic activity occurring within a specific space-magnitude interval. An `R-rule` can be viewed as a quasi-homogeneous cloud of narrow range magnitude events with wide spreading in time. Depending on the spatial window, the density of an `R-rule` varies considerably.

- `HP-Rule:` Similarly to an `AS-Rule`, a *historically precursory* rule `HP-Rule`, is a highly compact patch of low-magnitude events preceding a closely located strong earthquake.

- `P-rule:` Often, prior to a strong earthquake, the seismic activity in a medium magnitude range intensifies and becomes more clustered in space and time [63]. Although the generation of an earthquake is not always localised around its source, intense seismic activity that has *not* been associated with a strong earthquake may be potentially a *precursory* signal for future strong events, especially in regions where such patterns of behaviour have been historically observed.

- `U-rule:` A rule with *unknown* type represents a homogeneous cloud of events not fitting any of the the previous types.

7.7.4 *ANSS* Earthquake Clustering Rules

Tables 7.8-7.11 contain the complete set of cluster descriptors for the *ANSS* earthquake dataset discovered by $\mathcal{NOCEA}$. For illustrative purposes, clusters are separated by horizontal lines and are sorted on decreasing order of point coverage. Each rule is accompanied by ten fields, namely, `Rule`, `Cluster`, `Coverage( number of points - %)`, `Density`, `Time`, `Longitude`, `Latitude`, `Depth`, and `Magnitude`. Empty fields, appearing only for the `Time`-gene (see section 7.11 for an explanation), indicate irrelevant dimensions.

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnit.
$\mathcal{R}_2$	$\mathcal{C}_2$	993-2.87	7.4×10^{-3}	30/06/88-26/05/96	19.2-30.7	39.3-41.0	9-10	3.0-3.3
$\mathcal{R}_{11}$	$\mathcal{C}_2$	358-1.03	1.1×10^{-2}	30/06/88-26/05/96	20.1-21.8	37.3-39.3	9-10	3.0-3.4
$\mathcal{R}_{13}$	$\mathcal{C}_2$	233-0.67	1×10^{-2}	29/11/79-05/02/86	21.6-23.6	38.0-39.6	9-10	3.0-3.4
$\mathcal{R}_{18}$	$\mathcal{C}_2$	404-1.16	6.3×10^{-3}	17/11/78-27/09/96	19.9-24.0	37.8-39.3	9-10	3.4-3.6
$\mathcal{R}_{30}$	$\mathcal{C}_2$	228-0.65	7.2×10^{-3}	09/01/95-26/05/96	26.3-30.7	36.3-39.3	9-10	3.0-3.6
$\mathcal{R}_{50}$	$\mathcal{C}_2$	133-0.38	9.6×10^{-3}	08/03/89-07/12/91	26.3-28.0	37.6-39.3	9-10	3.0-3.6
$\mathcal{R}_{67}$	$\mathcal{C}_2$	115-0.33	6.8×10^{-3}	06/05/83-27/09/96	20.0-20.8	37.9-38.8	9-10	3.6-4.2
$\mathcal{R}_{85}$	$\mathcal{C}_2$	130-0.37	7.3×10^{-3}	15/08/92-09/01/95	29.2-30.9	36.8-39.3	9-10	3.0-3.6
$\mathcal{R}_{124}$	$\mathcal{C}_2$	51-0.14	8.9×10^{-3}	14/11/89-15/05/95	22.7-24.5	41.0-43.0	9-10	3.0-3.1
$\mathcal{R}_{154}$	$\mathcal{C}_2$	38-0.10	9×10^{-3}	30/06/88-27/09/96	19.8-21.4	41.0-42.1	9-10	3.2-3.3
$\mathcal{R}_{161}$	$\mathcal{C}_2$	41-0.11	1.1×10^{-2}	03/11/88-29/12/93	20.5-21.8	36.9-37.3	9-10	3.1-3.6
$\mathcal{R}_{191}$	$\mathcal{C}_2$	86-0.24	6.6×10^{-3}	23/07/90-26/05/96	21.8-24.2	38.5-39.3	9-10	3.0-3.4
$\mathcal{R}_{199}$	$\mathcal{C}_2$	49-0.14	1.4×10^{-2}	30/06/88-20/03/90	22.3-23.9	38.1-39.2	9-10	3.0-3.4
$\mathcal{R}_{215}$	$\mathcal{C}_2$	37-0.10	6.7×10^{-3}	23/07/90-07/12/91	28.0-30.7	37.6-39.3	9-10	3.0-3.3
$\mathcal{R}_{22}$	$\mathcal{C}_{17}$	565-1.63	9.4×10^{-4}	26/07/79-01/02/97	12.4-19.8	41.5-46.8	9-10	3.0-3.3
$\mathcal{R}_{26}$	$\mathcal{C}_{17}$	237-0.68	6.7×10^{-4}	13/03/78-01/02/97	9.40-14.0	42.3-45.1	9-10	3.3-3.8
$\mathcal{R}_{39}$	$\mathcal{C}_{17}$	151-0.43	4.4×10^{-4}	16/10/75-19/04/04	15.7-21.6	43.1-43.6	9-10	3.3-4.7
$\mathcal{R}_{44}$	$\mathcal{C}_{17}$	215-0.62	6.5×10^{-4}	17/06/98-22/11/01	12.4-23.3	36.9-47.0	9-10	3.0-3.3
$\mathcal{R}_{80}$	$\mathcal{C}_{17}$	103-0.29	6.5×10^{-4}	11/02/98-20/07/01	18.7-32.0	39.0-41.0	3-9	3.2-3.3
$\mathcal{R}_{125}$	$\mathcal{C}_{17}$	168-0.48	4.8×10^{-4}	30/06/88-05/03/00	16.9-22.1	40.9-43.1	9-10	3.3-4.2
$\mathcal{R}_{127}$	$\mathcal{C}_{17}$	83-0.23	6.1×10^{-4}	26/07/79-19/04/04	8.90-11.9	44.9-47.0	9-10	3.0-3.3
$\mathcal{R}_{157}$	$\mathcal{C}_{17}$	43-0.12	7.6×10^{-4}	30/06/88-26/05/96	22.1-24.7	41.0-42.9	9-10	3.1-3.6
$\mathcal{R}_{174}$	$\mathcal{C}_{17}$	47-0.13	4.7×10^{-4}	13/03/78-05/06/97	10.4-14.0	45.6-46.6	9-10	3.3-3.8
$\mathcal{R}_{178}$	$\mathcal{C}_{17}$	44-0.12	5.2×10^{-4}	17/06/98-05/03/00	9.90-14.0	42.4-47.0	9-10	3.3-4.2
$\mathcal{R}_{195}$	$\mathcal{C}_{17}$	36-0.10	6.9×10^{-4}	26/07/79-23/10/87	17.4-19.8	41.8-43.1	9-10	3.3-4.0
$\mathcal{R}_{236}$	$\mathcal{C}_{17}$	39-0.11	1×10^{-3}	22/11/01-19/04/04	13.1-18.4	43.5-47.0	9-10	3.0-3.3
$\mathcal{R}_{14}$	$\mathcal{C}_{12}$	642-1.85	1.5×10^{-4}		19.3-31.0	33.8-39.9	32-33	4.2-4.7
$\mathcal{R}_{21}$	$\mathcal{C}_{12}$	540-1.56	1.1×10^{-4}	29/03/02-19/04/04	20.3-28.0	34.3-37.7	5-33	3.0-4.1
$\mathcal{R}_{60}$	$\mathcal{C}_{12}$	83-0.23	1.2×10^{-4}	06/05/83-19/04/04	21.8-28.3	33.6-36.5	9-10	4.1-4.7
$\mathcal{R}_{62}$	$\mathcal{C}_{12}$	79-0.22	7.6×10^{-5}	29/03/02-19/04/04	19.2-28.2	37.7-39.1	10-33	3.6-4.2
$\mathcal{R}_{64}$	$\mathcal{C}_{12}$	80-0.23	7.4×10^{-5}	09/10/97-29/03/02	19.7-32.0	36.0-38.4	5-9	3.0-3.7
$\mathcal{R}_{75}$	$\mathcal{C}_{12}$	137-0.39	1.7×10^{-4}	30/06/88-17/06/98	22.7-33.1	34.0-39.5	32-33	3.0-3.5
$\mathcal{R}_{192}$	$\mathcal{C}_{12}$	36-0.10	1.7×10^{-4}	25/08/93-16/12/03	29.3-31.3	34.1-39.2	32-33	3.5-4.2
$\mathcal{R}_{210}$	$\mathcal{C}_{12}$	36-0.10	1.8×10^{-4}	05/09/71-06/08/80	22.8-30.2	37.9-39.6	32-33	3.6-4.2
$\mathcal{R}_{217}$	$\mathcal{C}_{12}$	37-0.10	1.8×10^{-4}	17/06/98-31/07/02	31.3-35.3	33.9-37.4	32-33	3.5-4.7
$\mathcal{R}_{15}$	$\mathcal{C}_{13}$	516-1.49	1×10^{-5}		19.2-30.4	37.5-39.7	16-32	3.5-4.7
$\mathcal{R}_{34}$	$\mathcal{C}_{13}$	208-0.60	8.6×10^{-6}	12/01/84-17/09/95	8.50-21.7	40.9-45.2	10-24	3.3-4.2
$\mathcal{R}_{41}$	$\mathcal{C}_{13}$	190-0.54	4.9×10^{-6}		17.6-31.6	39.9-40.9	16-36	3.6-4.7
$\mathcal{R}_{96}$	$\mathcal{C}_{13}$	59-0.17	1×10^{-5}	08/10/63-29/11/79	18.9-31.6	37.3-39.6	10-16	4.0-4.7
$\mathcal{R}_{102}$	$\mathcal{C}_{13}$	69-0.19	7.9×10^{-6}		19.9-21.0	37.2-40.0	10-41	4.7-5.5
$\mathcal{R}_{129}$	$\mathcal{C}_{13}$	53-0.15	5.9×10^{-6}	07/12/91-26/05/96	20.2-30.9	33.8-37.5	10-32	3.9-4.7
$\mathcal{R}_{142}$	$\mathcal{C}_{13}$	45-0.13	4.3×10^{-6}	05/09/71-11/02/98	18.4-27.7	39.5-40.9	16-37	3.0-3.5
$\mathcal{R}_{166}$	$\mathcal{C}_{13}$	36-0.10	7×10^{-6}	07/10/74-04/03/77	18.1-31.4	37.2-40.9	1-16	3.0-4.0
$\mathcal{R}_{184}$	$\mathcal{C}_{13}$	45-0.13	7.4×10^{-6}	12/01/84-19/04/04	7.00-16.6	45.2-46.7	10-16	3.0-4.2
$\mathcal{R}_{226}$	$\mathcal{C}_{13}$	10-0.02	4.3×10^{-6}		20.4-21.0	36.3-40.0	0-32	5.5-5.7

Table 7.8: Clustering Rules for the *ANSS* Earthquake Dataset (A)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnit.
$\mathcal{R}_4$	$\mathcal{C}_4$	567-1.63	1.7×10^{-3}	15/07/78-08/04/03	20.8-28.0	37.9-39.5	9-10	3.7-4.1
$\mathcal{R}_{35}$	$\mathcal{C}_4$	199-0.57	3.2×10^{-3}	09/12/80-17/09/95	19.7-21.6	37.2-37.9	9-10	3.6-4.7
$\mathcal{R}_{63}$	$\mathcal{C}_4$	87-0.25	2.9×10^{-3}	26/02/88-26/05/96	26.3-28.1	34.8-35.8	9-10	3.4-4.1
$\mathcal{R}_{131}$	$\mathcal{C}_4$	45-0.13	2.2×10^{-3}	26/02/88-22/04/93	26.3-28.0	36.3-37.9	9-10	3.6-4.1
$\mathcal{R}_{149}$	$\mathcal{C}_4$	37-0.10	2.4×10^{-3}	29/11/79-09/07/00	20.1-20.8	38.8-39.7	9-10	3.6-4.0
$\mathcal{R}_{173}$	$\mathcal{C}_4$	61-0.17	2.4×10^{-3}	26/02/88-09/01/95	27.3-28.8	35.8-37.5	9-10	3.1-3.6
$\mathcal{R}_{176}$	$\mathcal{C}_4$	49-0.14	1.6×10^{-3}	23/03/79-27/09/96	20.2-20.6	37.9-39.8	9-10	4.3-5.1
$\mathcal{R}_{12}$	$\mathcal{C}_{11}$	372-1.07	7.9×10^{-4}	05/02/86-29/03/02	21.8-28.3	33.7-35.9	32-33	3.5-4.2
$\mathcal{R}_{20}$	$\mathcal{C}_{11}$	401-1.15	6.4×10^{-4}	01/06/74-29/03/02	20.0-22.7	35.9-39.5	32-33	3.2-4.0
$\mathcal{R}_{61}$	$\mathcal{C}_{11}$	82-0.23	4.3×10^{-4}	10/01/72-05/02/86	21.1-28.4	33.8-35.9	32-33	3.9-4.2
$\mathcal{R}_{133}$	$\mathcal{C}_{11}$	86-0.24	4.2×10^{-4}	16/01/73-16/03/01	22.7-29.3	35.9-39.7	32-33	3.5-3.6
$\mathcal{R}_{189}$	$\mathcal{C}_{11}$	38-0.10	1.1×10^{-3}	29/08/82-26/05/96	26.8-28.1	35.9-37.0	32-33	3.6-4.2
$\mathcal{R}_{29}$	$\mathcal{C}_{22}$	229-0.66	1.2×10^{-3}	04/03/77-21/10/98	19.7-25.9	39.7-40.9	9-10	3.6-4.0
$\mathcal{R}_{31}$	$\mathcal{C}_{22}$	287-0.82	6×10^{-4}	03/11/88-19/04/04	28.0-31.7	35.8-41.5	9-10	3.6-4.1
$\mathcal{R}_{49}$	$\mathcal{C}_{22}$	119-0.34	1.1×10^{-3}	09/12/80-30/06/88	19.0-25.7	39.6-41.5	9-10	3.0-3.4
$\mathcal{R}_{55}$	$\mathcal{C}_{22}$	73-0.21	9.6×10^{-4}	07/10/74-19/04/04	22.1-28.0	39.5-41.0	9-10	4.0-4.1
$\mathcal{R}_{104}$	$\mathcal{C}_{22}$	72-0.20	1.2×10^{-3}	26/07/79-14/02/87	18.0-21.7	39.5-43.1	9-10	4.0-4.2
$\mathcal{R}_{147}$	$\mathcal{C}_{22}$	38-0.10	7.6×10^{-4}	13/03/78-29/11/79	22.3-31.2	38.4-41.2	9-10	3.0-3.4
$\mathcal{R}_{156}$	$\mathcal{C}_{22}$	103-0.29	1.4×10^{-3}	04/03/77-23/10/87	19.5-24.1	39.3-41.8	9-10	3.4-3.6
$\mathcal{R}_{10}$	$\mathcal{C}_{10}$	447-1.29	9.3×10^{-6}		23.6-28.4	34.1-35.7	33-79	3.5-4.7
$\mathcal{R}_{25}$	$\mathcal{C}_{10}$	228-0.65	1.4×10^{-5}	14/06/64-01/02/97	19.8-23.6	34.8-39.8	38-60	4.3-4.7
$\mathcal{R}_{54}$	$\mathcal{C}_{10}$	127-0.36	7.9×10^{-6}	09/05/72-19/04/04	26.4-31.8	35.7-36.4	33-79	3.7-4.7
$\mathcal{R}_{143}$	$\mathcal{C}_{10}$	51-0.14	1.2×10^{-5}	12/09/72-19/04/04	22.3-23.6	34.8-36.1	33-79	3.6-4.2
$\mathcal{R}_{43}$	$\mathcal{C}_{30}$	400-1.15	2.4×10^{-6}		21.0-31.2	33.8-41.1	4-39	4.7-5.2
$\mathcal{R}_{59}$	$\mathcal{C}_{30}$	79-0.22	3.1×10^{-6}	04/06/63-06/05/83	21.9-29.0	33.8-36.5	3-32	3.9-4.7
$\mathcal{R}_{99}$	$\mathcal{C}_{30}$	75-0.21	3.2×10^{-6}		21.4-41.7	41.1-46.6	32-33	3.0-4.9
$\mathcal{R}_{139}$	$\mathcal{C}_{30}$	105-0.30	4.5×10^{-6}		18.3-31.9	36.5-41.2	3-9	4.2-4.7
$\mathcal{R}_{169}$	$\mathcal{C}_{30}$	47-0.13	5.5×10^{-6}	15/02/63-06/08/80	39.6-45.1	37.9-39.9	33-52	4.4-5.2
$\mathcal{R}_{203}$	$\mathcal{C}_{30}$	49-0.14	6.7×10^{-6}		31.2-45.1	34-41.1	32-33	4.7-5.2
$\mathcal{R}_{205}$	$\mathcal{C}_{30}$	41-0.11	4.5×10^{-6}		24.9-30.9	36.4-37.4	33-43	3.5-4.7
$\mathcal{R}_6$	$\mathcal{C}_6$	787-2.27	1.8×10^{-2}	17/06/98-19/04/04	20.2-22.3	37.0-39.0	4-5	3.0-3.6
$\mathcal{R}_{36}$	$\mathcal{C}_{25}$	263-0.76	1.6×10^{-3}	16/03/01-16/12/03	23.3-24.7	38.4-39.0	0-41	3.0-3.6
$\mathcal{R}_{51}$	$\mathcal{C}_{25}$	115-0.33	1.5×10^{-3}	27/09/96-09/07/00	24.7-32.3	36.5-41.2	9-10	3.4-3.6
$\mathcal{R}_{103}$	$\mathcal{C}_{25}$	95-0.27	1.2×10^{-3}	11/02/98-16/03/01	22.3-30.7	37.2-39.0	4-5	3.0-3.6
$\mathcal{R}_{138}$	$\mathcal{C}_{25}$	44-0.12	1.8×10^{-3}	09/07/00-22/11/01	24.7-28.4	38.1-40.8	9-10	3.0-3.6
$\mathcal{R}_{180}$	$\mathcal{C}_{25}$	43-0.12	2.9×10^{-3}	09/10/97-01/11/99	27.0-30.1	39.0-41.0	3-7	3.0-3.1
$\mathcal{R}_{16}$	$\mathcal{C}_{14}$	461-1.33	6.2×10^{-5}	09/09/83-19/04/04	32.3-45.1	35.8-43.3	9-10	3.4-4.7
$\mathcal{R}_{155}$	$\mathcal{C}_{14}$	40-0.11	1×10^{-4}	20/03/90-09/01/95	28.3-37.5	33.2-35.8	9-10	3.0-4.2
$\mathcal{R}_{206}$	$\mathcal{C}_{14}$	37-0.10	9.6×10^{-5}	09/01/95-19/04/04	33.5-39.9	36.1-41.7	9-10	3.0-3.4
$\mathcal{R}_{17}$	$\mathcal{C}_{15}$	386-1.11	1.8×10^{-4}	08/07/77-19/04/04	22.4-32.3	36.5-41.2	9-10	4.1-4.7
$\mathcal{R}_{87}$	$\mathcal{C}_{15}$	68-0.19	2.9×10^{-4}	27/09/96-29/03/02	19.9-28.0	33.4-37.9	9-10	3.7-4.1
$\mathcal{R}_{128}$	$\mathcal{C}_{15}$	38-0.10	2×10^{-4}	09/07/00-29/03/02	23.4-27.6	34.7-36.1	10-32	3.5-3.8
$\mathcal{R}_{201}$	$\mathcal{C}_{15}$	44-0.12	1.9×10^{-4}	06/05/83-26/02/88	18.1-29.5	34.8-37.2	9-10	3.5-4.1
$\mathcal{R}_{32}$	$\mathcal{C}_{23}$	407-1.17	2.6×10^{-4}	22/11/01-19/04/04	19.2-26.8	39.0-40.6	0-31	3.0-3.6
$\mathcal{R}_{66}$	$\mathcal{C}_{23}$	75-0.21	1.9×10^{-4}	29/03/02-19/04/04	25.2-27.6	37.8-39.0	0-38	3.0-3.6
$\mathcal{R}_{163}$	$\mathcal{C}_{23}$	38-0.10	3.5×10^{-4}	11/02/98-19/04/04	20.1-22.7	37.7-39.8	31-42	3.0-3.1
$\mathcal{R}_3$	$\mathcal{C}_3$	302-0.87	3.6×10^{-3}	23/10/87-27/09/96	19.2-29.3	39.3-40.9	9-10	3.4-3.6
$\mathcal{R}_{65}$	$\mathcal{C}_3$	98-0.28	4.7×10^{-3}	20/06/87-15/08/92	24.2-26.3	38.2-39.3	9-10	3.0-3.6
$\mathcal{R}_{86}$	$\mathcal{C}_3$	62-0.17	3.6×10^{-3}	05/02/86-30/06/88	20.8-24.2	37.8-39.6	9-10	3.0-3.4
$\mathcal{R}_{141}$	$\mathcal{C}_3$	37-0.10	3.7×10^{-3}	09/12/80-05/02/86	23.6-24.8	38.2-39.6	9-10	3.0-3.4
$\mathcal{R}_9$	$\mathcal{C}_9$	396-1.14	3.7×10^{-5}	29/11/79-20/01/96	19.6-28.3	37.8-40.5	10-16	3.0-4.6
$\mathcal{R}_{148}$	$\mathcal{C}_9$	50-0.14	1.8×10^{-5}	19/09/84-17/09/95	20.2-26.8	37.9-39.5	16-32	3.0-3.5
$\mathcal{R}_{197}$	$\mathcal{C}_9$	44-0.12	2.2×10^{-5}	05/02/86-25/08/93	11.5-23.6	40.5-44.7	10-16	3.0-3.3
$\mathcal{R}_{37}$	$\mathcal{C}_{26}$	230-0.66	2.4×10^{-5}	17/06/98-16/03/01	22.7-34.1	33.5-41.4	10-37	3.0-3.5
$\mathcal{R}_{78}$	$\mathcal{C}_{26}$	90-0.26	1.3×10^{-5}	23/03/79-08/04/03	8.70-21.4	40.9-45.6	32-33	3.0-4.7
$\mathcal{R}_{193}$	$\mathcal{C}_{26}$	46-0.13	2.1×10^{-5}	17/06/98-22/11/01	19.4-22.7	35.4-42.9	10-32	3.1-3.5
$\mathcal{R}_{213}$	$\mathcal{C}_{26}$	50-0.14	2.1×10^{-5}	22/11/01-19/04/04	7.00-24.1	40.6-45.2	10-21	3.0-3.4
$\mathcal{R}_5$	$\mathcal{C}_5$	313-0.90	6.3×10^{-3}	25/04/82-08/04/03	9.40-12.4	44.0-44.9	9-10	3.0-3.3
$\mathcal{R}_{105}$	$\mathcal{C}_5$	54-0.15	5.2×10^{-3}	25/04/82-19/04/04	11.8-12.4	43.1-44.0	9-10	3.0-3.3
$\mathcal{R}_1$	$\mathcal{C}_1$	361-1.04	1.3×10^{-3}	29/03/02-19/04/04	20.1-22.4	37.7-39.0	5-31	3.0-3.6
$\mathcal{R}_{19}$	$\mathcal{C}_{16}$	316-0.91	1.5×10^{-3}	05/03/00-19/04/04	21.0-27.9	34.5-37.0	4-5	3.0-4.0
$\mathcal{R}_{159}$	$\mathcal{C}_{16}$	45-0.13	8.5×10^{-4}	21/10/98-05/03/00	20.5-28.4	34.6-37.0	4-5	3.2-3.9
$\mathcal{R}_{134}$	$\mathcal{C}_{79}$	294-0.84	2.2×10^{-1}	01/02/97-17/06/98	12.5-13.1	42.7-43.2	9-10	3.0-4.1
$\mathcal{R}_{136}$	$\mathcal{C}_{79}$	55-0.15	1.1×10^{-1}	01/02/97-17/06/98	12.6-13.1	42.8-43.2	9-10	4.1-4.7
$\mathcal{R}_0$	$\mathcal{C}_0$	348-1.00	3.9×10^{-2}	15/08/92-09/01/95	26.1-28.2	37.8-39.3	9-10	3.0-3.4

Table 7.9: Clustering Rules for the *ANSS* Earthquake Dataset (B)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnit.
$\mathcal{R}_{27}$	$\mathcal{C}_{20}$	229-0.66	2.7×10^{-4}	13/03/78-19/04/04	14.0-18.2	43.6-46.5	9-10	3.3-4.2
$\mathcal{R}_{53}$	$\mathcal{C}_{20}$	104-0.30	1.9×10^{-4}	12/06/75-01/02/97	9.90-14.0	41.6-47.0	9-10	3.8-4.2
$\mathcal{R}_{33}$	$\mathcal{C}_{24}$	258-0.74	1.8×10^{-5}	05/09/71-19/04/04	20.7-22.3	35.8-38.9	33-71	3.4-4.2
$\mathcal{R}_{113}$	$\mathcal{C}_{24}$	44-0.12	2×10^{-5}	16/03/01-19/04/04	20.3-27.7	35.1-39.6	42-60	3.0-3.4
$\mathcal{R}_{42}$	$\mathcal{C}_{29}$	169-0.48	3.4×10^{-3}	14/11/89-06/09/94	19.7-22.5	37.5-41.7	4-5	3.0-3.3
$\mathcal{R}_{111}$	$\mathcal{C}_{29}$	86-0.24	2.5×10^{-3}	25/08/93-20/01/96	19.7-22.2	37.3-41.2	4-5	3.3-3.8
$\mathcal{R}_{188}$	$\mathcal{C}_{29}$	39-0.11	6.3×10^{-3}	06/09/94-20/01/96	19.5-21.4	38.9-41.6	4-5	3.0-3.3
$\mathcal{R}_{46}$	$\mathcal{C}_{32}$	181-0.52	1.1×10^{-5}	06/05/83-19/04/04	7.00-22.8	41.2-47.0	3-9	3.3-3.8
$\mathcal{R}_{93}$	$\mathcal{C}_{32}$	55-0.15	4.2×10^{-6}	26/07/79-19/04/04	10.3-22.4	41.3-44.1	3-9	3.8-4.7
$\mathcal{R}_{200}$	$\mathcal{C}_{32}$	48-0.13	1.1×10^{-5}	23/02/99-16/03/01	7.0-35.6	37.1-46.3	0-3	3.1-3.8
$\mathcal{R}_{8}$	$\mathcal{C}_{8}$	270-0.78	3.9×10^{-3}	01/02/97-01/11/99	25.6-30.7	36.8-41.0	9-10	3.0-3.4
$\mathcal{R}_{24}$	$\mathcal{C}_{19}$	267-0.77	5.4×10^{-4}	26/02/88-27/09/96	21.8-26.3	33.8-37.8	9-10	3.0-4.1
$\mathcal{R}_{7}$	$\mathcal{C}_{7}$	254-0.73	1.3×10^{-2}	09/12/80-20/06/87	24.8-25.9	38.2-39.5	9-10	3.0-3.7
$\mathcal{R}_{77}$	$\mathcal{C}_{47}$	81-0.23	5.8×10^{-4}	09/01/95-27/09/96	25.1-30.4	34.7-40.0	4-5	3.0-4.0
$\mathcal{R}_{82}$	$\mathcal{C}_{47}$	71-0.20	3.6×10^{-4}	25/08/93-19/04/04	18.4-29.1	34.2-40.1	4-5	4.0-4.1
$\mathcal{R}_{151}$	$\mathcal{C}_{47}$	50-0.14	5.2×10^{-4}	17/06/98-04/12/02	21.2-29.2	37.2-39.5	4-5	3.6-4.0
$\mathcal{R}_{202}$	$\mathcal{C}_{47}$	47-0.13	4.6×10^{-4}	29/03/02-19/04/04	22.4-23.3	37.8-39.0	0-25	3.0-3.6
$\mathcal{R}_{83}$	$\mathcal{C}_{50}$	155-0.44	1.4×10^{-4}	15/07/78-31/07/02	14.0-16.9	36.9-43.1	9-10	3.3-4.2
$\mathcal{R}_{84}$	$\mathcal{C}_{50}$	84-0.24	7.5×10^{-5}	26/10/76-01/02/97	9.50-15.7	40.9-47.0	9-10	4.2-4.7
$\mathcal{R}_{68}$	$\mathcal{C}_{39}$	108-0.31	9.1×10^{-7}	14/06/64-12/01/84	7.00-23.8	40.9-47.0	10-27	3.5-4.7
$\mathcal{R}_{100}$	$\mathcal{C}_{39}$	71-0.20	1.6×10^{-6}	12/01/84-20/01/96	7.00-23.1	41.1-45.8	24-32	3.0-5.1
$\mathcal{R}_{144}$	$\mathcal{C}_{39}$	39-0.11	8.6×10^{-7}	29/06/65-01/11/99	10.7-19.9	35.2-47.1	10-24	4.8-5.1
$\mathcal{R}_{229}$	$\mathcal{C}_{39}$	19-0.05	7.1×10^{-7}		11.8-19.9	36.5-41.1	26-46	4.8-5.1
$\mathcal{R}_{40}$	$\mathcal{C}_{28}$	153-0.44	2.1×10^{-6}	25/09/73-19/04/04	21.5-29.0	34.3-37.7	79-108	3.5-4.6
$\mathcal{R}_{114}$	$\mathcal{C}_{28}$	42-0.12	3×10^{-6}		21.1-23.6	34.0-39.6	60-76	4.2-4.7
$\mathcal{R}_{212}$	$\mathcal{C}_{28}$	36-0.10	3.7×10^{-6}	16/12/69-19/04/04	22.3-23.6	36.1-38.4	33-79	3.5-4.2
$\mathcal{R}_{74}$	$\mathcal{C}_{45}$	158-0.45	3.5×10^{-7}		31.6-45.1	34.8-42.2	10-32	3.5-5.2
$\mathcal{R}_{218}$	$\mathcal{C}_{45}$	52-0.15	1.2×10^{-7}		34.2-45.1	35.9-43.5	4-37	5.2-6.2
$\mathcal{R}_{38}$	$\mathcal{C}_{27}$	197-0.56	2.6×10^{-1}	31/07/02-16/12/03	15.1-15.7	43.0-43.4	9-10	3.0-3.8
$\mathcal{R}_{45}$	$\mathcal{C}_{31}$	188-0.54	3×10^{-5}	17/08/81-09/01/95	19.5-29.5	37.9-41.2	5-9	3.0-4.2
$\mathcal{R}_{23}$	$\mathcal{C}_{18}$	185-0.53	9.8×10^{-5}		26.2-26.8	45.4-45.8	124-169	3.2-4.8
$\mathcal{R}_{52}$	$\mathcal{C}_{35}$	118-0.34	5.3×10^{-7}		21.0-32.9	34.1-41.7	2-41	5.2-5.7
$\mathcal{R}_{219}$	$\mathcal{C}_{35}$	21-0.06	2.7×10^{-7}		22.0-32.7	34.0-36.8	41-88	5.2-5.7
$\mathcal{R}_{228}$	$\mathcal{C}_{35}$	27-0.07	4.3×10^{-7}		10.2-21.0	37.0-46.5	4-9	4.8-5.5
$\mathcal{R}_{233}$	$\mathcal{C}_{35}$	11-0.03	4.2×10^{-7}		20.6-28.8	34.7-38.0	75-94	4.8-5.2
$\mathcal{R}_{47}$	$\mathcal{C}_{33}$	143-0.41	1.1×10^{-1}	23/03/79-26/07/79	18.5-19.5	41.7-42.4	9-10	3.0-4.9
$\mathcal{R}_{132}$	$\mathcal{C}_{78}$	50-0.14	6.9×10^{-5}	11/02/98-19/04/04	7.00-24.3	41.0-44.9	7-9	3.0-3.3
$\mathcal{R}_{137}$	$\mathcal{C}_{78}$	49-0.14	1.1×10^{-4}	30/06/88-20/07/01	10.8-19.1	42.1-47.0	4-5	3.0-3.3
$\mathcal{R}_{165}$	$\mathcal{C}_{78}$	35-0.10	1.7×10^{-4}	20/07/01-19/04/04	9.00-23.5	40.6-45.1	3-7	3.2-3.3
$\mathcal{R}_{106}$	$\mathcal{C}_{61}$	85-0.24	7.4×10^{-3}	15/08/92-09/01/95	26.6-29.2	37.7-39.8	4-5	3.0-3.3
$\mathcal{R}_{187}$	$\mathcal{C}_{61}$	36-0.10	5.5×10^{-3}	22/04/93-09/01/95	29.2-30.6	38.1-41.2	4-5	3.0-3.3
$\mathcal{R}_{69}$	$\mathcal{C}_{40}$	111-0.32	6.8×10^{-5}	20/02/65-23/03/79	9.20-21.4	40.9-45.0	32-33	3.9-4.7
$\mathcal{R}_{79}$	$\mathcal{C}_{48}$	107-0.30	4.1×10^{-2}	20/07/01-22/11/01	24.0-24.5	39.0-39.2	0-42	3.0-3.6
$\mathcal{R}_{119}$	$\mathcal{C}_{71}$	64-0.18	1.3×10^{-3}	25/04/82-29/03/02	7.50-8.90	43.2-45.2	9-10	3.0-3.3
$\mathcal{R}_{214}$	$\mathcal{C}_{71}$	36-0.10	6.6×10^{-4}	23/10/87-20/07/01	7.00-8.60	43.6-45.3	9-10	3.3-3.8
$\mathcal{R}_{48}$	$\mathcal{C}_{34}$	98-0.28	1.6×10^{-5}	14/02/87-07/12/91	22.5-30.0	33.8-37.5	12-32	3.9-4.7
$\mathcal{R}_{70}$	$\mathcal{C}_{41}$	97-0.28	4×10^{-1}	01/01/83-06/05/83	19.6-20.2	37.9-38.4	9-10	3.9-4.7
$\mathcal{R}_{179}$	$\mathcal{C}_{98}$	41-0.11	2.4×10^{-6}	08/07/77-19/04/04	15.9-17.3	37.1-40.3	10-39	3.0-4.7
$\mathcal{R}_{207}$	$\mathcal{C}_{98}$	45-0.13	5.1×10^{-6}		12.1-15.9	37.4-38.5	0-42	4.3-4.7
$\mathcal{R}_{152}$	$\mathcal{C}_{85}$	38-0.10	1.2×10^{-3}	06/08/80-04/12/02	10.0-11.8	43.1-44.0	9-10	3.0-3.3
$\mathcal{R}_{186}$	$\mathcal{C}_{85}$	44-0.12	1.5×10^{-3}	05/06/97-17/06/98	9.80-14.0	43.2-47.0	9-10	3.3-3.9
$\mathcal{R}_{88}$	$\mathcal{C}_{51}$	77-0.22	1.1	01/10/85-09/06/86	19.8-20.1	42.1-42.5	9-10	3.0-3.3
$\mathcal{R}_{71}$	$\mathcal{C}_{42}$	76-0.21	2.1×10^{-2}	06/09/94-26/05/96	20.1-22.5	37.9-38.9	4-5	3.0-3.3
$\mathcal{R}_{177}$	$\mathcal{C}_{97}$	45-0.13	1.6×10^{-7}		19.9-33.9	37.4-41.5	2-33	5.7-6.6
$\mathcal{R}_{222}$	$\mathcal{C}_{97}$	17-0.04	1.1×10^{-7}		12.5-21.4	41.5-47.0	4-33	5.7-6.2
$\mathcal{R}_{223}$	$\mathcal{C}_{97}$	11-0.03	1.9×10^{-7}		20.8-29.2	33.8-37.4	24-32	5.7-6.2
$\mathcal{R}_{91}$	$\mathcal{C}_{54}$	71-0.20	2.5×10^{-5}	26/07/79-19/04/04	8.60-22.3	41.1-46.8	9-10	4.7-5.2
$\mathcal{R}_{92}$	$\mathcal{C}_{55}$	71-0.20	5.9×10^{-1}	14/11/89-20/03/90	27.0-27.3	35.9-36.3	9-10	3.1-4.1
$\mathcal{R}_{122}$	$\mathcal{C}_{74}$	71-0.20	1.7×10^{-1}	30/06/88-03/05/94	21.8-22.3	38.0-38.5	9-10	3.0-3.1
$\mathcal{R}_{76}$	$\mathcal{C}_{46}$	70-0.20	2.7×10^{-4}	09/01/95-19/04/04	30.7-33.5	33.5-42.0	9-10	3.0-3.4
$\mathcal{R}_{58}$	$\mathcal{C}_{38}$	69-0.19	4.1×10^{-6}		14.6-15.7	38.5-40.0	258-326	3.5-4.8
$\mathcal{R}_{90}$	$\mathcal{C}_{53}$	69-0.19	2.9×10^{-4}	08/03/89-19/04/04	7.00-10.8	43.4-47.0	3-7	3.0-3.1
$\mathcal{R}_{95}$	$\mathcal{C}_{57}$	68-0.19	1.5×10^{-5}	17/05/84-19/04/04	7.00-15.9	32.8-36.9	9-10	3.0-5.2
$\mathcal{R}_{81}$	$\mathcal{C}_{49}$	67-0.19	1.8×10^{-4}	17/06/98-20/07/01	18.4-34.0	39.0-41.2	3-9	3.4-3.6

Table 7.10: Clustering Rules for the *ANSS* Earthquake Dataset (C)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnit.
$\mathcal{R}_{107}$	$\mathcal{C}_{62}$	63-0.18	1.2×10^{-6}	07/01/61-15/08/92	19.9-30.9	34.3-37.9	41-71	4.7-5.2
$\mathcal{R}_{108}$	$\mathcal{C}_{63}$	63-0.18	5.8×10^{-3}	31/07/02-19/04/04	20.2-21.3	37.5-38.9	2-4	3.0-3.7
$\mathcal{R}_{98}$	$\mathcal{C}_{59}$	62-0.17	1.6×10^{-1}	23/07/90-07/12/91	29.3-29.7	36.8-37.2	9-10	3.0-3.6
$\mathcal{R}_{101}$	$\mathcal{C}_{60}$	62-0.17	3.4×10^{-2}	26/02/88-06/09/94	20.4-21.6	37.4-37.8	9-10	3.4-3.6
$\mathcal{R}_{118}$	$\mathcal{C}_{70}$	62-0.17	1.8×10^{-2}	30/06/88-20/01/96	19.8-21.1	41.0-42.2	9-10	3.0-3.1
$\mathcal{R}_{89}$	$\mathcal{C}_{52}$	61-0.17	8.5×10^{-4}	22/11/01-19/04/04	10.8-16.4	40.3-44.9	3-7	3.0-3.1
$\mathcal{R}_{120}$	$\mathcal{C}_{72}$	61-0.17	2.3×10^{-5}	26/11/67-27/09/96	31.3-37.1	33.6-39.0	32-33	3.7-4.7
$\mathcal{R}_{112}$	$\mathcal{C}_{66}$	59-0.17	3×10^{-1}	27/09/96-01/02/97	32.0-32.5	34.2-34.7	32-33	3.9-4.7
$\mathcal{R}_{194}$	$\mathcal{C}_{104}$	57-0.16	1.9×10^{-3}	01/06/74-09/09/83	19.4-22.8	37.2-38.8	32-33	4.0-4.2
$\mathcal{R}_{135}$	$\mathcal{C}_{80}$	56-0.16	1.2×10^{-1}	03/11/88-01/02/97	21.8-22.3	38.1-38.5	9-10	3.2-3.3
$\mathcal{R}_{56}$	$\mathcal{C}_{36}$	54-0.15	2.2×10^{-1}	15/05/95-20/01/96	21.4-22.0	39.8-40.3	9-10	4.0-4.2
$\mathcal{R}_{94}$	$\mathcal{C}_{56}$	54-0.15	1.1	06/09/94-09/01/95	28.8-29.0	36.7-37.1	9-10	3.0-3.6
$\mathcal{R}_{97}$	$\mathcal{C}_{58}$	54-0.15	2.2	15/05/95-17/09/95	21.4-21.9	39.8-40.3	4-5	3.0-3.1
$\mathcal{R}_{115}$	$\mathcal{C}_{67}$	54-0.15	3×10^{-5}	07/01/61-07/12/91	19.4-23.7	37.3-39.6	33-38	4.3-4.7
$\mathcal{R}_{57}$	$\mathcal{C}_{37}$	53-0.15	1.3×10^{-1}	06/09/94-20/01/96	21.4-21.9	39.8-40.3	9-10	4.3-4.7
$\mathcal{R}_{146}$	$\mathcal{C}_{83}$	53-0.15	2.5×10^{-2}	16/03/01-22/11/01	25.5-25.8	38.3-38.7	25-40	3.2-3.8
$\mathcal{R}_{109}$	$\mathcal{C}_{64}$	52-0.15	3.4×10^{-2}	18/02/76-26/10/76	12.8-13.4	46.0-46.6	32-33	3.0-5.1
$\mathcal{R}_{126}$	$\mathcal{C}_{76}$	51-0.14	1.4×10^{-2}	17/06/98-19/04/04	20.2-21.2	37.0-39.2	4-5	3.7-3.8
$\mathcal{R}_{170}$	$\mathcal{C}_{93}$	51-0.14	1.5×10^{-4}	25/04/82-22/11/01	7.00-11.5	43.9-45.0	10-14	3.0-3.3
$\mathcal{R}_{150}$	$\mathcal{C}_{84}$	48-0.13	7.6×10^{-7}		21.5-26.6	35.1-38.4	108-137	3.4-4.6
$\mathcal{R}_{168}$	$\mathcal{C}_{92}$	48-0.13	2.2×10^{-3}	22/11/01-12/08/03	19.2-22.2	40.6-43.0	9-10	3.0-3.6
$\mathcal{R}_{209}$	$\mathcal{C}_{109}$	47-0.13	2.7×10^{-4}	16/10/75-19/04/04	19.5-21.0	39.8-42.6	9-10	4.2-4.7
$\mathcal{R}_{130}$	$\mathcal{C}_{77}$	46-0.13	5.7×10^{-5}	06/05/83-19/04/04	26.4-27.1	36.2-36.7	137-164	3.4-4.8
$\mathcal{R}_{140}$	$\mathcal{C}_{81}$	45-0.13	1.2×10^{-4}	08/03/89-09/01/95	22.2-30.8	34.7-38.4	4-5	3.3-4.0
$\mathcal{R}_{162}$	$\mathcal{C}_{89}$	45-0.13	1.9×10^{-6}	14/04/81-19/04/04	28.4-34.5	34.1-35.7	33-74	3.7-4.6
$\mathcal{R}_{164}$	$\mathcal{C}_{90}$	45-0.13	1.6×10^{-5}	06/11/77-19/04/04	31.2-45.1	36.9-42.0	9-10	4.7-5.2
$\mathcal{R}_{204}$	$\mathcal{C}_{107}$	36-0.10	1.4×10^{-6}	28/12/70-19/04/04	15.3-16.0	38.2-39.3	82-258	3.0-5.0
$\mathcal{R}_{227}$	$\mathcal{C}_{107}$	9-0.02	7.9×10^{-7}		14.6-15.8	38.2-40.2	233-295	5.0-5.7
$\mathcal{R}_{73}$	$\mathcal{C}_{44}$	44-0.12	4.4×10^{-5}	30/06/88-19/04/04	41.9-45.1	34.7-44.4	32-33	3.5-4.2
$\mathcal{R}_{121}$	$\mathcal{C}_{73}$	44-0.12	2.8	15/05/95-17/09/95	21.9-22.3	38.1-38.5	9-10	3.0-3.1
$\mathcal{R}_{196}$	$\mathcal{C}_{105}$	44-0.12	1.1×10^{-1}	05/06/97-21/10/98	20.5-21.3	37.0-37.6	32-33	4.0-4.2
$\mathcal{R}_{198}$	$\mathcal{C}_{106}$	44-0.12	2.5×10^{-3}	26/01/74-25/04/82	21.1-25.6	37.9-39.5	32-33	3.0-3.1
$\mathcal{R}_{123}$	$\mathcal{C}_{75}$	42-0.12	4.2	08/03/89-14/11/89	23.3-23.8	39.2-39.3	9-10	3.0-3.1
$\mathcal{R}_{167}$	$\mathcal{C}_{91}$	42-0.12	3.3×10^{-1}	05/03/00-09/07/00	11.7-12.1	44.1-44.5	9-10	3.3-4.1
$\mathcal{R}_{183}$	$\mathcal{C}_{101}$	42-0.12	6×10^{-5}	17/06/98-19/04/04	7.00-24.1	40.6-46.6	0-3	3.0-3.1
$\mathcal{R}_{117}$	$\mathcal{C}_{69}$	40-0.11	1	06/05/83-09/09/83	24.6-25.1	39.9-40.3	9-10	3.4-3.6
$\mathcal{R}_{145}$	$\mathcal{C}_{82}$	40-0.11	4×10^{-7}	06/07/62-26/11/90	11.0-21.0	40.0-47.0	11-33	5.1-5.7
$\mathcal{R}_{158}$	$\mathcal{C}_{87}$	40-0.11	3.8×10^{-5}	01/01/83-19/04/04	26.4-26.9	45.4-45.9	79-122	3.2-4.8
$\mathcal{R}_{116}$	$\mathcal{C}_{68}$	39-0.11	3.2×10^{-1}	26/05/96-27/09/96	27.0-27.5	35.9-36.3	32-33	3.6-4.2
$\mathcal{R}_{160}$	$\mathcal{C}_{88}$	39-0.11	1.1×10^{-3}	22/04/93-09/01/95	22.5-26.6	37.7-43.6	4-5	3.0-3.3
$\mathcal{R}_{175}$	$\mathcal{C}_{96}$	39-0.11	3.8×10^{-6}		19.3-24.9	38.9-39.8	33-60	3.7-4.3
$\mathcal{R}_{171}$	$\mathcal{C}_{94}$	38-0.10	1.6	31/07/02-04/12/02	13.5-13.9	38.3-38.5	4-5	3.0-3.3
$\mathcal{R}_{172}$	$\mathcal{C}_{95}$	38-0.10	1.2×10^{-3}	29/03/02-19/04/04	21.3-22.4	36.4-39.0	1-4	3.0-3.6
$\mathcal{R}_{185}$	$\mathcal{C}_{102}$	38-0.10	9.7×10^{-4}	12/07/89-25/08/93	19.9-21.2	37.0-41.2	4-5	3.3-3.9
$\mathcal{R}_{153}$	$\mathcal{C}_{86}$	37-0.10	9.2×10^{-7}	23/02/64-06/08/80	9.70-22.0	40.9-47.0	33-49	4.0-4.7
$\mathcal{R}_{181}$	$\mathcal{C}_{99}$	37-0.10	2.2×10^{-1}	09/12/80-14/04/81	22.8-23.5	37.9-38.3	32-33	3.6-4.2
$\mathcal{R}_{28}$	$\mathcal{C}_{21}$	36-0.10	1.9×10^{-4}	20/06/87-19/04/04	19.9-21.7	43.6-46.0	9-10	3.3-4.2
$\mathcal{R}_{182}$	$\mathcal{C}_{100}$	36-0.10	1.1×10^{-6}		18.5-27.3	39.8-40.5	37-64	3.0-4.7
$\mathcal{R}_{190}$	$\mathcal{C}_{103}$	36-0.10	1×10^{-7}	14/04/81-19/04/04	7.9-32.4	41.4-46.6	33-53	3.0-5.1
$\mathcal{R}_{208}$	$\mathcal{C}_{108}$	36-0.10	1×10^{-2}	17/06/98-22/11/01	19.6-23.6	39.8-40.7	4-5	3.0-3.1
$\mathcal{R}_{110}$	$\mathcal{C}_{65}$	35-0.10	3.3×10^{-2}	01/01/83-09/10/97	7.00-7.50	44.2-44.7	9-10	3.0-3.1
$\mathcal{R}_{211}$	$\mathcal{C}_{110}$	35-0.10	7.6×10^{-4}	27/09/96-19/04/04	20.0-21.4	36.8-39.3	9-10	4.1-4.7
$\mathcal{R}_{216}$	$\mathcal{C}_{111}$	35-0.10	3.1×10^{-3}	17/09/95-29/03/02	21.3-22.4	35.9-38.6	32-33	4.0-4.2
$\mathcal{R}_{224}$	$\mathcal{C}_{114}$	35-0.10	2×10^{-5}	15/02/63-14/04/81	8.00-21.4	42.4-45.2	31-34	4.8-5.1
$\mathcal{R}_{220}$	$\mathcal{C}_{112}$	24-0.06	1.2×10^{-5}		26.1-26.8	45.4-45.9	105-172	4.8-5.5
$\mathcal{R}_{232}$	$\mathcal{C}_{118}$	15-0.04	1.9×10^{-7}	07/01/61-25/08/93	30.9-45.1	33.3-37.9	39-70	4.8-5.2
$\mathcal{R}_{234}$	$\mathcal{C}_{119}$	13-0.03	8.3×10^{-5}	18/02/76-26/07/79	9.30-22.5	41.1-47.0	9-10	4.9-5.1
$\mathcal{R}_{235}$	$\mathcal{C}_{120}$	13-0.03	5.4×10^{-2}	29/08/82-12/01/84	19.9-20.2	37.9-38.4	9-10	4.8-5.2
$\mathcal{R}_{225}$	$\mathcal{C}_{115}$	12-0.03	2×10^{-9}		21.5-45.0	36.8-47.0	48-146	5.5-7.1
$\mathcal{R}_{221}$	$\mathcal{C}_{113}$	11-0.03	1.9×10^{-5}		26.5-27.2	36.3-36.9	144-171	4.8-5.2
$\mathcal{R}_{230}$	$\mathcal{C}_{116}$	10-0.02	2.4×10^{-7}		20.8-28.0	32.0-33.8	5-33	4.8-5.2
$\mathcal{R}_{231}$	$\mathcal{C}_{117}$	9-0.02	4.1×10^{-7}	08/02/75-19/04/04	12.6-16.1	38.5-40.1	258-487	4.8-5.0

Table 7.11: Clustering Rules for the *ANSS* Earthquake Dataset (D)

7.7.5 *ANSS* Earthquake Cluster Statistics

Table 7.12 provides statistics for the discovered clusters including: `Number of Rules`, `Number of Points`, `Coverage` and mean `Density` of the rules belonging to the cluster. Clusters are sorted on decreasing order of point coverage.

Cluster	#Rules	#Points	Coverage(%)	Density
$\mathcal{C}_{2}$	14	2896	8.372	7.8×10^{-3}
$\mathcal{C}_{12}$	9	1670	4.828	1.2×10^{-4}
$\mathcal{C}_{4}$	7	1045	3.021	2×10^{-3}
$\mathcal{C}_{22}$	7	921	2.662	8.9×10^{-4}
$\mathcal{C}_{30}$	7	796	2.301	3×10^{-6}
$\mathcal{C}_{25}$	5	560	1.619	1.5×10^{-3}
$\mathcal{C}_{15}$	4	536	1.549	1.9×10^{-4}
$\mathcal{C}_{3}$	4	499	1.442	3.8×10^{-3}
$\mathcal{C}_{26}$	4	416	1.203	1.9×10^{-5}
$\mathcal{C}_{1}$	1	361	1.044	1.3×10^{-3}
$\mathcal{C}_{79}$	2	349	1.009	1.9×10^{-1}
$\mathcal{C}_{20}$	2	333	0.963	2.4×10^{-4}
$\mathcal{C}_{29}$	3	294	0.85	3.3×10^{-3}
$\mathcal{C}_{8}$	1	270	0.781	3.9×10^{-3}
$\mathcal{C}_{7}$	1	254	0.734	1.3×10^{-2}
$\mathcal{C}_{50}$	2	239	0.691	1.1×10^{-4}
$\mathcal{C}_{28}$	3	231	0.668	2.4×10^{-6}
$\mathcal{C}_{27}$	1	197	0.569	2.6×10^{-1}
$\mathcal{C}_{18}$	1	185	0.535	9.8×10^{-5}
$\mathcal{C}_{33}$	1	143	0.413	1.1×10^{-1}
$\mathcal{C}_{61}$	2	121	0.35	6.7×10^{-3}
$\mathcal{C}_{48}$	1	107	0.309	4.1×10^{-2}
$\mathcal{C}_{34}$	1	98	0.283	1.6×10^{-5}
$\mathcal{C}_{98}$	2	86	0.249	3.3×10^{-6}
$\mathcal{C}_{43}$	1	81	0.234	8.2×10^{-5}
$\mathcal{C}_{42}$	1	76	0.22	2.1×10^{-2}
$\mathcal{C}_{54}$	1	71	0.205	2.5×10^{-5}
$\mathcal{C}_{74}$	1	71	0.205	1.7×10^{-1}
$\mathcal{C}_{38}$	1	69	0.199	4.1×10^{-6}
$\mathcal{C}_{57}$	1	68	0.197	1.5×10^{-5}
$\mathcal{C}_{62}$	1	63	0.182	1.2×10^{-6}
$\mathcal{C}_{59}$	1	62	0.179	1.6×10^{-1}
$\mathcal{C}_{70}$	1	62	0.179	1.8×10^{-2}
$\mathcal{C}_{72}$	1	61	0.176	2.3×10^{-5}
$\mathcal{C}_{104}$	1	57	0.165	1.9×10^{-3}
$\mathcal{C}_{36}$	1	54	0.156	2.2×10^{-1}
$\mathcal{C}_{58}$	1	54	0.156	2.2
$\mathcal{C}_{37}$	1	53	0.153	1.3×10^{-1}
$\mathcal{C}_{64}$	1	52	0.15	3.4×10^{-2}
$\mathcal{C}_{93}$	1	51	0.147	1.5×10^{-4}
$\mathcal{C}_{92}$	1	48	0.139	2.2×10^{-3}
$\mathcal{C}_{77}$	1	46	0.133	5.7×10^{-5}
$\mathcal{C}_{89}$	1	45	0.13	1.9×10^{-6}
$\mathcal{C}_{107}$	2	45	0.13	1.2×10^{-6}
$\mathcal{C}_{73}$	1	44	0.127	2.8
$\mathcal{C}_{106}$	1	44	0.127	2.5×10^{-3}
$\mathcal{C}_{91}$	1	42	0.121	3.3×10^{-1}
$\mathcal{C}_{69}$	1	40	0.116	1
$\mathcal{C}_{87}$	1	40	0.116	3.8×10^{-5}
$\mathcal{C}_{88}$	1	39	0.113	1.1×10^{-3}
$\mathcal{C}_{94}$	1	38	0.11	1.6
$\mathcal{C}_{102}$	1	38	0.11	9.7×10^{-4}
$\mathcal{C}_{99}$	1	37	0.107	2.2×10^{-1}
$\mathcal{C}_{100}$	1	36	0.104	1.1×10^{-6}
$\mathcal{C}_{108}$	1	36	0.104	1×10^{-2}
$\mathcal{C}_{110}$	1	35	0.101	7.6×10^{-4}
$\mathcal{C}_{114}$	1	35	0.101	2×10^{-5}
$\mathcal{C}_{118}$	1	15	0.043	1.9×10^{-7}
$\mathcal{C}_{120}$	1	13	0.038	5.4×10^{-2}
$\mathcal{C}_{113}$	1	11	0.032	1.9×10^{-5}

Cluster	#Rules	#Points	Coverage(%)	Density
$\mathcal{C}_{17}$	12	1731	5.004	6.6×10^{-4}
$\mathcal{C}_{13}$	10	1231	3.559	7.7×10^{-6}
$\mathcal{C}_{11}$	5	979	2.83	6.4×10^{-4}
$\mathcal{C}_{10}$	4	853	2.466	1×10^{-5}
$\mathcal{C}_{6}$	1	787	2.275	1.8×10^{-2}
$\mathcal{C}_{14}$	3	538	1.555	6.5×10^{-5}
$\mathcal{C}_{23}$	3	520	1.503	2.5×10^{-4}
$\mathcal{C}_{9}$	3	490	1.416	3.2×10^{-5}
$\mathcal{C}_{5}$	2	367	1.061	6.1×10^{-3}
$\mathcal{C}_{16}$	2	361	1.044	1.4×10^{-3}
$\mathcal{C}_{0}$	1	348	1.006	3.9×10^{-2}
$\mathcal{C}_{24}$	2	302	0.873	1.8×10^{-5}
$\mathcal{C}_{32}$	3	284	0.821	8.3×10^{-6}
$\mathcal{C}_{19}$	1	267	0.772	5.4×10^{-4}
$\mathcal{C}_{47}$	4	249	0.72	4.7×10^{-4}
$\mathcal{C}_{39}$	4	237	0.685	1×10^{-6}
$\mathcal{C}_{45}$	2	210	0.607	2.4×10^{-7}
$\mathcal{C}_{31}$	1	188	0.543	3×10^{-5}
$\mathcal{C}_{35}$	4	177	0.512	4.6×10^{-7}
$\mathcal{C}_{78}$	3	134	0.387	9.6×10^{-5}
$\mathcal{C}_{40}$	1	111	0.321	6.8×10^{-5}
$\mathcal{C}_{71}$	2	100	0.289	9.7×10^{-4}
$\mathcal{C}_{41}$	1	97	0.28	4×10^{-1}
$\mathcal{C}_{85}$	2	82	0.237	1.4×10^{-3}
$\mathcal{C}_{51}$	1	77	0.223	1.
$\mathcal{C}_{97}$	3	73	0.211	1.5×10^{-7}
$\mathcal{C}_{55}$	1	71	0.205	5.9×10^{-1}
$\mathcal{C}_{46}$	1	70	0.202	2.7×10^{-4}
$\mathcal{C}_{53}$	1	69	0.199	2.9×10^{-4}
$\mathcal{C}_{49}$	1	67	0.194	1.8×10^{-4}
$\mathcal{C}_{63}$	1	63	0.182	5.8×10^{-3}
$\mathcal{C}_{60}$	1	62	0.179	3.4×10^{-2}
$\mathcal{C}_{52}$	1	61	0.176	8.5×10^{-4}
$\mathcal{C}_{66}$	1	59	0.171	3×10^{-1}
$\mathcal{C}_{80}$	1	56	0.162	1.2×10^{-1}
$\mathcal{C}_{56}$	1	54	0.156	1.1
$\mathcal{C}_{67}$	1	54	0.156	3×10^{-5}
$\mathcal{C}_{83}$	1	53	0.153	2.5×10^{-2}
$\mathcal{C}_{76}$	1	51	0.147	1.4×10^{-2}
$\mathcal{C}_{84}$	1	48	0.139	7.6×10^{-7}
$\mathcal{C}_{109}$	1	47	0.136	2.7×10^{-4}
$\mathcal{C}_{81}$	1	45	0.13	1.2×10^{-4}
$\mathcal{C}_{90}$	1	45	0.13	1.6×10^{-5}
$\mathcal{C}_{44}$	1	44	0.127	4.4×10^{-5}
$\mathcal{C}_{105}$	1	44	0.127	1.1×10^{-1}
$\mathcal{C}_{75}$	1	42	0.121	4.2
$\mathcal{C}_{101}$	1	42	0.121	6×10^{-5}
$\mathcal{C}_{82}$	1	40	0.116	4×10^{-7}
$\mathcal{C}_{68}$	1	39	0.113	3.2×10^{-1}
$\mathcal{C}_{96}$	1	39	0.113	3.8×10^{-6}
$\mathcal{C}_{95}$	1	38	0.11	1.2×10^{-3}
$\mathcal{C}_{86}$	1	37	0.107	9.2×10^{-7}
$\mathcal{C}_{21}$	1	36	0.104	1.9×10^{-4}
$\mathcal{C}_{103}$	1	36	0.104	1×10^{-7}
$\mathcal{C}_{65}$	1	35	0.101	3.3×10^{-2}
$\mathcal{C}_{111}$	1	35	0.101	3.1×10^{-3}
$\mathcal{C}_{112}$	1	24	0.069	1.2×10^{-5}
$\mathcal{C}_{119}$	1	13	0.038	8.3×10^{-5}
$\mathcal{C}_{115}$	1	12	0.035	2×10^{-9}
$\mathcal{C}_{116}$	1	10	0.029	2.4×10^{-7}

Table 7.12: *ANSS* Earthquake Cluster Statistics

7.8 Arbitrary Density Rules

$\mathcal{NOCEA}$ has the remarkable property of being able to discover rules at *any* density level, provided of course that each feasible rule contains a minimum number of points, i.e. the sparsity level NT_s (section 5.3). This ability of $\mathcal{NOCEA}$ becomes evident in table 7.13 where the discovered rules are sorted in the descending order of their point density.

Typically, `AS-rules` accompanying strong earthquakes are by definition the most dense, since they are very narrowly bounded in both space and time. Some representative examples of relatively dense rules are $\mathcal{R}_{97}$ and $\mathcal{R}_{56}$ with density 2.16 and 0.225, respectively. The density of rules is measured in units of number of points per grid cell. $\mathcal{R}_{97}$ (3.0≤Mag.≤3.1) and $\mathcal{R}_{56}$ (4.0≤Mag.≤4.2) each cover 54 shallow aftershocks that occurred at focal depths 5km and 10km respectively, following the 13/05/1995 destructive earthquake in Kozani-Greece with magnitude 6.6 [86]. $\mathcal{R}_{218}$, on the other hand, is one of the sparser rules with density 1.51x10^{-7}, that is nearly six orders of magnitude smaller than $\mathcal{R}_{97}$. $\mathcal{R}_{218}$, an `R-rule` type, covers a major part of the regular large-magnitude seismicity (5.2≤Mag.≤6.2) in the EAF-Karviola junction, generated within a specific depth interval (4-37km) from the beginning of the catalogue. The 3-D projections of $\mathcal{R}_{97}$, $\mathcal{R}_{56}$ and $\mathcal{R}_{218}$ shown in figures 7.68-7.70, provide a clear sense of how large is the compactness between `AS-` and `R-rules` with large spatial window. For illustrative purposes, both the borders and the data points covered by a given rule are visualised with the same colour, which is different from other rules. Examples of varying density rules on artificial datasets can also be found in section 6.6.

7.9 Arbitrary Geometry and Size Rules

The ability to self-adjust well to the geometry and size of non-spherical clusters, is one of $\mathcal{NOCEA}$'s most appealing properties for real-world clustering. As a matter of fact, the number and combination of dimensions where clustering rules may exhibit large variances in size and geometry is arbitrary, and more importantly is directly inherited from the underlying data distribution.

ID	Rule	Density
0	$\mathcal{R}_{123}$	4.2
1	$\mathcal{R}_{121}$	2.75
2	$\mathcal{R}_{97}$	2.16
3	$\mathcal{R}_{171}$	1.58
4	$\mathcal{R}_{94}$	1.12
5	$\mathcal{R}_{88}$	1.07
6	$\mathcal{R}_{117}$	1
7	$\mathcal{R}_{92}$	5.92×10^{-1}
8	$\mathcal{R}_{70}$	4.04×10^{-1}
9	$\mathcal{R}_{167}$	3.28×10^{-1}
10	$\mathcal{R}_{116}$	3.25×10^{-1}
11	$\mathcal{R}_{112}$	2.95×10^{-1}
12	$\mathcal{R}_{38}$	2.57×10^{-1}
13	$\mathcal{R}_{56}$	2.25×10^{-1}
14	$\mathcal{R}_{134}$	2.23×10^{-1}
15	$\mathcal{R}_{181}$	2.2×10^{-1}
16	$\mathcal{R}_{122}$	1.67×10^{-1}
17	$\mathcal{R}_{98}$	1.61×10^{-1}
18	$\mathcal{R}_{57}$	1.32×10^{-1}
19	$\mathcal{R}_{135}$	1.17×10^{-1}
20	$\mathcal{R}_{196}$	1.15×10^{-1}
21	$\mathcal{R}_{136}$	1.15×10^{-1}
22	$\mathcal{R}_{47}$	1.08×10^{-1}
23	$\mathcal{R}_{235}$	5.42×10^{-2}
24	$\mathcal{R}_{79}$	4.15×10^{-2}
25	$\mathcal{R}_{0}$	3.95×10^{-2}
26	$\mathcal{R}_{109}$	3.44×10^{-2}
27	$\mathcal{R}_{101}$	3.4×10^{-2}
28	$\mathcal{R}_{110}$	3.26×10^{-2}
29	$\mathcal{R}_{146}$	2.45×10^{-2}
30	$\mathcal{R}_{71}$	2.11×10^{-2}
31	$\mathcal{R}_{6}$	1.84×10^{-2}
32	$\mathcal{R}_{118}$	1.81×10^{-2}
33	$\mathcal{R}_{199}$	1.39×10^{-2}
34	$\mathcal{R}_{126}$	1.36×10^{-2}
35	$\mathcal{R}_{7}$	1.34×10^{-2}
36	$\mathcal{R}_{11}$	1.14×10^{-2}
37	$\mathcal{R}_{161}$	1.05×10^{-2}
38	$\mathcal{R}_{13}$	1.01×10^{-2}
39	$\mathcal{R}_{208}$	1×10^{-2}
40	$\mathcal{R}_{50}$	9.59×10^{-3}
41	$\mathcal{R}_{154}$	9×10^{-3}
42	$\mathcal{R}_{124}$	8.85×10^{-3}
43	$\mathcal{R}_{106}$	7.41×10^{-3}
44	$\mathcal{R}_{2}$	7.36×10^{-3}
45	$\mathcal{R}_{85}$	7.28×10^{-3}
46	$\mathcal{R}_{30}$	7.2×10^{-3}
47	$\mathcal{R}_{67}$	6.83×10^{-3}
48	$\mathcal{R}_{215}$	6.72×10^{-3}
49	$\mathcal{R}_{191}$	6.59×10^{-3}
50	$\mathcal{R}_{188}$	6.34×10^{-3}
51	$\mathcal{R}_{5}$	6.33×10^{-3}
52	$\mathcal{R}_{18}$	6.32×10^{-3}
53	$\mathcal{R}_{108}$	5.84×10^{-3}
54	$\mathcal{R}_{187}$	5.53×10^{-3}
55	$\mathcal{R}_{105}$	5.21×10^{-3}
56	$\mathcal{R}_{65}$	4.71×10^{-3}
57	$\mathcal{R}_{8}$	3.94×10^{-3}
58	$\mathcal{R}_{141}$	3.67×10^{-3}
59	$\mathcal{R}_{86}$	3.62×10^{-3}
60	$\mathcal{R}_{3}$	3.59×10^{-3}
61	$\mathcal{R}_{42}$	3.42×10^{-3}
62	$\mathcal{R}_{35}$	3.16×10^{-3}
63	$\mathcal{R}_{216}$	3.1×10^{-3}
64	$\mathcal{R}_{180}$	2.89×10^{-3}
65	$\mathcal{R}_{63}$	2.88×10^{-3}
66	$\mathcal{R}_{198}$	2.55×10^{-3}
67	$\mathcal{R}_{111}$	2.52×10^{-3}
68	$\mathcal{R}_{149}$	2.45×10^{-3}
69	$\mathcal{R}_{173}$	2.39×10^{-3}
70	$\mathcal{R}_{168}$	2.22×10^{-3}
71	$\mathcal{R}_{131}$	2.21×10^{-3}
72	$\mathcal{R}_{194}$	1.94×10^{-3}
73	$\mathcal{R}_{138}$	1.84×10^{-3}
74	$\mathcal{R}_{4}$	1.71×10^{-3}
75	$\mathcal{R}_{176}$	1.58×10^{-3}
76	$\mathcal{R}_{36}$	1.55×10^{-3}
77	$\mathcal{R}_{186}$	1.53×10^{-3}
78	$\mathcal{R}_{19}$	1.53×10^{-3}
79	$\mathcal{R}_{51}$	1.46×10^{-3}
80	$\mathcal{R}_{156}$	1.44×10^{-3}
81	$\mathcal{R}_{119}$	1.31×10^{-3}
82	$\mathcal{R}_{1}$	1.29×10^{-3}
83	$\mathcal{R}_{172}$	1.23×10^{-3}
84	$\mathcal{R}_{104}$	1.23×10^{-3}
85	$\mathcal{R}_{29}$	1.22×10^{-3}
86	$\mathcal{R}_{152}$	1.2×10^{-3}
87	$\mathcal{R}_{103}$	1.16×10^{-3}
88	$\mathcal{R}_{189}$	1.11×10^{-3}
89	$\mathcal{R}_{160}$	1.07×10^{-3}
90	$\mathcal{R}_{49}$	1.06×10^{-3}
91	$\mathcal{R}_{236}$	1×10^{-3}
92	$\mathcal{R}_{185}$	9.67×10^{-4}
93	$\mathcal{R}_{55}$	9.59×10^{-4}
94	$\mathcal{R}_{22}$	9.42×10^{-4}
95	$\mathcal{R}_{159}$	8.48×10^{-4}
96	$\mathcal{R}_{89}$	8.46×10^{-4}
97	$\mathcal{R}_{12}$	7.91×10^{-4}
98	$\mathcal{R}_{147}$	7.62×10^{-4}
99	$\mathcal{R}_{211}$	7.58×10^{-4}
100	$\mathcal{R}_{157}$	7.57×10^{-4}
101	$\mathcal{R}_{195}$	6.87×10^{-4}
102	$\mathcal{R}_{26}$	6.69×10^{-4}
103	$\mathcal{R}_{214}$	6.62×10^{-4}
104	$\mathcal{R}_{44}$	6.51×10^{-4}
105	$\mathcal{R}_{80}$	6.45×10^{-4}
106	$\mathcal{R}_{20}$	6.37×10^{-4}
107	$\mathcal{R}_{127}$	6.1×10^{-4}
108	$\mathcal{R}_{31}$	6.05×10^{-4}
109	$\mathcal{R}_{77}$	5.77×10^{-4}
110	$\mathcal{R}_{24}$	5.39×10^{-4}
111	$\mathcal{R}_{151}$	5.23×10^{-4}
112	$\mathcal{R}_{178}$	5.18×10^{-4}
113	$\mathcal{R}_{125}$	4.8×10^{-4}
114	$\mathcal{R}_{174}$	4.66×10^{-4}
115	$\mathcal{R}_{202}$	4.65×10^{-4}
116	$\mathcal{R}_{39}$	4.41×10^{-4}
117	$\mathcal{R}_{61}$	4.35×10^{-4}
118	$\mathcal{R}_{133}$	4.18×10^{-4}
119	$\mathcal{R}_{82}$	3.63×10^{-4}
120	$\mathcal{R}_{163}$	3.52×10^{-4}
121	$\mathcal{R}_{87}$	2.91×10^{-4}
122	$\mathcal{R}_{90}$	2.87×10^{-4}
123	$\mathcal{R}_{27}$	2.75×10^{-4}
124	$\mathcal{R}_{76}$	2.72×10^{-4}
125	$\mathcal{R}_{209}$	2.7×10^{-4}
126	$\mathcal{R}_{32}$	2.57×10^{-4}
127	$\mathcal{R}_{128}$	1.96×10^{-4}
128	$\mathcal{R}_{201}$	1.91×10^{-4}
129	$\mathcal{R}_{28}$	1.89×10^{-4}
130	$\mathcal{R}_{53}$	1.86×10^{-4}
131	$\mathcal{R}_{66}$	1.85×10^{-4}
132	$\mathcal{R}_{217}$	1.84×10^{-4}
133	$\mathcal{R}_{210}$	1.83×10^{-4}
134	$\mathcal{R}_{81}$	1.81×10^{-4}
135	$\mathcal{R}_{17}$	1.77×10^{-4}
136	$\mathcal{R}_{192}$	1.68×10^{-4}
137	$\mathcal{R}_{165}$	1.68×10^{-4}
138	$\mathcal{R}_{75}$	1.65×10^{-4}
139	$\mathcal{R}_{170}$	1.51×10^{-4}
140	$\mathcal{R}_{14}$	1.5×10^{-4}
141	$\mathcal{R}_{83}$	1.37×10^{-4}
142	$\mathcal{R}_{60}$	1.2×10^{-4}
143	$\mathcal{R}_{140}$	1.19×10^{-4}
144	$\mathcal{R}_{21}$	1.12×10^{-4}
145	$\mathcal{R}_{137}$	1.06×10^{-4}
146	$\mathcal{R}_{155}$	9.95×10^{-5}
147	$\mathcal{R}_{23}$	9.82×10^{-5}
148	$\mathcal{R}_{206}$	9.56×10^{-5}
149	$\mathcal{R}_{234}$	8.35×10^{-5}
150	$\mathcal{R}_{72}$	8.23×10^{-5}
151	$\mathcal{R}_{62}$	7.57×10^{-5}
152	$\mathcal{R}_{84}$	7.53×10^{-5}
153	$\mathcal{R}_{64}$	7.45×10^{-5}
154	$\mathcal{R}_{132}$	6.86×10^{-5}
155	$\mathcal{R}_{69}$	6.77×10^{-5}
156	$\mathcal{R}_{16}$	6.16×10^{-5}
157	$\mathcal{R}_{183}$	6.02×10^{-5}
158	$\mathcal{R}_{130}$	5.7×10^{-5}
159	$\mathcal{R}_{73}$	4.4×10^{-5}
160	$\mathcal{R}_{158}$	3.75×10^{-5}
161	$\mathcal{R}_{9}$	3.74×10^{-5}
162	$\mathcal{R}_{45}$	3.04×10^{-5}
163	$\mathcal{R}_{115}$	3.03×10^{-5}
164	$\mathcal{R}_{91}$	2.53×10^{-5}
165	$\mathcal{R}_{37}$	2.36×10^{-5}
166	$\mathcal{R}_{120}$	2.32×10^{-5}
167	$\mathcal{R}_{197}$	2.19×10^{-5}
168	$\mathcal{R}_{193}$	2.11×10^{-5}
169	$\mathcal{R}_{213}$	2.06×10^{-5}
170	$\mathcal{R}_{113}$	2.04×10^{-5}
171	$\mathcal{R}_{224}$	1.96×10^{-5}
172	$\mathcal{R}_{221}$	1.92×10^{-5}
173	$\mathcal{R}_{148}$	1.85×10^{-5}
174	$\mathcal{R}_{33}$	1.8×10^{-5}
175	$\mathcal{R}_{164}$	1.65×10^{-5}
176	$\mathcal{R}_{48}$	1.58×10^{-5}
177	$\mathcal{R}_{95}$	1.46×10^{-5}
178	$\mathcal{R}_{25}$	1.44×10^{-5}
179	$\mathcal{R}_{78}$	1.27×10^{-5}
180	$\mathcal{R}_{143}$	1.19×10^{-5}
181	$\mathcal{R}_{220}$	1.16×10^{-5}
182	$\mathcal{R}_{200}$	1.09×10^{-5}
183	$\mathcal{R}_{46}$	1.08×10^{-5}
184	$\mathcal{R}_{15}$	1.05×10^{-5}
185	$\mathcal{R}_{96}$	1.02×10^{-5}
186	$\mathcal{R}_{10}$	9.33×10^{-6}
187	$\mathcal{R}_{34}$	8.55×10^{-6}
188	$\mathcal{R}_{102}$	7.92×10^{-6}
189	$\mathcal{R}_{203}$	7.88×10^{-6}
190	$\mathcal{R}_{54}$	7.85×10^{-6}
191	$\mathcal{R}_{184}$	7.36×10^{-6}
192	$\mathcal{R}_{166}$	6.97×10^{-6}
193	$\mathcal{R}_{129}$	5.85×10^{-6}
194	$\mathcal{R}_{169}$	5.51×10^{-6}
195	$\mathcal{R}_{226}$	5.26×10^{-6}
196	$\mathcal{R}_{207}$	5.09×10^{-6}
197	$\mathcal{R}_{41}$	4.9×10^{-6}
198	$\mathcal{R}_{205}$	4.52×10^{-6}
199	$\mathcal{R}_{139}$	4.45×10^{-6}
200	$\mathcal{R}_{142}$	4.27×10^{-6}
201	$\mathcal{R}_{93}$	4.18×10^{-6}
202	$\mathcal{R}_{58}$	4.15×10^{-6}
203	$\mathcal{R}_{175}$	3.79×10^{-6}
204	$\mathcal{R}_{212}$	3.74×10^{-6}
205	$\mathcal{R}_{99}$	3.24×10^{-6}
206	$\mathcal{R}_{59}$	3.06×10^{-6}
207	$\mathcal{R}_{114}$	2.98×10^{-6}
208	$\mathcal{R}_{43}$	2.44×10^{-6}
209	$\mathcal{R}_{179}$	2.38×10^{-6}
210	$\mathcal{R}_{40}$	2.11×10^{-6}
211	$\mathcal{R}_{162}$	1.86×10^{-6}
212	$\mathcal{R}_{100}$	1.6×10^{-6}
213	$\mathcal{R}_{204}$	1.37×10^{-6}
214	$\mathcal{R}_{107}$	1.15×10^{-6}
215	$\mathcal{R}_{182}$	1.06×10^{-6}
216	$\mathcal{R}_{153}$	9.17×10^{-7}
217	$\mathcal{R}_{68}$	9.06×10^{-7}
218	$\mathcal{R}_{144}$	8.55×10^{-7}
219	$\mathcal{R}_{227}$	7.93×10^{-7}
220	$\mathcal{R}_{150}$	7.59×10^{-7}
221	$\mathcal{R}_{229}$	7.14×10^{-7}
222	$\mathcal{R}_{223}$	5.77×10^{-7}
223	$\mathcal{R}_{230}$	5.5×10^{-7}
224	$\mathcal{R}_{52}$	5.31×10^{-7}
225	$\mathcal{R}_{145}$	4.78×10^{-7}
226	$\mathcal{R}_{228}$	4.32×10^{-7}
227	$\mathcal{R}_{233}$	4.25×10^{-7}
228	$\mathcal{R}_{231}$	4.13×10^{-7}
229	$\mathcal{R}_{74}$	3.49×10^{-7}
230	$\mathcal{R}_{219}$	2.71×10^{-7}
231	$\mathcal{R}_{177}$	2.16×10^{-7}
232	$\mathcal{R}_{232}$	1.95×10^{-7}
233	$\mathcal{R}_{222}$	1.9×10^{-7}
234	$\mathcal{R}_{218}$	1.51×10^{-7}
235	$\mathcal{R}_{190}$	1×10^{-7}
236	$\mathcal{R}_{225}$	2.4×10^{-9}

Table 7.13: Varying Density Rules Discovered by $\mathcal{NOCEA}$ in the *ANSS* Dataset

To elaborate on this issue consider some highly-dense `AS-rules` such as $\mathcal{R}_{97}$, $\mathcal{R}_{56}$, and $\mathcal{R}_{79}$, as well as some moderate-to-low density rules, e.g, $\mathcal{R}_{218}$, $\mathcal{R}_{95}$, and $\mathcal{R}_{230}$. A brief discussion regarding $\mathcal{R}_{97}$, $\mathcal{R}_{56}$, and $\mathcal{R}_{218}$ can be found in section 7.8. $\mathcal{R}_{79}$ captures an uncommon pattern of aftershock activity that is not confined around the hypocenter of the mainshock. In short, $\mathcal{R}_{79}$ covers 107 events of the intense aftershock activity associated with the 26/06/2001 strong earthquake in Skyros Island, N. Aegean Sea, Greece, with magnitude 6.5 [85]. Unlike typical `AS-rules`, e.g. $\mathcal{R}_{97}$ and $\mathcal{R}_{56}$, $\mathcal{R}_{79}$ is *remarkably elongated* along the depth axis with a span of over 42km (0-42km), as shown in figure 7.68.

As mentioned earlier, significant differences in size and geometry are not limited only to a single dimension. For instance, consider $\mathcal{R}_{218}$, $\mathcal{R}_{95}$, and $\mathcal{R}_{230}$. $\mathcal{R}_{230}$ delineates a time consistent (`R-rule`) seismogenic source of moderate magnitude (4.8≤Mag.≤5.2) extending to shallow depths (5km≤Depth≤33km) through coastal northeastern Libya and the Mediterranean Sea south of Crete and can be viewed as an extension of the *Hellenic Arc* to the south. R_{95} covers mainland Tunisia as well as the sea between Tunisia and Sicily. This region is seismically active since directly beneath it lies the Africa-Eurasia plate boundary [77]. The geometry of R_{95} is characterised by *a*) thin concentration (9km≤Depth≤ 10km) along the depth axis, *b*) widespread magnitude interval (3.0≤Mag.≤5.2), and *c*) shorter time span of approximately twenty years, compared to $\mathcal{R}_{230}$ and $\mathcal{R}_{218}$. $\mathcal{R}_{230}$ is geometrically a) very confined along two dimensions, i.e. latitude and magnitude, b) widespread in time and depth, and c) average range spreading along longitude. Finally, $\mathcal{R}_{218}$, is a "bulky" rule, i.e. it has a relatively large volume, though its magnitude interval is not extremely widespread.

Figures 7.68-7.70 show 3-D projections of these rules in spatial [`Longitude`×`Latitude`×`Depth`], spatial-temporal [`Longitude`×`Latitude`×`Time`] and spatial-magnitude [`Longitude`×`Latitude`×`Magnitude`] subspaces, illustrating $\mathcal{NOCEA}$'s ability to discover rules with wide diversity in size and geometry.

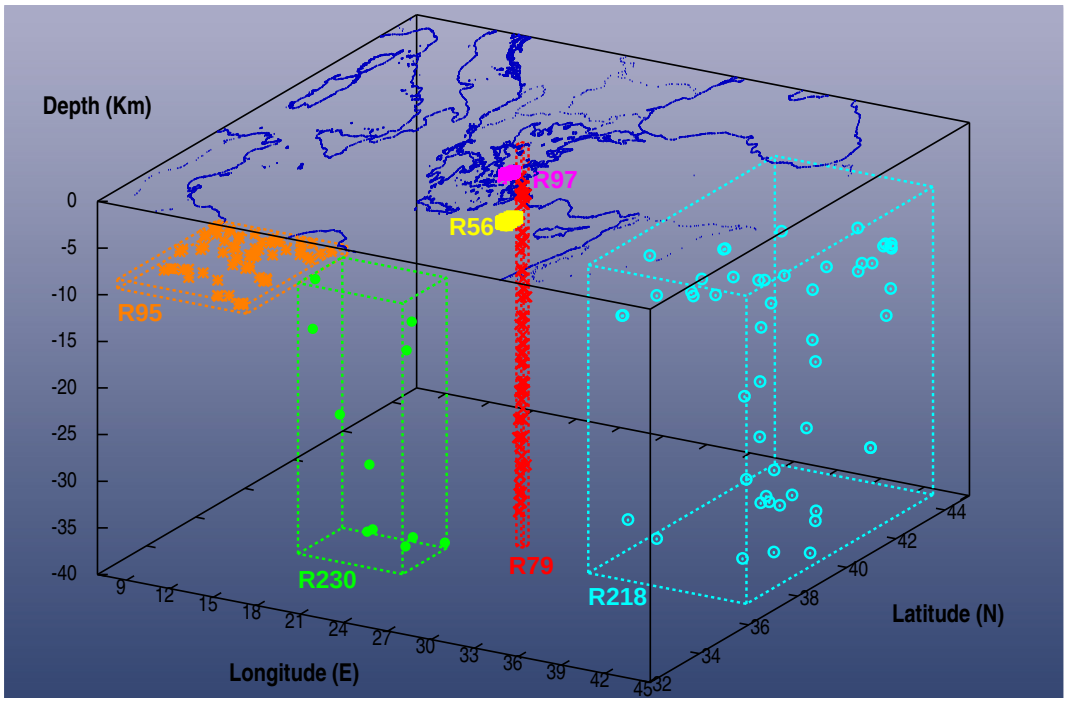


Figure 7.68: Arbitrary Density, Size and Geometry Rules in [Long.×Lat.×Depth]

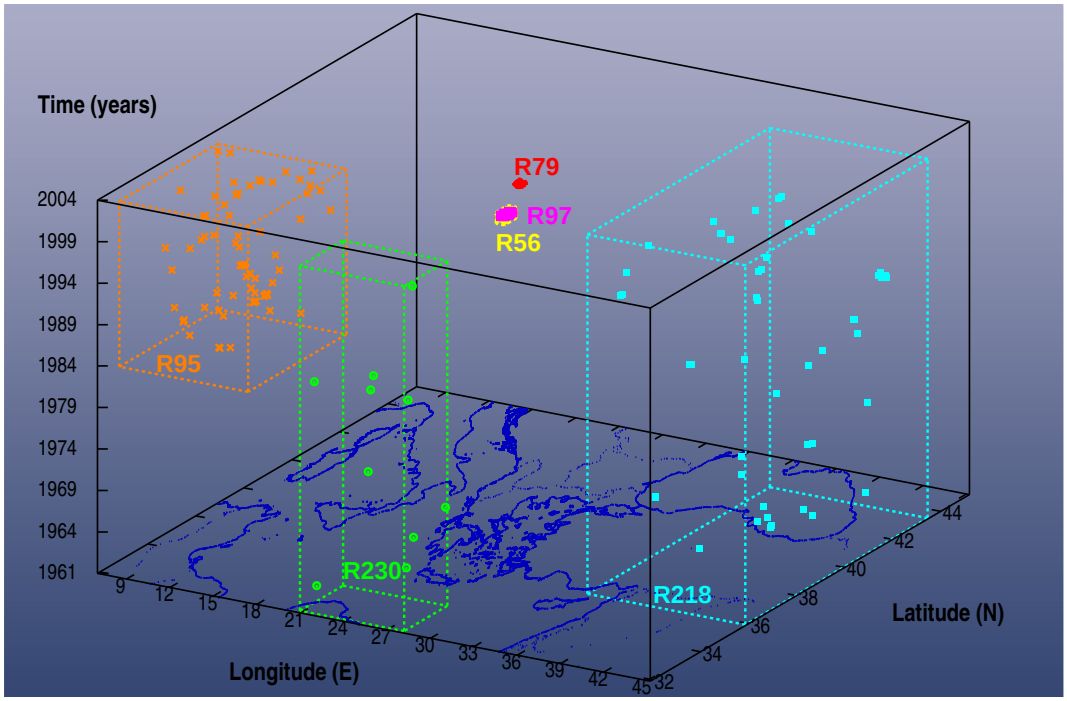


Figure 7.69: Arbitrary Density, Size and Geometry Rules in [Long.×Lat.×Time]

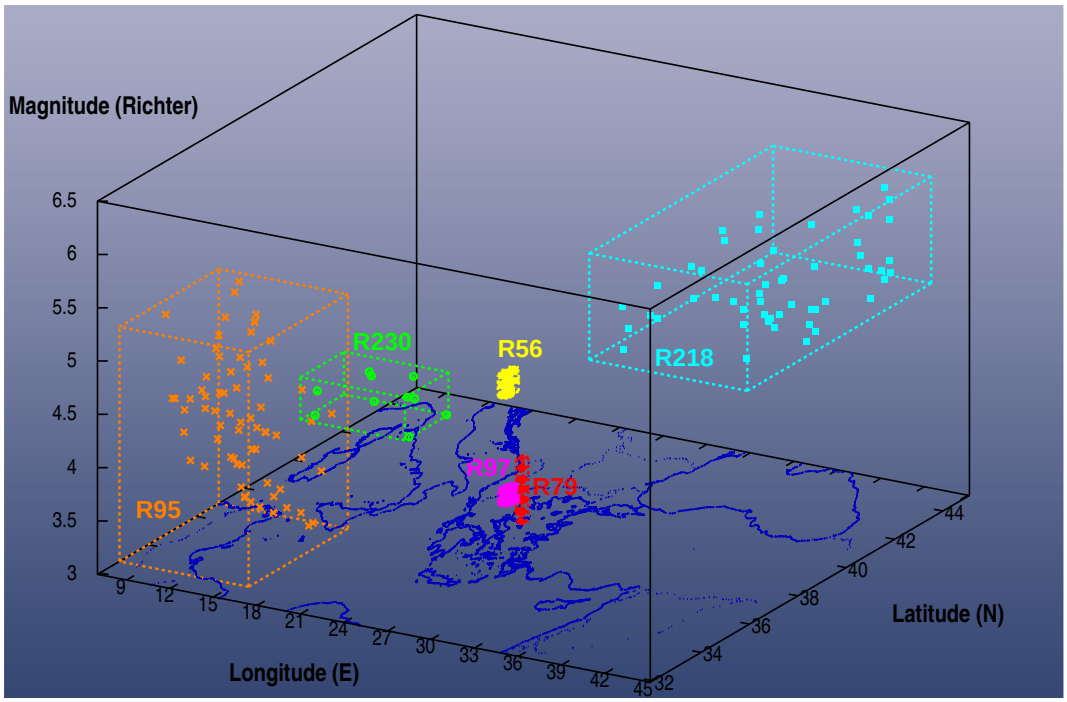


Figure 7.70: Arbitrary Density, Size and Geometry Rules in [Long.×Lat.×Mag.]

7.10 Arbitrary Data Coverage Rules

Figure 7.71 plots a classical frequency histogram (section 3.2) for the data coverage of rules, where the dissection, i.e. allocation, of observations (the data coverage values of the discovered rules) into bins is based on a uniform step of 0.02%.

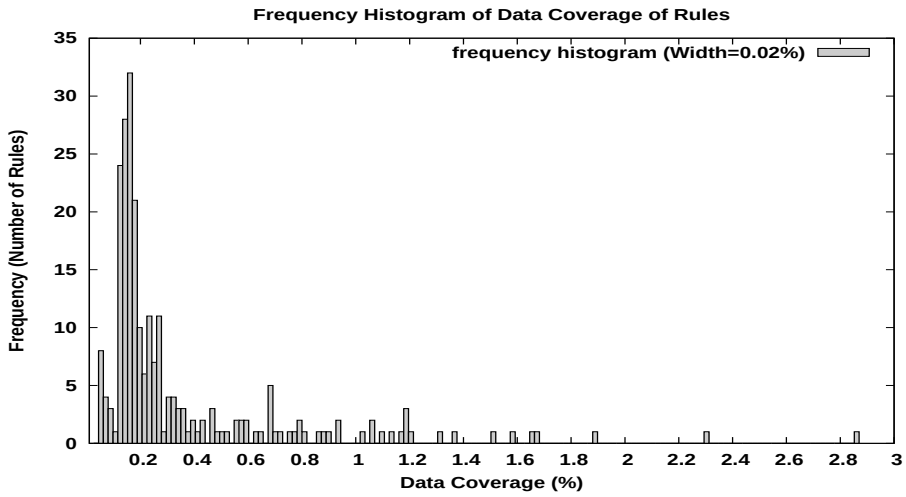


Figure 7.71: Frequency Histogram for the Data Coverage of Rules

The frequency histogram illustrates a number of interesting issues:

- The highly concentrated structure observed around the mode value (approximately at 0.2%) is certainly not due to any preference bias, but it rather reflects $\mathcal{NOCEA}$'s intrinsic ability to directly inherit the size-geometry of rules, and consequently their point coverage, from the underlying data distribution. In the *ANSS* earthquake case study it is the complex nature of earthquake dynamics along with the discontinuity of the faulting zones that result in the formation of a multitude of earthquake patches with tiny point size.

- Clearly, the histogram exhibits a highly positive skew, verifying our intuition that $\mathcal{NOCEA}$ can discover rules with arbitrarily wide variances in data coverage.

- Finally, as is often the case, especially in real-world high-dimensional clustering problems, some clusters may comprise a very small fraction, e.g. 0.2%, of the total points. The analysis of the histogram suggests that $\mathcal{NOCEA}$ has the ability to reveal such tiny structures, regardless of their geometry and density.

7.11 Subspace Clustering

The main goal of subspace clustering is to identify and retain only relevant features (dimensions) of the clustering while pruning away those where the points are very widespread. $\mathcal{NOCEA}$ performed effective subspace clustering for the *ANSS* earthquake dataset. More specifically, in tables 7.8-7.11, empty fields in the antecedent part of the rules correspond to irrelevant feature-genes (see section 5.4.1) that are detected by $\mathcal{NOCEA}$'s post-processing simplification algorithm of section 5.11. It can be easily discerned that the vast majority of rules are embedded in the full 5-D dimensional space, with few exceptions involving solely the temporal dimension. This finding is not surprising for two reasons:

- Due to the discontinuous nature of the faulting zones in our study [77], strain - the main source of earthquakes - accumulates at different rates at different spatial neighbourhoods. Thereby, there are no rules with adequately large spatial window, which, in turn, does not permit dropping any spatial condition in the antecedent part of the rules.

- Due to the logarithmic nature of the G-R power law, clustering rules are always bounded in relatively small intervals along the magnitude axis.

7.11.1 Interpretation of Subspace Clustering

This section discusses the interpretation of the presence of irrelevant features in the *ANSS* dataset.

Figure 7.72 depicts the projection of six rules, namely, $\mathcal{R}_{218}$, $\mathcal{R}_{97}$, $\mathcal{R}_{220}$, $\mathcal{R}_{221}$, $\mathcal{R}_{95}$, and $\mathcal{R}_{169}$, in the 3-D [`Longitude`×`Latitude`×`Time`] subspace. These rules are mainly characterised by their varying time window (interval), and are being deliberately chosen to explain the meaning of subspace clustering from an earthquake analysis perspective.

Clearly, an `R-rule` such as $\mathcal{R}_{218}$, $\mathcal{R}_{220}$ (Vrancea-Romania) or $\mathcal{R}_{221}$ (Kos Island, Greece), delineates a consistent trait of seismic activity with specific spatio-magnitude behaviour over time. Obviously, the more confined the spatio-magnitude window of an `R-rule`, the more precise the prediction about future earthquakes due to the given rule is. From an earthquake prediction point of view, the average value of the time frequency histogram of a rule comprising an irrelevant time-gene can be used to determine the recurrence period of earthquakes with specific magnitude occurring within the spatial region that is fully specified by the antecedent part of that rule. For instance, given that $\mathcal{R}_{218}$ covers 52 events from 1961 to 2004, and its land surface is approximately 1022084km^2, the repetition time of relatively shallow depth (4-37km) earthquakes in this region having magnitude in the range [5.2-6.2] is at least $\left(\frac{2004-1961}{52}\right)\left(\frac{1022084}{100\times100}\right)\approx$84.5 years per 100×100km^2 of land surface. In contrast, `AS-rules`, e.g. $\mathcal{R}_{97}$ are of limited usefulness for long-term earthquake prediction purposes because of their

localised nature and short lifetime.

Between the two extremes, `AS-` and `R-rules`, lie intermediate-duration `U-rules`. Depending on the period(s) of quiescence of `U-rules` one can extract different types of seismic patterns. For instance, consider $\mathcal{R}_{95}$ and $\mathcal{R}_{169}$, two `U-rules` with similar time span, but very different period of activity. $\mathcal{R}_{95}$ is a currently active (17/05/84≤Time≤19/04/04) seismogenic zone in mainland Tunisia as well as the sea between Tunisia and Sicily. Regardless of the reasons for $\mathcal{R}_{95}$'s quiescence from 1961 to 1984, e.g. catalogue incompleteness, it has the potential to become an `R-rule` with a dropped time-gene in the near future, provided of course that the seismic activity in that region continues with the same rate. In contrast, $\mathcal{R}_{169}$'s activity located in Karviola-Turkey ended in the early eighties (15/02/63≤Time≤06/08/80), and therefore the seismic hazard due to $\mathcal{R}_{169}$ should be viewed only as a part of the generalised seismic excitation that was observed in that region during the sixties and seventies, e.g. Varto (1966), Bingol (1971), Lice (1975), Caldiran-Muradiye (1977) [1].

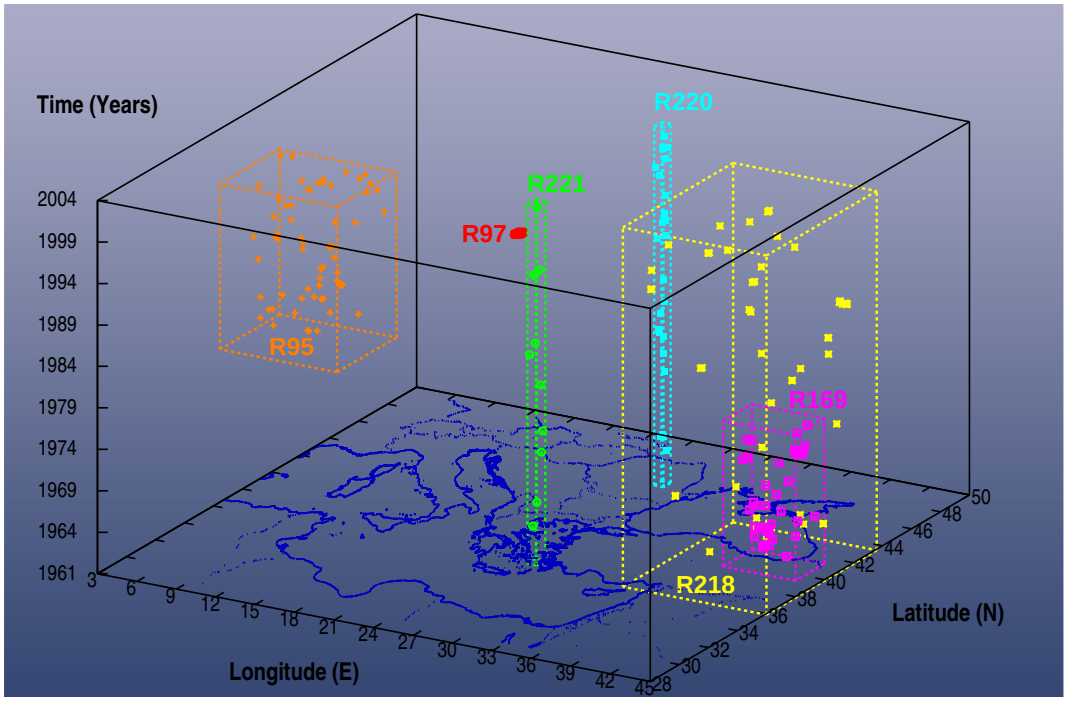


Figure 7.72: Example Rules With and Without Irrelevant Time-Dimension Projected in [Long.×Lat.×Time]

7.11.2 Generalisation and Histogram Smoothing as Prerequisites for Subspace Clustering

This section elaborates the vital role of generalisation (section 5.9) and histogram smoothing via KDE (section 3.3) in subspace clustering.

Following the discussion in the previous section, 7.11.1, it is not difficult to understand the merit of introducing smoothness to the frequency histograms. More specifically, without smoothing, the jagged histograms that correspond to non-relevant dimensions of very low density rules, e.g. time-dimension in $\mathcal{R}_{218}$, $\mathcal{R}_{220}$, or $\mathcal{R}_{221}$, will be split into multiple smaller segments by the homogeneity operator, as described in section 5.7. Excessive fragmentation of low density rules along irrelevant dimensions has two serious drawbacks:

- Formation of spurious (sparse) rules that are subsequently discarded.
- Severe distortion of the elongated shape, which is a prerequisite for declaring a dimension as irrelevant during subspace clustering (section 5.11).

Finally, generalisation also contributes to subspace clustering by building up as generic rules as possible, thus enabling the formation of widespread genes when possible. For instance, without generalisation, $\mathcal{NOCEA}$ would fail to detect the regular seismic pattern demarcated by $\mathcal{R}_{218}$, if the stochastic evolutionary search creates multiple rules inside that region.

7.12 Arbitrary-Shaped Clusters

The complex nature of the intra-continental collision process along with the geological heterogeneity of the earth's crustal outer layer results in the formation of complex spatio-temporal-magnitude cluster structures in the *ANSS* dataset [64]. It is essential to reveal these structures to gain a deeper insight into these intrinsically complex phenomena. In $\mathcal{NOCEA}$, the body of an arbitrary-shaped cluster is approximated using a set of axis-parallel and disjoint rules forming a relatively homogeneous spatio-temporal-magnitude pathway (see section 5.12). The density of points along the neighbourhoods that collectively define such a pathway exhibits only a marginal variation.

It is evident from the cluster-descriptor tables 7.8-7.11 that the *ANSS* earthquake dataset, as expected, comprises many arbitrary-shaped clusters of varying numbers of rules, point coverage, density, and geometry. Figures 7.73, 7.74, and 7.75 depict seven arbitrary-shaped clusters, namely, $\mathcal{C}_2$, $\mathcal{C}_{10}$, $\mathcal{C}_{11}$, $\mathcal{C}_{12}$, $\mathcal{C}_{14}$, $\mathcal{C}_{16}$, and $\mathcal{C}_{17}$, in the 3-D projections [`Longitude`× `Latitude`×`Depth`], [`Longitude`× `Latitude`×`Time`], and [`Longitude`× `Latitude`×`Magnitude`], respectively. Points belonging to the same cluster are plotted with the same colour. One can easily observe the wide diversity in the size and geometry of the non-convex clusters. Notice that $\mathcal{NOCEA}$ found many other clusters with far more complex shapes but they are harder to visualise. Another interesting observation is that non-convex clusters may diffuse (spread) inside the feature space $\mathcal{F}$ in arbitrary directions. Finally, there are different degrees of overlapping among clusters in different subsets of dimensions at different data localities (neighbourhoods). Although there are subspaces, e.g. [`Longitude`×`Latitude`×`Time`], and [`Longitude`× `Latitude`×`Magnitude`], where clusters are becoming blurred, $\mathcal{NOCEA}$ can easily distinguish these clusters as it always operates in the full-dimensional space, which, in turn, guards against artifacts formed in lower dimensional projections as in [7, 74].

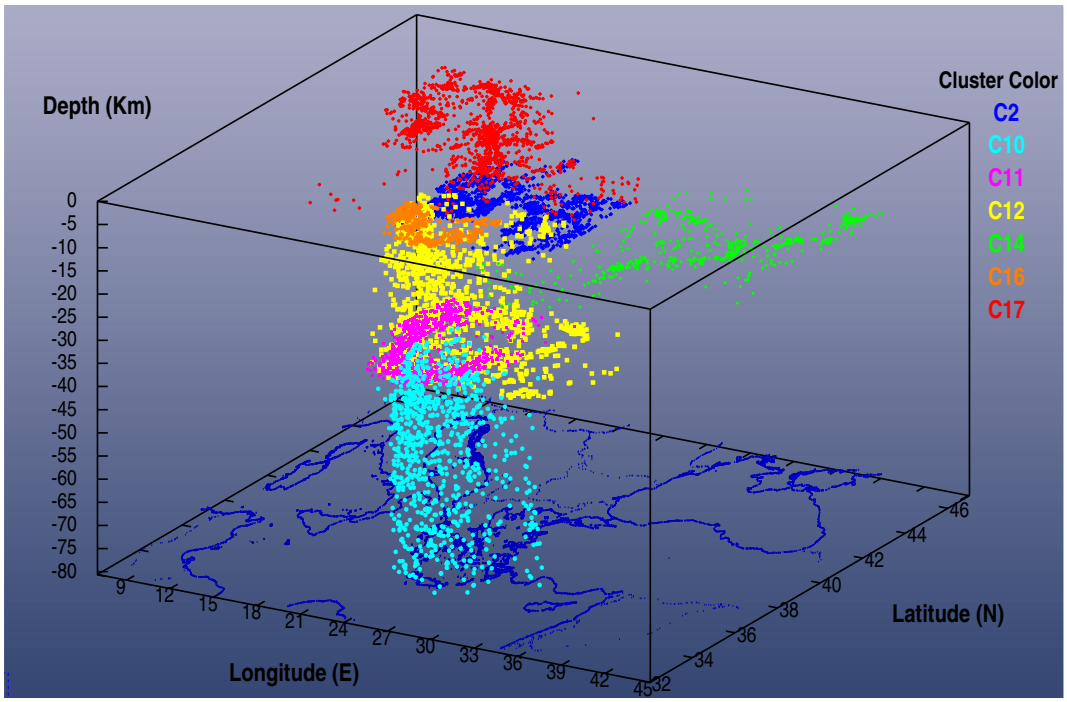


Figure 7.73: Arbitrary-Shaped Clusters Projected in [Long.×Lat.×Depth]

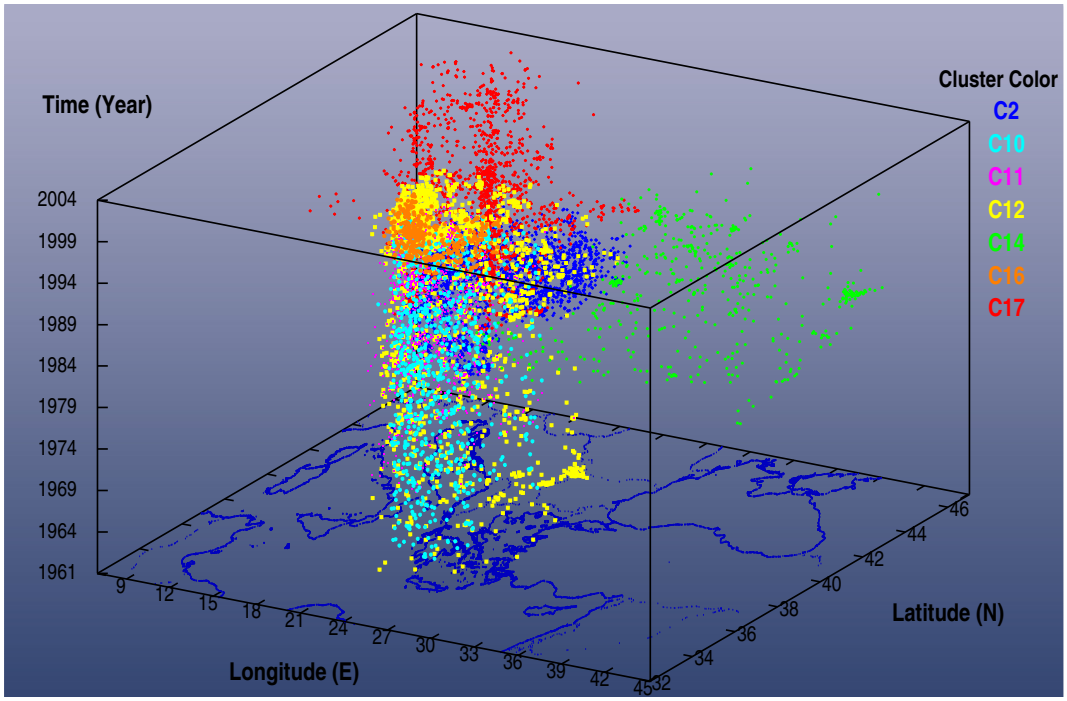


Figure 7.74: Arbitrary-Shaped Clusters Projected in [Long.×Lat.×Time]

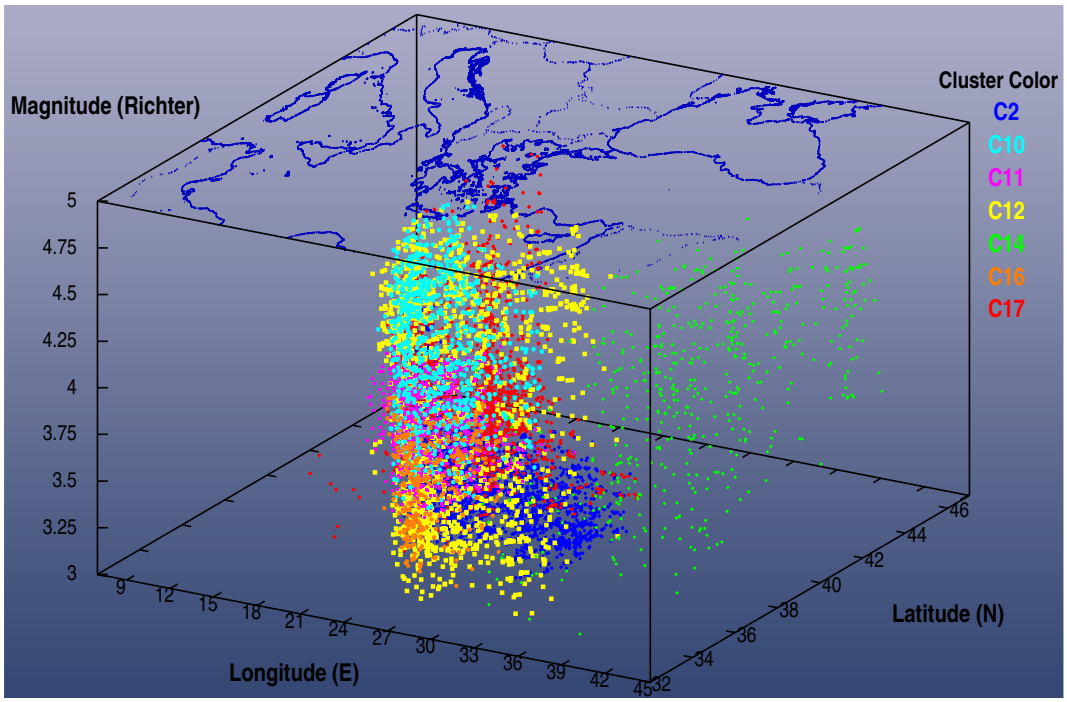


Figure 7.75: Arbitrary-Shaped Clusters Projected in [Long.×Lat.×Mag.]

7.12.1 Interpreting an Arbitrary-shaped Cluster

The backbone of a non-convex cluster may be arbitrarily complex as discussed in section 7.12. Likewise, the subset of dimensions and the data localities where the geometry of a non-convex cluster fluctuates considerably, are also arbitrary. It is worth noting that rules belonging to the same cluster may also overlap and/or have very different densities in some subspaces of the feature space $\mathcal{F}$. In our earthquake clustering context, the data pathway defined by the rules of an arbitrary-shaped cluster reflects the spatio-temporal-magnitude evolution of seismic activity associated with the given cluster. For instance, consider the cluster $\mathcal{C}_4$ whose body is made up of seven rules, i.e. $\mathcal{R}_4$, $\mathcal{R}_{35}$, $\mathcal{R}_{63}$, $\mathcal{R}_{131}$, $\mathcal{R}_{149}$, $\mathcal{R}_{173}$, and $\mathcal{R}_{176}$ (see table 7.9).

The epicentres of the earthquakes in $\mathcal{C}_4$ are widely distributed along an arc extending from the Ionian Sea, in the west, to the Taurides mountains -Turkey, in the east, through the central Aegean, as shown in figure 7.76. Notably, $\mathcal{C}_4$'s geometry exhibits no fluctuations along the third spatial dimension, i.e. depth, since all the above rules are characterised by the same focal depth (9-10km).

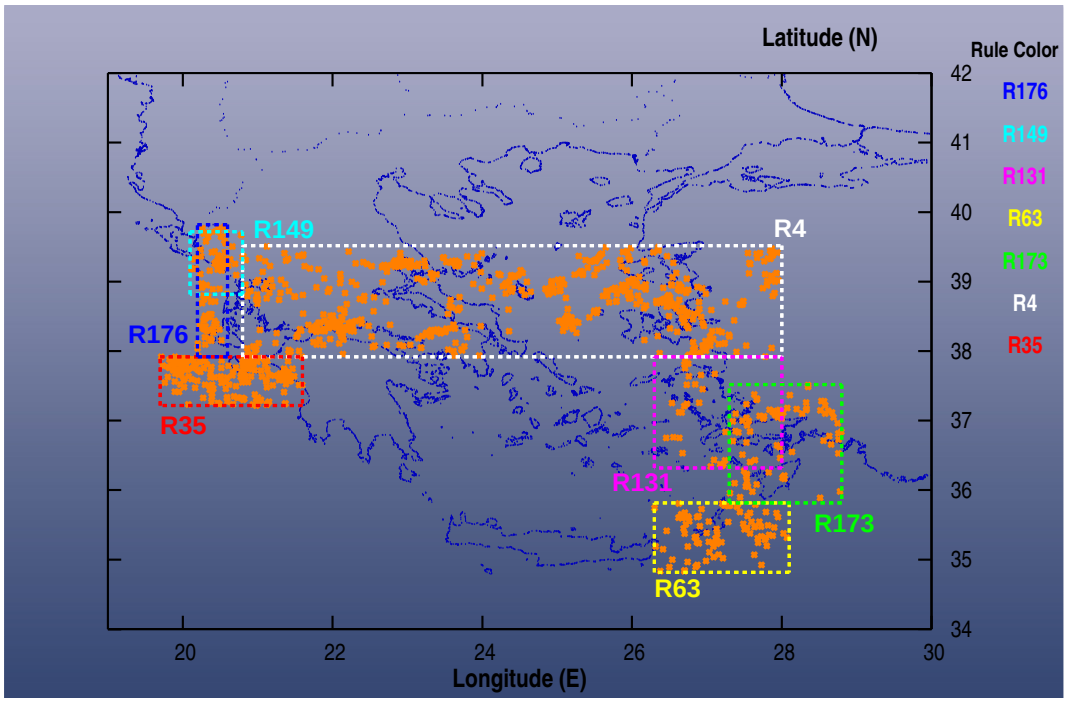


Figure 7.76: The backbone of an Arbitrary-Shaped Cluster ($\mathcal{C}_4$) Projected in [Long.×Lat.]

Figure 7.78 clearly shows that the seismic activity due to $\mathcal{C}_4$ does not belong to a distinct class of events, but instead the range of magnitudes of the rules constituting $\mathcal{C}_4$ differs considerably. Finally, significant temporal fluctuations are only evident (see figure 7.77) in the south-eastern part of the cluster near Rhodes and the Taurides mountains on the south-western coast of Turkey.

Isolating complex patterns of seismic activity, and presenting them as comprehensible summaries in the form of DNF (Disjunctive Normal Form) expressions, helps seismologists to better understand the phenomenon. Finally, the extraction and interpretation of potential causal relationship(s) between the seismicity associated with the rules of $\mathcal{C}_4$ go beyond the scope of the thesis.

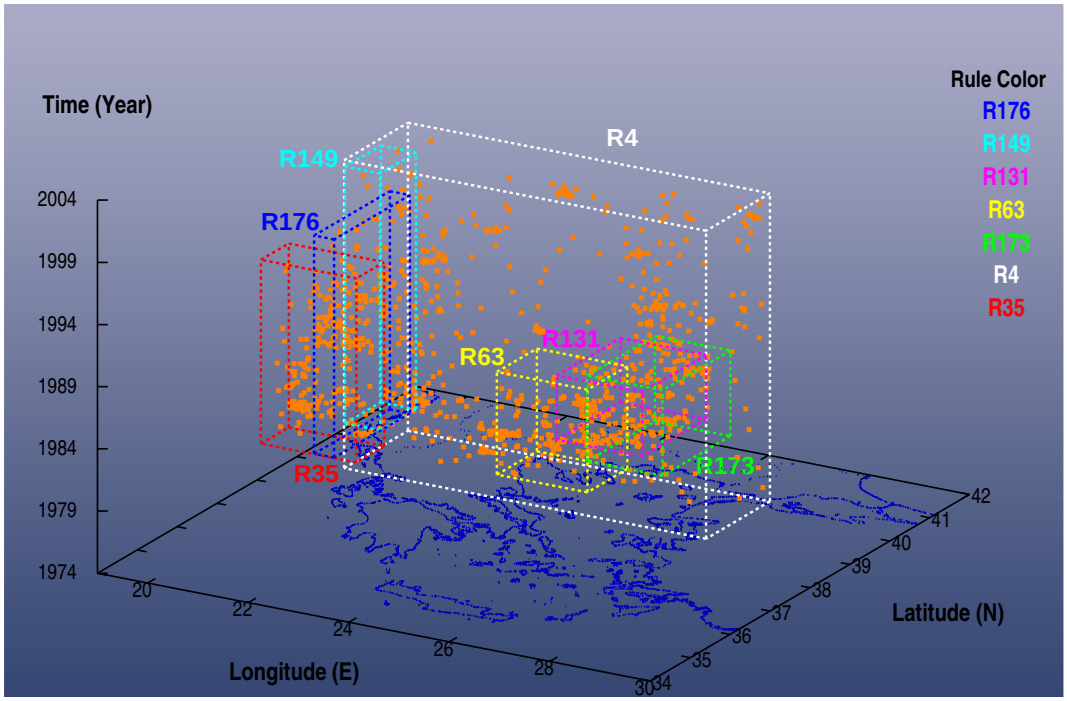


Figure 7.77: The backbone of an Arbitrary-Shaped Cluster ($\mathcal{C}_4$) Projected in [Long.×Lat.×Time]

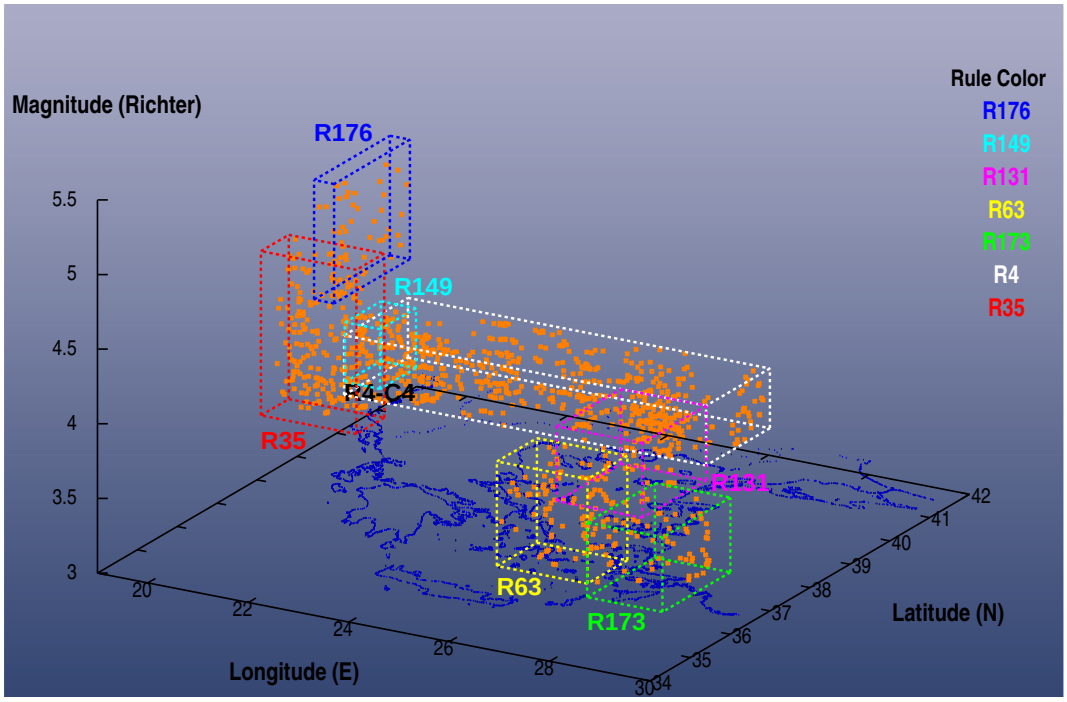


Figure 7.78: The backbone of an Arbitrary-Shaped Cluster ($\mathcal{C}_4$) Projected in [Long.×Lat.×Mag.]

7.13 Seismic Hazard Estimation

This section describes how the seismicity knowledge discovered by $\mathcal{NOCEA}$ can be exploited to compile improved seismic hazard maps.

7.13.1 Seismic Hazard Objectives

Hazard assessment is to evaluate, for a certain place, how frequent and how strong earthquakes will be felt, in order to take measures minimising as much as possible the severity of aftershock damage [1, 3]. In general, a seismic-hazard analysis attempts to address four important questions:

- Delineation of the seismic sources (*where*?)
- Rate of earthquake recurrence (*how often*?)
- Magnitudes-frequency distribution (*how strong*?)
- Derivation of an attenuation relationship (e.g. *what ground motion should be expected at a given distance from an earthquake's origin*?)

Usually, the main objective of a seismic-hazard analysis is to evaluate the probability of exceeding a particular level of ground motion at a site during a specific time interval, e.g. 50 years. In contrast, the goal of hazard-assessment in this thesis is *to evaluate, for a particular site of interest, the repetition time of a shaking with magnitude exceeding a specified level.*

7.13.2 Characterisation of Seismic Sources

The first stage of seismic hazard assessment, the characterisation of seismic sources, involves obtaining robust answers to three important questions: *where*, *how often* and *how strong* earthquakes are likely to be [1, 3]. The clustering rules constituting $\mathcal{NOCEA}$'s discovered knowledge quantify with great accuracy all these aspects of a potential seismic source. In particular, the spatial-shape of rule(s) directly determines the geometry of the seismic source. The boundaries of the time-gene delineate the period of activity, while the average value of the

corresponding frequency histogram determines the recurrence time. Finally, the magnitude-gene specifies the upper and lower magnitude of the source, as well as the average expected magnitude.

7.13.3 Characterisation of Attenuation Relationship

Estimates of expected ground motion or magnitude at a given distance from an earthquake of specific magnitude are the second element of earthquake hazard assessment [1, 3]. These estimates are usually complex equations, called *attenuation relationships*, which express ground motion or magnitude as a function of magnitude and distance from the earthquake's origin. In general, the rate at which the strength of shaking decreases with distance from an earthquake's hypocenter and magnitude varies regionally.

Obtaining robust attenuation laws requires a deep understanding of regional geo-tectonics, which is beyond the scope of the thesis. Consequently, the following over-simplified attenuation model is adopted to ease our analysis:

- By definition (recall from section 5.4 that feasible rules have quasi-homogeneous data distribution) earthquakes associated with a given rule have an equal probability of occurring at any Longitude×Latitude cell of the rectangular area demarcated by that rule.

- Any event is *independent* of the occurrence of all other events.

- To obtain a time resistant hazard estimate short lifetime rules, i.e. period of activity $\leq$15 years, are neglected.

- Seismic waves are propagated such that *any* focal depth earthquake with magnitude M_E occurring at the epicentral location $E(\mathtt{x}_E,\mathtt{y}_E)$ ($\mathtt{x}$: longitude and $\mathtt{y}$: latitude) generates a shaking in a surface site $A(\mathtt{x}_A,\mathtt{y}_A)$ of magnitude M_A that is given by a *static distance-decreasing attenuation law*:

$$\boxed{M_A = M_E * e^{-a*d(E,A)}} \tag{7.36}$$

  where $d(E, A)$ is the surface Euclidean distance between the sites E and A, while a is a constant, i.e. a=0.002.

7.13.4 Computation of Repetition Time

The last element of hazard assessment is the actual computation for a particular site of interest, of the repetition time for a shaking exceeding a specified level of magnitude [1, 3]. This task involves summing up the incremental contributions of vibrations from all seismic sources.

In practice the hazard mass, i.e. number of earthquakes, of each rule is initially distributed over equi-width hazard categories, that is `[3.0-3.5)`, `[3.5-4.0)`,... depending on the intersection of each interval with the magnitude gene of the rule. The hazard mass within each category is then rounded to the center of the corresponding interval, and finally is equally assigned to the Longitude×Latitude cells covered by the rule.

Next, for each cell of the surface grid we compute, using the attenuation law, the number of events in its neighbourhood that could generate at least $\mathtt{M}_{min}$ magnitude shaking at that cell. Finally, the repetition time at a given cell is derived by dividing the mean lifetime of all rules affecting that cell by the total number of events causing at least $\mathtt{M}_{min}$ magnitude shaking at that cell. Recall that in order to obtain a hazard estimation that is relatively resistant to large variations over time, rules with short lifetime, i.e. period of activity<15 years, such as time-confined `AS-rules` are neglected.

Figures 7.79-7.80 depict the hazard for various levels of magnitudes. Briefly, concerning high magnitude shaking, (i.e. `5.0<Mag.`) the northern Aegean Sea and the surrounding lands (north-central Greece, west-central Turkey) appear to be by far the most frequently vibrated regions in our study. As the magnitude threshold $\mathtt{M}_{min}$ decreases finer structures are revealed because more rules are taken into consideration when compiling the hazard map. As expected, the hazard maps in figures 7.80(a-c) show that the Ionian Islands, the Corinthian Gulf, the Hellenic Arc and the Vrancea region in Romania are the most frequently affected regions from low-to-moderate (`Mag.<4.5`) earthquakes. Similar hazard maps have been compiled using complex seismological methods, e.g. [13].

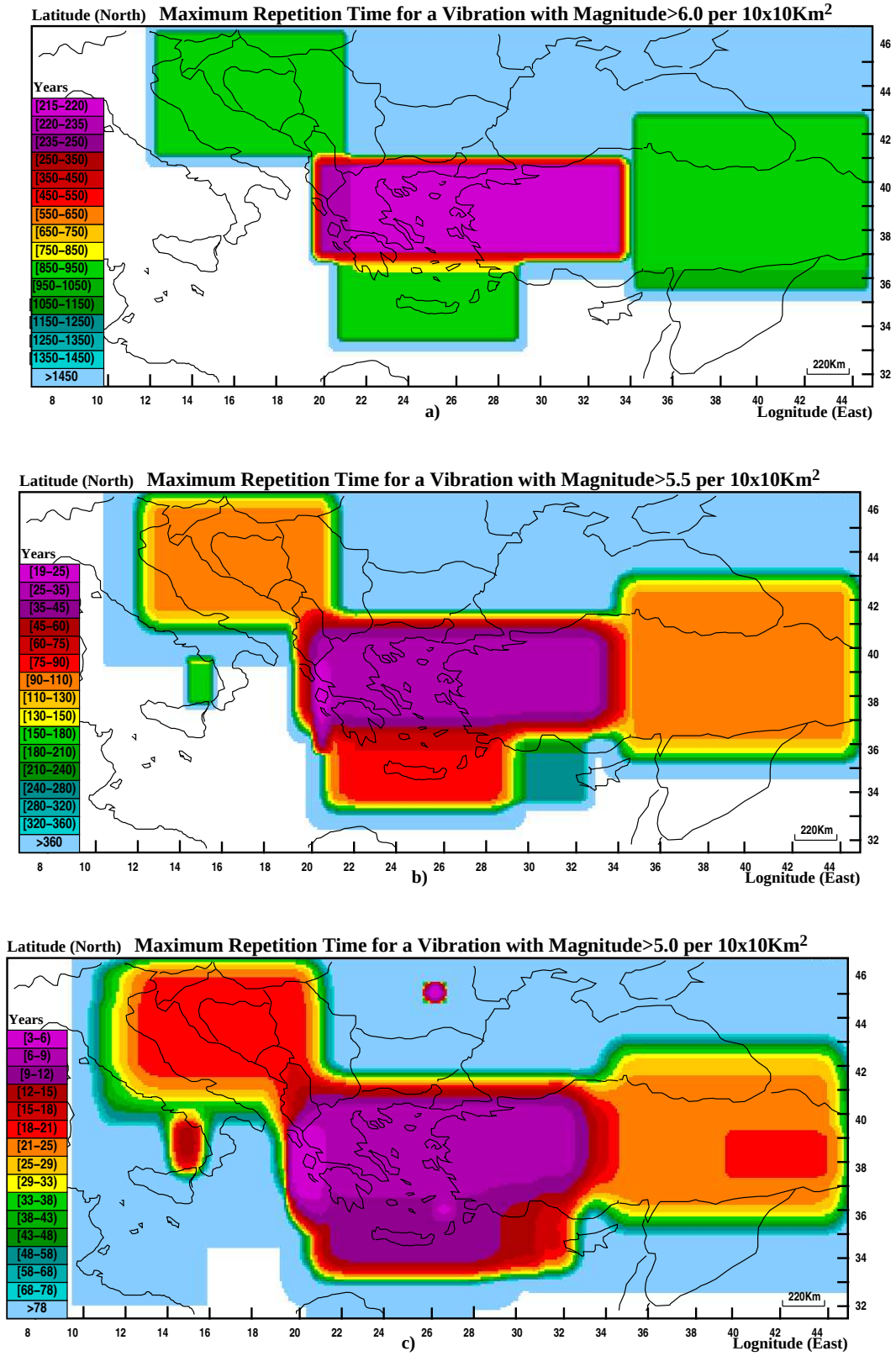


Figure 7.79: Hazard Maps for Vibrations with Magnitute Exceeding 5.0 on the Richter Scale

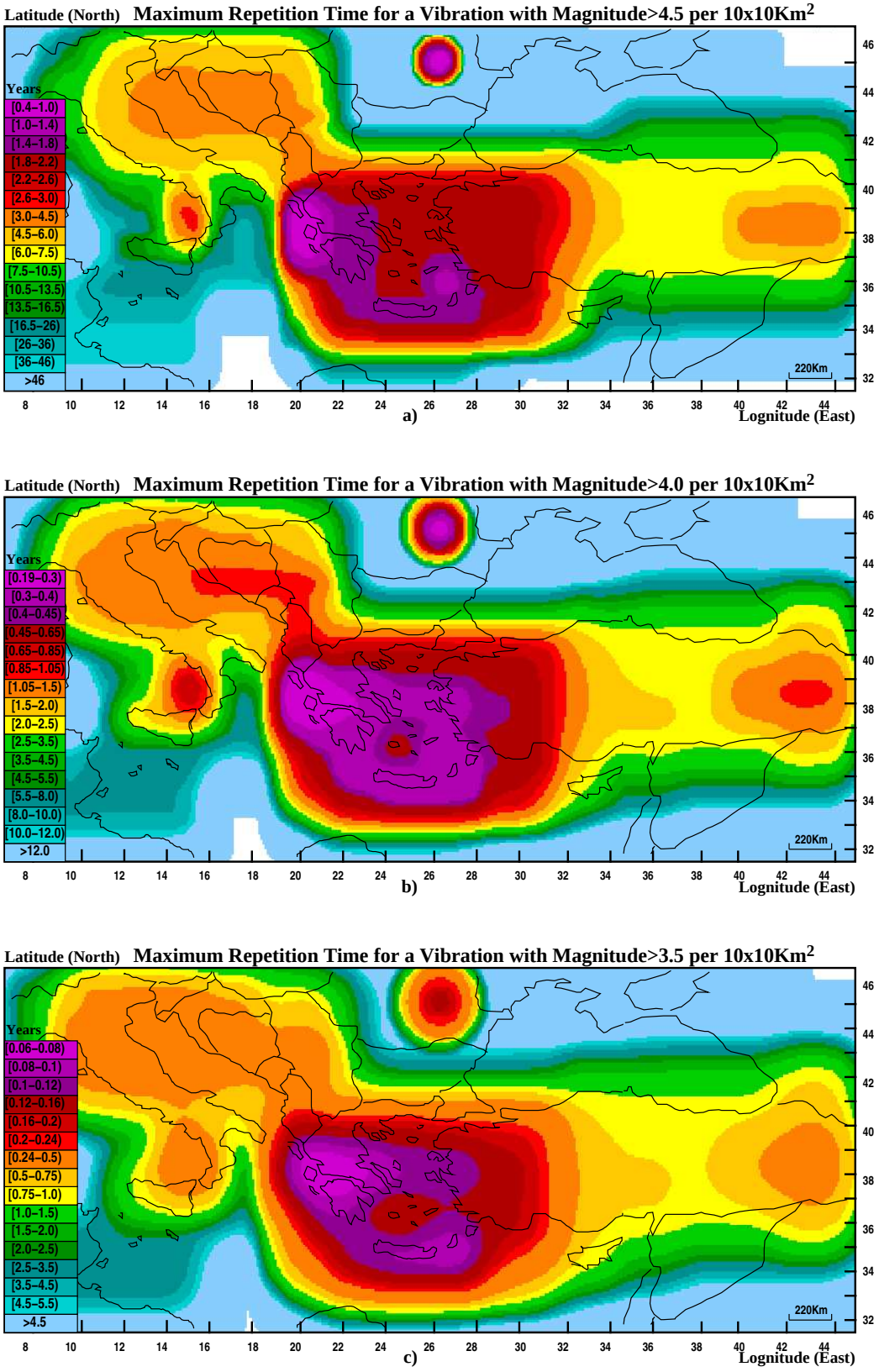


Figure 7.80: Hazard Maps for Vibrations with Magnitute Exceeding 3.5 on the Richter Scale

7.14 Evolution of High-Magnitude Seismicity

In this section we study the historic evolution of moderate-to-high-magnitude seismicity. The basis for such an analysis is the spatio-temporal distribution of events covered by *hazardous* rules, that is, rules whose upper bound of magnitude-gene exceeds an input threshold (i.e. 5.0≤Mag.). Reliable models describing the evolution of high-magnitude seismicity are essential tools in construction design for earthquake resistance [1]. Table 7.14 summarises the hazardous rules found by $\mathcal{NOCEA}$ for the *ANSS* catalogue. For illustrative purposes, rules threatening the same area are grouped together, although a rule may affect several areas, e.g. $\mathcal{R}_{177}$ extends from the Ionian Sea (west) to central Turkey (east).

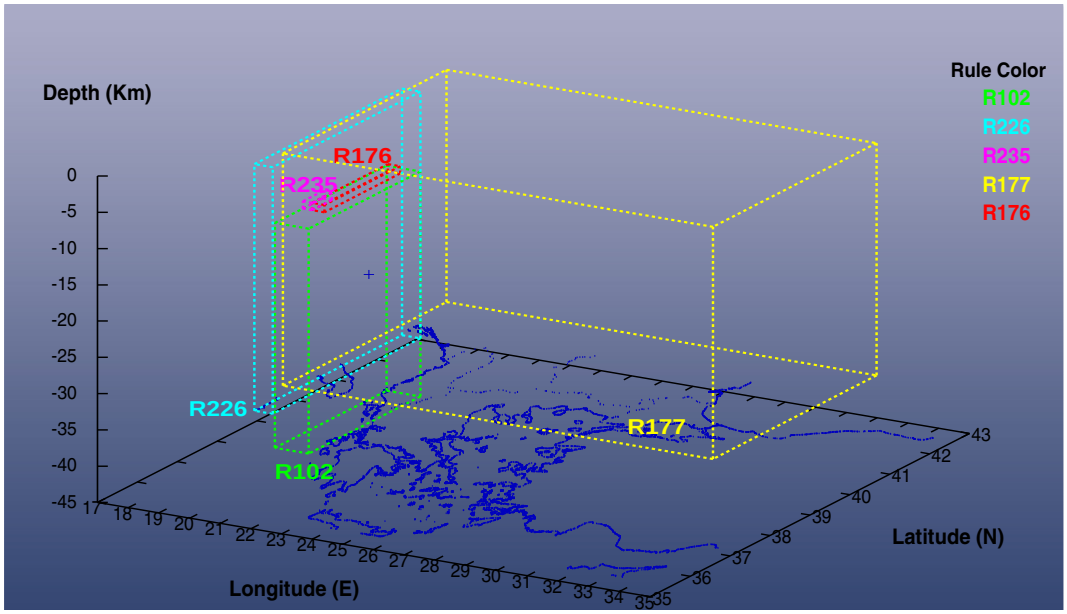
Rule	Type	Coverage	Density	Time	Longitude	Latitude	Depth	Magnit.
Ionian Sea								
$\mathcal{R}_{102}$	R-rule	69-0.19	7.9×10^{-6}		19.9-21.0	37.2-40.0	10-41	4.7-5.5
$\mathcal{R}_{226}$	R-rule	10-0.02	4.3×10^{-6}		20.4-21.0	36.3-40.0	0-32	5.5-5.7
$\mathcal{R}_{235}$	AS-rule	13-0.03	5.4×10^{-2}	29/08/82-12/01/84	19.9-20.2	37.9-38.4	9-10	4.8-5.2
$\mathcal{R}_{176}$	U-rule	49-0.14	1.6×10^{-3}	23/03/79-27/09/96	20.2-20.6	37.9-39.8	9-10	4.3-5.1
Coastal Northeastern Africa								
$\mathcal{R}_{230}$	R-rule	10-0.02	2.4×10^{-7}		20.8-28.0	32.0-33.8	5-33	4.8-5.2
$\mathcal{R}_{95}$	P-rule?	68-0.19	1.5×10^{-5}	17/05/84-19/04/04	7.00-15.9	32.8-36.9	9-10	3.0-5.2
Romania-Vrancea								
$\mathcal{R}_{220}$	R-rule	24-0.06	1.2×10^{-5}		26.1-26.8	45.4-45.9	105-172	4.8-5.5
$\mathcal{R}_{190}$	U-rule	36-0.10	1×10^{-7}	14/04/81-19/04/04	7.9-32.4	41.4-46.6	33-53	3.0-5.1
Northern and Southeastern Turkey								
$\mathcal{R}_{74}$	R-rule	158-0.45	3.5×10^{-7}		31.6-45.1	34.8-42.2	10-32	3.5-5.2
$\mathcal{R}_{164}$	U-rule	45-0.13	1.6×10^{-5}	06/11/77-19/04/04	31.2-45.1	36.9-42.0	9-10	4.7-5.2
$\mathcal{R}_{169}$	AS-rule	47-0.13	5.5×10^{-6}	15/02/63-06/08/80	39.6-45.1	37.9-39.9	33-52	4.4-5.2
$\mathcal{R}_{203}$	R-rule	49-0.14	6.7×10^{-6}		31.2-45.1	34-41.1	32-33	4.7-5.2
$\mathcal{R}_{218}$	R-rule	52-0.15	1.2×10^{-7}		34.2-45.1	35.9-43.5	4-37	5.2-6.2
$\mathcal{R}_{232}$	U-rule	15-0.04	1.9×10^{-7}	07/01/61-25/08/93	30.9-45.1	33.3-37.9	39-70	4.8-5.2
Greece, Aegean Sea, West Coastal Turkey								
$\mathcal{R}_{43}$	R-rule	400-1.15	2.4×10^{-6}		21.0-31.2	33.8-41.1	4-39	4.7-5.2
$\mathcal{R}_{52}$	R-rule	118-0.34	5.3×10^{-7}		21.0-32.9	34.1-41.7	2-41	5.2-5.7
$\mathcal{R}_{107}$	U-rule	63-0.18	1.2×10^{-6}	07/01/61-15/08/92	19.9-30.9	34.3-37.9	41-71	4.7-5.2
$\mathcal{R}_{177}$	R-rule	45-0.13	1.6×10^{-7}		19.9-33.9	37.4-41.5	2-33	5.7-6.6
$\mathcal{R}_{219}$	R-rule	21-0.06	2.7×10^{-7}		22.0-32.7	34.0-36.8	41-88	5.2-5.7
$\mathcal{R}_{221}$	R-rule	11-0.03	1.9×10^{-5}		26.5-27.2	36.3-36.9	144-171	4.8-5.2
$\mathcal{R}_{223}$	R-rule	11-0.03	1.9×10^{-7}		20.8-29.2	33.8-37.4	24-32	5.7-6.2
$\mathcal{R}_{233}$	R-rule	11-0.03	4.2×10^{-7}		20.6-28.8	34.7-38.0	75-94	4.8-5.2
Italy, Adriatic Sea, Dalmatian Coast and Albania								
$\mathcal{R}_{91}$	U-rule	71-0.20	2.5×10^{-5}	26/07/79-19/04/04	8.60-22.3	41.1-46.8	9-10	4.7-5.2
$\mathcal{R}_{100}$	U-rule	71-0.20	1.6×10^{-6}	12/01/84-20/01/96	7.00-23.1	41.1-45.8	24-32	3.0-5.1
$\mathcal{R}_{109}$	AS-rule	52-0.15	3.4×10^{-2}	18/02/76-26/10/76	12.8-13.4	46.0-46.6	32-33	3.0-5.1
$\mathcal{R}_{144}$	U-rule	39-0.11	8.6×10^{-7}	29/06/65-01/11/99	10.7-19.9	35.2-47.1	10-24	4.8-5.1
$\mathcal{R}_{145}$	U-rule	40-0.11	4×10^{-7}	06/07/62-26/11/90	11.0-21.0	40.0-47.0	11-33	5.1-5.7
$\mathcal{R}_{190}$	U-rule	36-0.10	1×10^{-7}	14/04/81-19/04/04	7.9-32.4	41.4-46.6	33-53	3.0-5.1
$\mathcal{R}_{222}$	R-rule	17-0.04	1.1×10^{-7}		12.5-21.4	41.5-47.0	4-33	5.7-6.2
$\mathcal{R}_{224}$	U-rule	35-0.10	2×10^{-5}	15/02/63-14/04/81	8.00-21.4	42.4-45.2	31-34	4.8-5.1
$\mathcal{R}_{227}$	R-rule	9-0.02	7.9×10^{-7}		14.6-15.8	38.2-40.2	233-295	5.0-5.7
$\mathcal{R}_{228}$	R-rule	27-0.07	4.3×10^{-7}		10.2-21.0	37.0-46.5	4-9	4.8-5.5
$\mathcal{R}_{229}$	R-rule	19-0.05	7.1×10^{-7}		11.8-19.9	36.5-41.1	26-46	4.8-5.1
$\mathcal{R}_{234}$	AS-rule	13-0.03	8.3×10^{-5}	18/02/76-26/07/79	9.30-22.5	41.1-47.0	9-10	4.9-5.1

Table 7.14: Hazardous Rules in the *ANSS* Earthquake Dataset

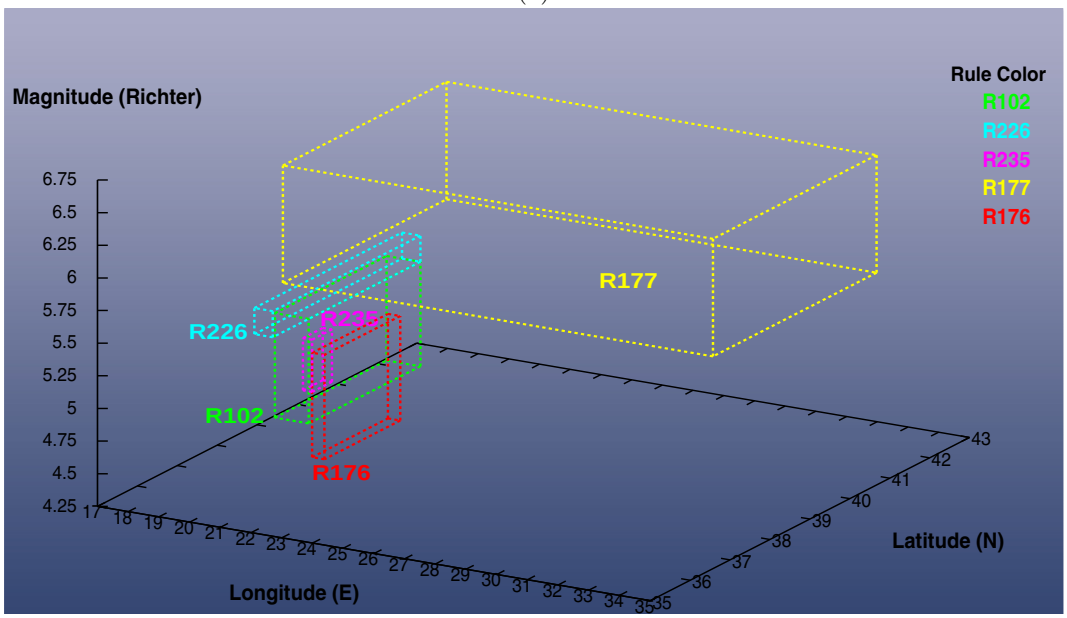
Although the area studied is very broad, most of the moderate-to-high magnitude seismicity related to the collision boundaries of the tectonic plates appears to be relatively stable, with an evident exception occurring in the mid-eighties in a region that extends from north central Italy, in the west, to the Adriatic Sea, Dalmatian Coast, Bulgaria and Romania to the east. Note that hazardous rules corresponding to high-magnitude aftershock activity are neglected from further analysis due to their extremely confined time window, e.g. $\mathcal{R}_{109}$ (6.5M, 06/05/76, Friuli, NE Italy), $\mathcal{R}_{235}$ (7.0M, 17/01/83, Kephalonia Island, Ionian Sea, Greece).

7.14.1 Ionian Sea

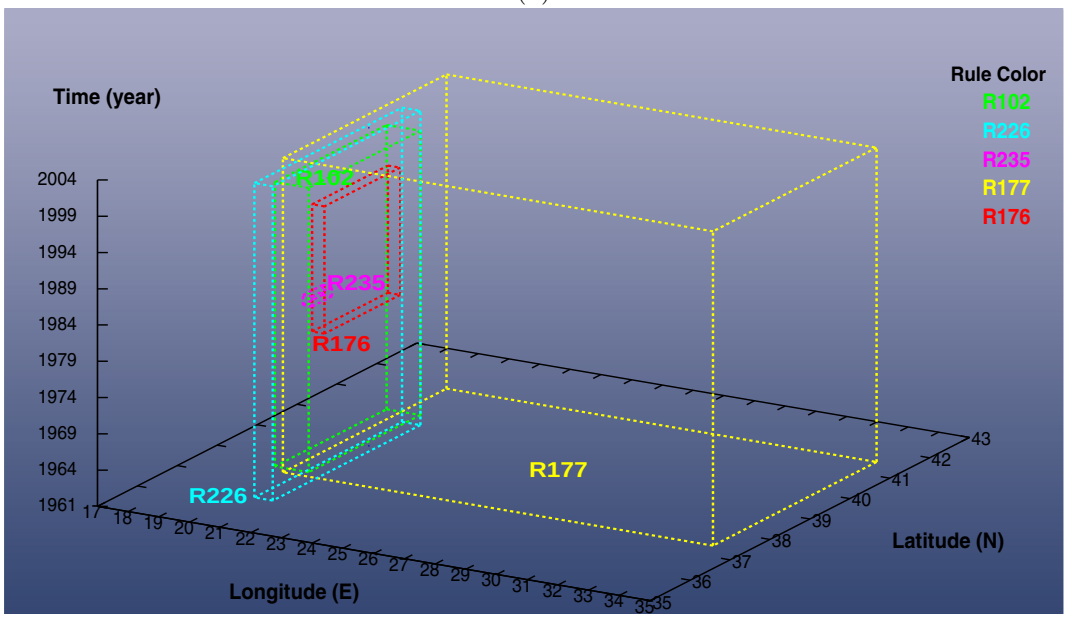
In the Ionian Sea, $\mathcal{NOCEA}$ discovered three `R-rules` ($\mathcal{R}_{102}$, $\mathcal{R}_{177}$ and $\mathcal{R}_{226}$), one `AS-Rule` ($\mathcal{R}_{235}$) and one `U-Rule` ($\mathcal{R}_{176}$) (see figures 7.81(a-b)). Both the spatio-temporal location and short lifetime of $\mathcal{R}_{235}$ indicate that this rule was part of the aftershock activity following the 17/01/83 destructive earthquake (7.0M) which occurred near Kephalonia Island, Ionian Sea. The type of $\mathcal{R}_{176}$ is ambiguous because *a*) it is spatially located in the vicinity of the 17/01/83 mainshock hypocenter *b*) it covers low-to-moderate magnitude events that happened for a limited time window (23/03/79 - 27/09/96), and *c*) it was activated prior to the 17/01/83 mainshock. Whether $\mathcal{R}_{176}$ was a precursory signal (`HP-rule`) or part of the aftershock activity or both is unknown to us. Notably the events covered by both $\mathcal{R}_{176}$ and $\mathcal{R}_{235}$ are concentrated in a thin (1km) slice at depth 9-10km, which is exactly at the same depth as the fault whose rupture caused the 17/01/83 mainshock. $\mathcal{R}_{226}$ is an extremely hazardous rule due to its high magnitude (5.5-5.7M) and relatively shallow depth (0-32km). $\mathcal{R}_{226}$ poses a continuous hazard with the recurrence interval of a strong earthquake (5.5<Mag.) being approximately 4.4 years. $\mathcal{R}_{102}$ is slightly deeper (10-41km) than $\mathcal{R}_{226}$ and can be viewed as the extension of the latter towards lower magnitudes (4.7-5.5M). Remarkably, the repetition time for an event with magnitude greater than 5.0 due to $\mathcal{R}_{102}$ is only 0.989 years. Finally, the source of the strongest events (5.7-6.6M) in the area is located by the southwestern part of rule $\mathcal{R}_{177}$ that is discussed in section 7.14.6.



(a)



(b)



(c)

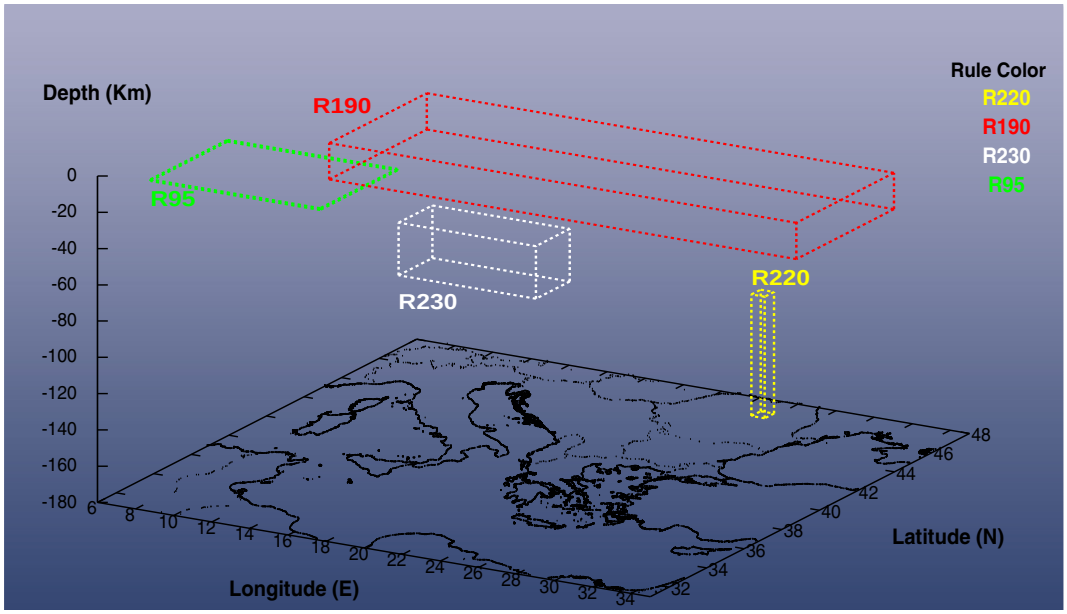
Figure 7.81: Hazardous Rules in the Ionian Sea

7.14.2 Coastal Northeastern Africa

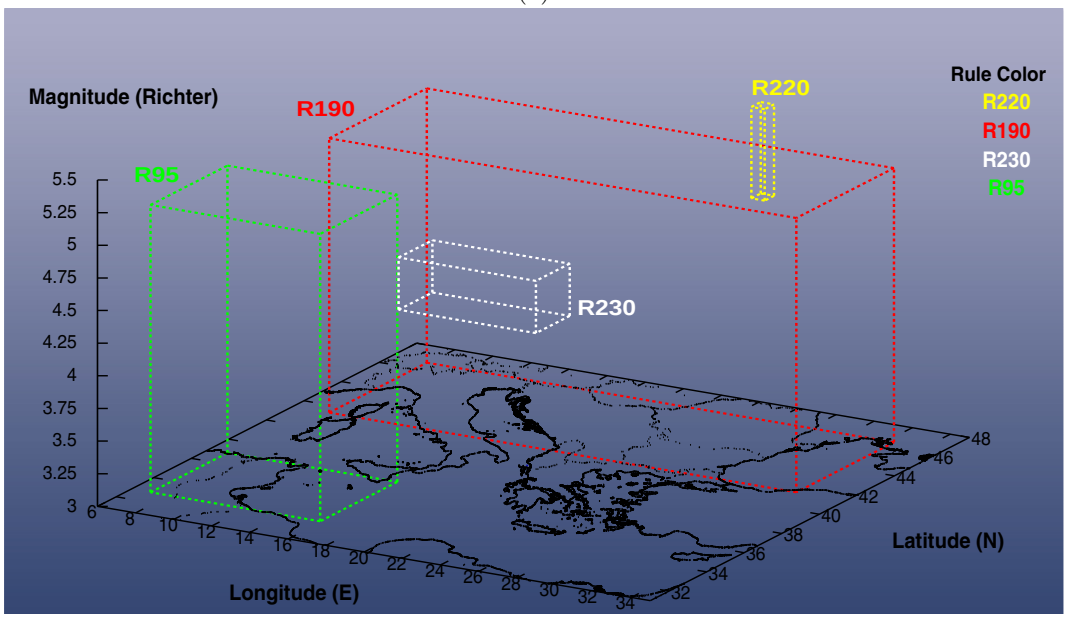
The coastal area of northeastern Africa is considerably less active compared to the other regions studied in our analysis. This region is mainly characterised by two hazardous rules with different properties (figure 7.82). The first seismogenic zone, an `R-rule` ($\mathcal{R}_{230}$), includes coastal Libya and the Mediterranean Sea south of Crete and can be viewed as an extension of the *Hellenic Arc* to the south. $\mathcal{R}_{230}$ describes shallow depth earthquakes (5-33km) of moderate magnitudes (4.8-5.2M) with the time interval between two events with magnitude greater than 5.0 being approximately 21.5 years per $250\times250\text{km}^2$. The second rule ($\mathcal{R}_{95}$) covers mainland Tunisia as well as the sea between Tunisia and Sicily. This region is relatively active since directly beneath it lies the Africa-Eurasia plate boundary [77]. $\mathcal{R}_{95}$ is characterised by its relatively recent activation 17/05/84 along a thin slice (9-10km) with uniformly distributed magnitudes in the range 3.0-5.2. Perhaps the most interesting observation, to the best of our knowledge, is that based on the recorded events in our catalogue, no major (5.5<Mag.) earthquake has been observed close to $\mathcal{R}_{95}$ up until the end of the catalogue (19/04/2004). The recent low-magnitude seismic excitation along this region naturally raises the question whether $\mathcal{R}_{95}$ can be interpreted as a precursory signal (`P-rule`) for a future major earthquake. Another possible explanation for the relatively recent seismic excitation of $\mathcal{R}_{95}$ could be the incompleteness of the catalogue itself.

7.14.3 Romania

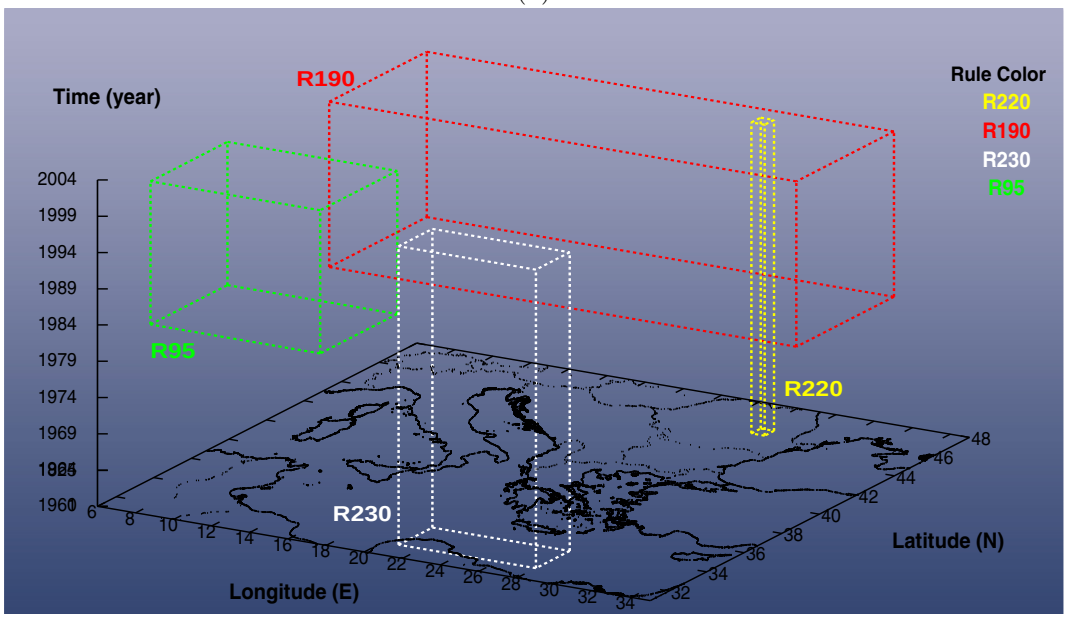
The highest seismic hazard in the North Balkans is attributed to a complex intra-continental collision process that is remarkably confined in the region of Vrancea, Romania. $\mathcal{NOCEA}$ located the source ($\mathcal{R}_{220}$) of regularly occurring strong earthquakes (4.8-5.5) deep under the surface at focal depths 105-172km (figure 7.82). This finding is in accordance with studies presented in [16]. Closer to the surface (33-53km) there is another layer ($\mathcal{R}_{190}$) (section 7.14.5) of lower seismicity (3.0-5.1) appearing consistently active only after 1981. In fact, $\mathcal{R}_{190}$ is not localised around Vrancea, but extends to the west up to the Italian peninsula.



(a)



(b)

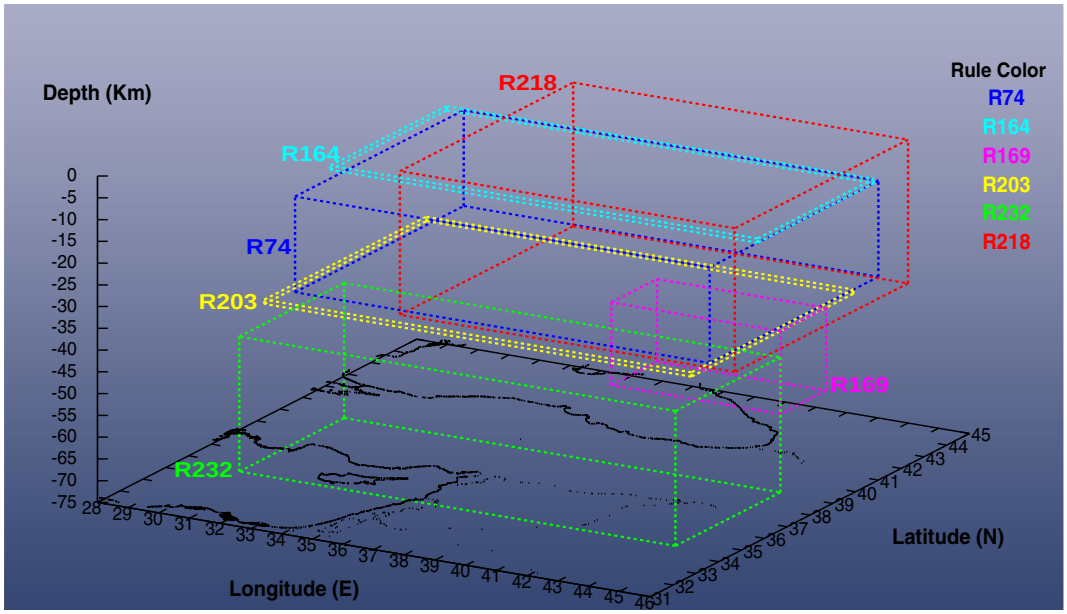


(c)

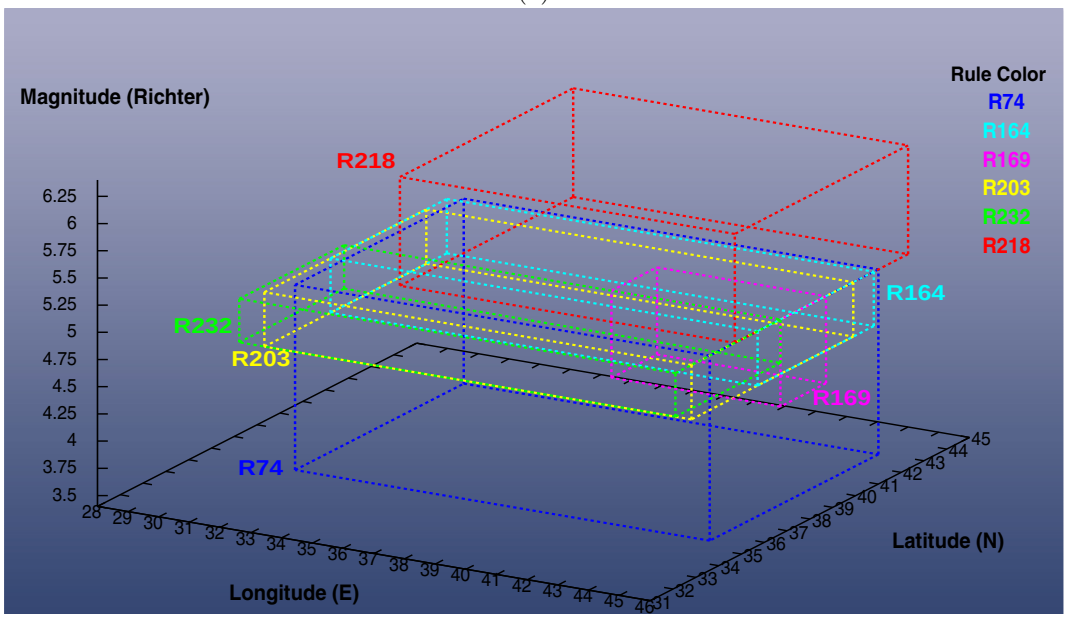
Figure 7.82: Hazardous Rules in Romania and Coastal-Northeastern Africa

7.14.4 Northern-Southeastern Turkey

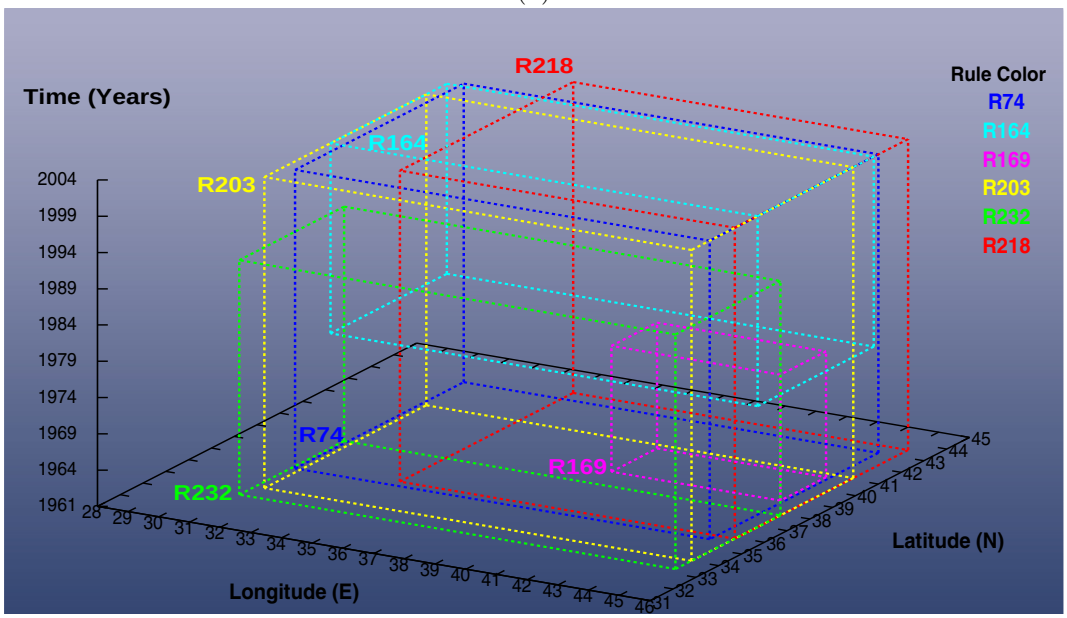
The most prominent tectonic features in Turkey are the *North Anatolian Fault* (*NAF*) in the north, and *East Anatolian Fault* (*EAF*) in the southeast [77]. NAF runs from about 30^{o}E up to its junction with EAF near Karliova at 41^{o}E. $\mathcal{NOCEA}$ discovered six hazardous rules ($\mathcal{R}_{74}$, $\mathcal{R}_{164}$, $\mathcal{R}_{169}$, $\mathcal{R}_{203}$, $\mathcal{R}_{218}$, and $\mathcal{R}_{232}$) affecting this region (see figure 7.83). $\mathcal{R}_{169}$ is located exactly along the junction of EAF and NAF near Karliova. It was active only from 15/02/63 to 06/08/80. During that period of time the area enclosed by $\mathcal{R}_{169}$ was experiencing a sequence of severe earthquakes, e.g. Varto (1966), Bingol (1971), Lice (1975), Caldiran-Muradiye (1977) [1]. In two occasions, Varto (1966) and Caldiran-Muradiye (1977)), the hypocenters of the mainshocks were located in the upper-depth bound of $\mathcal{R}_{169}$. Hence, $\mathcal{R}_{169}$ can be classified as an extended `AS-rule` that was generated by multiple mainshocks. $\mathcal{R}_{203}$, an `R-rule`, is in essence the regular activity generated by the EAF at the focal depth of (32-33km) and is perpendicular to $\mathcal{R}_{169}$ along the depth axis. At least three severe earthquakes at Varto (1966), Dinar (1995) and Adana (1998) had their hypocenters inside $\mathcal{R}_{203}$. $\mathcal{R}_{74}$, another `R-rule`, lies exactly above $\mathcal{R}_{203}$ along the depth axis and covers low-to-moderate magnitude events 3.5-5.2M generated from both NAF and EAF within focal depths 10-32km. $\mathcal{R}_{218}$ is an `R-rule` covering the moderate-to-high seismicity (5.2-6.2M) of EAF and Karviola junction faults. The hypocenter of events of $\mathcal{R}_{218}$ are more widespread compared to $\mathcal{R}_{203}$ (1-41km). The focal mechanisms enclosed by $\mathcal{R}_{218}$ generate an event with magnitude greater than 5.0, 5.5 and 6.0, on average, every 0.846, 1.209 and 4.231 years, respectively. $\mathcal{R}_{232}$ covers events that were generated from 07/01/61 to 25/08/93 by the extension of the EAF zone to the Mediterranean Sea in the Gulf of Iskenderun near Cyprus and Syria. Noticeably, nowadays the focal mechanisms of both $\mathcal{R}_{232}$ and $\mathcal{R}_{169}$ experience a period of quiescence. Perhaps the most evident change in the seismicity of this region occurred between 1977 and 1980 where relatively moderate magnitude (4.7-5.2M) earthquakes appeared to migrate gradually from Karviola junction ($\mathcal{R}_{169}$) to the west along a spatial pathway delineated by rule $\mathcal{R}_{164}$. $\mathcal{R}_{164}$ extends along NAF and EAF and is closer to the surface (9-10km) than $\mathcal{R}_{169}$ (33-52km).



(a)



(b)

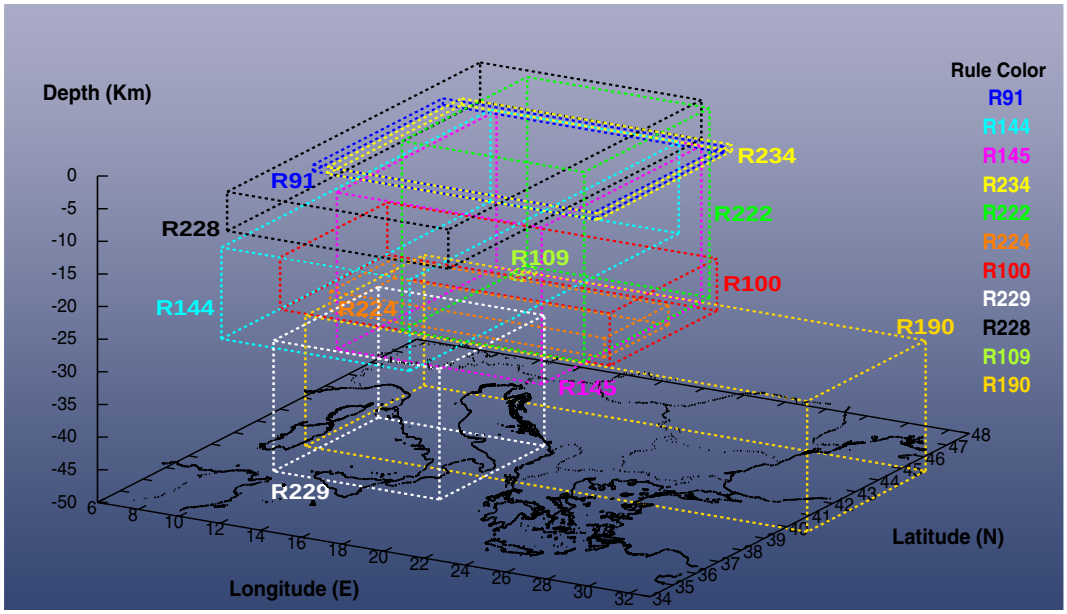


(c)

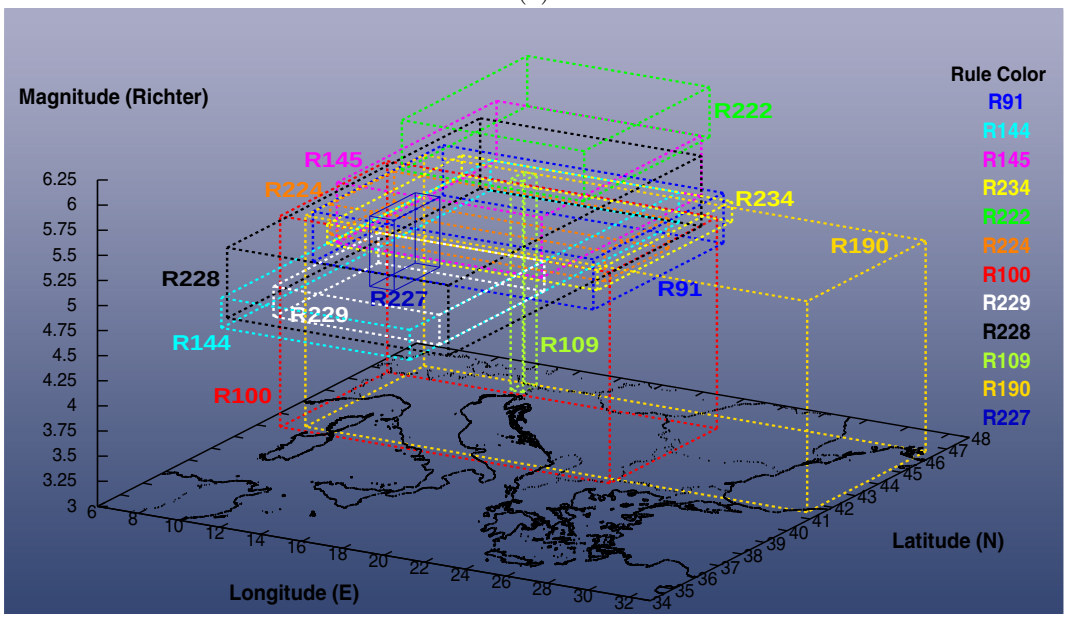
Figure 7.83: Hazardous Rules in Northern and Southeastern Turkey

7.14.5 Italy, Adriatic Sea, Dalmatian Coast and Albania

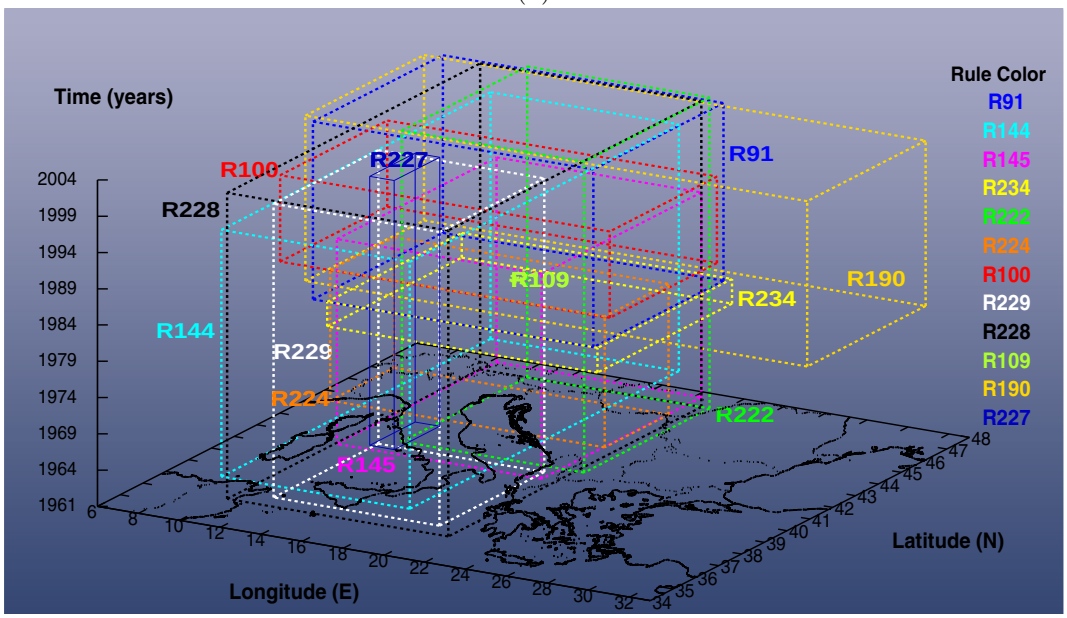
The active tectonics of the Adriatic Sea and the surrounding lands, including the Italian Peninsula, Dalmatian Coast and Albania, are mainly attributed to the continental collision between the Adriatic block and Eurasia, running along the Dalmatian Coast, in the east, and the Italian peninsula in the west [77]. Our results indicate that this region experiences a period of great instability regarding the focal mechanisms that rupture relatively strong earthquakes. $\mathcal{NOCEA}$ revealed twelve spatio-temporal zones ($\mathcal{R}_{91}$, $\mathcal{R}_{100}$, $\mathcal{R}_{109}$, $\mathcal{R}_{144}$, $\mathcal{R}_{145}$, $\mathcal{R}_{190}$, $\mathcal{R}_{222}$, $\mathcal{R}_{224}$, $\mathcal{R}_{227}$, $\mathcal{R}_{228}$, $\mathcal{R}_{229}$, and $\mathcal{R}_{234}$) of relatively homogeneous moderate-to-high seismicity, as shown in figures 7.84(a-c). The high magnitude seismicity (5.7-6.2M) in the north-central part of this region is captured by a relatively shallow (4-33km) `R-rule` ($\mathcal{R}_{222}$), which is apparently the extension of $\mathcal{R}_{177}$ (the most hazardous rule in the Aegean Sea and surrounding lands) to northwest along the collision zone between Eurasia and Africa. The focal mechanisms of $\mathcal{R}_{222}$ trigger an earthquake with magnitude greater than 5.7 on average every 2.588 years. Further down the magnitude scale (4.8-5.5) appears a very shallow (4-9km) `R-rule`, namely $\mathcal{R}_{228}$, which extends along the entire window (including the Ionian Sea as well) as opposed to $\mathcal{R}_{222}$ that covers the north-central part of the window. The area of the Calabrian Arc, Southern Apennines and southeastern Tyrrhenian Sea is regularly affected by a deep (233-295km) `R-rule`, namely $\mathcal{R}_{227}$, which covers events with magnitudes reaching up to 5.7 (5.0-5.7M). This region is further threatened by two recently ($\approx$1975) activated rules, $\mathcal{R}_{204}$ (3.0-5.0M), $\mathcal{R}_{231}$ (4.8-5.0M) adhering spatially to $\mathcal{R}_{227}$. Noticeably, these rules are located along a deep fault with an almost diagonal southeast-northwest direction, running approximately from 15^oE, 38^oN 82km ($\mathcal{R}_{204}$) to 12^oE, 40^oN 487km ($\mathcal{R}_{231}$). $\mathcal{R}_{229}$, another `R-rule`, covers low-to-moderate events (4.8-5.1) whose hypocenters are scattered within a crustal layer of length 20km (26-46km) from Sicily and the South Apennines in the west to the southwestern Albanian Coast in the East. Clearly, $\mathcal{R}_{109}$ and $\mathcal{R}_{234}$ are `AS-rules` due to their short lifetime. For the remaining hazardous rules there is a rather complex sequence of events at different depths, whose interpretation is beyond the scope of this thesis.



(a)



(b)



(c)

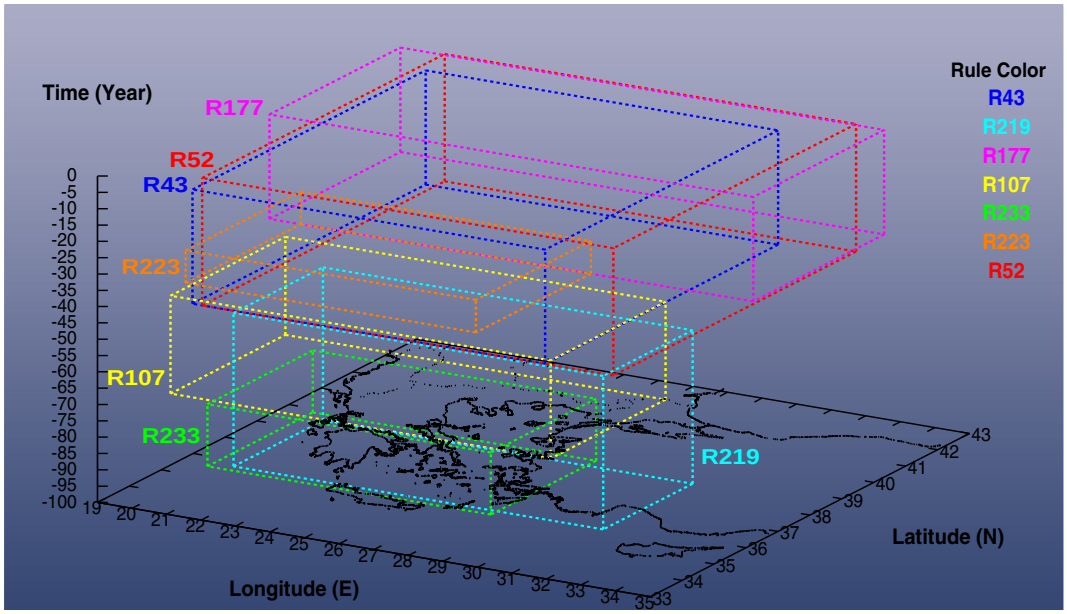
Figure 7.84: Hazardous Rules in Italy, Adriatic Sea, Dalmatian Coast and Albania

7.14.6 Greece, Aegean Sea, and West Coastal Turkey

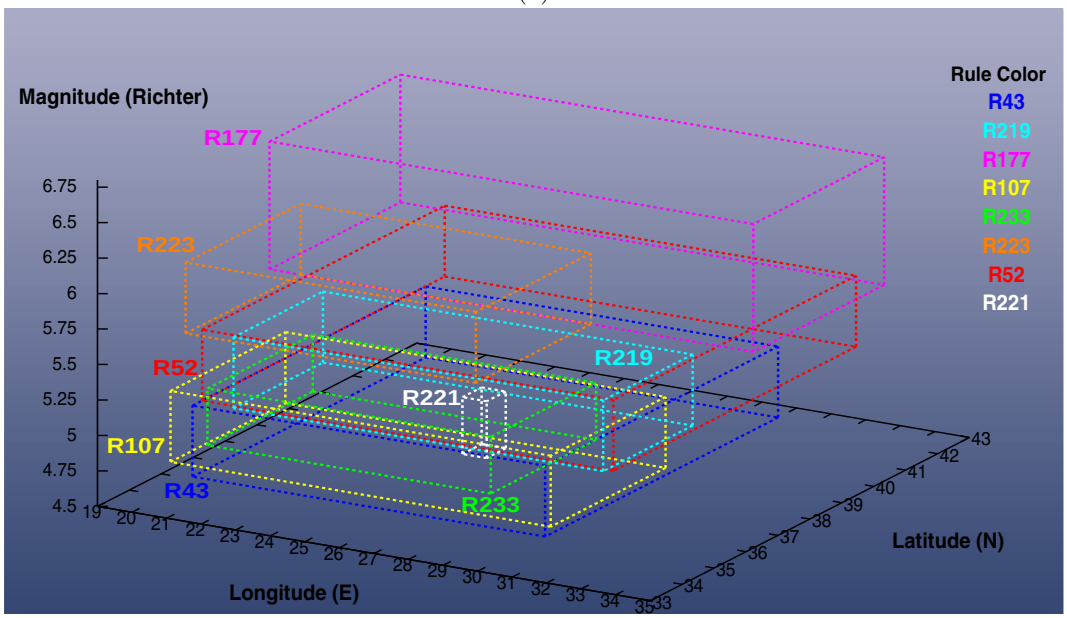
The Aegean Sea and the surrounding lands, including mainland Greece and West Coastal Turkey, appear to be seismically the most stable among all regions included in our study. This region is characterised by both shallow and intermediate depth strong earthquakes whose spatio-temporal distribution has changed little over the last 43 years (1961-2004) (see figures 7.85(a-c)). In particular, among the eight hazardous rules ($\mathcal{R}_{43}$, $\mathcal{R}_{52}$, $\mathcal{R}_{107}$, $\mathcal{R}_{177}$, $\mathcal{R}_{219}$, $\mathcal{R}_{221}$, $\mathcal{R}_{223}$, and $\mathcal{R}_{233}$) only $\mathcal{R}_{107}$ appears inactive after 15/08/92, while the others exhibit a typical `R-rule` behaviour. In fact, even though $\mathcal{R}_{219}$ is classified as a `R-rule`, its focal mechanisms have recently shown signs of seismic recession since $\mathcal{R}_{219}$ does not extend beyond 21/10/1998. Note that $\mathcal{R}_{107}$ and $\mathcal{R}_{219}$, both covering faults of the *Hellenic Arc*, have a large overlap in the spatio-temporal subspace while touching along the magnitude dimension. Therefore, this indicates a decreasing trend in the moderate (4.7-5.5M) seismicity of the *Hellenic Arc* generated at intermediate depths 41-88km.

From figures 7.85(a-b) it is evident that shallow-to-intermediate depth strong earthquakes have epicentres that are located mostly in the southern Aegean Sea along the *Hellenic Arc*. $\mathcal{R}_{177}$ is formed by a patch of extremely hazardous and very shallow (2-33Km) earthquakes with magnitudes reaching up to 6.6 degrees on the Richter scale (5.7-6.6). $\mathcal{R}_{177}$ "affects" the Ionian Isles, north-central Greece, *Northern Aegean Trough* and the northwestern coast of Turkey. It is worth mentioning that some of the most destructive earthquakes which occurred in this region (Greece: Thessaloniki-1978, Korinthos-1981, Aigion-1995, Kozani-1995, Athens-1999, Turkey: Gediz-1970, Dinar-1995, Marmara Sea-1999) [1, 2] were originated either from inside $\mathcal{R}_{177}$ or had their hypocenters located in close vicinity to $\mathcal{R}_{177}$. On average, the faults enclosed by $\mathcal{R}_{177}$ generate an earthquake with magnitude greater than 5.5 and 6.0 every 0.978 and 1.464 years, respectively. In the southern Aegean, the source of high-magnitude seismicity (5.7-6.2), are faults that are mainly located along the *Hellenic Arc*, south of Crete, at focal depths of

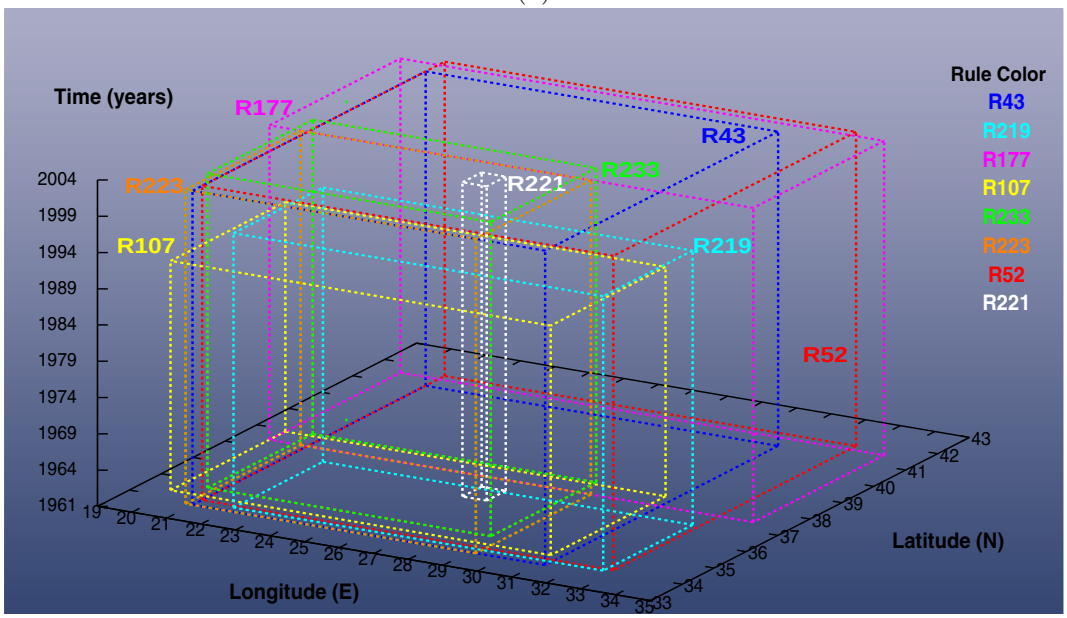
24-32km, which is exactly the region enclosed by $\mathcal{R}_{223}$. Further down in the magnitude dimension (5.2-5.7) the spatial distribution of events is slightly different. More specifically, events with magnitudes 5.2-5.7 form clearly two seismogenic zones. The first zone, that is $\mathcal{R}_{52}$, extends throughout the entire Aegean Sea, mainland Greece and west Turkey at focal depths of 2-41km, while the second zone, that is $\mathcal{R}_{219}$, is an orthogonal dipping of $\mathcal{R}_{52}$ towards depths of 41-88km along the *Hellenic Arc*. The less hazardous rules in terms of magnitude (4.7-5.2M) are $\mathcal{R}_{43}$, $\mathcal{R}_{107}$ and $\mathcal{R}_{233}$, forming a similar trend as far as the spatial distribution is concerned. In particular, $\mathcal{R}_{43}$, one of the most highly populated rules, has its 400 events scattered throughout the entire Aegean Sea and the surrounding lands at focal depths ranging from 4 to 39km. $\mathcal{R}_{107}$ and $\mathcal{R}_{233}$ are deeper extensions of $\mathcal{R}_{43}$ occurring only along the *Hellenic Arc* at depths of 41-71km and 75-94km, respectively. Finally, $\mathcal{R}_{221}$ is an extremely confined rule in every spatial dimension. $\mathcal{R}_{221}$ covers events whose epicentres are located in the southeastern part of the *Volcanic Arc* near Kos, while their hypocenters are concentrated in a confined spatial volume deep (144-171km) in the crustal layer.



(a)



(b)



(c)

Figure 7.85: Hazardous Rules in Greece, Aegean Sea, and West Coastal Turkey

7.15 Effectiveness and Efficiency Evaluation of $\mathcal{NOCEA}$ on Artificial and Real-World Datasets

In this section we evaluate the efficiency and effectiveness of $\mathcal{NOCEA}$ by conducting a variety of experiments using a mixture of synthetic and real-world datasets. Experiments with similar purposes are also reported in section 6.10 (chapter 6), but here the datasets are far more complex, being based on real-world data. It is worth noting that in both sets of experiments, $\mathcal{NOCEA}$ behaves similarly as far as the scalability and accuracy are concerned.

In particular, the goals of the experiments are to assess:

- **Efficiency:** Determine scalability with respect to:
  - *Size of the database (i.e. number of records)*
  - *Dimensionality of the data*
  - *Average dimensionality of the clusters*
  - *Number of processors in a parallel architecture*
- **Effectiveness:** Test if $\mathcal{NOCEA}$ recovers correctly and accurately clusters that are embedded in some subspaces of a high dimensional space.

7.15.1 Synthetic-Real-World Data Generator

Various data generators have been recently proposed to produce clusters embedded in subspaces of high dimensional spaces for evaluation purposes [5, 7, 74, 83]. The main disadvantage of these approaches is that the structure of the resulting clusters is both artificial and far less complex than real world cases. Additionally, despite these techniques being parameterised in the number of the desired clusters, the evaluation studies in [5, 7, 74, 83] were based on a limited number of clusters, i.e. 5. To address these limitations the generator described in [5] was modified to enforce the creation of complex, real-world type structures consisting of numerous, i.e. 237, clusters based on the discovered knowledge from the *ANSS* earthquake catalogue.

Recall from section 7.7 that $\mathcal{NOCEA}$ partitioned the 34593 points of the five-dimensional *ANSS* seismic catalogue into 237 homogeneous rules, while the level of background noise is approximately 23% (7911 points). The main idea of the proposed generator is to create clusters that can be approximated by one rule by embedding each *ANSS* rule into higher dimensional spaces. The first five dimensions of a cluster in the augmented spaces are directly inherited from the coordinates of the corresponding rule. By doing this we generate synthetic datasets with realistic characteristics such as *a*) real-world structural complexity *b*) numerous clusters *c*) clusters with diversity in size, density, geometry, and data coverage, and *d*) real-world non-uniformly distributed noise.

Dimensionality of Clusters

Let d denote the desired number of dimensions without considering the five features of the earthquake dataset. Hereafter the latter will be referred to as e-features. The range of values was set to [0, 100] for all artificially generated attributes. Similar to generators that are described in [5, 83] and in section 6.5, the number of relevant dimensions associated with a given rule is determined by a realisation of a Poisson random variable with mean μ with the additional constraint that this number must be at most d.

Determination of the Bounded Dimensions

The next step is to determine the bounded dimensions associated with each rule. Following the recommendation of [5, 83], when generating the *(i+1)*th rule, approximately half of its bounded dimensions are chosen from among the bounded dimensions of the ith rule, while the remaining are generated at random. This technique was introduced to model the fact that often different clusters share subsets of correlated dimensions. To ensure no dependency on the order by which rules are processed during this stage, the list of rules is randomly permutated.

Variance of the Bounded Dimensions

If a rule is to be embedded in k dimensions the algorithm selects k variance values independently on each other. Given a spread parameter r and a scale factor s' uniformly distributed in [1, s] the variance is then $(r*s')^2$, where r=s=2 [5].

Centers of the Bounded Dimensions

The coordinate of the *central* or *anchor* point along each bounded dimension of a rule is randomly drawn from [0, 100].

Number of Points

In [5] the number of points (N_i) assigned to the ith cluster is proportional to a realisation of an exponential random variable. However, this technique results in all clusters having reasonably similar size [83]. To force imbalance in the cluster sizes, the authors in [83] proposed computing initially (N_i) in a similar manner, but then setting for each $i \leq k/2$, $N_i = \delta N_i$ and $N_{i+k/2} = N_{i+k/2} + (1\text{-}\delta)N_i$, for $\delta \in \{0.2, 0.33, 0.5\}$. However, to the best of our knowledge, the most comprehensive studies [5, 7, 74, 83] evaluating the performance of clustering algorithms in high dimensional space were based on a limited number of clusters, i.e. 5, which is often not a realistic choice to simulate real-world examples. In contrast, our generator creates a fairly large number of clusters, i.e. 237, with a rich diversity in size, geometry, and data coverage. Consequently, the resultant clusters are significantly sparser in high dimensional spaces compared with other approaches. This introduces an additional challenge: to distinguish between clusters and produce correct cluster descriptors. In fact, the number of points assigned to a given rule in the augmented space is proportional to the coverage of its parental rule in the original space. Since we are interested in transmitting the original structures in the augmented space, the number of points of each original rule is simply multiplied by an integer replication factor to obtain the size of the corresponding rule in the full dimensional space.

Types of Data Distribution

One of the main goals of this section is to investigate $\mathcal{NOCEA}$'s performance under different distributions of the points in the subspace of bounded dimensions. Currently our generator supports the following types of data distributions:

- *Uniform:* For those attributes that define the subspace where the rule is embedded, the value is drawn independently at random from the uniform distribution within the range $[\overline{x}\text{-}3\sigma, \overline{x}+3\sigma]$, where $\overline{x}$ and σ are the mean

value and standard deviation in each dimension, respectively.

- *Normal:* For the ith bounded dimension of a rule, the coordinates of the points projected onto dimension i follow a normal distribution with mean $\overline{x}$ at the respective coordinate of the center point and variance determined as explained earlier.

- *Uniform_Ellipse:* Similar to *Uniform* distribution, the algorithm samples independently at random k values, one for each bounded dimension in the respective range, with the additional constraint that this point must be enclosed by the kth dimensional hyper-ellipse, centred on the anchor point of that rule and with a 3σ-length axis in each bounded dimension.

- *Normal_Ellipse*: The only difference from a *Uniform_Ellipse* is that the points in each dimension are drawn independently from a normal distribution as described earlier.

Generating Data

Having completed the previous stages the algorithm generates the points associated with a given rule as follows: Recall that in total there will be (d+5) dimensions in the full-dimensional space. Let (x_1, x_2, x_3, x_4, x_5) be the coordinates of a given point P in the subspace defined by the e-features of the rule $\mathcal{R}$ that covers that point. For each point P of $\mathcal{R}$ the algorithm creates randomly an appropriate number of new points P' in the close vicinity of P such that the coordinates of the new points in the subspace defined by the e-features are ($N(x_1, w_1)$, $N(x_2, w_2)$, $N(x_3, w_3)$, $N(x_4, w_4)$, $N(x_5, w_5)$), where w_i is the bin width in the ith dimension ($i \in [1, 5]$), while $N(a, b)$ is a normal distribution centred on a with standard deviation b. The coordinates of points in the non-bounded dimensions are then generated independently at random within [0, 100]. Finally, for all the remaining attributes the algorithm randomly selects one type of distribution and then it appropriately creates the coordinates of points depending on the data distribution as described earlier. As far as the noise is concerned, the procedure is identical. Note that, unlike other generators, our approach does not create a perfectly uniform noise, since in the subspace defined by the e-features, the distribution of

noisy points is directly inherited from the earthquake catalogue.

In all the experiments reported in sections 7.15.2-7.15.6 twenty independent and randomly initialised runs were performed for all datasets. The reported measurements of execution time and recall-accuracy are based on an arithmetic average of the clustering results over the twenty different random runs.

7.15.2 Scalability With Database Size

Figure 7.86 depicts the scalability of $\mathcal{NOCEA}$ as the size of the database increases from 0.5 to 25 million records. Each dataset has 20 dimensions including the five e-features and 237 single-rule clusters embedded on average in some 10-dimensional subspace as described in section 7.15.1. Note that most of the original rules were already embedded in the first five dimensions (e-features). Additionally, recall from section 7.15.1 that the level of noise is approximately 23% of the database size. For this set of experiments, the distribution of points for the bounded dimensions of all clusters follow a uniform distribution. $\mathcal{NOCEA}$ successfully locates all input clusters within the course of 50 generations, on average.

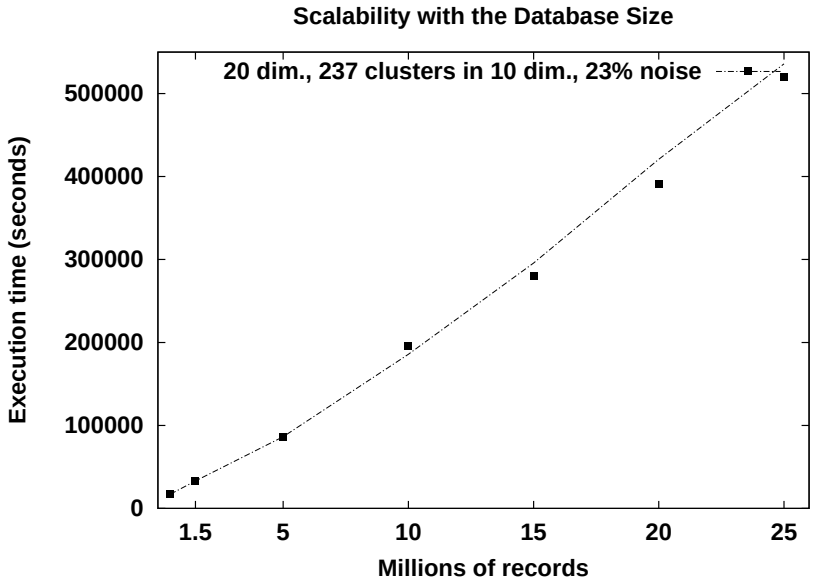


Figure 7.86: Execution Time *vs.* Number of Records

Figure 7.86 shows that the execution time scales almost *linearly* with the

database size. This is because, given the relatively low dimensionality of the datasets, the execution time is dominated by the construction of the density histograms which, in turn, is a task with linear complexity in the database size. Similar scalability behaviour was reported in other studies [7, 74]. However these studies were based on small numbers of clusters, usually five. Performance could be improved by replacing the current linear data-search mechanism that is employed by our system with a faster hyper-rectangular query mechanism (*kd*-trees) [26].

7.15.3 Scalability With Dimensionality of Data

Figure 7.87 shows the scalability of $\mathcal{NOCEA}$ as the dimensionality of the feature space increases from 20 to 200. In this series of experiments, each dataset has 3*34593=103779 records distributed over 273 clusters being embedded in some 10-dimensional subspace.

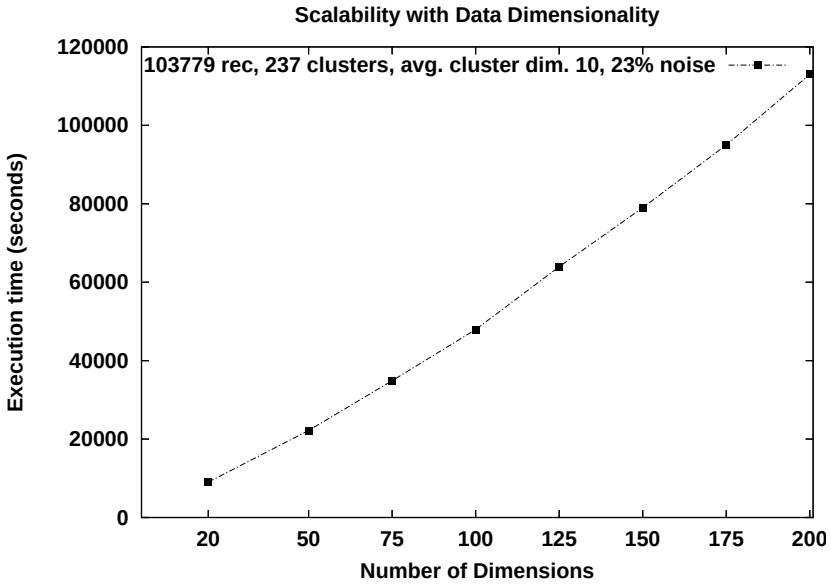


Figure 7.87: Execution Time *vs.* Number of Dimensions

Clearly, the curve exhibits a *super-linear* trend. This behaviour is mainly due to the fact that for a given rule-set, $\mathcal{NOCEA}$ must build at least one density histogram for each rule in every dimension. Additionally, as the dimensionality increases the application of the genetic operators becomes increasingly more expensive due to the constraint of producing individuals without overlapping rules.

Note that both tasks (construction of density histograms and constraint checking) are of linear complexity with data dimensionality.

7.15.4 Scalability With Cluster Dimensionality

Figure 7.88 shows $\mathcal{NOCEA}$'s scalability as the average dimensionality of hidden clusters increases from 10 to 50 in a 100-dimensional space. In each case, the dataset has 3*34593=103779 records containing as before 273 clusters, and 23% noise.

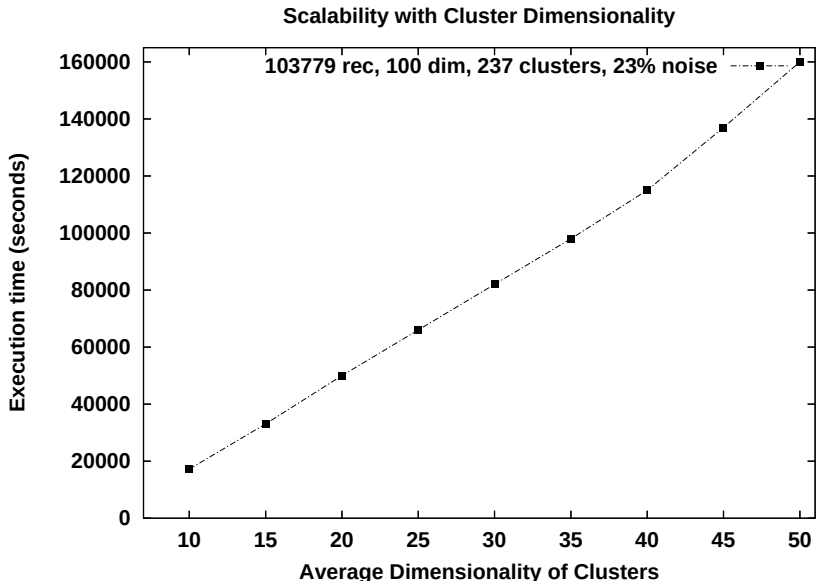


Figure 7.88: Execution Time *vs.* Avg. Cluster Dimensionality

The *super-linear* speed-up in the execution time with the dimensionality of the hidden clusters is explained as follows: the higher the dimensionality of the hidden clusters the more likely it is for the evolutionary search operators to produce non-homogeneous candidate rules along the bounded dimensions. Hence, extra repairing operations are required to obtain a feasible (homogeneous) rule-set. The computational overhead introduced by the additional repairing operations for a given cluster is generally proportional to the dimensionality of that cluster. Additionally, as the dimensionality of hidden clusters increases, more space becomes available (uncovered) for the mutation operator to grow existing rules

or to produce new ones. Despite the fact that no access to the database is required when mutating a genome, the cost of applying the mutation operator may be substantial, especially for high dimensional spaces or numerous clusters. In fact, not surprisingly, given the relatively large number of clusters (237) and the moderate size of the datasets used in this section, when the dimensionality of hidden clusters exceeds the value of 30, mutating a single genome becomes more expensive than evaluating a genome by a factor of 0.6.

7.15.5 Scalability With Task Parallelism

In this section we study the scalability of $p\mathcal{NOCEA}$ (section 5.13) under various *task parallelisation* schemes.

$p\mathcal{NOCEA}$ supports task parallelism for the most expensive genetic operations such as, repairing-evaluation (E), mutation (M), recombination (R) and generalisation (G). Figure 7.89 compares the speedups achieved for various parallelisations of $p\mathcal{NOCEA}$ using a synthetic dataset that was generated as described in section 7.15.1. The 100-dimensional dataset contains 242151 points (23% noise) forming 237 uniform-clusters being embedded on average in some 30-dimensional space.

Each parallelisation scheme is characterised by a combination of capital letters denoting the genetic operations that were parallelised under that scheme. Let p be the number of processors in $p\mathcal{NOCEA}$. The speedup of $p\mathcal{NOCEA}$ with p processors over the sequential $\mathcal{NOCEA}$ with one processor is defined as t_1/t_p, where t_1 is the execution time of the sequential single-processor $\mathcal{NOCEA}$ while t_p is the execution time of $p\mathcal{NOCEA}$ with p processors [43].

All measurements have been performed on a network of homogeneous *PE*s (Processing Elements) (Intel(R) Xeon(TM) CPU 3.06GHz, 512 KB cache, 2GB of RAM) running Fedora Core 2.0. The remote PEs were connected with the coordinator machine through a 100Mb/s Ethernet cable while the communication protocol was RMI (Remote Method Invocation) being implemented in $Java^{TM}2$ Standard Edition 1.4.2_05.

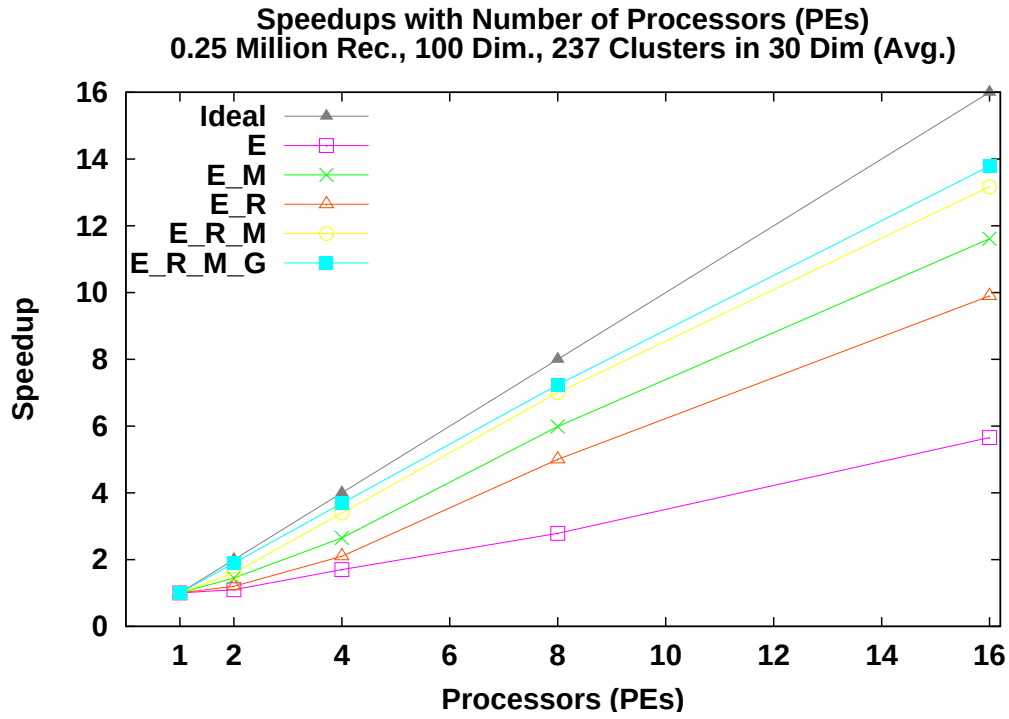


Figure 7.89: *Speedups* for Various Parallelisations of $p\mathcal{NOCEA}$

Not surprisingly, the fully-parallelised version of $p\mathcal{NOCEA}$, that is $E\_R\_M\_G$, gives the best results, with a speedup of 13.789 on 16 PEs. Regarding the other schemes, it is unsafe to draw any general conclusion since the relative cost between the genetic operations is heavily dependent on the dataset itself. Hence, despite the coordination-communication overhead introduced due to task parallelism, a fully-parallelised $p\mathcal{NOCEA}$ is strongly recommended.

7.15.6 Effectiveness Evaluation

The goal of this section is twofold:

- to assess the accuracy of $\mathcal{NOCEA}$ in recovering the boundaries and subspace in which each cluster has been embedded
- to investigate the quality of the clustering results under various data distributions

In all the above experiments, regardless of the data distribution, each discovered cluster was correctly located in its original subspace using a single hyper-rectangular rule. Experiments on artificial datasets with similar results are also reported in section 6.10.3. Since by definition $\mathcal{NOCEA}$ seeks relatively homogeneous clusters, the very low density tails (if any) of a uni-dimensional histogram must be separated from the main part of the distribution. This is exactly the case of non-uniform clusters, such as normal and ellipsoid. Bearing in mind that in our experiments the data were distributed among a multitude (237) of clusters, the data coverage of the homogeneous hyper-rectangular "core" of a non-uniform cluster may not always exceeds the fixed sparse threshold T_s. Consequently, even though $\mathcal{NOCEA}$ has the ability to locate such clusters, the pruning of non-sparse regions eliminates those candidate rules capturing the core of very small clusters. This problem is proportionately exaggerated with the dimensionality of the clusters because the number of points being missed out in the tails of a bounded dimension is added to the total loss.

As a result of separating the very low density boundaries of a cluster from its denser core, $\mathcal{NOCEA}$ missed some (on average 5 clusters over twenty different and randomly initialised runs) non-uniform clusters of very small coverage, e.g. 0.02%, especially when the dimensionality of the hidden clusters was relatively high, e.g. greater than 10. Clearly, more research in the future is required to tackle this problem.

7.16 Summary

This chapter has presented a thorough evaluation of $\mathcal{NOCEA}$ on a challenging real-world problem, that is a detailed study of the seismicity along the African-Eurasian-Arabian tectonic plate boundary.

The experimental results suggest that $\mathcal{NOCEA}$ meets the important DM clustering criteria as follows. $\mathcal{NOCEA}$ discovered highly homogeneous and complex geometry clusters that were reported in the interpretable form of disjoint and axis-aligned rectangular rules; the output has minimised for ease of comprehension.

$\mathcal{NOCEA}$ discovered rules/clusters of arbitrary density, geometry, and data coverage. It has been shown that $\mathcal{NOCEA}$ is able to perform effective subspace clustering on a particularly sparse dataspace. Representative examples of clusters with irrelevant features were reported and interpreted. The experiments showed that $\mathcal{NOCEA}$ self-adjusted well to the characteristics of the dataset, and did not require auxiliary information regarding the number and dimensionality of clusters.

This Chapter has also investigated the efficiency and effectiveness of $\mathcal{NOCEA}$ under varying database size, data dimensionality, cluster dimensionality and data distribution, on synthetic datasets being based on the *ANSS* earthquake dataset. Additionally, it has been shown that task parallelism of the most expensive genetic operators has the potential to reach a speed up of 13.8 on 16 processors.

From an earthquake analysis point of view, $\mathcal{NOCEA}$ revealed complex seismicity patterns that can aid seismologists to better understand the phenomenon. Finally, it has been shown that it is possible to exploit the discovered knowledge to compile precise hazard maps, and to interpret the evolution of high magnitude seismicity whose social impact is the most severe.

Part IV

Conclusion

Chapter 8

Conclusions

Capsule

This Chapter summarises the thesis results, and concludes that evolutionary algorithms have great potential as search mechanisms to effectively and efficiently mine high quality clustering knowledge from massive and high dimensional databases. The limitations of the work are discussed, and a number of further research directions are identified.

8.1 Summary

8.1.1 Research Challenges

Driven by advances in data collection and storage, increasingly large and complex datasets are being stored. Parkinson's law of data, a corollary of Parkinson's law states that "*...data expands to fill the space available for storage...*". In fact, data doubles about every year, but useful information seems to be decreasing [31]. The area of KDD has arisen over the last decade to address this problem, and it has become not only an important research area, but also one with large potential in the real world. Intelligent DM techniques are being developed to semi-automate the process of mining nuggets of hidden knowledge, and extract them in forms that can be readily utilised in areas such as decision support and forecasting. Clustering - perhaps the most challenging descriptive DM task - seeks to identify homogeneous clusters of data points based on the values of their attributes.

Based on a substantial literature survey (Chapter 2) we argue that current clustering techniques do not address all of the key criteria (section 2.3.2) for DM clustering adequately, although considerable work has been done in addressing each requirement separately. Furthermore, after decades of claiming that EAs are powerful optimisation techniques well-suited for problems with large and complex search spaces, no EA-based system has adequately exploited the advantages (section 2.4.11) of EAs to tackle realistic large-scale clustering problems. The survey also shows that most clustering techniques suffer from the infamous *curse of dimensionality phenomenon* (section 2.3.2): due to the sparsely filled feature space the data are "lost in space" and the effectiveness of clustering algorithms that depend critically on distance or density measures degenerate rapidly with increasing dimensionality [55].

8.1.2 A Novel Clustering Methodology

The fundamental question addressed by this thesis is: *can a stochastic search cluster large high-dimensional datasets, and extract knowledge that conforms to the important requirements for DM clustering*? Experimental results on both artificial datasets (chapter 6) and real-world (chapter 7) datasets lead us to conclude that it can.

The thesis has developed a novel three-phase clustering methodology (chapters 3 - 5) that utilises the intrinsic search parallelism and stochastic nature of EAs to efficiently mine *disjoint* and *axis-aligned hyper-rectangular* clustering *rules* with *homogeneous* data distribution from *massive* and *high* dimensional *numerical* databases.

Quantisation: Firstly, a sophisticated quantisation algorithm ($\mathcal{TSQ}$) (Chapter 4) imposes a uniform multi-dimensional grid onto the dataspace to reduce the search combinations. $\mathcal{TSQ}$ quantises the dataspace using a novel statistical analysis that reflects the local data distribution. It determines an appropriate grid resolution that enables the discrimination of clusters, while preserving accuracy and acceptable computational cost.

Clustering: Secondly, a novel EA ($\mathcal{NOCEA}$) (Chapter 5) discovers high quality clustering rules using several novel semi-stochastic genetic operators, an integer-valued encoding scheme reflecting the quantised dataspace, and a simple data coverage maximisation fitness function. The evolutionary optimisation in $\mathcal{NOCEA}$ has a simple well-defined goal, which, however, provides enough quality information to drive the selective pressure of the EA: *maximise the total point coverage with an arbitrary-length feasible rule-set.* A feasible solution (section 5.4.1) comprises rules that are disjoint, syntactically valid, non-sparse, and have homogeneous data distribution. Specialised genetic operators, i.e. recombination (section 5.8), mutation (section 5.10), and homogeneity (section 5.7) maintain feasibility of individuals in the population, while generalisation (section 5.9) improves knowledge comprehensibility and reduces computation by making rule-sets as short and generic as possible.

Both $\mathcal{TSQ}$ and $\mathcal{NOCEA}$ rely on a novel statistical analysis ($\mathcal{UDA}$) (Chapter 3) identifying flat density regions ($\mathcal{U}$-regions) in univariate smooth histograms. Neighbouring $\mathcal{U}$-regions co-existing at different density levels are of great importance as they indicate the existence of distinct cluster structures in higher dimensional spaces; $\mathcal{U}$-regions help to generate accurate cluster "signatures", as often their boundaries coincide with the edges of the actual clusters.

Knowledge Simplification and Cluster Formation: Thirdly, a post processing simplification phase performs subspace clustering (section 5.11), and assembles the clusters (section 5.12).

Task Parallelism: The thesis has also explored task parallelism for several genetic operations to improve scalability when the data to be mined is massive (section 5.13).

8.1.3 Thesis Achievements

The thesis has demonstrated that it is possible to deliver the following desiderata for DM clustering using the proposed methodology:

- Effective treatment of high dimensionality and exceptional resistance to the curse of dimensionality; precise discrimination of clusters even in very sparse high-dimensional spaces (e.g. 200 dimensions) (sections 6.10.3 and 7.15).
- End-user comprehensibility/interpretability of the clustering results (sections 6.3, 7.7.4, 7.11.1, and 7.12).
- Ability to discover clusters embedded in arbitrary subspaces of high dimensional data (sections 6.10, 7.11, and 7.15.6).
- Linear scalability with database size (sections 6.10.2 and 7.15.2), and both data (sections 6.10.2 and 7.15.3) and cluster (sections 6.10.2 and 7.15.4) dimensionality.
- Substantial potential for task parallelism (section 5.13) achieving a speed up of 13.8 on 16 processors (section 7.15.5).
- Ability to discover highly-homogeneous clusters of arbitrary density (sections 6.6, 7.7.5 and 7.8), geometry (sections 6.4, 7.7.4, 7.9, and 7.12), and data coverage (section 7.10).
- Insensitivity to order of data input (section 6.7) and initialisation (section 6.8).
- Substantial resistance to uniform background noise (section 6.5).
- Minimal requirements for *a priori* knowledge (e.g. automatic determination of the optimal number of clusters and the subspace where each cluster is embedded, on the fly) and no presumptions of any canonical distribution for the input data (section 6.9).

8.2 Limitations

The current work has the following limitations, although most could be addressed in further work.

- Like any EA, $\mathcal{NOCEA}$ is slower and requires more space compared to conventional single-solution clustering techniques, as it evolves multiple solutions.
- The current implementation of $\mathcal{NOCEA}$ stores the entire dataset in the main memory. Compounding this limitation, the entire dataset must be replicated in the local memory of every PE. However, future work could include investigating data distribution among different processors or secondary storage to handle arbitrarily large and high dimensional datasets.
- The thesis has explored coarse-grained task parallelism with minimal communication coordination overhead where a single genetic operation constitutes a thread (a sequential unit of computation that is entirely executed in a single processor) (section 5.13). However, with the current coarse-grained parallel architecture no speedup improvement is possible when the number of available processors exceeds the total number of threads.
- To enable the discovery of homogeneous clusters the repair operator has been designed to separate the central densely-populated body (uniform core) of a cluster from its low density boundaries. This means that some peripheral cluster points are naturally "lost" (sections 6.10.2 and 7.15.6). The total loss is proportional to the dimensionality of the cluster and inversely proportional to the kurtosis (i.e. point concentration around the center) of the cluster points along the bounded dimensions. Hence, non-uniform clusters with small data coverage may be missed by $\mathcal{NOCEA}$ as the data coverage of the uniform core of such clusters may not always exceed the sparsity level (section 5.3).
- Sparse parts of arbitrary-shaped clusters sticking out from the backbone of such clusters are lost due to the elimination of sparse rules (section 5.6.1).
- Finally, $\mathcal{NOCEA}$ can not cope with categorical (nominal) attributes.

8.3 Future Research Directions

There are several avenues to extend this research and address the limitations identified in the previous section.

Arbitrary-Oriented Rules: The first avenue is to enhance the knowledge representation language (section 5.3) by including the capability to rotate the hyper-rules. Evolving arbitrary-oriented rules allows the approximation of non-convex clusters with fewer rules, but there is a price to be paid, i.e. an increase in the size of the search space as more search combinations are possible.

Alternative Geometric Rules: Another possible avenue is to introduce more complex geometric shapes, such as hyper- spheres, ellipses, trapezoids, or polyhedra. Again, one must carefully balance the trade-off between the expressive power of the representation language and the size of the search space (section 5.6.1. Additionally, more advanced and thus expensive, genetic operators are required to manipulate such complex structures.

Adaptive Grids: The third avenue to be explored is the construction of non-uniform grids where the bin width in each dimension varies depending on the local data distribution. Adaptive grids will allow $\mathcal{NOCEA}$ to delineate cluster boundaries more accurately.

Data Parallelism: Many real-world clustering problems are intrinsically multi-dimensional and massive in size. Hence it may not always be feasible to fit the entire dataset in the main memory of a single machine. To handle large amounts of data it is vital to explore data parallelism with data distributed among different processors [43, 44], and the use of secondary storage/in-memory DBMS.

Fine Grain Task Parallelism: Advanced genetic operators may produce substantial computations, especially for high dimensional datasets with a multitude of clusters (section 7.15.5). This thesis has explored coarse grain task parallelism, i.e. at a high level of abstraction where individual tasks like mutation, recombination, generalisation, and repairing, are assigned to different processors,

but each task is entirely executed in a single processor. Further investigation might explore finer grain parallelism by enabling several processing units to work simultaneously on the same individual task.

Advanced Data Structures: It may be possible to improve scalability by replacing the current, linear-scan data mechanism with a more sophisticated and faster data structure, e.g. kd-trees [26].

Island Model EA: A very promising avenue for future research would be to convert the *single-population* $\mathcal{NOCEA}$ to an *island-model* parallel EA [12]. This can be simply done by geometrically decomposing the feature space $\mathcal{F}$ into non-overlapping neighbourhoods (*islands*), and instantiating an ordinal $\mathcal{NOCEA}$ within each island. Local $\mathcal{NOCEA}$s will run independently of each other against the portion of the dataset assigned to each island. The complete solution to the clustering problem is assembled by combining the best solution from each island.

The anticipated gains from an island-model $\mathcal{NOCEA}$ are twofold: *a*) more flexible search that can be easily adapted to the local characteristics of the data, e.g. adaptive grid resolution, and *b*) low-communication data and task parallelism.

Evolution of Overlapping Partitions: Future research might explore the evolution of individuals where rule overlapping is permitted (see section 5.6.1). This idea reduces the complexity of the genetic operators, enables approximating arbitrary-shaped clusters with fewer and more generic rules, and eliminates the loss of the sparse parts of non-convex clusters (see section 8.2). However, there are several drawbacks that are discussed in detail in section 5.6.1.

Minimisation of the Loss of Peripheral Points: Finally, future research must tackle the problem of missing peripheral cluster points (see section 8.2). One possible solution might be to allow lower density tails to be included in the main part of the distribution if there are not other clusters in close proximity. However, the formation of clusters (section 5.12) becomes more complicated as two adjacent rules of similar density with large touch do not necessarily indicate the existence of an arbitrary-shaped cluster.

Bibliography

[1] Advanced National Seismic System (ANSS), URL: http://www.anss.org/.

[2] U.S. Geological Survey, URL: http://www.usgs.gov/.

[3] Schweizerischer Erdbebendienst (SED) - Swiss Seismological Service, URL: http://www.seismo.ethz.ch/.

[4] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973:420–431, 2001.

[5] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J. Park. Fast algorithms for projected clustering. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(2):61–72, 1999.

[6] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proc. of the 2000 ACMSIGMOD International Conference on Management of Data (SIGM'99)*, volume 29(2), pages 70–81. ACM Press, 2000.

[7] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 94–105, 1998.

[8] M. Ankerst, M. M. Breunig, H-P. Kriegel, and J. Sander. *OPTICS*: Ordering points to identify the clustering structure. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGM'99)*, volume 28,2 of *SIGMOD Record*, pages 49–60. ACM Press, 1999.

[9] G. P. Babu and M. N. Murty. A near-optimal initial seed selection in k-means algorith using a genetic algorithm. *Pattern Recognition Letters*, 14:763–769, 1993.

[10] T. Bäck. Evolution strategies: An alternative evolutionary algorithm. In *Artificial Evolution*, pages 3–20. Springer, 1996.

[11] T. Bäck, D. B. Fogel, and T. Michalewicz (Eds.). *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing and Oxford University Press, 2000.

[12] T. Bäck, D. B. Fogel, and T. Michalewicz (Eds.). *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing and Oxford University Press, 2000.

[13] I. D. Banitsiotou, T. M. Tsapanos, V. M. Margaris, and P. M. Hatzidimitriou. Estimation of the seismic hazard parameters for various sties in greece using a probabilistic approach. *Natural Hazards and Earth System Sciences*, (4):399–405, 2004.

[14] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, New Jersey, 1961.

[15] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

[16] C. A. Benetatos and A. Kiratzi. Stochastic strong ground motion simulation of the intemediate depth earthquakes: the cases of the 30 may 1990 vrancea (romania) and of the 22 january 2002 karpathos island (greece) earthquakes. *Soil Dynamics and Earthquake Engineering*, (24):1–9, 2004.

[17] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[18] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbour" meaningful? In *Proc. 7th Int. Conf. Data Theory*, volume 1540, pages 217–235. LNCS Springer Verlag, 1999.

[19] K. Blekas and A. Stafylopatis. Real-coded genetic optimization of fuzzy clustering. In *Fourth European Congress on Intelligent Techniques and Soft Computing - EUFIT'96*, volume 1, pages 461–465. Verlag Mainz, 1996.

[20] E. Bonsma, M. Shackleton, and R. Shipman. Eos - an evolutionary and ecosystem research platform. *BT Technology Journal*, 18(14):24–31, 2000.

[21] B. Bozkaya, J. Zhang, and E. Erkut. An effective genetic algorithm for the p-median problem. Technical Report Research Report No. 97-2, Research Papers in Management Science, Department of Finance and Management Science, Faculty of Business, University of Alberta, Canada, 1997.

[22] E. Cantu-Paz and C. Kamath. *On the use of evolutionary algorithms in data mining. In Abbass, H., Sarker, R., and Newton, C. (Eds.) Data Mining: a Heuristic Approach , pp. 48-71.* Hershey, PA: IDEA Group Publishing, 2002.

[23] C-H Cheng, A. W. Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 84–93. ACM Press, 1999.

[24] I. Csiszar and J. Korner. *Information Theory: Coding Theorems for Discrete Memoryless System*. Academic Press, 1981.

[25] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[26] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry Algorithms and Applications*. Springer-Verlag, Berlin Heidelberg, 2000.

[27] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[28] K. Deb and D. E. Goldberg. A comparison of selection schemes used in genetic algorithms. In *Proc of Foundations of Genetic Algorithms 1 (FOGA-1)*, pages 69–93, 1991.

[29] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Duxbury Press, 4th edition, 1995.

[30] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[31] M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003.

[32] M. Ester, H-P Kriegel, J.Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[33] M. Ester, H-P. Kriegel, J. Sander, and X. Xu. Density-connected sets and their application for trend detection in spatial databases. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*. AAAI Press, 1997.

[34] V. Estivill-Castro. Hybrid genetic algorithms are better for spatial clustering. In *Proc. of the Pacific Rim International Conference on Artificial Intelligence (PRICAI'00)*, pages 424–434, 2000.

[35] V. Estivill-Castro and A. Murray. Spatial clustering for data mining with genetic algorithms. Technical Report FIT-TR-1997-10, Faculty of Information Technology, Queensland University of Technology, September 01 1997.

[36] V. Estivill-Castro and A. Murray. Hybrid optimization for clustering in data mining. Technical Report 2000-01, Department of Computer Science and Software Engineering, University of Newcastle, 23 February 2000.

[37] B. S. Everitt. *Cluster Analysis*. Edward Arnold, 1993.

[38] E. Falkenauer. *Genetic algorithms and grouping problems*. Wiley, 1998.

[39] P. Fänti, J. Kivijärvi, T. Kaukoranta, and O. Nevalainen. Genetic algorithms for large-scale clustering problems. *The Computer Journal*, 40:547–554, 1997.

[40] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in knowledge discovery and data mining*. AAAI Press/The MIT Press, 1996.

[41] L. J. Fogel. Autonomous automata. *Industrial Research*, 1962.

[42] J. J. Fortier and H. Solomon. Clustering procedures. *In Multivariate Analysis*, pages 439–506, 1966. New York.

[43] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorightms*. Springer-Verlag, 2002.

[44] A. A. Freitas and S. H. Lavington. *Mining Very Large Databases with Parallel Processing*. Kluwer Academic Publishers, Boston, 1998.

[45] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[46] A. Ghozeil and D. B. Fogel. Discovering patterns in spatial data using evolutionary programming. In *Proc. of 1996 the First Annual Conference on Genetic Programming*, pages 521–527. MIT Press, 1996.

[47] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[48] S. Guha, R. Rastogi, and K. Shim. *CURE*: An efficient clustering algorithm for large databases. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 73–84. ACM Press, 1998.

[49] L. O. Hall, I. B. Özyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Trans. on Evolutionary Computation*, 3(2):103–112, 1999.

[50] L. C. Hamilton. *Modern Data Analyis: A First Course in Applied Statistics*. Brooks/Cole, 1990.

[51] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[52] J. Han, M. Kamber, and A. K. Tung. *Spatial Clustering Methods in Data Mining: A Survey.In Miller, H. and Han, J. Geographic Data Mining and Knowledge Discover*. Taylor and Francis, 2001.

[53] D. J. Hand. *Construction and Assessment of Classification Rules*. Wiley, 1997.

[54] J. Hartigan and M. Wong. Algorithm as136: A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[55] A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. of the 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 506–517. Morgan Kaufmann, 1999.

[56] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. 1998 International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 58–65, 1998.

[57] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[58] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[59] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comp. Surveys*, 31(3):264–323, 1999.

[60] R. E. Jensen. A dynamic programming algorithm for cluster analysis. *Operations Research*, 17:1034–1057, 1969.

[61] G. Karypis, E-H. Han, and V. Kumar. *Chameleon*: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[62] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[63] V. I. Keilis-Borok. Intermediate-term earthquake prediction. *Proc. of the National Academy of Sciences*, (93):3748–3755, 1996.

[64] A. Kiratzi and C. B. Papazachos. Active crustal deformation from the azores triple junction to middle east. *Tectonophysics*, 000000(243):1–24, 1995.

[65] J. Kivijärvi, P. Fänti, and O. Nevalainen. Efficient clustering with a self-adaptive genetic algorithm. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics (SCI2000)*, pages 241–246, 2000.

[66] K. Krishna and M. N. Murty. Genetic k-means algorithm. *Systems, Man and Cybernetics, Part B, IEEE Transactions*, 29(3):433–439, 1999.

[67] G. F. Luger and W. A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving (3nd edition)*. Addison Wesley Longman, Inc., 1998.

[68] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Statist., Prob.*, pages 281–297, 1967.

[69] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.

[70] L. Meng, Q. H. Wu, and Z. Z. Yong. A faster genetic clustering algorithm. In *Real-World Applications of Evolutionary Computing, EvoWorkshops 2000: EvoIASP, EvoSCONDI, EvoTel, EvoSTIM, EvoROB, and EvoFlight, Edinburgh, Scotland, UK, April 17, 2000, Proceedings*. Springer, 2000.

[71] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.

[72] B. L. Milenova and M. M. Campos. O-cluster: Scalable clustering of large high dimensional data sets. In *Proc. of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, pages 290–297. *IEEE* Computer Society, 2002.

[73] C. A. Murthy and N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17(8):825–832, 1996.

[74] H. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive datasets. In *Proc. of the 1st SIAM ICDM*, Chicago IL, USA, 2001.

[75] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. of 1994 Int. Conf. on Very Large Data Bases (VLDB'94)*, pages 144–155, 1994.

[76] S. Openshaw and PJ Taylor. The modifiable areal unit problem. In N. Wrigley and R. J. Bennet, editors, *Quantitative Geography: A British View*. Routledge and Kegan Paul, London, 1981.

[77] C. B. Papazachos and A. Kiratzi. A detailed study of the active crustal deformation in the aegean and surrounding area. *Tectonophysics*, 000000(253):129–154, 1996.

[78] Y. Park and M. Song. A genetic algorithm for clustering problems. In *Proc. of the 3rd Annual Conference of Genetic Programming*, pages 568–575. Morgan Kaufmann, 1998.

[79] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, 2004.

[80] M. J. Pazzani. Comprehensible knowledge discovery: Gaining insight from data. In *First Federal Data Mining Conference and Exposition, Washington, DC*, pages 73–80, 1997.

[81] M. J. Pazzani. Knowledge discovery from data? *IEEE Intelligent Systems*, 15:10–12, 2000.

[82] J. Periaux and G. Winter, editors. *Genetic Algorithms in Engineering and Computer Science*. John Wiley, 1995.

[83] C. M. Procopiuc, M. Jones, P. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, June 3–6, 2002,Madison, WI, USA*, pages 418–427, 2002.

[84] I. Rechenberg. Cybernetic solution parth of an experimental problem. *Library Translation 1122, Royal Aircraft Establishment, Farnborough, UK*, 1965.

[85] Z. Roumelioti, A. Kiratzi, and N. Melis. Relocation of the 26 july 2001 skyros island (greece) earthquake sequence using the double-difference technique. *Physics of the Earth and Plentary Interiors*, (138):231–239, 2003.

[86] Z. Roumelioti, A. Kiratzi, N. Theodoulidis, and C. Papaioannou. S-wave spectral analysis of the 1995 kozani-grevena (nw greece) aftershock sequence. *Journal of Seismology*, (6):219–236, 2002.

[87] I. A. Sarafis, P. W. Trinder, and A.M.S Zalzala. Mining comprehensible clustering rules with an evolutionary algorithm. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'03), Chicago-USA*. LNCS Springer-Verlag, 2003.

[88] I. A. Sarafis, P. W. Trinder, and A.M.S Zalzala. Towards effective subspace clustering with an evolutionary algorithm. In *Proc. of the IEEE Congress on Evolutionary Computation (CEC03), Canberra, Australia*, 2003.

[89] I. A. Sarafis, P. W. Trinder, and A.M.S Zalzala. A rule-based evolutionary algorithm for efficient and effective clustering on massive high-dimensional databases. *International Journal of Applied Soft Computing (ASOC), Elsevier Science*, 2005 (Invited Paper) (Accepted for Publication).

[90] P. Scheunders. A genetic c-means clustering algorithm applied to color image quantization. *Pattern Recognition*, 30(6):859–866, 1997.

[91] H-P Schwefel. Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik. *Diplomarbeit, University of Berlin*, 1965.

[92] D. W. Scott. *Multivariate Density Estimation*. Wiley, New York, 1992.

[93] G. Sheikholeslami, S. Chatterjee, and A. Zhang. *WaveCluster*: A wavelet based clustering approach for spatial data in very large databases. *Journal of Very Large Data Bases (VLDB)*, 8(4):289–304, 2000.

[94] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[95] R. Srikanth, R. George, N. Warsi, D. Prabhu, F. E. Petry, and B. P. Buckles. A variable-length genetic algorithm for clustering and classification. *Pattern Recognition Letters*, 16(8):789–800, 1995.

[96] G. R. Terrell. The maximal smoothing principle in density estimation. *Journal of the American Statistical Association*, 85(410):470–477, 1990.

[97] M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman & Hall, 1995.

[98] W. Wang, J. Yang, and R. Muntz. *STING*: A statistical information grid approach to spatial data mining. In *Proc. of the 23rd Int. Conf. on Very Large Data Bases (VLDB'97)*, pages 186–195. Morgan Kaufmann, 1997.

[99] T. Zhang, R. Ramakrishnan, and M. Livny. *BIRCH*: an efficient data clustering method for very large databases. In *Procc of the ACM SIGMOD Int. Conf. on Management of Data*, volume 25, 2 of *ACM SIGMOD Record*, pages 103–114. ACM Press, 1996.

Index