Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Soft Computing xxx (2006) xxx–xxx
Applied Soft Computing
www.elsevier.com/locate/asoc

NOCEA: A rule-based evolutionary algorithm for efficient and effective clustering of massive high-dimensional databases

Ioannis A. Sarafis [a,*], Phil W. Trinder [a], Ali M.S. Zalzala [b]

[a] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, United Kingdom
[b] Technology and Research Solutions FZ-LIC, P.O. Box 500735, Dubai, United Arab Emirates

Received 4 January 2005; received in revised form 16 December 2005; accepted 19 January 2006

Abstract

Clustering is a descriptive data mining task aiming to group the data into homogeneous groups. This paper presents a novel evolutionary algorithm (NOCEA) that efficiently and effectively clusters massive numerical databases. NOCEA evolves individuals of variable-length consisting of 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. A novel quantisation algorithm imposes a regular multi-dimensional grid structure onto the data space to reduce the search combinations. Due to quantisation the boundaries of the intervals are encoded as integer values. The evolutionary search is guided by a simple data coverage maximisation function. The enormous data space is effectively explored by task-specific recombination and mutation operators producing candidate solutions with no overlapping rules. A parsimony generalisation operator shortens the discovered knowledge by replacing adjacent rules with more generic ones. NOCEA employs a special homogeneity operator that enforces quasi-uniform data distribution in the space enclosed by the candidate rules. After convergence the discovered knowledge undergoes simplification to perform subspace clustering, and to assemble the clusters. Results using real-world datasets are included to show that NOCEA has several attractive properties for clustering including: (a) comprehensible output in the form of disjoint and homogeneous rules, (b) the ability to discover clusters of arbitrary shape, density, size, and data coverage, (c) ability to perform effective subspace clustering, (d) near linear scalability with the database size, data and cluster dimensionality, and (e) substantial potential for task parallelism (speedup of 13.8 on 16 processors). A real-world example is a detailed study of the seismicity along the African–Eurasian–Arabian plate boundaries.

Keywords: Data mining; Clustering; Evolutionary algorithms; Rule extraction; Earthquake analysis

1. Introduction

The importance of collecting data related to business or scientific activities to achieve competitive advantage is widely recognised. Driven by advances in data collection and storage, increasingly large and high dimensional datasets are being stored. ‘‘Data expands to fill the space available for storage’’ (Parkinson’s law of data). In fact, data doubles about every year, but useful information seems to be decreasing [14]. Without special tools, human analysts can no longer make sense of such enormous volumes of data that require processing to make informed decisions. The area of knowledge discovery on databases (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 [14,19,28]. Intelligent data mining (DM)—the core stage of KDD—techniques are being developed to semi-automate the process of mining nuggets of hidden knowledge from massive databases, and extract them in forms that can be readily utilised in areas such as decision support.

Clustering—perhaps the most challenging descriptive DM task—aims to find the intrinsic structure of a collection of data points by partitioning them into homogeneous clusters based on the values of their attributes [32,35]. Clustering high dimensional data is especially challenging mainly due to the inherent sparsity of the dataspace. Most clustering techniques suffer from the infamous curse of dimensionality phenomenon: in moderate-to-high dimensional spaces almost all pairs of points are about as far away as average [3,9] 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

* Corresponding author. Tel.: +44 131 451 3435.
E-mail addresses: ceeis@macs.hw.ac.uk (I.A. Sarafis), P.W.Trinder@hw.ac.uk (P.W. Trinder), zalzala@teresol.com (Ali M.S. Zalzala).

1568-4946/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2006.01.011

effectiveness of clustering algorithms that depend critically on distance or density measures, degenerate rapidly with increasing dimensionality [30]. 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 relevant features to clustering (subspace clustering), noise tolerance, insensitivity to initialisation and order of data input, and minimal requirements for domain knowledge [19,28].

Evolutionary algorithms (EAs) are optimisation techniques inspired by the abstractions of Darwinian evolution [7,8,24,39]. In nature, individuals best suited to competition for scarce resources survive. 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 the fitness function. From an implementation viewpoint, EAs are highly parallel procedures and can be easily and conventionally used in parallel systems. 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.

1.1. Feature space and clustering rules

Let $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_d\}$ be a set of attributes with bounded, totally ordered numerical domains and $\mathcal{F} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_d$ be a d-dimensional feature or data space, where $\mathcal{A}_1, \ldots, \mathcal{A}_d$ 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 $\mathcal{P} = \{p_1, \ldots, p_N\}$ consists of a set of d-dimensional records or points, while N denotes the size of $\mathcal{P}$. Each data point p_i, is a vector containing d numerical values $p_i = [p_{i1}, \ldots, p_{id}]$ such that p_{ij} is drawn from the domain $[a_j, b_j]$ of $\mathcal{A}_j$ attribute.

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 paper a clustering rule $\mathcal{R}$ defined in the continuous space $\mathcal{F}$ is knowledge representation in the form:

$$\mathcal{R}: \quad \text{IFcond}_1 \wedge \cdots \wedge \text{cond}_d \ \text{THEN cluster\_label}$$

The antecedent (IF) part of $\mathcal{R}$ consists of a logical conjunction of d conditions, one for each feature, whereas the conclusion (THEN) part contains the cluster label. The semantics of this kind of clustering rule is: 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. (10,000 $\leq$ income $<$ 25,000). 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 $\mathcal{R} = [l_1, u_1) \times \cdots \times [l_d, u_d)$, $i = 1, \ldots, d$, where $l_i, u_i \in \mathbb{R}$ denote the lower and upper bounds of $\mathcal{R}$ in the ith dimension, respectively.

1.2. Our contributions

This paper proposes a novel rule-based EA methodology for DM clustering with the following three phases: firstly, a sophisticated quantisation algorithm (TSQ) (Section 5) imposes a uniform multi-dimensional grid onto the dataspace to reduce the search combinations for clustering. TSQ determines an appropriate grid resolution that enables the discrimination of clusters, while preserving accuracy and acceptable computational cost. Secondly, a novel rule-based EA (NOCEA) (Section 6) discovers high quality hyper-rectangular clustering rules using several novel semi-stochastic genetic operators, an integer-valued encoding scheme, and a simple data coverage maximisation fitness function. Each individual comprises a variable number of disjoint and axis-aligned hyper-rectangular clustering rules with homogeneous data distribution. Both TSQ and NOCEA rely on a novel statistical analysis (UDA) (Section 4) identifying flat density regions ($\mathcal{U}$-regions) in univariate histograms. $\mathcal{U}$-regions detected in orthogonal univariate projections are ''signatures'' of clusters existing in higher dimensional spaces. Thirdly, a post-processing simplification phase (Section 7) that performs subspace clustering and assembles the clusters. The paper also explores task parallelism (Section 6.7) for several genetic operations to improve scalability.

The proposed methodology meets the following desiderata for DM clustering:

- Effective treatment of high dimensionality and exceptional resistance to the curse of dimensionality; precise discrimination of clusters even in very sparse high-dimensional spaces.
- End-user comprehensibility of the results.
- Ability to discover clusters embedded in arbitrary subspaces of high dimensional data.

- Linear scalability with database size, and both data and cluster dimensionality.
- Substantial potential for task parallelism achieving a speed up of 13.8 on 16 processors.
- Ability to discover homogeneous clusters of arbitrary density, geometry, and data coverage.
- Insensitivity to initialisation and order of data.
- Substantial resistance to uniform noise.
- Minimal requirements for a priori knowledge (e.g. automatic determination of the optimal number of clusters and the subspace where each cluster is embedded) and no presumptions of any canonical distribution for the input data.
- Operating on the full dimensional space guards against artifacts formed by the joint projection of multiple clusters in lower dimensional spaces.
- Introduction of a simple non-distance or density based clustering criterion.
- Stochastic traversal of the search space that easily avoids local optima.

1.3. Structure

The remainder of the paper is organised as follows: Section 2 surveys related work in DM and EA literature. Section 3 overviews briefly NOCEAs architecture. Section 4 presents UDA. In Section 5, a new quantisation algorithm (TSQ), is proposed. Section 6 describes in depth various aspects of the evolutionary optimisation within the NOCEA system. Section 7 presents post-processing routines that simplify the discovered knowledge and form the genuine clusters. Section 8 discusses the parameter settings. The experimental results are reported in Section 9. Finally, our conclusions and directions for future research are presented in Section 10. A more detailed discussion of the work presented in this paper can be found in [47].

2. Related work

2.1. Conventional clustering algorithms

Clustering techniques for DM have been studied extensively in recent years, e.g. [13,17,28,29,32,33,35]. Existing clustering algorithms can be broadly classified into four categories: *partitioning*, *hierarchical*, *density-based*, and *grid-based*.

- *Partitioning techniques* organise the data into k clusters so as the distance of each point to the closest cluster representative point (k-means) is minimised [33,35,42]. Despite being popular, partitioning algorithms: (a) are not suitable to discover non-convex clusters, (b) are sensitive to the outliers and the initial selection of cluster seeds, (c) require the number of clusters in advance, (d) cannot deal with clusters of different sizes and densities, and (e) produce poor quality cluster descriptors.
- *Hierarchical techniques* produce a nested sequence (dendrogram) of clusters. The hierarchy can be formed in top-down (divisive) or bottom-up (agglomerative) fashion. Each iteration involves merging or splitting clusters based on some distance function that measures the similarity between clusters. The main disadvantages of hierarchical techniques are their inability to perform corrections once a decision (split or merge) has been executed and their high computational complexity, typically, quadratic in the number of data points (N). BIRCH [57] is a divisive clustering algorithm with complexity $O(N)$ that can work with limited amount of memory. However, BIRCH is sensitive to the order of data input and does perform well only when the clusters are hyper-spherical and have similar sizes. CURE [25] is an agglomerative method with $O(N)$ complexity that is achieved using a combination of random sampling and partitioning. The representative points of a cluster are generated by selecting a fixed number of well-scattered points from the cluster and shrinking them towards the center of the cluster by a specified fraction. CURE has the ability to discover non-spherical clusters of different sizes and densities, but is sensitive to the input parameters. Similar to CURE, CHAMELEON [34] is an agglomerative clustering algorithm, which tries to improve the clustering quality using more elaborate criteria (inter-connectivity and closeness of clusters) when merging two clusters. Experimental results have shown that CHAMELEON is more effective than CURE in discovering arbitrary-shaped clusters but it has quadratic $O(N^2)$ complexity.
- *Density-based techniques* group neighbouring data points into clusters based on a local condition of density rather than the distance between data points. DBSCAN [15] seeks for core objects, that is, points whose ϵ-neighbourhood (ϵ: radius) contain at least MinPts points. A sequence 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 considered as 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 does not work well in high-dimensional spaces and is quite sensitive to the user-defined parameters ϵ and MinPts. OPTICS [6], an extension of DBSCAN, creates an ordering of points in the databases and stores some additional distance information, allowing the extraction of all density-based clustering for any lower value of the user-specified radius ϵ. OPTICS has the same complexity as that of DBSCAN. DENCLUE [31] uses an influence function to describe the impact of a point about its neighbourhood while the over all density of the data space is the sum of influence functions from all data points. Clusters are determined using density attractors that are local maxima of the overall density function. To compute the sum of influence functions a grid structure is used. DENCLUE (a) scales well ($O(N)$), (b) can find arbitrary-shaped clusters, and (c) can deal with noise, but it is quite sensitive to the input parameters.
- *Grid-based techniques* quantize the data space into a multi-dimensional grid and aggregate the points into cells. The

basic idea is to discard low-density cells, and then combine adjacent high-density cells to form clusters. The classical tradeoff is between resolution and complexity. These methods are reasonably fast for low-to-moderate dimensional datasets and can discover arbitrary-shaped clusters. However, since the number of cells grows exponentially with the dimensionality it is both difficult and expensive to find high-density adjacent cells in high dimensional spaces. CLIQUE works with high dimensional datasets by building clusters in appropriate subspaces of the original data space [5]. Each dimension is divided into equal-width bins resulting a uniform grid. A cell is dense if it contains more than a user-specified number of points. CLIQUE adopts a bottom-up approach of discovering q-dimensional dense cells based on $(q-1)$-dimensional dense cells. CLIQUE scales linearly with the size of the dataset and quadratic with the dimensionality. MAFIA [41], is a modification of CLIQUE that runs faster and finds better quality clusters. The main modification is the introduction of an adaptive interval size based on the actual data distribution. Due to adaptive intervals MAFIA scales linearly with the size and dimensionality of the dataset and delineates cluster boundaries more accurately than CLIQUE. WaveCluster [52], a multi-resolutional technique, maps the data onto a multi-dimensional grid, it then applies a wavelet transformation to the original feature space and finally finds dense regions in the transformed space. The wavelet transformation makes the clusters more distinguishable and removes outliers. WaveCluster (a) detects clusters at various levels of accuracy, (b) handles large datasets efficiently, (c) finds arbitrary-shaped clusters, and (d) is insensitive to noise/outliers and order of data input. WaveCluster is suitable only for low-dimensional datasets.

2.2. Clustering with EAs

There have been many attempts to use EAs for clustering, e.g. [16,18,23,26,38,49,48,50,54]. An excellent introductory discussion regarding the application of EAs in clustering can be found in [20]. In [26] the individuals represent the coordinates of the centers of the k desired clusters and the evolutionary algorithm, called GGA, relocates the cluster centers in a way that the k-means (HCM) and fuzzy k-means (FCM) criterion functions are minimised. Experimental results have shown that GGA achieves both avoidance of local extreme and minimal sensitivity to initialisation, but its execution time can take up to two orders of magnitude more than FCM/HCM. To improve both efficiency and effectiveness various hybrid methods have been proposed [50,16]. These methods either use the best solution obtained by a complete run of EA as initial seed for an iterative method such as HCM/FCM, or use special mutation operators that perform few standard HCM/FCM iteration steps to obtain a locally optimised offspring. Finally, it is worth noting that EAs have also been used to discover more comprehensible cluster descriptors such as ellipsoids and boxes [23,48,49,54].

3. NOCEA overview

Non-overlapping clustering with evolutionary algorithms (NOCEA) 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 NOCEA is shown in Fig. 1. NOCEA includes several pre-and post-processing stages to prepare the raw data and simplify the discovered knowledge, respectively. 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, the TSQ quantisation algorithm (Section 5) imposes a regular multi-dimensional grid onto the dataspace to reduce the search combinations. Thereby, the bounds of rules are encoded as integer values. Like most EAs, NOCEA begins with an initial population of individuals whose chromosomes are independently initialised with a single, randomly generated rule. Next, a task-specific genetic operator, the repair operator (Section 6.6) shrinks the boundaries of rules or split 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) (Section 6.2), 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 6.5), generalisation (Section 6.3), and mutation (Section 6.4). 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

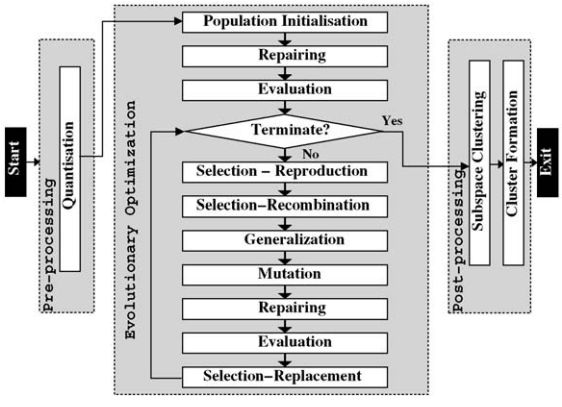

Fig. 1. Architecture of NOCEA.

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. After convergence, a post-processing routine performs subspace clustering (Section 7.1) removing redundant conditions from the antecedent part of the rules. Finally, adjacent rules with similar densities are grouped together to assemble (Section 7.2) clusters, and report them in disjunctive normal form (DNF) expressions.

4. Uniform-region discovery algorithm—UDA

This section presents the novel uniform-region discovery algorithm (or UDA) to analyse smooth univariate density histograms. The goal of 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. UDA combines standard histogram smoothing techniques, i.e. Kernel density estimation, with new heuristics that perform a fine localised analysis of the data distribution. 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. UDA is extensively used for both data quantisation (Section 5) and clustering (Section 6).

For instance, consider the density histograms of the candidate rule $\mathcal{R}$, as shown in Fig. 2. 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 (dashed lines) pass through the boundaries of $\mathcal{U}$-regions.

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 UDA 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 $[l_{ij}, u_{ij}]$ of jth rule covers k bins ($u_{ij} - l_{ij} + 1 = k$). Additionally, let $f_{l_{ij}}, \ldots, f_{u_{ij}}$ be the k smoothed histogram values, i.e. number of points contained in each bin, that corresponds 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. UDA 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, $\mathrm{UCL} = \max(f_{l_{ij}}, \ldots, f_{u_{ij}})$. The baseline lower control limit (LCL), which specifies the splitting density level is then computed as:

$$\mathrm{LCL} = T_{\mathrm{h}} \times \mathrm{UCL} \tag{1}$$

where the user-specified homogeneity threshold $T_{\mathrm{h}} \in (0, 1]$ controls the desired degree of uniformity. If all histogram values exceed LCL then D is deemed $\mathcal{U}$-region and it is kept intact. For instance, UDA detects only one $\mathcal{U}$-region along the

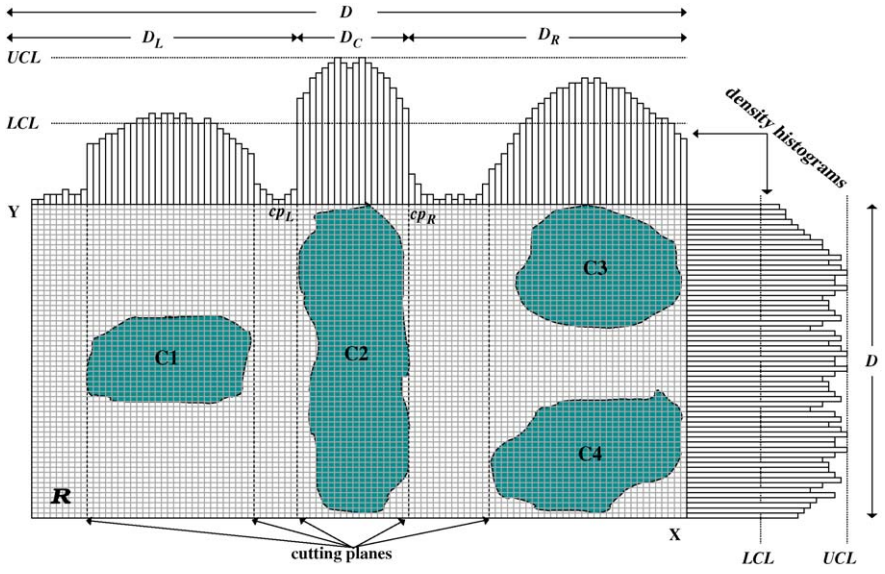

Fig. 2. Motivation of UDA.

vertical-axis in Fig. 2. Notice that a $\mathcal{U}$-region detected in a uni-dimensional histogram may be the result of the joint projection of multiple clusters. If UDA 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 Fig. 2. 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 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 deemed $\mathcal{U}$-region and it is not analysed further. The above procedure is recursively applied to each newly formed boundary region independently until no more splitting sites occur. In essence, as UDA proceeds, $\mathcal{U}$-regions are gradually detected at decreasingly low level of density. To suppress the subsequent formation of sparse rules and to reduce computation, UDA instantly discards all sparse regions including those that are $\mathcal{U}$-regions. A region is sparse if it covers less than NT_s points, where N is the total number of points and T_s is the sparsity threshold (Section 8).

5. Two stage quantisation algorithm—TSQ

Grid-based clustering techniques include a pre-processing step, called quantisation that imposes a multi-dimensional grid structure onto the data space [5,41]. The grid is formed by partitioning the domain of each dimension into non-overlapping intervals or bins of uniform or variable size [28,29]. Quantisation reduces the search combinations by aggregating together points mapping into the same cell, and the classical tradeoff 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 computational complexity as dimensionality increases, the data space is quantised by considering each dimension independently. Unfortunately, any uni-dimensional analysis can only produce an approximation of the optimal resolution simply because it disregards all inter-attribute correlations existing in higher dimensional spaces. Most grid-based 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.

In this section, a two stage quantisation algorithm (TSQ) (Sections 5.1 and 5.2), is proposed to support the clustering of large and high dimensional numerical datasets. The quantised data are subsequently analysed by the evolutionary-based clustering algorithm NOCEA. TSQ quantizes the dataspace using a novel statistical analysis of uni-dimensional 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, TSQ has no specific bias toward coarse resolutions, because NOCEA can operate on relatively fine grids as it attempts to produce highly homogeneous rather than highly dense clusters.

The choice of the width of the bins 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 (under-quantisation). 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 (over-quantisation).

Terrell [55] suggested a practical data-based rule for setting the upper bound on bin width for univariate histograms. In particular, the bin width w should be directly proportional to the standard deviation σ of the univariate sample and inversely proportional to $N^{1/3}$, N: sample size [51,55]:

$$w \leq 3.729\sigma N^{-1/3} \tag{2}$$

A naive quantisation would be to directly apply Eq. (2) in each dimension independently to derive an over-smoothed estimator of the bin width. While simple, this approach has weaknesses. In particular, although w changes at a rate inversely proportional to $N^{1/3}$, this rate is much faster for relatively small datasets, e.g. $N < 50{,}000$. However, for moderate-to-large datasets the factor $N^{-1/3}$ exhibits only a marginal variation. Given that massive datasets are common in DM, and w is more sensitive to changes in σ compared to N, it becomes clear why the effect of σ on w is crucial.

A major limitation of Eq. (2) 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 [27]. Very coarse resolutions attributed to the harmful effect of outliers in formulae (2), 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 (2). 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. These issues are addressed in Sections 5.1 and 5.2.

5.1. Gross statistical analysis

5.1.1. Step 1: removal of outliers

To guard against the harmful effects of outliers in highly skewed distributions, 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. Hence, TSQ focuses on data lying inside the outlier-free interval E [27] rather than the entire domain $[a, b]$:

$$E = [\max(a, (Q_1 - 1.5\text{IQR})), \min(b, (Q_3 + 1.5\text{IQR}))] \quad (3)$$

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

5.1.2. 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 Eq. (2) due to the impact on σ. Hence, it is vital to guard against significant data discontinuities. 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 $x_1, \ldots, x_k$ 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 as [11]:

$$H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i) \quad (4)$$

Let $b_1, \ldots, b_k$ 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:

$$H = -\sum_{i=1}^{k} d_i \log_2 d_i \quad (5)$$

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 TSQ to quantify significant data discontinuities in univariate samples. If the data in E from formulae (3) 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:

$$\epsilon = \delta\left(\frac{N}{N_E}\right) l_E \quad (6)$$

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

Using ϵ as initial resolution, TSQ constructs two conventional density histograms, one for the target dimension and one for a uniform distribution both defined in E. The density histogram is a step function with height for each bin being the proportion of the sample E contained in that bin. It then computes the entropy for both histograms using Eq. (5). 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. TSQ incorporates quantitative information related to both data discontinuities and concentration by modifying ϵ by a factor r_H:

$$\epsilon' = r_H \epsilon \quad (7)$$

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).

5.1.3. Step 3: smoothing via Kernel density estimation

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 Appendix A. 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 (A.2)]. However, for non-normal data distributions, e.g. multi-modal or heavily skewed distributions, the statistical performance of formulae (A.2) is poor [51,53,56].

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 E into a finite set of disjoint intervals containing a relatively small percentage, e.g. 5%, of the total (N) 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 jth 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 Eq. (A.2)) for the ith interval in jth 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 TSQ bandwidth scalarisation is:

$$h_j = \sum_{i=1}^{k} \left(\frac{N_{ij}}{N_E} \right) h_{kj} \tag{8}$$

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

5.2. Detailed statistical analysis

5.2.1. Step 1: detection of quasi-uniform regions

Let us assume that during the gross statistical analysis stage the outlier-free interval E is partitioned into m uniform bins of size ϵ' determined by Eq. (7). TSQ then employs the UDA (Section 4) algorithm to obtain all non-sparse $\mathcal{U}$-regions along the smoothed frequency histogram of E.

5.2.2. Step 2: quantisation of quasi-uniform regions

The rationale of partitioning the original smoothed histogram with UDA is to enable a more detailed analysis within the $\mathcal{U}$-regions identified. In particular, Terrell's quantisation rule (Eq. (2)) 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..." [51]. However, an extremely fine resolution computed by Eq. (2), even if it is valid from a statistical point of view, incurs high computational costs for clustering, especially for high dimensional datasets [5,41]. Therefore, it is necessary to set a lower bound on w that yields a reasonable compromise between efficiency and effectiveness. Recall that the provisional bin width ϵ' given by Eq. (7) is a particularly robust estimator of the lower bound of w. Hence, 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:

$$w_{ij} = \max(\epsilon', (3.729\sigma_{ij}N_{ij}^{-1/3})) \tag{9}$$

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.

5.2.3. 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 increases substantially the complexity of NOCEA. Therefore, TSQ uses a simple weighted-sum method to scalarise the set of locally optimal bin widths into a single global value. 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. In particular, 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 equisized intervals of length w_j that is determined by the following weighted-sum expression:

$$w_j = \sum_{i=1}^{k} \left(\frac{N_{ij}}{\text{total}_{N_j}} \right) w_{kj} \tag{10}$$

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

6. Evolutionary optimisation in NOCEA

This section describes in depth the part of NOCEA system that is responsible for the evolutionary optimisation. In particular, Section 6.1 presents the semantics of the individual representation and explains why NOCEA adopts an integer encoding scheme instead of binary or floating. The fitness function that measures the quality of candidate solutions is presented in Section 6.2. Section 6.3 presents the generalisation operator that strives to produce shorter rule-sets. Sections 6.4 and 6.5 describe the mutation and recombination operators, respectively, that are responsible to introduce variation in the population. Section 6.6 presents the repair operator that enforces the formation of homogeneous rules. Finally, Section 6.7 describes a preliminary parallelisation of NOCEA to improve scalability.

6.1. Individual representation

Since clustering is all about summarising data distributions, we adopt clustering rules (Section 1) as a readily interpretable structure to describe the discovered knowledge. NOCEA evolves individuals of variable-length, which consist of disjoint and axis-aligned hyper-rectangular rules. Each fixed-length rule, in turn, is composed of d feature genes, where each gene encodes an interval-like condition related to one feature. The ith feature-gene ($i = 1, \ldots, d$) of jth rule is subdivided into two fields, namely lower (l_{ij}) and upper (u_{ij}) bound. During quantisation, the domain $[a_i, b_i]$, $i = 1, \ldots, d$, of the ith dimension is partitioned into m_i disjoint intervals of uniform length w_i. The boundaries for the intervals of the rules in ith dimension ($i = 1, \ldots, d$) are encoded as integer values from the discrete domain $[0, \ldots, (m_j - 1)]$. 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 $(u_{c1} + 1) \leq l_{c2}$. The obvious advantage of the variable-length genotype is the transfer of control over the optimal number of rules from humans to the genetic search mechanisms. Additionally, since rules are disjoint there is no need to encode the consequent part.

Fig. 3(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 $w_{\text{Income}} = 25$ and $w_{\text{Expenditure}} = 20$ be the bin width for income and expenditure, respectively. Fig. 3(b) shows the structure of the genotype corresponding to the candidate solution of Fig. 3(a) that has three rules $\mathcal{R}_1, \mathcal{R}_2$, and $\mathcal{R}_3$. Fig. 3(c) depicts the conventional binary representation of these rules using five-bit precision for both dimensions.

There are several reasons for using an integer-valued representation rather than floating-point or binary. Although a real-valued representation enables arbitrary-precision solutions to be found, it generates prohibitively large search spaces for problems of high dimensionality. There are also several reasons to abandon binary encoding in NOCEA. 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 Fig. 3(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. 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 Fig. 3(a), NOCEA needs at least five bits to cover this range. However, a five-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

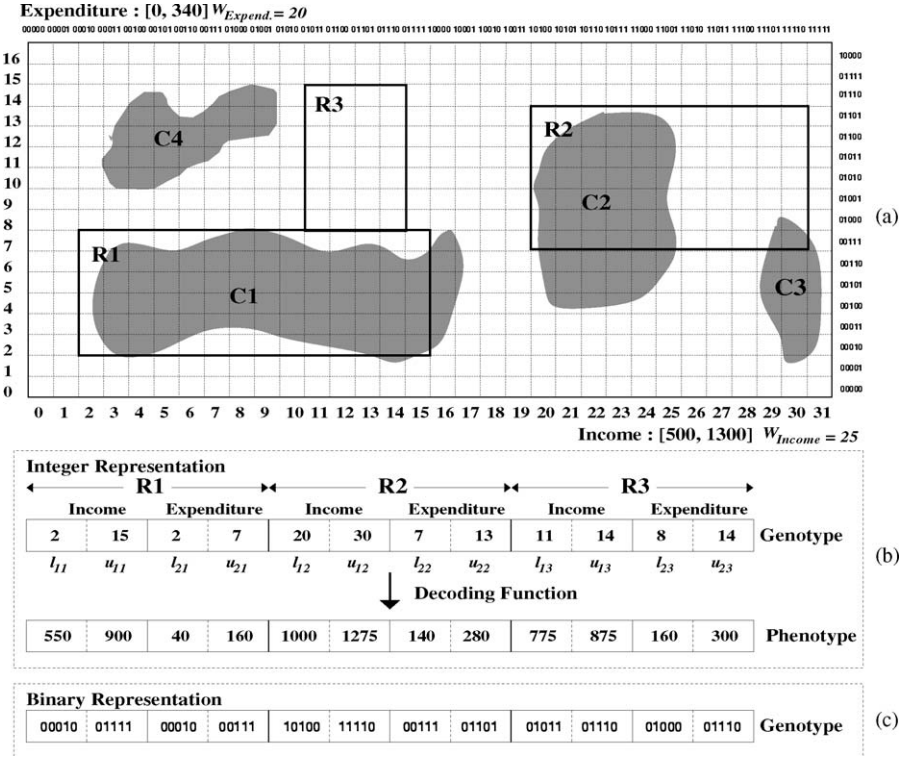

Fig. 3. Integer-valued representation in NOCEA.

rule bound can take are from a minimum length integer-valued sequence $0, \ldots, (m-1)$, where m denotes the number of bins in each dimension.

In this paper, a feasible solution always complies with the following requirements: (a) the upper bound of all rules must be at least equal to its associated lower bound for all dimensions, (b) all hyper-rectangular rules are axis-aligned and disjoint, (c) the d-dimensional region enclosed by a feasible rule must have a relatively homogeneous distribution of points (Section 6.6). For example, rule $\mathcal{R}_2$ in Fig. 3(a) is not homogeneous, even though it is semantically valid and axis-parallel, and (d) the point coverage of a feasible rule must be statistically significant to minimise the risk of over-fitting (i.e. to cover very few instances) the data. Hence, sparse rules (Section 4) are eliminated, because they reflect spurious relationships that are unlikely to occur in unseen data.

6.2. 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 individual in solving the target problem [24,39]. This value is the quantitative information the EA uses to guide the search. 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 general, the choice of the fitness function is closely related to the problem at hand.

In this paper a simple and robust fitness function is proposed to guide the evolutionary search. In our clustering context, the fitness function f: $\mathcal{S} \rightarrow (0, 1]$ ($\mathcal{S}$: the set of all possible feasible solutions) 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 $\mathcal{I}$:

$$f(\mathcal{I}) = \left(\frac{1}{N}\sum_{i=1}^{k} N_i\right) \tag{11}$$

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

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. Aiming to maximise data coverage with 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 f are:

- *Not a distance-based clustering criterion*. 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.
- *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 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.
- *Resistance to curse of dimensionality*. The curse of dimensionality phenomenon (Section 1) has no impact on f because the concepts of sparsity and point proximity are not encapsulated in f.
- *Bounded range*. Due to the disjointness of rules in the chromosome, the range of the fitness function 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.
- *No preference bias with respect to the structural characteristics of a clustering solution*. Clearly, formulae (11) 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. This also allows the exploration of more search combinations. 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.

6.3. 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 because it strives (a) to minimise the length of individuals by replacing 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 (Section 7.1). 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 them and additionally, there is one dimension where they are contiguous.

Generalisation, also improves scalability because computation heavily depends on the total number of rules that are processed in each generation. Generalisation is applied with a small probability to a pair of adjacent rules at time satisfying some conditions, and produces a single and hopefully more generic rule. Let us assume that the pair of ith and jth rules undergo generalisation along the gth dimension. The original rules are put together in an incomplete generalisation G that must not overlap with neighbouring rules. To achieve this, the

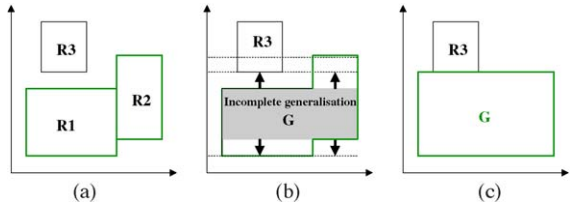

Fig. 4. Generalisation principle. (a) Before; (b) expansion; (c) after.

operator firstly determines the backbone of the rule G that will eventually replace the two rules. The lower (l_k) and upper (u_k) bounds of G are:

$$[l_k, u_k] = \begin{cases} [\min(l_{ki}, l_{kj}), \max(u_{ki}, u_{kj})], & \text{if } k = g \\ [\max(l_{ki}, l_{kj}), \min(u_{ki}, u_{kj})], & \text{otherwise} \end{cases} \quad (12)$$

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 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 Fig. 4(a) yields initially the incomplete generalisation G shown as grey-shadowed region in Fig. 4(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 Fig. 4(c) comprises fewer and more generic rules compared to the solution in Fig. 4(a).

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 generalisations in Fig. 5(a–c), where the relative darkness indicates the density of the clusters. In the first case—Fig. 5(a), the density of rules $\mathcal{R}_1$ and $\mathcal{R}_2$ differs significantly and hence the resulting generalisation, which inherits this difference, is non-homogeneous. In the second case—Fig. 5(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 the generalisation is unsuccessful 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. $\mathcal{R}_7$ and $\mathcal{R}_8$ under generalisation and additionally there exist other rules, e.g. $\mathcal{R}_5$ and $\mathcal{R}_6$ in close proximity, as shown in Fig. 5(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 NOCEA to converge into an optimal solution.

To reduce the severity of the side-effects associated with generalisation, 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

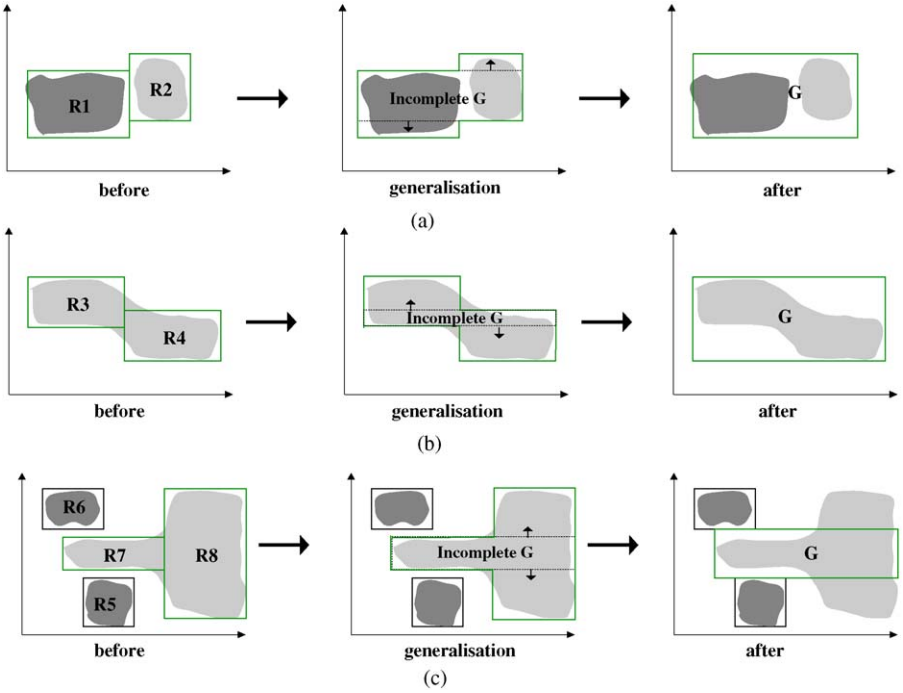

Fig. 5. Pitfalls of unconstrained generalisation.

rules $\mathcal{R}_i$ and $\mathcal{R}_j$, are generalised only if the following conditions are true for every dimension $l = 1, \ldots, d$, excluding g (g: touching dimension for $\mathcal{R}_i$ and $\mathcal{R}_j$):

$$\frac{\min(D_i, D_j)}{\max(D_i, D_j)} \geq T_{\mathrm{h}} \quad \text{and} \quad \frac{R_{li} \cap R_{lj}}{u_{lm} - l_{lm}} \geq T_{\mathrm{g}}, \quad m \in \{i, j\} \tag{13}$$

where D_i, D_j denote the density of ith and jth rule, respectively, while $R_{li} \cap R_{ij} = \min(u_{li}, u_{lj}) - \max(l_{li}, l_{lj}) + 1$ is the length of the intersection between the two rules in lth dimension. The homogeneity T_{h} and generalisation $T_{\mathrm{g}} \in (0, 1]$ thresholds are discussed in Section 8. The first condition prevents generalising rules with very different densities, while the second permits generalisation only when rules have proper alignment and similar sizes.

6.4. Mutation operators

Mutation serves to prevent premature loss of population diversity by randomly sampling new points in the search space [7]. Typically, 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. NOCEA has two novel semi-stochastic mutation operators, namely, Grow- and Seed-Mutation.

6.4.1. 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, making, thus, the individuals fitter [48,49]. Given 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, improving thus comprehensibility and reducing computational complexity. The general form of Grow-Mutation can be written as:

$$\mu = \mu' + U \tag{14}$$

where μ' and μ denote the value of a gene, i.e. left or right bound, before and after Grow-Mutation, respectively, where U is a uniform discrete random variable in $[0, \mu_{\max}]$ for the upper bound, and $[-\mu_{\max}, 0]$ for the lower bound. $\mu_{\max}$ represents the maximum amount of modification for a valid expansion that does not produces overlapping rules. Fig. 6 depicts the algorithm for determining $\mu_{\max}$ if the upper bound u_{ij} of jth rule is grow-mutated in ith dimension. The derivation of $\mu_{\max}$ for the lower bound is the dual procedure.

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

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

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

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

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

m_i: total number of bins in ith dimension

Fig. 6. Computing $\mu_{\max}$ for an upper Grow-Mutation.

Fig. 7 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 $\mu_{\max}$ with the algorithm in Fig. 6, $\mathcal{R}_1$ is randomly expanded to the right within the rectangle (abcd) that is demarcated by the dashed lines, as shown in Fig. 7(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 ($\mu_{\max}$), Grow-Mutation may create non-homogeneous rules, e.g. $\mathcal{R}_1$ in Fig. 7(b). In such cases, the repair operator (Section 6.6) enforces feasibility on the candidate solutions, as depicted in Fig. 7(c). Clearly the data coverage of $\mathcal{R}_1$ has been increased by the Grow-Mutation.

6.4.2. Seed-Mutation

Due to the constraint of evolving disjoint rule-sets, Grow-Mutation is incapable of assuring that every uncovered region of the feature space $\mathcal{F}$ is accessible to NOCEA. This limitation is evident in Fig. 8(a) where NOCEA has reached a deadlock in increasing the coverage of the individual. 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 $\mathcal{R}_4$. NOCEA escapes from such a local optimum using 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 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} (m_k: total number of bins in the kth dimension):

$$[\mathrm{lb}_k, \mathrm{ub}_k] = \begin{cases} [(u_{ij} + 1), (m_k - 1)], & \text{if } k = i \\ [l_{kj}, u_{kj}], & \text{otherwise} \end{cases} \tag{15}$$

If the bounding box contains at least one uncovered cell the algorithm selects randomly 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 bounds in a specific dimension are also processed in a random order. The rationale behind the large-scale expansion of the seed is to increase the probability of locating a non-sparse rule.

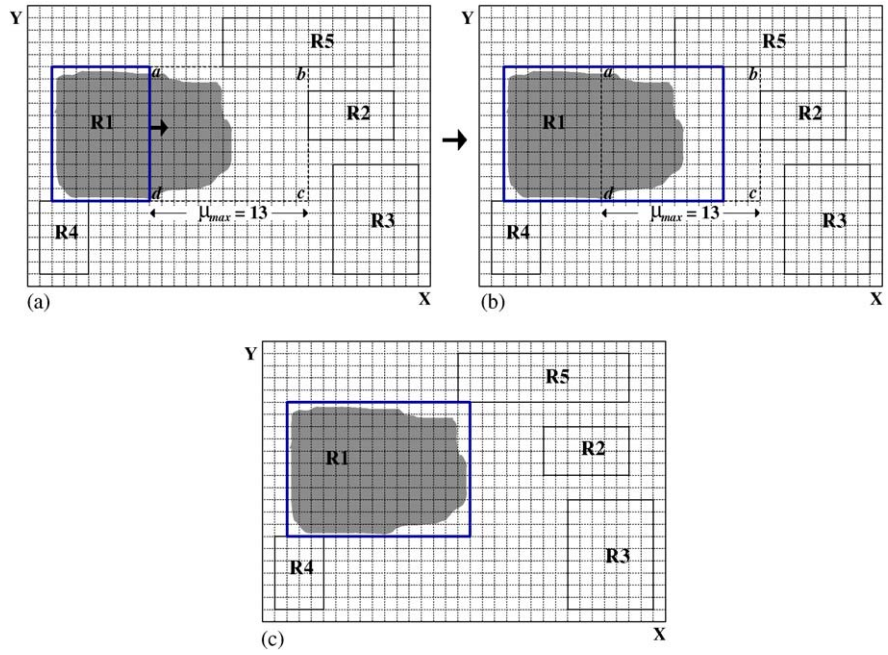

Fig. 7. Local tuning with Grow-Mutation/repairing. (a) Before Grow-Mutation; (b) after Grow-Mutation; (c) after repairing.

Fig. 8(b–d) shows how NOCEA breaks the deadlock (perform both local fine-tuning and discovery of new clusters) using 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 Fig. 8(c)) inside a previously unreachable region. The subsequent repairing of the expanded seed yields two new homogeneous rules $\mathcal{R}_5$ and $\mathcal{R}_6$ as shown in Fig. 8(d).

6.4.3. 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 $\in$ [1, k], Feature $\in$ [1, d], Bound $\in$ [lower, upper], denote the rule, feature, and bound, respectively, undergoing mutation whose type is specified in the field Type $\in$ [grow, seed]. The list of mutation events is shuffled to assure randomness in the order by which bounds, features and rules are processed. Notice that a scheduled mutation event may be heavily affected or even prevented by preceding mutation(s). This is because, any form of mutation must yield non-lethal variations, i.e. solutions with

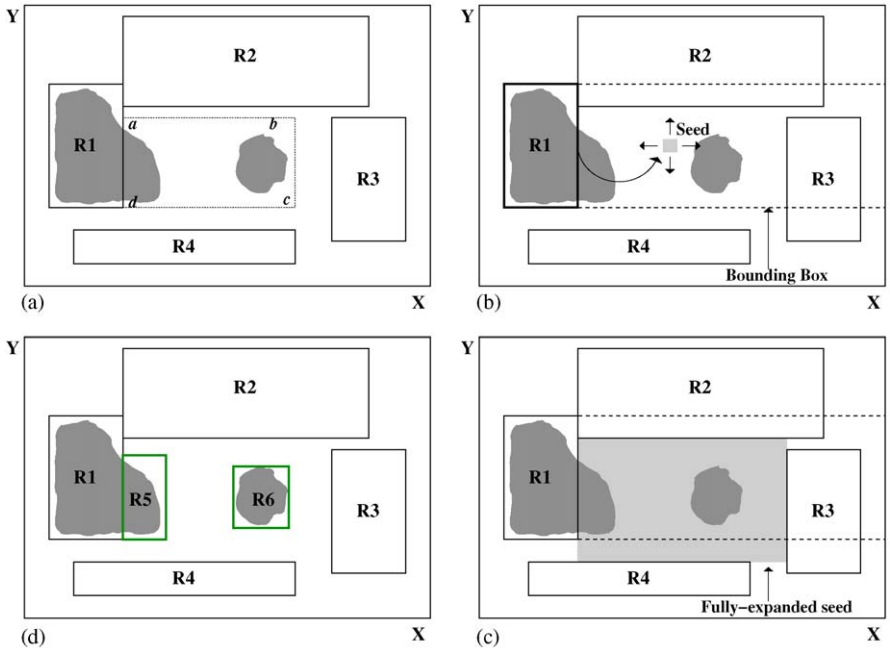

Fig. 8. Seed-Mutation example. (a) A limitation of Grow-Mutation to explore the rectangular area *abcd*; (b) seed-generation; (c) seed-expansion; (d) seed-repairing.

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.

Fig. 9. Scheduling and executing mutations in NOCEA.

disjoint and syntactically valid rules. In NOCEA, mutations are scheduled and executed in the following manner (Fig. 9).

6.5. Recombination operator

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 [7,8]. As the EA is unaware of what characteristics account for the good performance, the best it can do is to recombine characteristics at random. However, due to the selective pressure of EAs, poorly performing offspring will not survive for long. NOCEAs fitness function 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 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, i.e. a solution comprising 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 this paper disruption after crossover is the splitting of parental rules to create non-lethal offspring.

This section presents NOCEAs novel recombination scheme, the overlaid rule crossover (ORC) operator whose functionality is geared towards: (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, 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 ORC are explained with the help of the example depicted in Fig. 10, where the colour of a rule indicates its parental origin. ORC operates on two parent solutions at time and creates two non-lethal offspring in a way that rule disruption is minimised. The main stages in ORC are:

(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, ORC achieves propagating each rule present in the parental chromosomes in at least one offspring. For instance, Fig. 10 shows that each offspring inherits all rules from its primary parent.

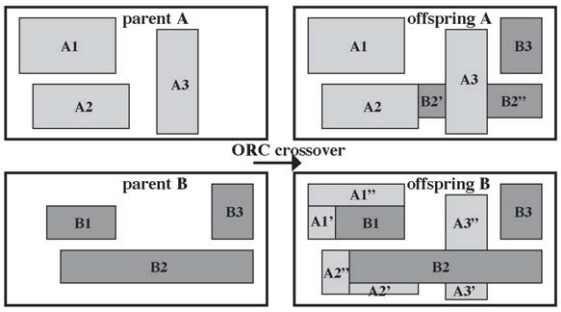

Fig. 10. ORC recombination example.

(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 Fig. 10, receives unaltered rule $\mathcal{B}_3$ from its secondary parent B, since such an operation yields a non-lethal solution. 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 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 untill now ORC has a purely deterministic behaviour.

(3) *Resolving rule overlapping*. What follows next is the splitting algorithm of Fig. 11 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

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

Fig. 11. Algorithm for resolving rule overlapping in ORC.

bound of an offspring rule that intersect 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 splitting algorithm completes all the newly formed non-sparse rules (if any) are copied into the offspring, enriching its genetic material and consequently improving its performance.

From an exploratory viewpoint, 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, NOCEA has the exceptional ability to reach this optimal point of the search space in a single ORC operation.

In short, the salient features of ORC are:

- *Non-lethal variations*. Despite its semi-stochastic functionality, ORC always guarantees the generation of non-lethal offspring.
- *Beneficial variations*. ORC improves the mean performance of the population since the offspring are always at least as fit as their primary parents.
- *No positional bias*. ORC has no positional bias [7] because the transmission of rule-genes to offspring is absolutely independent of their relative positions on the parental chromosomes.
- *Distributional bias*. ORC has a distributional bias [7] in the sense that the expected number of rules that are transmitted to an offspring is bounded minimally by the number of rules of its primary parent. Concerning the secondary parent, 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.

6.6. 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 repair operator relies on the UDA algorithm (Section 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 [4,5,10,40,41]. 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 $\mathcal{U}$-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 $T_s \in (0, 1]$. NOCEAs fitness function is totally blind to the quality of the clustering results, solely seeking to maximise data coverage. In particular, the proposed fitness function lacks any bias that would yield:

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

In essence, since there is no constraint to prevent rules from growing arbitrarily, 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. For instance, recall the candidate rule $\mathcal{R}$, as shown in Fig. 2. Undoubtedly, $\mathcal{R}$ is non-homogeneous, hence infeasible, as it encloses four distinct clusters, in addition to noise.

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 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 [30]. 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. In particular, all dimensions of a candidate rule $\mathcal{R}$ undergo the following processing stages with a random order.

6.6.1. Construction of smoothed frequency histogram

Initially, NOCEA computes the smoothed frequency histogram along the orthogonal univariate projection of the current dimension using the binned Kernel density estimation

(KDE) with the boundary correction as described in Appendix A. It is vital 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 [51,53,56], 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, NOCEA employs a binned version of the KDE method as explained in Appendix A. Henceforth, the term density or frequency histogram will refer to a binned KDE histogram as defined in Appendix A, 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 (A.2)] as described in Appendix A. However, for non-normal data distributions, e.g. multi-modal or heavily skewed distributions, the statistical performance of formulae (A.2) is poor [51,53,56].

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

(i) Split dimension into k (e.g. $k = 4$) equi data coverage segments.
(ii) Apply the oversmoothing normal reference rule [formulae (A.2)] 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.
(iii) Compute a provisional bandwidth h by scalarasing the local bandwidths as weighted-sum $h = \sum_{i=1}^{k}\left(N_{(i)}/\left(N_{(1)} + \cdots + N_{(k)}\right)\right)h_{(k)}$.
(iv) Using the bandwidth found in step (iii), construct a smooth frequency histogram based on the binned KDE with boundary correction as explained in Appendix A.
(v) Apply the UDA (Section 4) to the smooth frequency histogram obtained in step (iv) to locate non-sparse $\mathcal{U}$-regions.
(vi) If no non-sparse $\mathcal{U}$-regions can be found in step (v), set the bandwidth (h) to the value found in step (iii) and exit. Otherwise, repeat steps (iii)–(iv) to the newly formed $\mathcal{U}$-regions to compute the final bandwidth.

Having determined the smoothing bandwidth (h), NOCEA builds the final smooth frequency histogram for the current dimension.

6.6.2. Detection of cutting planes/histogram splitting

Then, the repair operator reapplies 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 the same processing stages, as described in Sections 6.6.1–6.6.2, 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.

An example of repairing is shown in Fig. 12, where the candidate non-homogeneous rule $\mathcal{R}$ of Fig. 12(a) is hierarchically decomposed into a set of disjoint-feasible rules (Fig. 12(b)) using axis-aligned cutting planes that are denoted by dashed lines. Evidently, as repairing progresses the refined rules become increasingly more homogeneous.

6.7. Task parallelism—pNOCEA

This section explores the use of task parallelism to speed up NOCEA when the data to be mined is massive [20,21]. The core idea of parallel version of NOCEA (pNOCEA), 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. pNOCEA achieves a speedup of 13.8 on 16 processors (Section 9.10.5). Fig. 13 depicts the architecture of pNOCEA, where several processor–memory–disk units are attached on a communication network, and coordinated by a central master machine.

- *Data placement*. pNOCEA implements a share-nothing architecture where each remote processor (PE) has direct access only to its local memory (M), as shown in Fig. 13. Currently each local memory contains a replica of the entire dataset (D). There is no need to migrate incomplete tasks between PEs because all tasks involving access to the data, i.e. generalisation, recombination, and repairing, can be completed on a single PE.
- *Granularity*. Typically, a thread is a sequential unit of computation that is entirely executed in a single PE without interruption. Granularity, in parallel processing, is the average computation cost of a thread. 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. pNOCEA 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 solution, constitutes a sequential thread of computation that is entirely executed in a single PE.
- *Load balancing*. 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 PEs have to explicitly ask for work. During the various stages of a single generation, the coordinator machine

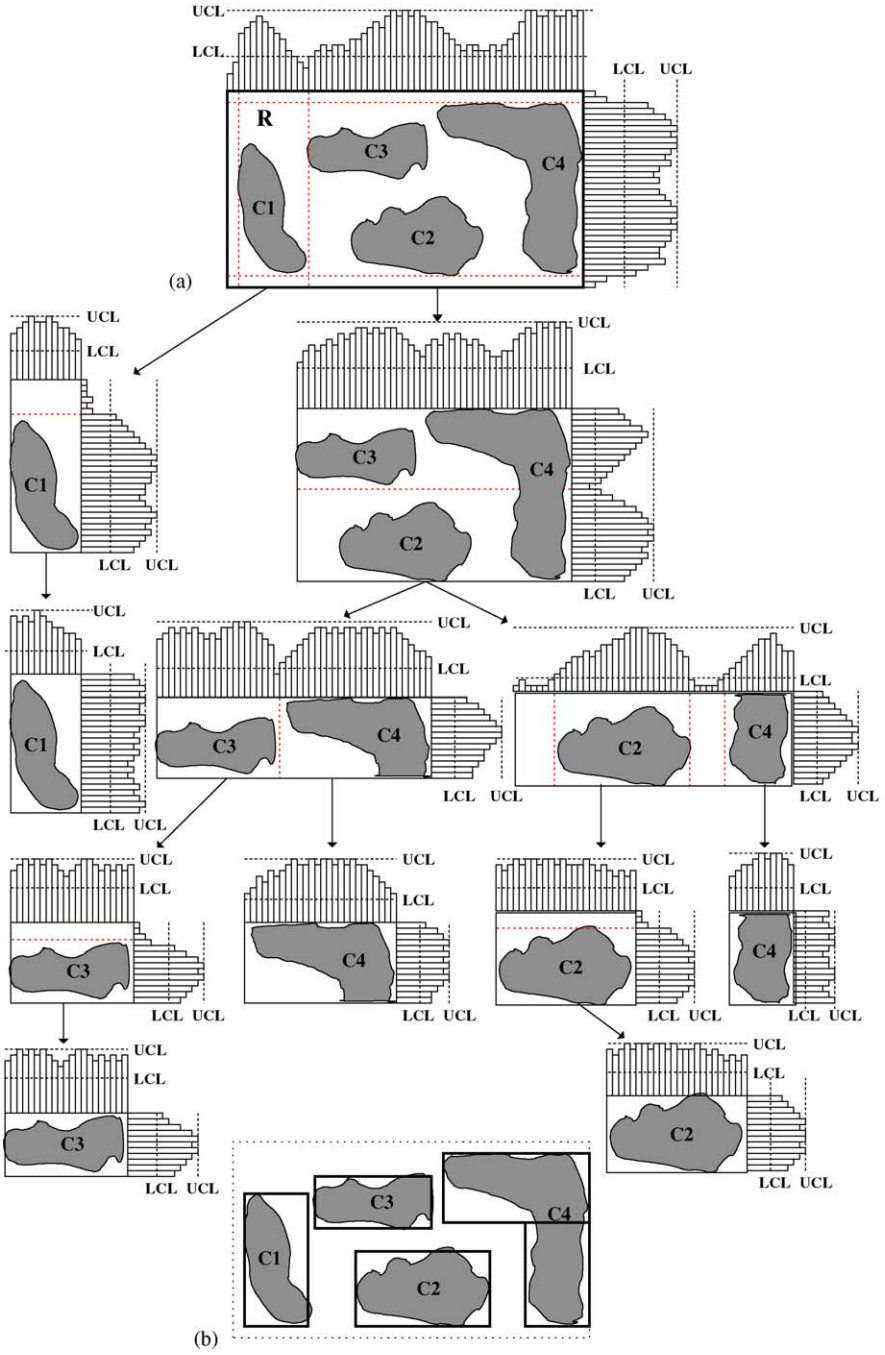

Fig. 12. The repairing of a non-homogeneous rule.

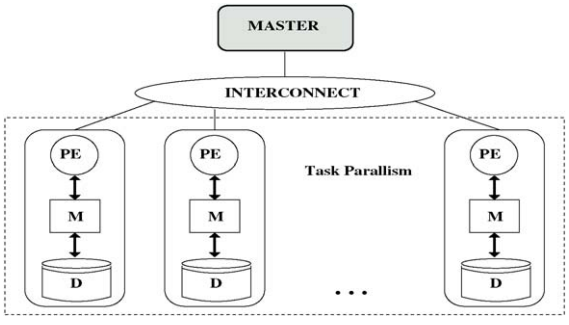

Fig. 13. Parallel pNOCEA architecture.

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.

- *Communication model*. It is vital to minimise the communication cost in a parallel system. Due to the coarse-grained granularity of pNOCEA, the average computation cost of threads is significantly higher compared to transmission cost. Furthermore, since each thread is entirely executed in one PE without interruption, the master 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. The actual communication is modelled via message passing, i.e. remote method invocation (RMI), between different PEs.

7. Post-processing discovered knowledge

7.1. 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 [3,9]. 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 three-dimensional dataset of Fig. 14, 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 Fig. 14 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 $X \times Y$. Various dimensionality reduction techniques, e.g. principal components analysis [22] 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.

NOCEA is absolutely insensitive to the presence of irrelevant features in high dimensional spaces, as opposed to traditional clustering techniques [30]. This is because NOCEA attempts to maximise both the homogeneity and data coverage of rules rather than to optimise some distance or density based criterion function. Hence, NOCEA is unusual in operating in the full-dimensional space, thereby avoiding artifacts produced by the joint projection of clusters in subspaces.

In practice, 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 NOCEA discovered the following rule-set for the dataset shown in Fig. 14:

$$\mathcal{R}_1: \quad \text{IF } (5 \le X \le 45) \wedge (0 \le Y \le 60) \wedge (30 \le Z \le 70) \text{ THEN } \mathcal{C}_1$$

$$\mathcal{R}_2: \quad \text{IF } (0 \le X \le 100) \wedge (80 \le Y \le 90) \wedge (0 \le Z \le 100) \text{ THEN } \mathcal{C}_2$$

$$\mathcal{R}_3: \quad \text{IF } (75 \le X \le 85) \wedge (20 \le Y \le 30) \wedge (0 \le Z \le 100) \text{ THEN } \mathcal{C}_3$$

Examination of the rules reveals that the information encapsulated within some specific conditions, e.g. $(0 \le X \le 100)$ 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

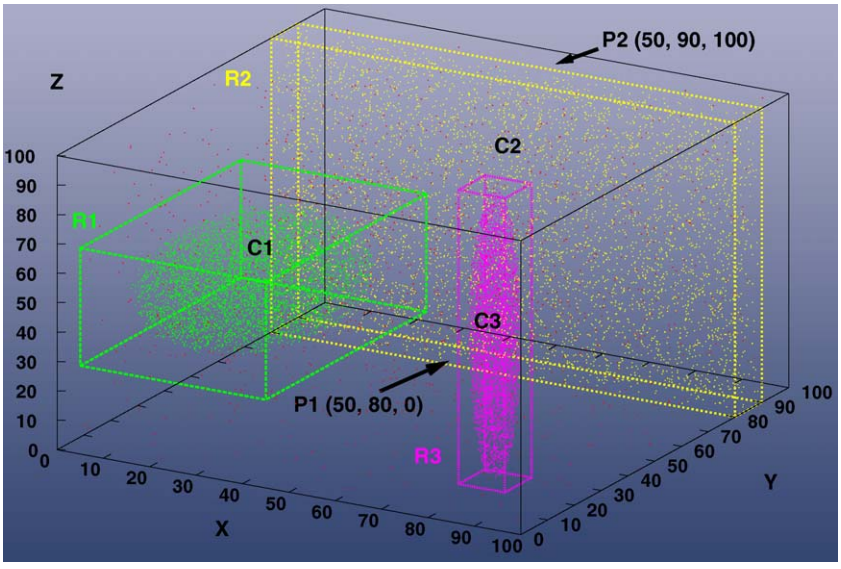

Fig. 14. Clusters embedded in different subspaces.

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, NOCEA compares the length of the rule in that dimension with the spreading of points along the entire dimension. Recall from Section 5.1 that an outlier-resistant estimator of the spreading of points in the ith dimension is the length l_E of the interval $E = [\max(a_i, (Q_{1i} - 1.5\mathrm{IQR}_i)), \min(b_i, (Q_{3i} + 1.5\mathrm{IQR}_i))]$, where Q_{1i}, Q_{3i}, and IQR_i denote the first quartile, third quartile, and the inter-quartile 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:

$$T_r \leq \left(\frac{u_{ij} - l_{ij}}{l_E} \right) \tag{16}$$

where l_{ij} and u_{ij} denotes the real-valued decoded values for the lower and upper bounds of the jth rule in the ith dimension. The setting for the input threshold $T_r \in (0, 1]$ is discussed in Section 8.

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 the simplification analysis, the rules in our example reduce to a more informative knowledge:

$\mathcal{R}_1$: IF $(5 \leq X \leq 45) \wedge (0 \leq Y \leq 60) \wedge (30 \leq Z \leq 70)$ THEN $\mathcal{C}_1$
$\mathcal{R}_2$: IF $(80 \leq Y \leq 90)$ THEN $\mathcal{C}_2$
$\mathcal{R}_3$: IF $(75 \leq X \leq 85) \wedge (20 \leq Y \leq 30)$ 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 [14,19,28]. Examples of irrelevant features in real-world seismic data along with their interpretation can be found in Section 9.7.

7.2. Assembling clusters

This section describes a bottom-up post-processing algorithm that assembles the genuine clusters from the discovered rules. Often real world databases contain correlated subsets of dimensions that lead to points getting aligned along arbitrary shapes in lower dimensional spaces. Clusters with non-convex geometry require multiple rules to obtain an accurate and homogeneous descriptor.

In this paper, 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 disjunctive normal form (DNF) expression, where each disjunct represents an axis-parallel rule. Once NOCEA converges, the genome of the best individual undergoes the following bottom-up grouping algorithm filling the consequent part of the rules with a cluster identifier.

Initially each rule belongs to a distinct cluster. Each step of the grouping 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 8.

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.

Fig. 15 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.

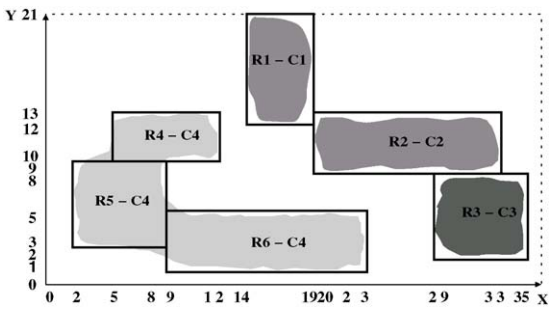

Fig. 15. Capturing non-convex clusters with rule-sets.

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

$$\begin{aligned}
&\text{IF } (14 \le X \le 19) \wedge (12 \le Y \le 21) \text{ THEN cluster } \mathcal{C}_1\\
&\text{IF } (20 \le X \le 33) \wedge (9 \le Y \le 13) \text{ THEN cluster } \mathcal{C}_2\\
&\text{IF } (29 \le X \le 35) \wedge (2 \le Y \le 8) \text{ THEN cluster } \mathcal{C}_3\\
&\text{IF } [(9 \le X \le 23) \wedge (1 \le Y \le 5)]\\
&\quad \vee [(2 \le X \le 8) \wedge (3 \le Y \le 9)]\\
&\quad \vee [(5 \le X \le 12) \wedge (10 \le Y \le 13)] \text{ THEN cluster } \mathcal{C}_4
\end{aligned}$$

8. Parameter settings

This section discusses the default parameter settings in NOCEA.

- *Population size and termination*. The default population size is 50. NOCEA terminates if at least one of the following conditions is true: when the number of generations that have been executed exceeds a pre-specified 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 $(d-1)$ 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. NOCEA implements a typical k-fold ($k = 4$) tournament selection scheme. In particular, each time an individual is requested for reproduction, k 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 (tournament size).
- *Recombination*. The recombination rate is the probability that recombination (instead of reproduction) is used to create new genomes. NOCEA applies the overlaid rule crossover (ORC) operator (Section 6.5) 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 6.3), 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 6.3 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, NOCEA adopts a middle-ground stochastic approach with variable T_g whose value for a given generalisation is drawn from a normal distribution $T_g = N(\mu,\ \sigma) : (0 \le T_g \le 1)$, 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 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 6.4.1) or seed (Section 6.4.2) with an equal probability.
- *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 6.6) 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 9.10.6). 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. NOCEA implements a simple elite-preserving replacement, where the best performing individual of the current population is directly copied to the new population. 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 7.1) 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

Table 1
Default parameter settings in NOCEA

Parameter name	Value
Population size	50
Generations	300
Termination condition	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	One random rule
Replacement strategy	Elitist (elite size = 1)
Sparsity threshold (T_s)	0.5% when $d < 5$, 0.1% otherwise
Homogeneity threshold (T_h)	0.3
Subspace threshold (T_r)	0.9
Generalisation threshold (T_g)	$N(0.65, 0.1)$
Clustering threshold (T_c)	0.2

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 7.2) 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 paper, 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 1 summarises the default settings for both EA- and clustering-related parameters used by NOCEA.

9. Evaluation

This section presents a real-world application of 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 geotectonics and seismicity associated with this region, is provided. The section 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 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. Finally, the section 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.

9.1. How are earthquakes generated-measured?

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 70 km 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.

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:

$$\log(n) = a - bM \tag{17}$$

where n is the number of earthquakes with seismic energy radiated greater than M (magnitude), while a and b are constants.

9.2. The ANSS earthquake catalogue

9.2.1. African–Eurasian–Arabian plate boundary

The Aegean sea and the surrounding area, which extends from the Italian peninsula, in the west (7°W), to the Karviola junction of the North (NAF) and East (EAF) Anatolian Faults in east Turkey (41°E), experience a rapid and intense crustal deformation [37,43]. 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 Fig. 16 delineates the Africa–Eurasia–Arabia plate collision zone.

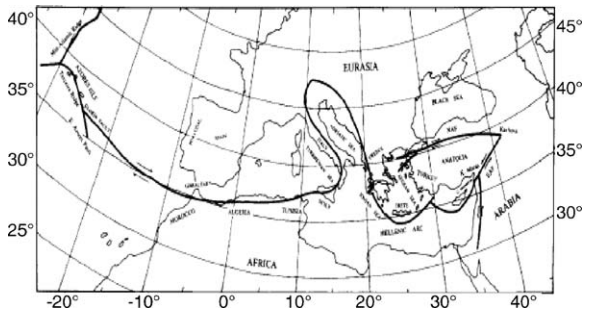

Fig. 16. Africa–Eurasia–Arabia plate boundary [37].

9.2.2. ANSS dataset description

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, 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 American Advanced National Seismic System (ANSS) (http://www.anss.org/) donated the earthquake dataset that is used in the paper. It contains 34,593 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 epicenter coordinates, one temporal dimension (time), and finally by the amount of energy (magnitude) that was released. Table 2 summarises the characteristics of the dataset. NOCEA operates on a [time × longitude × latitude × depth × magnitude] axis-aligned rectangular grid with resolution [l25 days × 0.1° × 0.1° × l.0 km × 0.1 Ric]. The third column represents the bin width computed by the TSQ quantisation algorithm (Section 5) while the last column shows the precision of the recorded measurements.

9.2.3. Visual clustering of the ANSS dataset

This section provides a preliminary human-eye clustering of the ANSS dataset.

Table 2
Various characteristics of the ANSS earthquake dataset

Size = 34,593	Domain	#Bins	Bin width	Precision
Time (days)	01/1961–04/2004	126	125	1
Longitude (°E)	7–45	381	0.1	0.1
Latitude (°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

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 [37]. However, as shown in Section 9.3, not all complex structures in a multidimensional space can be always extracted by a non fully-dimensional projective clustering method. Fig. 20 displays the spatial [longitude × latitude × depth] distribution of earthquakes. Fig. 19(a–e) depict the uni-dimensional frequency histograms for all dimensions, while Figs. 17 and 18 illustrate some pair-wise

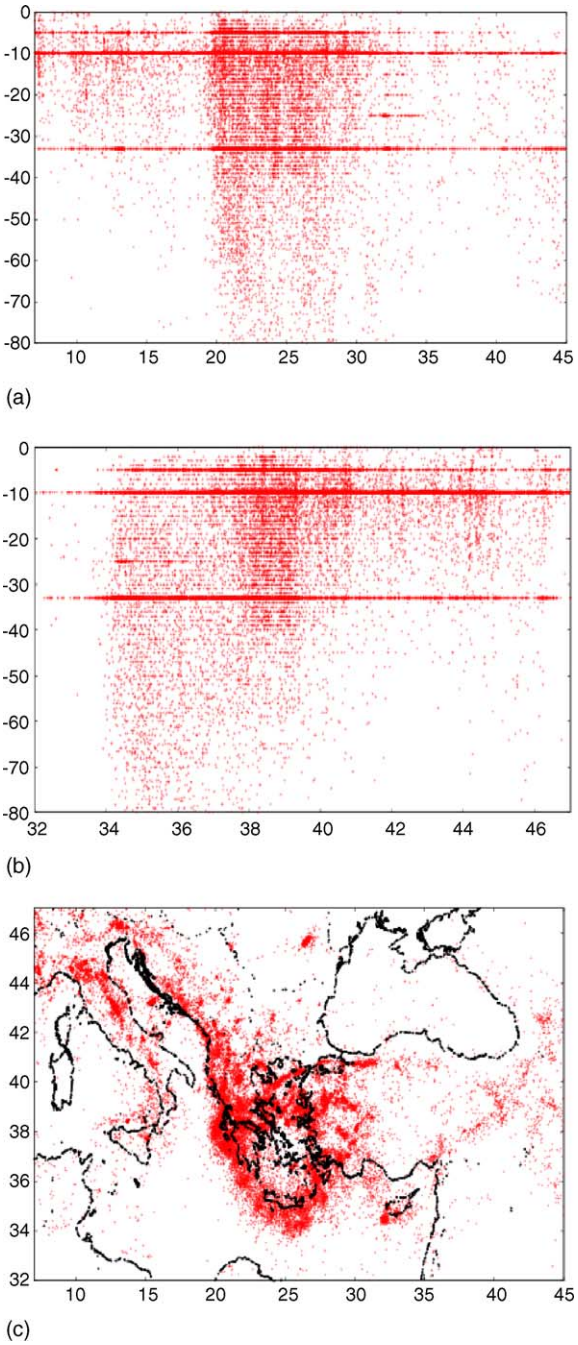

Fig. 17. Pair-wise projections of seismic events (A). (a) Longitude × depth; (b) latitude × depth; (c) longitude × latitude.

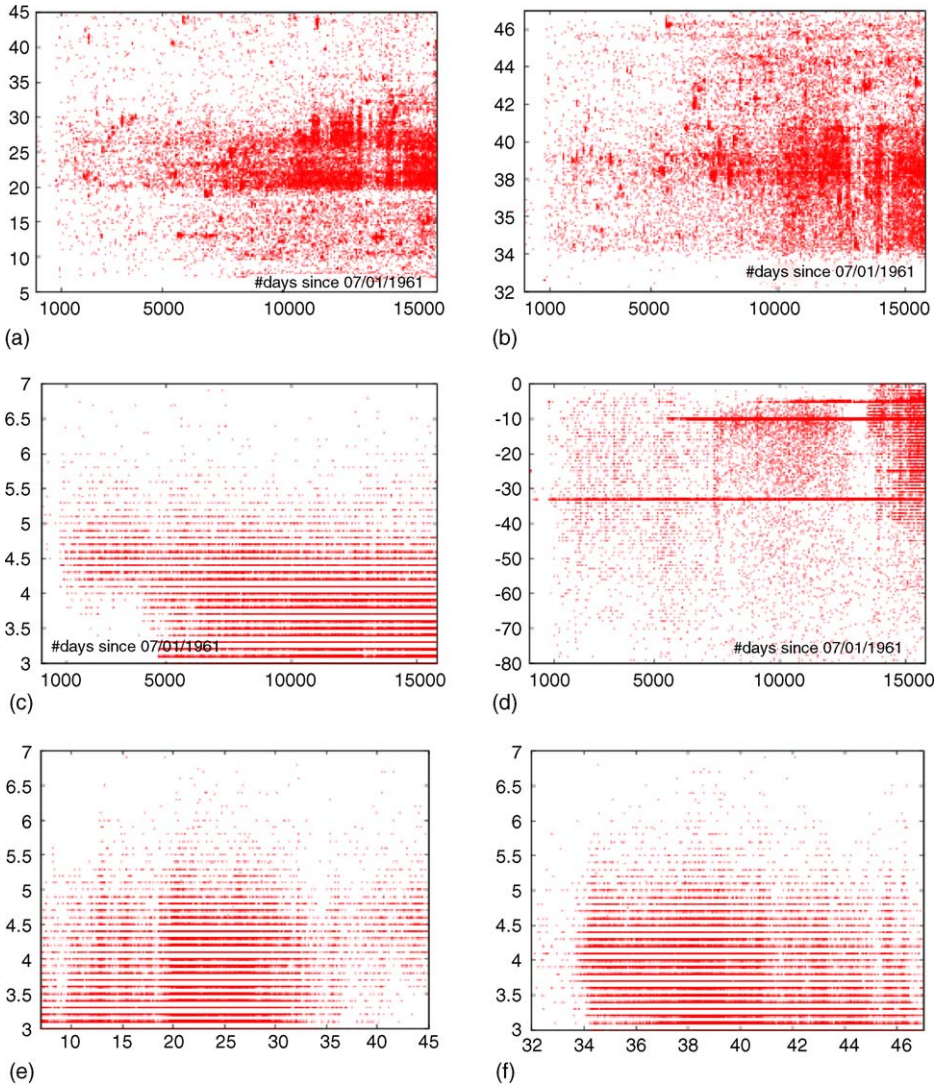

Fig. 18. Pair-wise projections of seismic events (B). (a) Time × longitute; (b) time × latitude; (c) time × magnitude; (d) time × depth; (e) longitude × magnitude; (f) latitude × magnitude.

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 (Fig. 17(c)), particularly in the Aegean Sea, forming variable-length slices along the depth axis as clearly shown in Figs. 20 and 17(a and b). These events are located mainly close to the surface 0–50 km. Another interesting observation that can be easily discerned from Figs. 19(e) and 17(a and b) is that the vast majority of seismic activity is confined along three shallow, remarkably thin (l km), horizontal slices located at focal depths 2–3 km, 9–10 km, and 32–33 km that are highlighted with purple, dark-blue, and cyan colour in Fig. 20, respectively. These 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 (Fig. 19(d)). This finding obeys the G–R power-law (see Section 9.1), 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 (Fig. 19(a)) is attributed to the incompleteness of the instrument measurements in the past, and does not reflect a chronologically increasing seismic excitation.

9.3. Evolutionary-based clustering of ANSS

This section presents the knowledge discovered by 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.

9.3.1. Analysing the ANSS dataset

NOCEA discovered 237 highly-homogeneous hyper-rectangular rules forming 121 distinct clusters over the course of 300 generations. Some clusters have quite complex spatio-

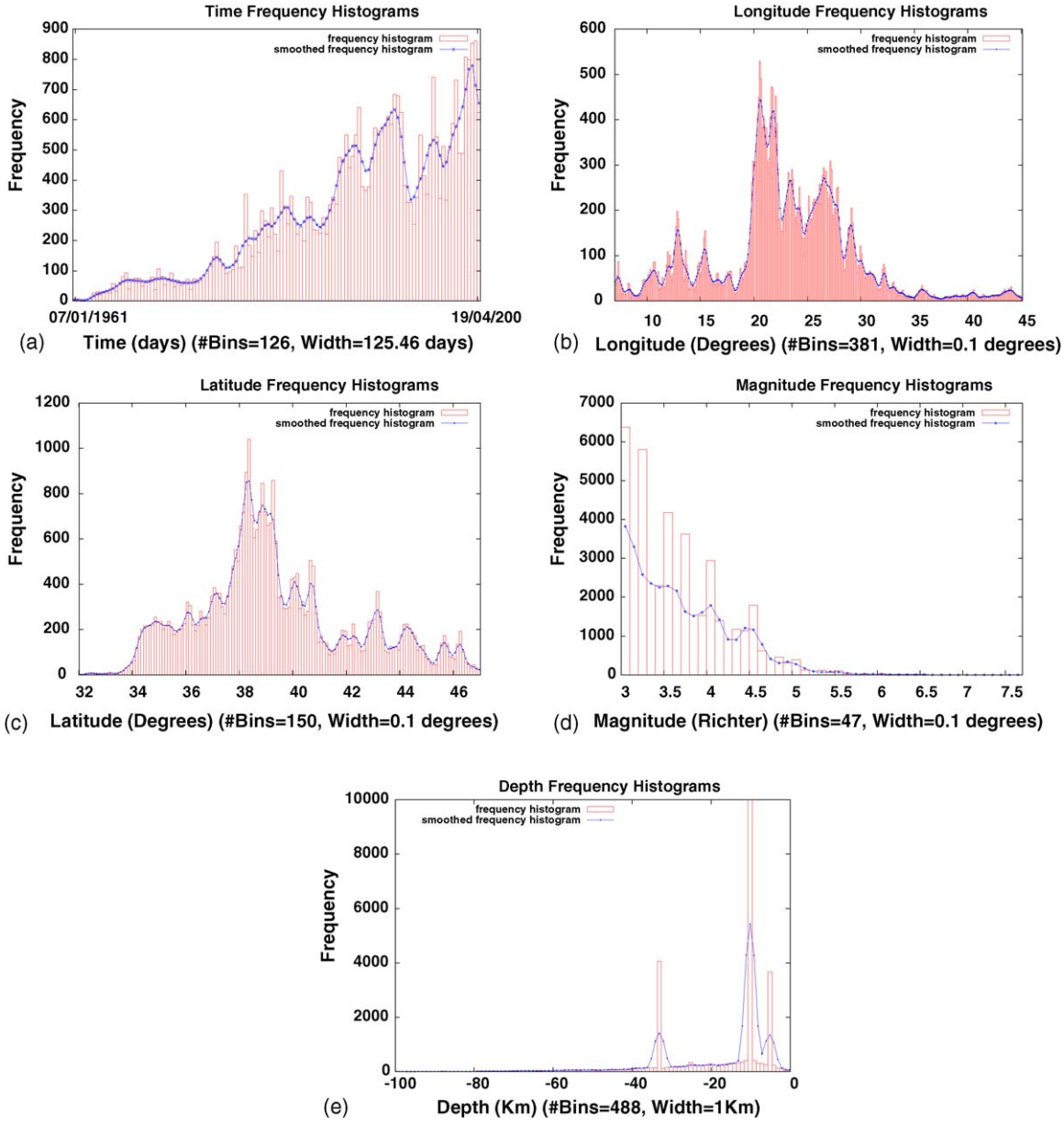

Fig. 19. Uni-dimensional frequency histograms for the earthquake dataset.

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 ($N_{\text{total}} = 34{,}593$), while the remaining points (23%) were considered non-uniform background noise by NOCEA. Fig. 21 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 NOCEA was performing local fine-tuning. The complete set of cluster descriptors is given in Appendix B.

Not surprisingly, most of the discovered clusters, especially those with arbitrary shapes, are not distinguishable in most non-fully-dimensional projections (Section 9.2.3), mainly due to the large degree of overlapping among the points of these clusters in lower dimensional projections. 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.

9.3.2. Parameter settings

Table 3 summarises the configuration settings for both the EA- and clustering-related parameters used by NOCEA to mine the ANSS dataset. All experiments reported in Section 9 have been performed on a Linux Fedora Core 2 workstation with an Intel(R) Xeon(TM) CPU 3.06 GHz processor, 512KB cache, 2GB of DRAM, and 9GB of IDE disk.

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

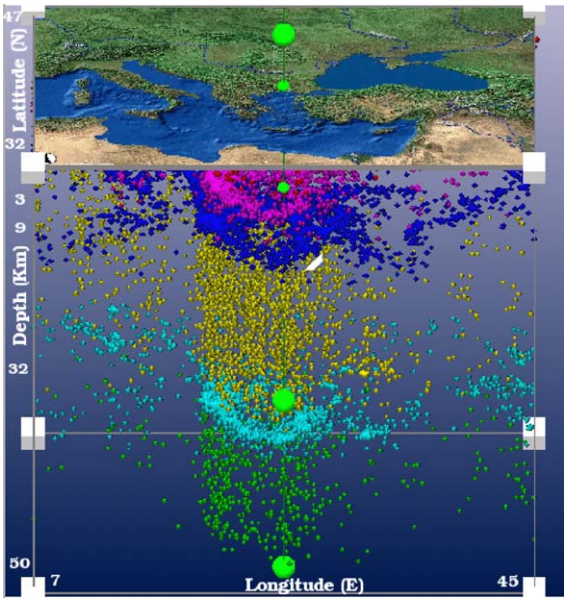

Fig. 20. Spatial [longitude × latitude × depth] distribution.

Table 3
Parameter settings for the ANSS earthquake dataset

Parameter name	Value
Population size	50
Generations	300
Termination condition	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	A randomly generated rule
Replacement strategy	Elitist (elite size = 1)
Sparsity threshold (T_s)	0.1% (0.025% high-magnitude rules)
Homogeneity threshold (T_h)	0.3
Subspace clus. threshold (T_r)	0.9
Generalisation threshold (T_g)	$N(0.65, 0.1)$
Clustering threshold (T_c)	0.2

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, 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. NOCEA terminates after a pre-specified 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.

As far as the clustering-related parameters of NOCEA are concerned, the following standard configuration (Section 8) 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

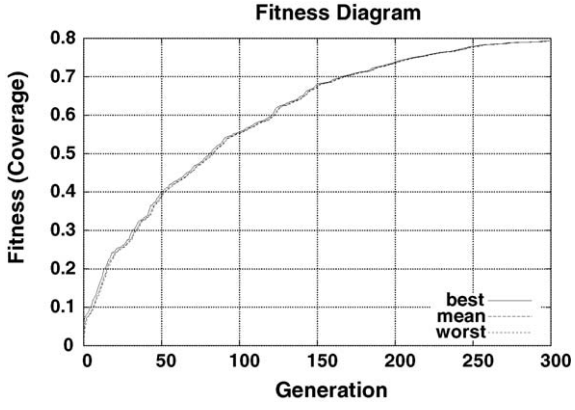

Fig. 21. Fitness diagram for the best–mean–worst solution.

and its extensions (see Section 9.10.1) 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 9.1). 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.

9.3.3. 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 [36]. Although the generation of an earthquake is not always localised around its source, intense seismic activity that has not been associated with a

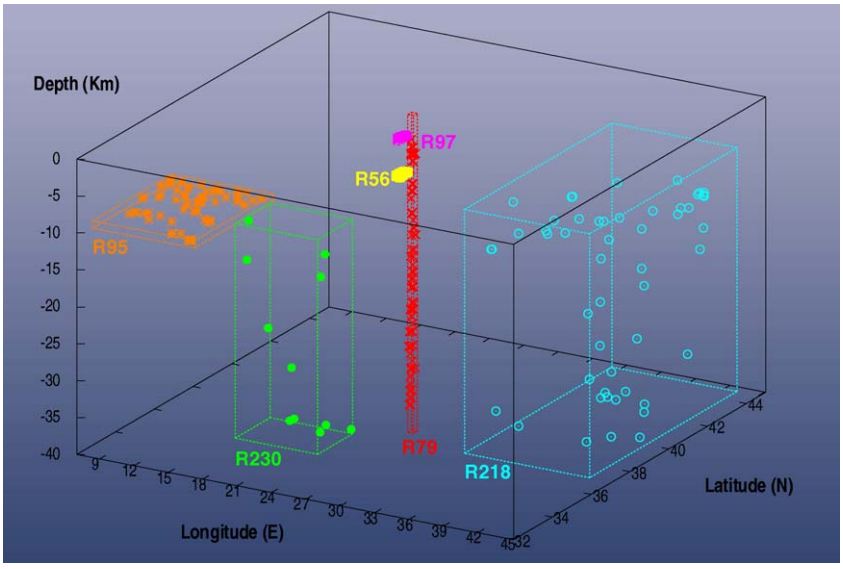

Fig. 22. Arbitrary density, size, and geometry rules in [longitude × latitude × depth].

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 previous types.

9.4. Arbitrary density rules

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.

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. $\mathcal{R}_{97}$ ($3.0 \leq$ magnitude ≤ 3.1) and $\mathcal{R}_{56}$ ($4.0 \leq$ magnitude ≤ 4.2) each cover 54 shallow aftershocks that occurred at focal depths 5 and 10 km, respectively, following the 13/05/1995 destructive earthquake in Kozani–Greece with magnitude 6.6 [46]. $\mathcal{R}_{218}$, on the other hand, is one of the sparser rules with density 1.51×10^{-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 ($5.2 \leq$ magnitude ≤ 6.2) in the EAF–Karviola junction, generated within a specific depth interval (4–37 km) from the beginning of the catalogue. The 3-D projections of $\mathcal{R}_{97}$, $\mathcal{R}_{56}$, and $\mathcal{R}_{218}$ shown in Figs. 22–24, 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

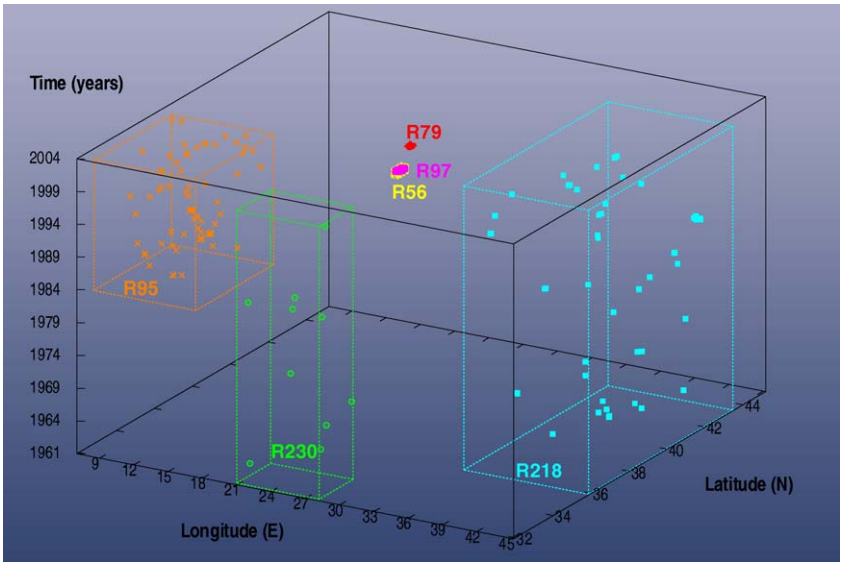

Fig. 23. Arbitrary density, size, and geometry rules in [longitude × latitude × time].

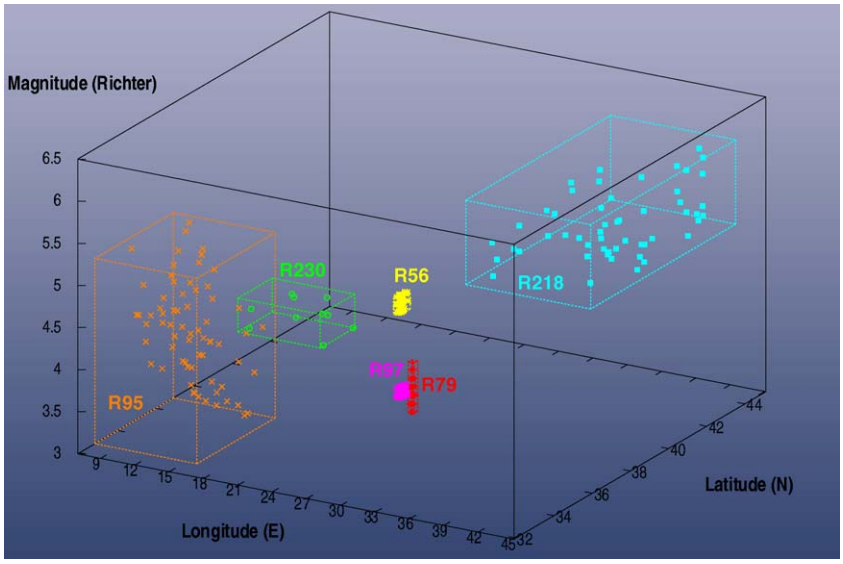

Fig. 24. Arbitrary density, size, and geometry rules in [longitude × latitude × magnitude].

are visualised with the same colour, which is different from other rules.

9.5. Arbitrary geometry and size rules

The ability to self-adjust well to the geometry and size of non-spherical clusters, is one of NOCEAs most appealing properties for real-world clustering. In 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. 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 $\mathcal{R}_{230}$. Rules $\mathcal{R}_{95}$, $\mathcal{R}_{56}$, and $\mathcal{R}_{218}$ are briefly discussed in Section 9.4. $\mathcal{R}_{79}$ 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 [45]. 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 42 km (0–42 km), as shown in Fig. 22. 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 ($4.8 \leq$ magnitude ≤ 5.2) extending to shallow depths ($5 \leq$ depth ≤ 33 km) through coastal north-eastern Libya and the Mediterranean Sea south of Crete and can be viewed as an extension of the Hellenic Arc to the south. $\mathcal{R}_{95}$ 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 [43]. The geometry of $\mathcal{R}_{95}$ is characterised by (a) thin concentration ($9 \leq$ depth ≤ 10 km) along the depth axis, (b) widespread magnitude interval ($3.0 \leq$ magnitude ≤ 5.2), and (c) shorter time span of approximately 20 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. Figs. 22–24 shows 3-D projections of these rules in spatial [longitude × latitude × depth], spatial-temporal [longitude × latitude × time] and spatial-magnitude [longitude × latitude × magnitude] subspaces, illustrating NOCEAs ability to discover rules with wide diversity in size and geometry.

9.6. Arbitrary data coverage rules

Fig. 25 plots a classical frequency histogram [51,53,56] 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%. The 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

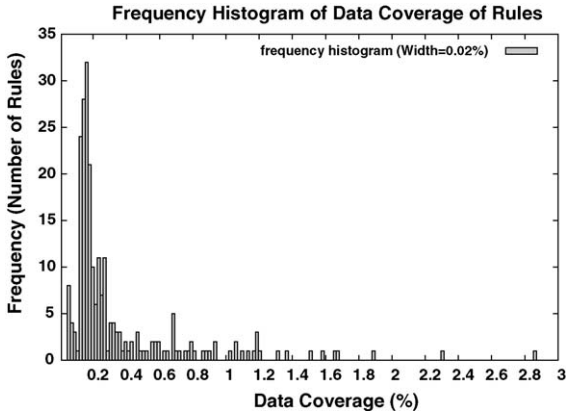

Fig. 25. Frequency histogram for the data coverage of rules.

preference bias, but it rather reflects NOCEAs intrinsic ability to directly inherit the size-geometry of rules, and consequently their point coverage, from the underlying data distribution. In the ANSS example 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 that 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 NOCEA has the ability to reveal such tiny structures, regardless of their geometry and density.

9.7. 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. NOCEA performed effective subspace clustering for the ANSS earthquake dataset. More specifically, in Tables B1–B4, empty fields in the antecedent part of the rules correspond to irrelevant feature-genes that are detected by NOCEAs

Table B1
Clustering rules for the ANSS earthquake dataset (A)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnitude
$\mathcal{R}_2$	$\mathcal{C}_2$	993–2.87	7.4×10^{-3}	30/06/1988–26/05/1996	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}	30/06/1988–26/05/1996	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}	29/11/1979–05/02/1986	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}	17/11/1978–27/09/1996	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}	09/01/1995–26/05/1996	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}	08/03/1989–07/12/1991	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}	06/05/1983–27/09/1996	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}	15/08/1992–09/01/1995	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}	14/11/1989–15/05/1995	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}	30/06/1988–27/09/1996	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}	03/11/1988–29/12/1993	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}	23/07/1990–26/05/1996	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}	30/06/1988–20/03/1990	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}	23/07/1990–07/12/1991	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}	26/07/1979–01/02/1997	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}	13/03/1978–01/02/1997	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}	16/10/1975–19/04/2004	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}	17/06/1998–22/11/2001	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}	11/02/1998–20/07/2001	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}	30/06/1988–05/03/2000	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}	26/07/1979–19/04/2004	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}	30/06/1988–26/05/1996	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}	13/03/1978–05/06/1997	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}	17/06/1998–05/03/2000	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}	26/07/1979–23/10/1987	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}	22/11/2001–19/04/2004	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}	29/03/2002–19/04/2004	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}	06/05/1983–19/04/2004	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}	29/03/2002–19/04/2004	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}	09/10/1997–29/03/2002	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}	30/06/1988–17/06/1998	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}	25/08/1993–16/12/2003	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}	05/09/1971–06/08/1980	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}	17/06/1998–31/07/2002	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}	12/01/1984–17/09/1995	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}	08/10/1963–29/11/1979	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}	07/12/1991–26/05/1996	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}	05/09/1971–11/02/1998	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}	07/10/1974–04/03/1977	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}	12/01/1984–19/04/2004	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 B2
Clustering rules for the ANSS earthquake dataset (B)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnitude
$\mathcal{R}_4$	$\mathcal{C}_4$	567–1.63	1.7×10^{-3}	15/07/1978–08/04/2003	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}	09/12/1980–17/09/1995	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}	26/02/1988–26/05/1996	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}	26/02/1988–22/04/1993	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}	29/11/1979–09/07/2000	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}	26/02/1988–09/01/1995	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}	23/03/1979–27/09/1996	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}	05/02/1986–29/03/2002	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}	01/06/1974–29/03/2002	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}	10/01/1972–05/02/1986	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}	16/01/1973–16/03/2001	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}	29/08/1982–26/05/1996	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}	04/03/1977–21/10/1998	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}	03/11/1988–19/04/2004	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}	09/12/1980–30/06/1988	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}	07/10/1974–19/04/2004	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}	26/07/1979–14/02/1987	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}	13/03/1978–29/11/1979	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}	04/03/1977–23/10/1987	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}	14/06/1964–01/02/1997	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}	09/05/1972–19/04/2004	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}	12/09/1972–19/04/2004	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}	04/06/1963–06/05/1983	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}	15/02/1963–06/08/1980	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}	17/06/1998–19/04/2004	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}	16/03/2001–16/12/2003	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}	27/09/1996–09/07/2000	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}	11/02/1998–16/03/2001	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}	09/07/2000–22/11/2001	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}	09/10/1997–01/11/1999	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}	09/09/1983–19/04/2004	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}	20/03/1990–09/01/1995	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}	09/01/1995–19/04/2004	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}	08/07/1977–19/04/2004	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}	27/09/1996–29/03/2002	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}	09/07/2000–29/03/2002	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}	06/05/1983–26/02/1988	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}	22/11/2001–19/04/2004	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}	29/03/2002–19/04/2004	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}	11/02/1998–19/04/2004	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}	23/10/1987–27/09/1996	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}	20/06/1987–15/08/1992	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}	05/02/1986–30/06/1988	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}	09/12/1980–05/02/1986	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}	29/11/1979–20/01/1996	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}	19/09/1984–17/09/1995	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}	05/02/1986–25/08/1993	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}	17/06/1998–16/03/2001	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}	23/03/1979–08/04/2003	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}	17/06/1998–22/11/2001	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}	22/11/2001–19/04/2004	7.00–24.1	40.6–45.2	10–21	3.0–3.4

Table B2 (*Continued*)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnitude
$\mathcal{R}_5$	$\mathcal{C}_5$	313–0.90	6.3×10^{-3}	25/04/1982–08/04/2003	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}	25/04/1982–19/04/2004	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}	29/03/2002–19/04/2004	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}	05/03/2000–19/04/2004	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}	21/10/1998–05/03/2000	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}	01/02/1997–17/06/1998	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}	01/02/1997–17/06/1998	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}	15/08/1992–09/01/1995	26.1–28.2	37.8–39.3	9–10	3.0–3.4

Table B3
Clustering rules for the ANSS earthquake dataset (C)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnitude
$\mathcal{R}_{27}$	$\mathcal{C}_{20}$	229–0.66	2.7×10^{-4}	13/03/1978–19/04/2004	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}	12/06/1975–01/02/1997	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}	05/09/1971–19/04/2004	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}	16/03/2001–19/04/2004	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^{-5}	14/11/1989–06/09/1994	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}	25/08/1993–20/01/1996	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}	06/09/1994–20/01/1996	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}	06/05/1983–19/04/2004	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}	26/07/1979–19/04/2004	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}	23/02/1999–16/03/2001	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^{-5}	01/02/1997–01/11/1999	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}	26/02/1988–27/09/1996	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}	09/12/1980–20/06/1987	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}	09/01/1995–27/09/1996	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}	25/08/1993–19/04/2004	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}	17/06/1998–04/12/2002	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}	29/03/2002–19/04/2004	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}	15/07/1978–31/07/2002	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}	26/10/1976–01/02/1997	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^{-1}	14/06/1964–12/01/1984	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}	12/01/1984–20/01/1996	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}	29/06/1965–01/11/1999	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}	25/09/1973–19/04/2004	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}	16/12/1969–19/04/2004	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}	31/07/2002–16/12/2003	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}	17/08/1981–09/01/1995	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^{-1}		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}	23/03/1979–26/07/1979	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}	11/02/1998–19/04/2004	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}	30/06/1988–20/07/2001	10.8–19.1	42.1–47.0	4–5	3.0–3.3

Table B3 (*Continued*)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnitude
$\mathcal{R}_{165}$	$\mathcal{C}_{78}$	35–0.10	1.7×10^{-4}	20/07/2001–19/04/2004	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}	15/08/1992–09/01/1995	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}	22/04/1993–09/01/1995	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}	20/02/1965–23/03/1979	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}	20/07/2001–22/11/2001	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}	25/04/1982–29/03/2002	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}	23/10/1987–20/07/2001	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}	14/02/1987–07/12/1991	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}	01/01/1983–06/05/1983	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}	08/07/1977–19/04/2004	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}	06/08/1980–04/12/2002	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}	05/06/1997–17/06/1998	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	01/10/1985–09/06/1986	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}	06/09/1994–26/05/1996	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}	26/07/1979–19/04/2004	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}	14/11/1989–20/03/1990	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}	30/06/1988–03/05/1994	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}	09/01/1995–19/04/2004	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}	08/03/1989–19/04/2004	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}	17/05/1984–19/04/2004	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}	17/06/1998–20/07/2001	18.4–34.0	39.0–41.2	3–9	3.4–3.6

post-processing simplification algorithm of Section 7.1. It can be easily discerned that the vast majority of rules are embedded in the full 5-D 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 [43], strain (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 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.

The remainder of this section discusses the interpretation of the presence of irrelevant features in the ANSS dataset. Fig. 26 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 1,022,084 km^2, the repetition time of relatively shallow depth (4–37 km) earthquakes in this region having magnitude in the range [5.2–6.5] is at least $[(2004-1961)/52][1,022,084/(100 \times 100)] \approx 84.5$ years per 100 km × 100 km 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

Table B4
Clustering rules for the ANSS earthquake dataset (D)

Rule	Cluster	Coverage	Density	Time	Longitude	Latitude	Depth	Magnitude
$\mathcal{R}_{107}$	$\mathcal{C}_{62}$	63–0.18	1.2×10^{-6}	07/01/1961–15/08/1992	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}	31/07/2002–19/04/2004	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}	23/07/1990–07/12/1991	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}	26/02/1988–06/09/1994	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}	30/06/1988–20/01/1996	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}	22/11/2001–19/04/2004	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}	26/11/1967–27/09/1996	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}	27/09/1996–01/02/1997	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}	01/06/1974–09/09/1983	19.4–22.8	37.2–38.8	32–33	4.0–4.2
$\mathcal{R}_{135}$	$\mathcal{C}_{80}$	56–0.15	1.2×10^{-1}	03/11/1988–01/02/1997	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}	15/05/1995–20/01/1996	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	06/09/1994–09/01/1995	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	15/05/1995–17/09/1995	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}	07/01/1961–07/12/1991	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}	06/09/1994–20/01/1996	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}	16/03/2001–22/11/2001	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}	18/02/1976–26/10/1976	12.8–13.4	46.0–46.6	32–33	3.0–51
$\mathcal{R}_{126}$	$\mathcal{C}_{76}$	51–0.14	1.4×10^{-2}	17/06/1998–19/04/2004	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}	25/04/1982–22/11/2001	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}	22/11/2001–12/08/2003	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}	16/10/1975–19/04/2004	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}	06/05/1983–19/04/2004	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}	08/03/1989–09/01/1995	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}	14/04/1981–19/04/2004	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}	06/11/1977–19/04/2004	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}	28/12/1970–19/04/2004	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}	30/06/1988–19/04/2004	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	15/05/1995–17/09/1995	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}	05/06/1997–21/10/1998	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}	26/01/1974–25/04/1982	21.1–25.6	37.9–39.5	32–33	3.0–3.1
$\mathcal{R}_{123}$	$\mathcal{C}_{75}$	44–0.12	4.2	08/03/1989–14/11/1989	23.3–23.8	39.2–39. 3	9–10	3.0–3.1
$\mathcal{R}_{167}$	$\mathcal{C}_{91}$	44–0.12	3.3×10^{-1}	05/03/2000–09/07/2000	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}	17/06/1998–19/04/2004	7.00–24.1	40.6–46.6	0–3	3.0–3.1
$\mathcal{R}_{117}$	$\mathcal{C}_{69}$	42–0.12	1	06/05/1983–09/09/1983	24.6–25.1	39.9–40. 3	9–10	3.4–3.6
$\mathcal{R}_{145}$	$\mathcal{C}_{82}$	42–0.12	4×10^{-7}	06/07/1962–26/11/1990	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}	01/01/1983–19/04/2004	26.4–26.9	45.4–45. 9	79–122	3.2–4.8
$\mathcal{R}_{116}$	$\mathcal{C}_{68}$	40–0.11	3.2×10^{-1}	26/05/1996–27/09/1996	27.0–27.5	35.9–36. 3	32–33	3.6–4.2
$\mathcal{R}_{160}$	$\mathcal{C}_{88}$	40–0.11	1.1×10^{-3}	22/04/1993–09/01/1995	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
$\mathcal{R}_{171}$	$\mathcal{C}_{94}$	39–0.11	1.6	31/07/2002–04/12/2002	13.5–13.9	38.3–38. 5	4–5	3.0–3.3
$\mathcal{R}_{172}$	$\mathcal{C}_{95}$	39–0.11	1.2×10^{-3}	29/03/2002–19/04/2004	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}	12/07/1989–25/08/1993	19.9–21.2	37.0–41. 2	4–5	3.3–3.9
$\mathcal{R}_{153}$	$\mathcal{C}_{86}$	38–0.10	9.2×10^{-7}	23/02/1964–06/08/1980	9.70–22.0	40.9–47.0	33–49	4.0–4.7
$\mathcal{R}_{181}$	$\mathcal{C}_{99}$	38–0.10	2.2×10^{-1}	09/12/1980–14/04/1981	22.8–23.5	37.9–38. 3	32–33	3.6–4.2
$\mathcal{R}_{28}$	$\mathcal{C}_{21}$	37–0.10	1.9×10^{-4}	20/06/1987–19/04/2004	19.9–21.7	43.6–46. 0	9–10	3.3–4.2
$\mathcal{R}_{182}$	$\mathcal{C}_{100}$	37–0.10	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}	14/04/1981–19/04/2004	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}	17/06/1998–22/11/2001	19.6–23.6	39.8–40.7	4–5	3.0–3.1
$\mathcal{R}_{110}$	$\mathcal{C}_{65}$	36–0.10	3.3×10^{-2}	01/01/1983–09/10/1997	7.00–7.50	44.2–44.7	9–10	3.0–3.1
$\mathcal{R}_{211}$	$\mathcal{C}_{110}$	36–0.10	7.6×10^{-4}	27/09/1996–19/04/2004	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}	17/09/1995–29/03/2002	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}	15/02/1963–14/04/1981	8.00–21.4	42.4–45.2	31–34	4.8–5.1
$\mathcal{R}_{220}$	$\mathcal{C}_{112}$	35–0.10	1.2×10^{-5}		26.1–26.8	45.4–45.9	105–172	4.8–5.5
$\mathcal{R}_{232}$	$\mathcal{C}_{118}$	35–0.10	1.9×10^{-7}	07/01/1961–25/08/1993	30.9–45.1	33.3–37.9	39–70	4.8–5.2
$\mathcal{R}_{234}$	$\mathcal{C}_{119}$	24–0.06	8.3×10^{-5}	18/02/1976–26/07/1979	9.30–22.5	41.1–47.0	9–10	4.9–5.1
$\mathcal{R}_{235}$	$\mathcal{C}_{120}$	15–0.04	5.4×10^{-2}	29/08/1982–12/01/1984	19.9–20.2	37.9–38.4	9–10	4.8–5.2
$\mathcal{R}_{225}$	$\mathcal{C}_{115}$	13–0.03	2×10^{-9}		21.5–45.5	36.8–47.0	48–146	5.5–7.1
$\mathcal{R}_{221}$	$\mathcal{C}_{113}$	13–0.03	1.9×10^{-5}		26.5–27.2	36.3–36.9	144–171	4.8–5.2
$\mathcal{R}_{230}$	$\mathcal{C}_{116}$	12–0.03	2.4×10^{-7}		20.8–28.0	32.0–33.8	5–33	4.8–5.2
$\mathcal{R}_{231}$	$\mathcal{C}_{117}$	11–0.03	4.1×10^{-7}	08/02/1975–19/04/2004	12.6–16.1	38.5–40.1	258–487	4.8–5.0

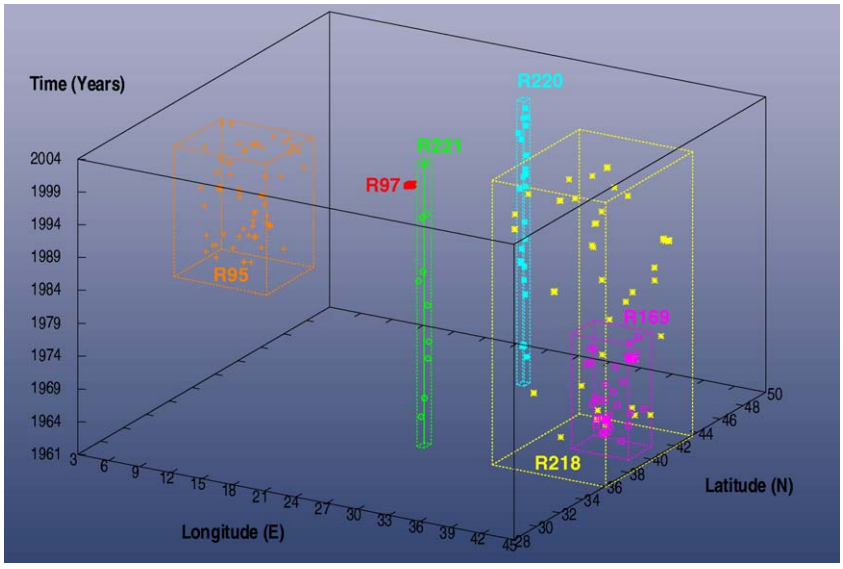

Fig. 26. Example rules with and without irrelevant time-dimension projected in [longitude × latitude × time].

similar time span, but very different period of activity. $\mathcal{R}_{95}$ is a currently active (17/05/1984 ≤ time ≤ 19/04/2004) 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 (15/02/1963 ≤ time ≤ 06/08/1980), 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 1960s and 1970s, e.g. Varto (1966), Bingol (1971), Lice (1975), Caldiran-Muradiye (1977) [1].

But what is the role of generalisation (Section 6.3) and histogram smoothing via KDE (Appendix A) in subspace clustering? Following the discussion above, 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 6.6. 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 7.1).

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, 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.

9.8. 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 [37]. It is essential to reveal these structures to gain a deeper insight into these intrinsically complex phenomena. In 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 7.2). 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 B1–B4 that the ANSS earthquake dataset, as expected, comprises many arbitrary-shaped clusters of varying numbers of rules, point coverage, density, and geometry. Figs. 27–29 depicts 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 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 in arbitrary directions. Finally, there are different degrees of overlapping among clusters in different subsets of dimensions at different data neighbourhoods. Although there are subspaces, e.g. [longitude × latitude × time], and [longitude × latitude × magnitude], where clusters are becoming blurred, NOCEA can easily distinguish them as it always operates in the full-

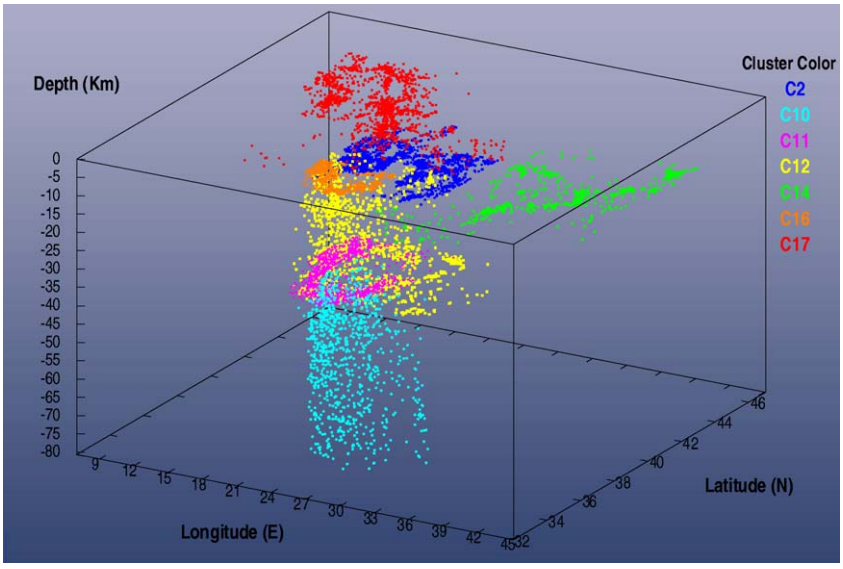

Fig. 27. Arbitrary-shaped clusters projected in [longitude × latitude × depth].

dimensional space, which, in turn, guards against artifacts formed in lower dimensional projections as in [5,41].

9.9. Interpreting an arbitrary-shaped cluster

The backbone of a non-convex cluster may be arbitrarily complex as discussed in Section 9.8. 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 $\mathcal{R}_{176}$. The epicenters 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 Fig. 30. 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 (9–10 km). Fig. 31 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 (Fig. 32) 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 expressions, helps seismologists to better understand the phenomenon. Finally, the extraction and interpretation of potential causal relationship(s) between the

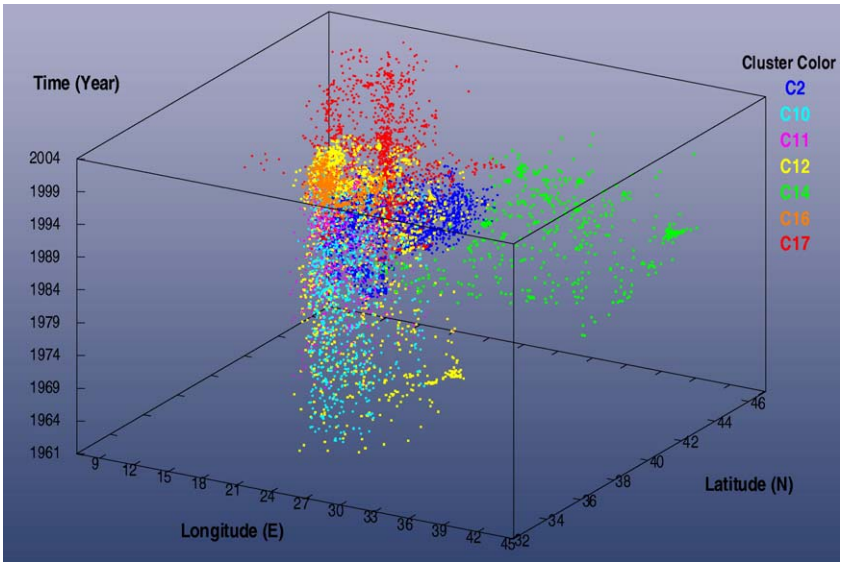

Fig. 28. Arbitrary-shaped clusters projected in [longitude × latitude × time].

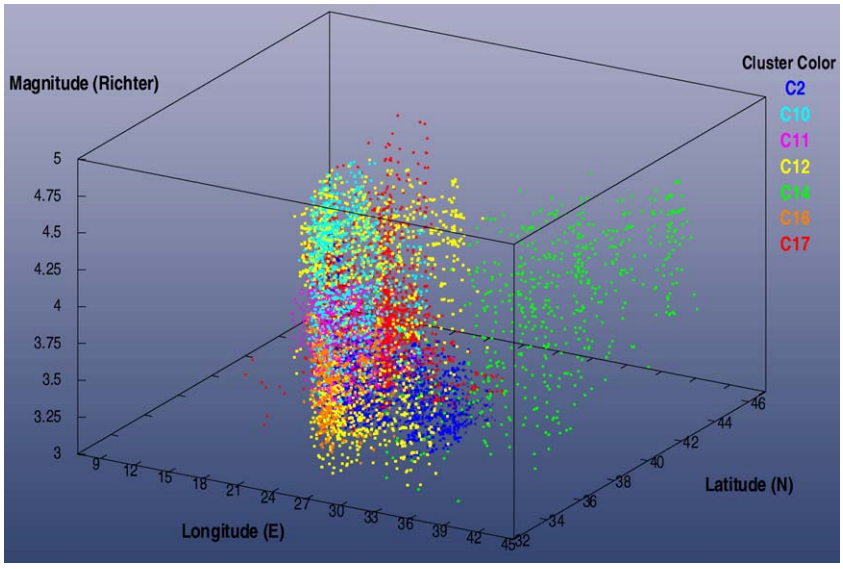

Fig. 29. Arbitrary-shaped clusters projected in [longitude × latitude × magnitude].

seismicity associated with the rules of $\mathcal{C}_4$ go beyond the scope of the paper.

9.10. Effectiveness and efficiency evaluation

In this section we evaluate the efficiency and effectiveness of NOCEA by conducting a variety of experiments using a mixture of synthetic and real-world datasets. In particular, the goals of the experiments are to assess:

- Efficiency: determine scalability with:
  - 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 NOCEA recovers correctly and accurately clusters that are embedded in some subspaces of a high dimensional space.

9.10.1. 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 [4,5,41,44]. 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 [4,5,41,44] were based on a limited number of clusters, i.e. 5. To address these limitations the generator described in [4] was modified to enforce the creation

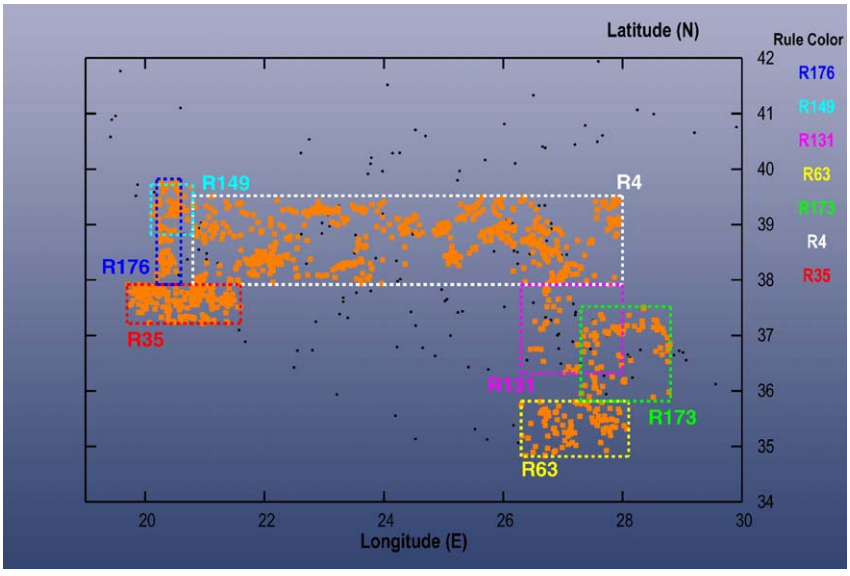

Fig. 30. The backbone of an arbitrary-shaped cluster ($\mathcal{C}_4$) projected in [longitude × latitude].

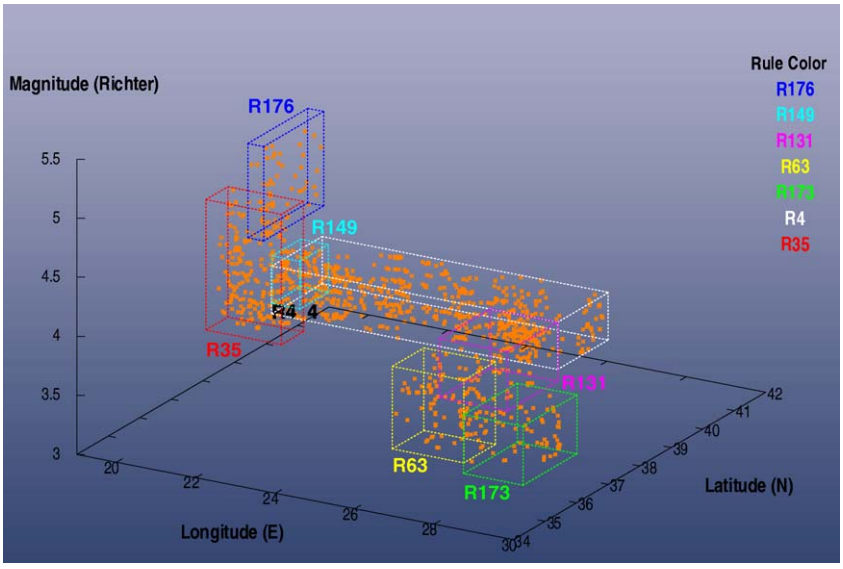

Fig. 31. The backbone of an arbitrary-shaped cluster ($\mathcal{C}_4$) projected in [longitude × latitude × magnitude].

of complex, real-world type structures consisting of numerous, i.e. 237, clusters based on the discovered rules from the ANSS catalogue.

Recall from Section 9.3 that NOCEA partitioned the 34,593 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 [4,44], 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 [4,44], 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

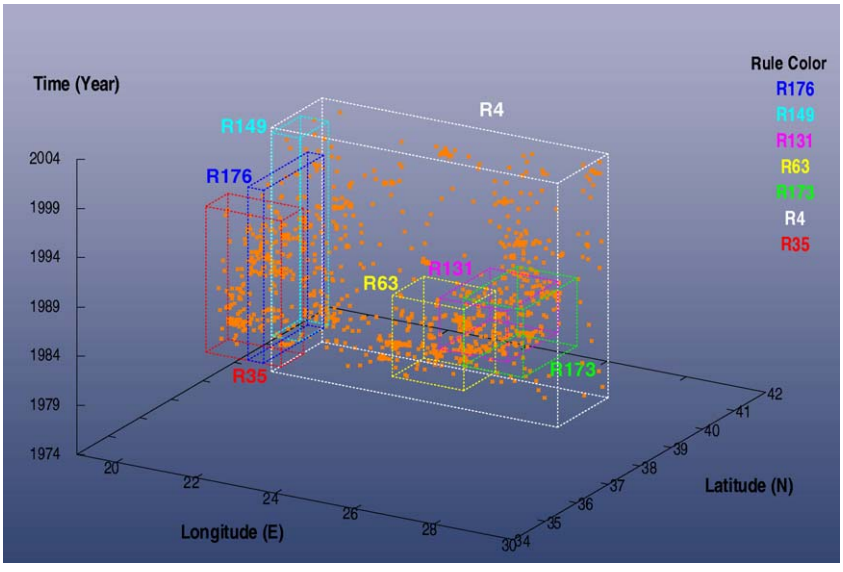

Fig. 32. The backbone of an arbitrary-shaped cluster ($\mathcal{C}_4$) projected in [longitude × latitude × time].

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 $(r \times s')^2$, where $r = s = 2$ [4].

Centers of 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 [4] 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 [44]. To force imbalance in the cluster sizes, the authors in [44] proposed computing initially (N_i) in a similar manner, but then setting for each $i \leq k/2$, $N_i = \delta N_i$ and $N_{i+k/2} = N_{i+k/2} + (1-\delta)N_i$, for $\delta \in \{0.2, 0.33, 0.5\}$. However, to the best of our knowledge, the most comprehensive studies [4,5,41,44] 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 NOCEAs performance under different distributions of the points in the subspace of bounded dimensions. Currently our generator supports the following 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 $[\bar{x} - 3\sigma, \bar{x} + 3\sigma]$, where $\bar{x}$ 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 $\bar{x}$ at the respective coordinate of the center point and variance determined above.
- *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, centered 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 centered 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 9.10.2–9.10.6 20 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 20 different random runs.

9.10.2. Scalability with database size

Fig. 33 depicts the scalability of 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 9.10.1. Note that most of the original rules were already embedded in the first five dimensions (e-features). Additionally, recall from Section 9.10.1 that the level of noise is approximately 23% of the

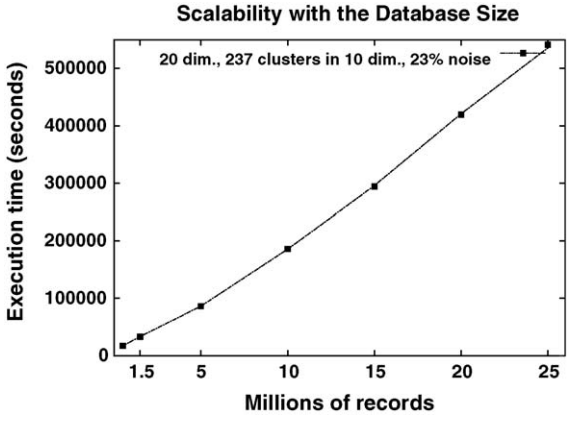

Fig. 33. Execution time vs. number of records.

database size. For this set of experiments, the distribution of points for the bounded dimensions of all clusters follow a uniform distribution. NOCEA successfully locates all input clusters within the course of 50 generations, on average.

Fig. 33 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 [5,41]. 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) [12].

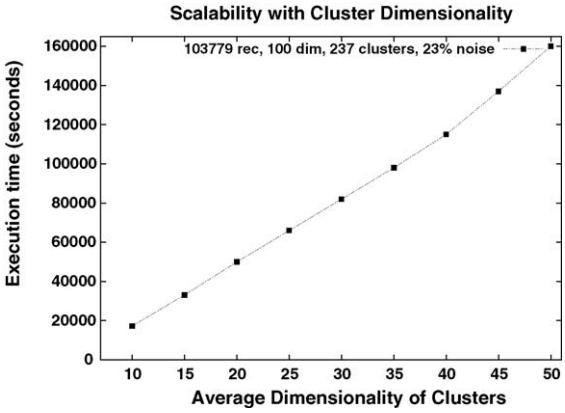

Fig. 35. Execution time vs. average cluster dimensionality.

9.10.3. Scalability with dimensionality of data

Fig. 34 shows the scalability of NOCEA as the dimensionality of the feature space increases from 20 to 200. In this series of experiments, each dataset has $3 \times 34{,}593 = 103{,}779$ records distributed over 273 clusters being embedded in some 10-dimensional subspace.

Clearly, the curve exhibits a super-linear trend. This behaviour is mainly due to the fact that for a given rule-set, 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.

9.10.4. Scalability with cluster dimensionality

Fig. 35 shows NOCEAs scalability as the average dimensionality of hidden clusters increases from 10 to 50 in a 100-dimensional space. In each case, the dataset has $3 \times 34{,}593 = 103{,}779$ records containing as before 273 clusters, and 23% noise.

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.

9.10.5. Scalability with task parallelism

In this section we study the scalability of pNOCEA (Section 6.7) under various task parallelisation schemes. pNOCEA supports task parallelism for the most expensive genetic operations such as, repairing-evaluation (E), mutation (M), recombination (R), and generalisation (G). Fig. 36 compares

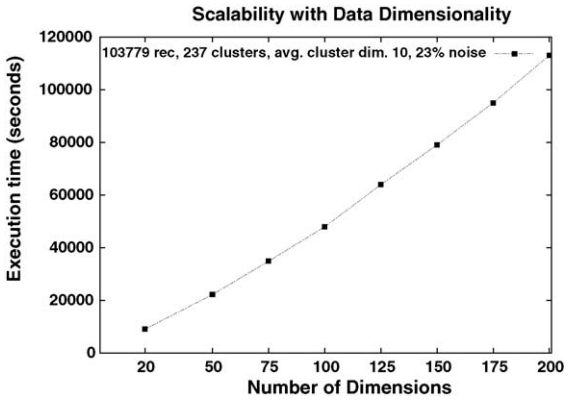

Fig. 34. Execution time vs. number of dimensions.

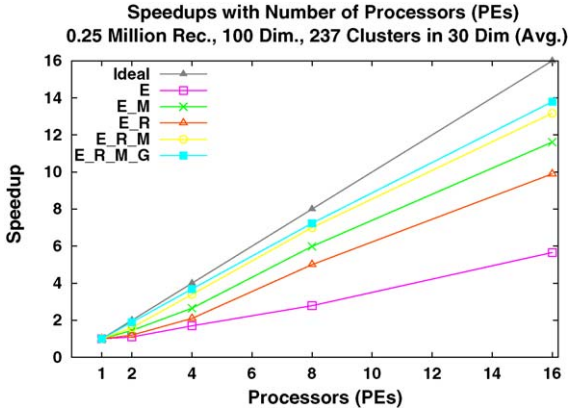

Fig. 36. Speedups for various parallelisations of pNOCEA.

the speedups achieved for various parallelisations of pNOCEA using a synthetic dataset that was generated as described in Section 9.10.1. The 100-dimensional dataset contains 242,151 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 pNOCEA. The speedup of pNOCEA with p processors over the sequential NOCEA with one processor is defined as t_1/t_p, where t_1 is the execution time of the sequential single-processor NOCEA while t_p is the execution time of pNOCEA with p processors [20].

All measurements have been performed on a network of homogeneous processing elements (PEs) (Intel(R) Xeon (TM) CPU 3.06 GHz, 512KB 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 remote method invocation (RMI) being implemented in Java™ 2 Standard Edition 1.4.2_05.

Not surprisingly, the fully-parallelised version of pNOCEA, 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 pNOCEA is strongly recommended.

9.10.6. *Effectiveness evaluation*

The goal of this section is two-fold:

- To assess the accuracy of 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. Since by definition 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 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, NOCEA missed some (on average 5 clusters over 20 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.

10. Conclusions and future work

This paper summarises the work developed in [47] which 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 paper 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 (reported in [47]) and real-world datasets lead us to conclude that it can.

We have developed a novel three-phase clustering methodology 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 databases.

The proposed methodology meets the following desiderata for DM clustering (see [47]):

- Effective treatment of high dimensionality and exceptional resistance to the curse of dimensionality; precise discrimination of clusters even in very sparse high-dimensional spaces.
- End-user comprehensibility of the results.
- Ability to discover clusters embedded in arbitrary subspaces of high dimensional data.
- Linear scalability with database size, and both data and cluster dimensionality.
- Substantial potential for task parallelism achieving a speed up of 13.8 on 16 processors.
- Ability to discover homogeneous clusters of arbitrary density, geometry, and data coverage.
- Insensitivity to initialisation and order of data.
- Substantial resistance to uniform noise.
- Minimal requirements for a priori knowledge (e.g. automatic determination of the optimal number of clusters and the subspace where each cluster is embedded) and no presumptions of any canonical distribution for the input data.
- Operating on the full dimensional space guards against artifacts formed by the joint projection of multiple clusters in lower dimensional spaces.
- Introduction of a simple non-distance or density based clustering criterion.
- Stochastic traversal of the search space that easily avoids local optima.

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. The discovered rules can aid seismologists in gaining a deeper insight into the phenomenon and allow them to improve the reliability of their estimates.

There are several avenues to extend this research and address its limitations:

- Data-parallelism to mine arbitrary-size datasets.
- Fine grain task-parallelism to enable several processors to work simultaneously on the same task.
- Introduction of alternative geometry and orientation rules to capture more complex trends.
- Use of non-uniform grids based on local distribution to delineate cluster bounds more accurately.
- Self-adaptation of clustering-related parameters.

Finally, ongoing research focuses on interpreting the earthquake clustering rules and applying the proposed methodology in climate datasets.

Acknowledgments

The authors gratefully acknowledge:

- The provision of the EoS evolutionary platform by British Telecommunications (BT) Intelligent Systems Laboratory (http://www.research.btexact.com/islab/FTGpublic/EosPlatform.html).
- The donation of the earthquake dataset by the American Advanced National Seismic System (ANSS) (http://www.anss.org/).
- The invaluable comments and suggestions on boundary Kernel theory by Professor Chris Jones from the Open University (http://www.open.ac.uk/).

Appendix A. Kernel density estimation

This section describes how the Kernel density estimation (KDE) method constructs a reasonably smooth approximation of the real density.

A.1. Kernel smoothing

KDE is a non-parametric technique for density estimation in which a known density function, the kernel K, is averaged across the observed data points to create a smooth approximation of the real density. Given n observations $X_1, \ldots, X_n$ 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..." [56] 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 [56]. The opposite should occur in regions where there are relatively few observations.

Normal	Epanechnikov
$K(v) = \frac{1}{\sqrt{2\pi}} e^{\frac{-v^2}{2}}, \; v \in \mathbb{R}$	$K(v) = \begin{cases} \frac{3}{4}(1-v^2), & \text{if } \lvert v\rvert < 1 \\ 0, & otherwise \end{cases}$

Fig. A1. The Normal and Epanechnikov kernel functions.

The shape of the bump is determined by a mathematical kernel K function. K 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 [51,53,56]. The kernel density estimate at point x is:

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

Fig. A1 illustrates two commonly used kernel functions, the Normal and Epanechnikov [51,53,56].

A.2. Automatic bandwidth selection

It has been widely recognised that the choice of the bandwidth h—rather than the shape of K—is crucial to the quality of the KDE, as it controls the amount of smoothness in the estimate of the density function [51,53,56]. 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.

Fig. A2 shows three kernel density estimates based on a sample of size $n = 1000$ from a density that is a mixture of three Gaussian distributions $N(0,1), N\left(-(3/2), (1/3)^2\right)$, and $N\left((3/2), (1/4)^2\right)$ with probabilities (1/2), (1/4), and (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, the value $h = 0.4$ reaches a good

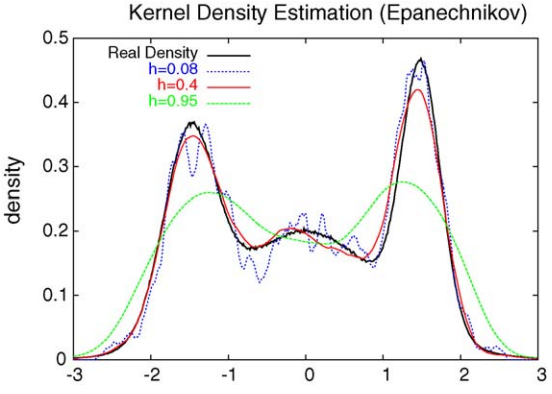

Fig. A2. Epanechnikov KDE for various smoothing.

compromise since both the essential structure of the distribution is recovered while most local artifacts are smoothed away.

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. If σ denotes the standard deviation of the data then h is [51]:

$$h = 1.144\sigma n^{-1/5} \tag{A.2}$$

Formulae (A.2) suggests a non-parametric way of computing the optimal bandwidth to be used with the normal kernel, and is expected to produce reasonable smoothing when the data distribution is close to normal. The optimal bandwidth ($h_{\text{Epan.}}$) for the Epanechnikov kernel can be computed using the normal scale bandwidth rule as starting point and then rescaling the obtained normal bandwidth (h_{Normal}) from Eq. (A.2) as [51]:

$$h_{\text{Epan.}} \approx 2.214 \times h_{\text{Normal}} \tag{A.3}$$

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 under-smoothing in areas with only sparse observations while over-smoothing 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 [51,53,56]. 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 pre-compute multiple bandwidths. To address these challenges, a new method for computing an optimal h that reflect the local data distribution is proposed (see Sections 5 and 6).

A.3. Binned Kernel density estimation

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 [formulae (A.1)] is very inefficient [53,56]. Consider, for example, the problem of obtaining a kernel density estimate over a mesh of M grid points, $g_1, \ldots, g_{\text{M}}$. Indeed, given a set of n observations, computing the KDE over the mesh of M points would require $O(nM)$ kernel evaluations [56]. This number can be much reduced if one uses kernels with compact 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 non-zero. 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 [56]. 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 [56].

Let K be a symmetric kernel with finite support confined on $[-t, t]$ $(t > 0)$. Additionally, let $[l, u]$ denotes 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 [56], the new binning algorithm proceeds by partitioning each one of the original bins (m) into $(p + 1)$ disjoint sub-intervals using p equispaced splitting sites $(1 \leq p)$. 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, $l = g_1 < \cdots < g_{\text{M}} = u$ with spacing $(w/(p + 1))$. Following the recommendation that M should be set to moderate values [56] (e.g. $M = 100$) so as to obtain a reasonable approximation of the exact density estimate, p is automatically computed as follows:

$$p = \max(1, \lceil (M - m - 1)/m \rceil) \tag{A.4}$$

The basic idea of binning KDE methods rely on rounding each observation by the nearest point from 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 $(i = 1, \ldots, M)$ has an associated bin count c_i, which 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 [56]:

$$f(j,h) = \frac{1}{nh}\sum_{i=1}^{M} K\left(\frac{g_j - g_i}{h}\right)c_i, \qquad j = 1, \ldots, M \tag{A.5}$$

Given that c_i is zero outside $[1, M]$ it follows that Eq. (A.5) can be rewritten as:

$$f(j,h) = \sum_{z=1-M}^{M-1} c_{j-z}k_z, \qquad j = 1, \ldots, M \tag{A.6}$$

where the kernel weight k_z is defined as:

$$k_z = \frac{1}{nh}K\left(\left(\frac{1}{h}\right)\frac{(u-l)z}{M-1}\right), \qquad z \in \mathbb{Z} \tag{A.7}$$

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 non-zero, that is $z = 0, \ldots, L$ where:

$$L = \min(\lfloor th(M-1)/(u-1) \rfloor, M-1) \tag{A.8}$$

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

$$f(j) = \sum_{z=-L}^{L} c_{j-z} k_z, \qquad j = 1, \ldots, M \tag{A.9}$$

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.

A.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 [56]. 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 [51,53,56].

In this paper, the boundary problem is tackled by employing special boundary kernels that are a linear multiple of the basic kernel K [formulae (A.1)] [56]. 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 the Epanechnikov kernel with support confined to $[-1, 1]$.

When estimating the density for grid points lying inside the main part of the distribution (h, $wm - h$) the ordinary binned kernel can be safely used because, centered 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 $g_j = \alpha h$ 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 [56]:

$$K^{\mathrm{L}}(v, \alpha) = \begin{cases} \dfrac{\nu_{2,\alpha}(K) - \nu_{1,\alpha}(K) v}{\nu_{0,\alpha}(K)\nu_{2,\alpha}(K) - \nu_{1,\alpha}(K)^2} K(v) & \text{if } (-1 < v < \alpha) \\ 0 & \text{otherwise} \end{cases} \tag{A.10}$$

where, $v = (g_j - g_i)/h$ and $\nu_{r,\alpha}(K) = \int_{-1}^{\alpha} x^r K(x)\,\mathrm{d}x$.

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:

$$\nu_{0,\alpha}(K) = \tfrac{3}{4}\alpha + \tfrac{1}{2} - \tfrac{1}{4}\alpha^3, \quad \nu_{1,\alpha}(K) = -\tfrac{3}{16}\alpha^4 - \tfrac{3}{16} + \tfrac{3}{8}\alpha^2,$$
$$\nu_{2,\alpha}(K) = -\tfrac{3}{20}\alpha^5 + \tfrac{1}{10} + \tfrac{1}{4}\alpha^5 \tag{A.11}$$

After the boundary correction the binned kernel density estimator at $g_j = \alpha h$ $(0 \leq \alpha < 1)$ point is [56]:

$$f(j, h, \alpha) = \frac{1}{nh} \sum_{i=1}^{M} K^{\mathrm{L}}(v, \alpha) c_i, \qquad j = 1, \ldots, M \tag{A.12}$$

where, $v = (g_j - g_i)/h$.

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

A.5. KDE for frequency histogram smoothing

The binned KDE with boundary correction 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.

Appendix B. ANSS earthquake clustering rules

Tables B1–B4 contain the complete set of cluster descriptors for the ANSS earthquake dataset discovered by NOCEA. For illustrative purposes, clusters are separated by a double space white line and are sorted on decreasing order of point coverage. Each rule is accompanied by 10 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 9.7 for an explanation), indicate irrelevant dimensions.

References

[1] Advanced National Seismic System (ANSS). URL: http://www.anss.org/.
[2] U.S. Geological Survey. URL: http://www.usgs.gov/.
[3] C. Aggarwal, A. Hinneburg, D. Keim, On the surprising behavior of distance metrics in high dimensional space, Lect. Notes Comput. Sci. 1973 (2001) 420–431.
[4] C.C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, J. Park, Fast algorithms for projected clustering, SIGMOD Rec. (ACM Special Interest Group on Management of Data) 28 (2) (1999) 61–72.
[5] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proceedings of the 1998 ACM-SIGMOD International Conference on Management of Data (SIGMOD'98), 1998, pp. 94–105.
[6] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, OPTICS: Ordering points to identify the clustering structure, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGM'99), vol. 28, 2 of SIGMOD Record, ACM Press, 1999, pp. 49–60.
[7] T. Bäck, D.B. Fogel, T. Michalewicz (Eds.), Evolutionary Computation 1: Basic Algorithms and Operators, Institute of Physics Publishing/Oxford University Press, 2000.
[8] T. Bäck, D.B. Fogel, T. Michalewicz (Eds.), Evolutionary Computation 2: Advanced Algorithms and Operators, Institute of Physics Publishing/Oxford University Press, 2000.
[9] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is ''nearest neighbour'' meaningful? in: Proceedings of the 7th International Conference on Data Theory, vol. 1540, LNCS, Springer-Verlag, 1999, pp. 217–235.
[10] C.-H. Cheng, A.W. Fu, Y. Zhang, Entropy-based subspace clustering for mining numerical data, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, 1999, pp. 84–93.

[11] I. Csiszar, J. Korner, Information Theory: Coding Theorems for Discrete Memoryless System, Academic Press, 1981.
[12] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry Algorithms and Applications, Springer-Verlag, 2000.
[13] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, 1973.
[14] M.H. Dunham, Data Mining: Introductory and Advanced Topics, Prentice-Hall, 2003.
[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, Density-connected sets and their application for trend detection in spatial databases, in: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97), AAAI Press, 1997.
[16] V. Estivill-Castro, Hybrid genetic algorithms are better for spatial clustering, in: Proceedings of the Pacific Rim International Conference on Artificial Intelligence (PRICAI'00), 2000, pp. 424–434.
[17] B.S. Everitt, Cluster Analysis, Edward Arnold, 1993.
[18] E. Falkenauer, Genetic Algorithms and Grouping Problems, Wiley, 1998.
[19] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, AAAI Press/The MIT Press, 1996.
[20] A.A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer-Verlag, 2002.
[21] A.A. Freitas, S.H. Lavington, Mining Very Large Databases with Parallel Processing, Kluwer Academic Publishers, Boston, 1998.
[22] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, 1990.
[23] A. Ghozeil, D.B. Fogel, Discovering patterns in spatial data using evolutionary programming, in: Proceedings of 1996 the First Annual Conference on Genetic Programming, MIT Press, 1996, pp. 521–527.
[24] D.E. Goldberg, Genetic Algorithms in Search, Optimisation, and Machine Learning, Addison-Wesley, 1989.
[25] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'98), ACM Press, 1998, pp. 73–84.
[26] L.O. Hall, I.B. Özyurt, J.C. Bezdek, Clustering with a genetically optimised approach, IEEE Trans. Evol. Comput. 3 (2) (1999) 103–112.
[27] L.C. Hamilton, Modern Data Analysis: A First Course in Applied Statistics, Brooks/Cole, 1990.
[28] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, 2000.
[29] J. Han, M. Kamber, A.K. Tung, Spatial clustering methods in data mining: a survey, in: H. Miller, J. Han (Eds.), Geographic Data Mining and Knowledge Discover, Taylor and Francis, 2001.
[30] A. Hinneburg, D. Keim, Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering, in: Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99), Morgan Kaufmann, 1999, pp. 506–517.
[31] A. Hinneburg, D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Proceedings of the 1998 International Conference on Knowledge Discovery and Data Mining, 1998, pp. 58–65.
[32] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, 1988.
[33] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comp. Surveys 31 (3) (1999) 264–323.
[34] G. Karypis, E.-H. Han, V. Kumar, Chameleon: hierarchical clustering using dynamic modeling, IEEE Comput. 32 (8) (1999) 68–75.
[35] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990.
[36] V.I. Keilis-Borok, Intermediate-term earthquake prediction, Proc. Natl. Acad. Sci. (93) (1996) 3748–3755.
[37] A. Kiratzi, C.B. Papazachos, Active crystal deformation from the azores triple junction to middle east, Tectonophysics (243) (1995) 1–24.
[38] K. Krishna, M.N. Murty, Genetic *k*-means algorithm, Syst. Man Cybernet. Part B: IEEE Trans. 29 (3) (1999) 433–439.
[39] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1996.
[40] B.L. Milenova, M.M. Campos, O-cluster: scalable clustering of large high dimensional data sets, in: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), IEEE Computer Society, 2002, pp. 290–297.
[41] H. Nagesh, S. Goil, A. Choudhary, Adaptive grids for clustering massive datasets, in: Proceedings of the First SIAM ICDM, Chicago, IL, USA, 2001.
[42] R. Ng, J. Han, Efficient and effective clustering method for spatial data mining, in: Proceedings of 1994 International Conference on Very Large Data Bases (VLDB'94), 1994, pp. 144–155.
[43] C.B. Papazachos, A. Kiratzi, A detailed study of the active crystal deformation in the aegean and surrounding area, Tectonophysics (253) (1996) 129–154.
[44] C.M. Procopiuc, M. Jones, P. Agarwal, T.M. Murali, A Monte Carlo algorithm for fast projective clustering, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Madison, USA, June 3–6, (2002), pp. 418–427.
[45] Z. Roumelioti, A. Kiratzi, N. Melis, Relocation of the 26 July 2001 Skyros island (Greece) earthquake sequence using the double-difference technique, Phys. Earth Plentary Interiors (138) (2003) 231–239.
[46] Z. Roumelioti, A. Kiratzi, N. Theodoulidis, C. Papaioannou, S-wave spectral analysis of the 1995 Kozani–Grevena (NW Greece) aftershock sequence, J. Seismol. (6) (2002) 219–236.
[47] I.A. Sarafis, Data mining clustering of high dimensional databases with evolutionary algorithms, PhD Thesis, Heriot-Watt University, Edinburgh, UK, 2005.
[48] I.A. Sarafis, P.W. Trinder, A.M.S. Zalzala, Mining comprehensible clustering rules with an evolutionary algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03), Chicago, USA, LNCS, Springer-Verlag, 2003.
[49] I.A. Sarafis, P.W. Trinder, A.M.S. Zalzala, Towards effective subspace clustering with an evolutionary algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03), Canberra, Australia, 2003.
[50] P. Scheunders, A genetic *c*-means clustering algorithm applied to color image quantisation, Pattern Recogn. 30 (6) (1997) 859–866.
[51] D.W. Scott, Multivariate Density Estimation, Wiley, New York, 1992.
[52] G. Sheikholeslami, S. Chatterjee, A. Zhang, WaveCluster: a wavelet based clustering approach for spatial data in very large databases, J. Very Large Data Bases (VLDB) 8 (4) (2000) 289–304.
[53] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman & Hall, 1986.
[54] R. Srikanth, R. George, N. Warsi, D. Prabhu, F.E. Petry, B.P. Buckles, A variable-length genetic algorithm for clustering and classification, Pattern Recogn. Lett. 16 (8) (1995) 789–800.
[55] G.R. Terrell, The maximal smoothing principle in density estimation, J. Am. Stat. Assoc. 85 (410) (1990) 470–477.
[56] M.P. Wand, M.C. Jones, Kernel Smoothing, Chapman & Hall, 1995.
[57] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, vol. 25, 2 of ACM SIGMOD Record, ACM Press, 1996, pp. 103–114.