Evaluation of Marine Engine Load Cycles using Machine Learning Methods

Dimitrios Vekios

Diploma Thesis

School of Naval Architecture and Marine Engineering
National Technical University of Athens

Supervisor: Assistant Prof. George Papalambrou

Committee Member : Prof. N. Kyrtatos
Committee Member : Prof. G. Zarafonitis

October 2019

Aknowledgements

This work has been carried out at the Laboratory of Marine Engineering (LME) at the School of Naval Architecture and Marine Engineering of the National Technical University of Athens, under the supervision of Assistant Professor George Papalambrou.

I owe my greatest appreciation to my thesis supervisor Assistant Professor George Papalambrou, for giving me the chance and motivation to work on this topic. I would also like to thank him for his patience, continuous support and immense knowledge. His guidance helped me in all the time working on this thesis.

I would also like to thank PhD candidate Nikolaos Planakis for providing me with valuable insight and supporting me throughout the entire process. His help and trust allowed me to gain valuable experience, while his knowledge and advice contributed to the completion of this thesis.

I take this opportunity to express gratitude to all LME fellow members for their help and support.

I am also sincerely grateful to my family and friends, for the unceasing encouragement, support, motivation and attention throughout my studies.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture. This accomplishment would not have been possible without them. Thank you.

Abstract

This diploma thesis focuses on marine engine load cycles analysis using data analytics and machine learning methods. Loading cycles of vessels is a complex field of study that requires consideration of several variables and whose study results can lead to decisions with high economic and environmental impact. The main objective of this thesis is to create a tool that is based on data analytics and machine learning and can properly classify loading cycles into groups based on their similarity and create representative load cycles, which can be used for better understanding of the loading patterns that appear during propulsion.

The thesis is divided in two parts. The first part discusses the existing literature on automotive driving cycles, a field that has acquired more attention compared to the study of marine loading cycles. The first part, also, contains the theoretical background and the literature review of data analytics and machine learning concepts, as well as an in-depth analysis of the ones that are used in the practical part, such as data preprocessing, clustering, aggregating and classification algorithms. The theoretical research aims to define the machine learning algorithms that are applied in the second part, which constitutes the main product of the thesis.

The second part of the thesis follows four main steps. The first step focuses on the selection/extraction of the loading cycles' variables, upon which clustering and classification models will be applied, by calculating the correlation matrix of variables. Pre-processing is, then, applied in the dataset in order to prepare it and make it suitable for algorithmic usage. Furthermore, hierarchical clustering is performed to group the preprocessed timeseries based on similarity criteria. To conclude the number of clusters appropriate for this thesis' scope, an evaluation measure (average silhouette score) is used that provides an indication of the optimal number of clusters.

In the third step, an averaging (or aggregating) algorithm is developed to provide optimal representative loading cycles (or templates) that represent sufficiently the timeseries of each cluster. For this purpose, different distance methods of data analytics are used for templates creation and, then, these methods are assessed based on their results.

Finally, the fourth step uses the results of the second step (clustering). More precisely, a machine learning classification algorithm is used to classify the timeseries dataset, which is split into train and test sets. Based on the cross-validation of the outputs of the classifier and the clustering, the accuracy of the classifier is calculated and is, then, assessed. Then, the classifier is modified to achieve most accurate results.

In short, the whole model, developed in Python, which is the main product of this thesis, can advise the type (cluster) of a given loading cycle, can divide given loading cycles into groups (clusters) and create representatives of each cluster.

Contents

Chapter 1

Introduction

1.1 Problem statement

Diesel engines are the vast majority of the engine installations of the vessels. In recent years, changes in the legal standards and the increasing environmental concern, as well as the continuous desire for better engine performance, have forced manufacturers of marine engines to implement new technologies related to the propulsion power units of vessels, aiming at lower exhaust emission and enhanced power management. To lower the exhaust gas emissions and minimize the propulsion/steaming costs, ship owners and operators have also been forced to achieve the optimal load of operation of the engines.

As a result, Research and Development on engine's performance has been increasing on a fast pace. New innovative technologies have been proposed towards the goals stated above. One of them is the use of a hybrid diesel electric configuration that could be used to assist the engine operation during transient loading conditions and enhance the total performance of the vessel's powerplant. Such a configuration is proposed by many manufacturers.

In general, the development of new technologies includes a loop of continuous testing, evaluation of the test results and improvement. This way, their operation is improved until desired results are achieved. In addition, it is of crucial importance that the test conditions of the experimental facilities are as close as possible to the conditions of real world applications. This way, the experimental results can faithfully represent the relevant real world application, while keeping the error at minimum levels. The importance of the above mentioned requirements, is amplified when the application requires large capital and human factor overhead, which is also the case for maritime industry. Therefore, in experimental and testbed facilities, where marine engines are tested, one critical factor for the output is the load cycle that will be applied to the engine.

Different types of vessels operate in various conditions. For example, marine power-plants in cargo vessels usually operate in steady state sea going condition. On the other hand, some vessel types operate in coastal areas and within port range, such as passenger vessels, yachts, tugs, special purpose vessels, etc. The power units of these vessels operate at various load points, which may be out of their optimal performance range. Therefore, the load cycles differ from vessel to vessel and from region to region and depend on several other factors, which makes their representation a greatly challenging issue.

1.2 Literature review

Marine driving cycles have not been developed and investigated in the same extent as automotive driving cycles. There is not sufficient literature about marine diesel engines loading cycles that can be used for further scientific research relative to this thesis' goal. However, extensive literature material exists for automotive driving cycles. For this reason, concepts of automotive driving cycles analysis are examined and taken into consideration to then be applied in marine loading cycles.

A driving cycle, or driving schedule, is represented by vehicle speed versus time. Some examples are the European certification driving cycle for light-duty vehicles, the New European Driving Cycle, NEDC [2], the FTP75 used in the United States and the JC08 driving cycle used in Japan. The two latter certification driving cycles are used for test approval to control that the vehicle manufacturers follow the legislation.

Driving cycles are important components for evaluating vehicles and play a fundamental role in vehicle design since the driving cycle affects the cost, fuel consumption, and the emissions of vehicles as discussed in [3] [4] [5]. In the past, driving cycles have mainly been used to assess exhaust gas emissions of vehicles [6] [7] [10] [11], but they are also used to evaluate different control strategies for vehicles as in [12] [13], as well as, in vehicle design and sizing of components in [14] [15].

In [16] [17] he certification driving cycles are sometimes criticized to not be representative of real-world driving as which means that using them for evaluation and design could be questioned. A representative driving cycle usually means that several statistical measures are close to real-world driving conditions.

In [8] and [9] is discussed that by using only a single driving cycle when designing a vehicle there is a considerable risk that the vehicle performance is optimized for this specific driving cycle and the resulting design may be non-robust and have sub-optimal performance for other driving cycles.

Based on the above, two problems were identified and these are regarding the representativeness of driving cycles and sub-optimization issues. To solve them more representative driving cycles are needed to both increase the robustness and reduce the risk for sub-optimization. Today the vehicle manufacturers use several different driving cycles with the objective that they will cover and represent real- world driving.

There are several approaches to generate new driving cycles that are representative for a certain region of interest. The Markov chain approach is used in [19], [20] [17] to extract typical behavior from large amounts of operational data. There have been many proposals of new driving cycles that are representative of a certain region of interest in [21] [22] [23] and [18]. Another approach [16] compared the existing standard driving cycles with objective methods, e.g., correlation analysis or automatic clustering, and chooses a combination of driving cycles that can be representative. However, comparison of driving cycles is still difficult to perform and the lack of a definition of representative driving cycles enables subjective judgments.

More recently a Markov chain approach has been used for generating representative driving cycles from real-world driving data in a compact way, see [19] [20] [1] [17].

In [24], P.Nyberg, E. Frisk, and L. Nielsen present an approach to generate driving cycles with certain desired equivalence properties from a database of real-world driving cycles. A key contribution of their work is an algorithm that combines the two objectives above. A formulation with Markov processes is used to obtain a condensed and effective characterization of the database, and to generate candidate driving cycles. To solve the problem of finding representative driving cycles with equivalence properties regarding excitation, they used the Markov chain approach in [19] [1] [17]. Added to that is a method transforming a candidate to an equivalent driving cycle with desired excitation. The method is a general approach. The result is a new method that combines the generation of driving cycles using real-world driving cycles with the concept of equivalent driving cycles. Their method partly relies on work presented in [25], [26].

As far as pattern recognision is concerned, Berndt and Clifford in [29] and Chen and Nascimento among others in [30] have investigated pattern recognision in timeseries using data analytics and machine learning methods. Paparrizos and Gravano in [28] use k-shape algorithm to cluster timeseries, while Skyler Seto among others in [27] presents modifications to DTW as timeseries similarity measure, as well as a template-based approach for Human Activity Recognition.

1.3 Structure of data

In this thesis, raw data from several vessels during engine's operation available at LME Laboratory are used. More precisely, data available for the analysis contain the relative signals of several variables of the engine in relation to time prior to the engine's stoppage. There is no extra information about the vessel and the engine, except that the data come from different vessels with different engines. These data are raw data as measured in relation to time by sensors on board, which have been normalised, with values in the range [0,1]. An example is presented below in figure 1.1

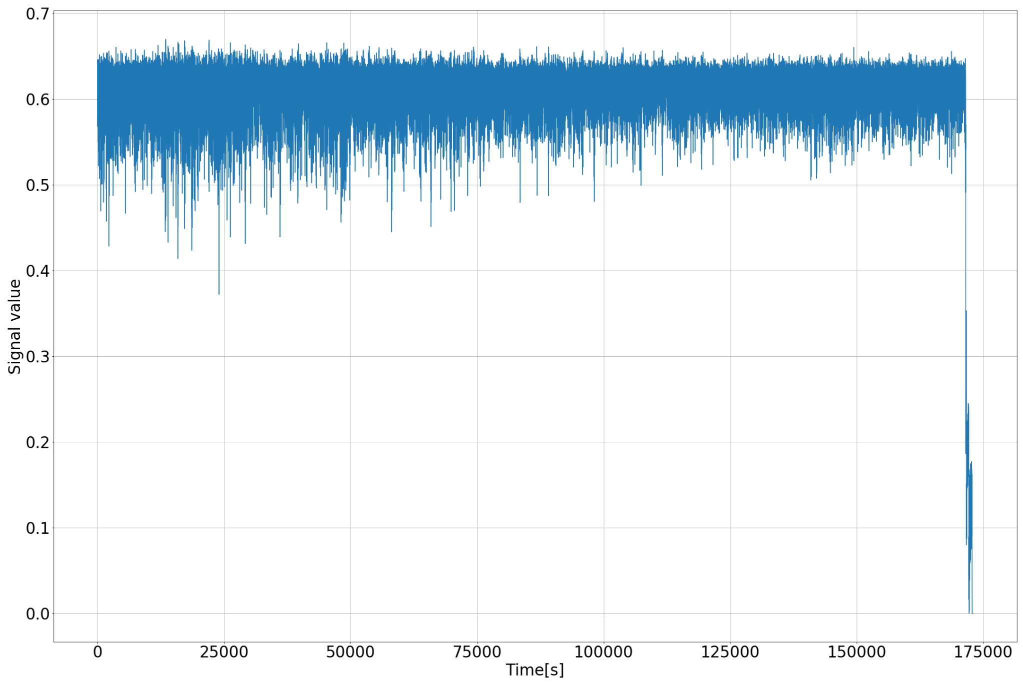


Figure 1.1: Engine's variable signal sample

1.4 Thesis Structure

This thesis' goal is to create a useful tool that can provide solutions to the problems stated in section 1.1. Specifically, in this thesis, the main goal is the creation of representative load cycles, hereinafter also reffered as templates, of a vessel during maneuvering and a model that can classify given load cycles. These templates will be created by processing and handling of above mentioned real world data and will then be available as input in experimental facilities. Such a facility is the hybrid-electric powertrain of Laboratory of Marine Engineering (LME). The created templates aim to model real world conditions as faithfully as possible and to provide a realistic input in experiments. As seen in literature, a common method towards loading cycle creation in automotive sector used Markov chain and statistical measures. For this thesis, data analytics and machine learning methods related to pattern recognision are utilized to provide optimal results in large datasets.

Further to a relative discussion [1], the creation of representative loading cycles of vessels, will contribute to optimization of engine's operation and will assist in patterns identification and categorization. As described in section 1.1, during operation different load transients will be requested to the engine, e.g. when changing course, trying to keep constant vessel position or reversing the vessel's direction. A group of realistic general loading cycles that resemble these situations will be a useful tool in order to test and benchmark different engine control strategies.

In the first part of this thesis, the theoretical background of the algorithms and methods used is presented. The second part is the practical part and includes preprocessing of available data, clustering and templates creation as well as classification. The second part is therefore divided in four main methodological steps. In the first step, the available raw data will be preprocessed in order to become suitable for algorithm input and, additionally, the relationship between the different engine's variables will be investigated to achieve dimensionality reduction. Then, in the second step, the preprocessed data in the form of timeseries will be clustered in groups using data analytics concepts. During this step, the selection of clustering method and number of clusters will also be determined. In the third step, for each of the clusters created, a representative load cycle (also called template) will be created using aggregate method of data analytics. Finally, the fourth step aims to create a classifier that will be trained and tested on the results of clustering through machine learning.

[1]Xavier Llamas, Lars Eriksson, Linkoping, Sweden

Chapter 2

Theoretical background

2.1 Time series clustering

Introduction

Clustering is used to classify large quantities of data when there is no previous knowledge about classes [31]. Emerging concepts (e.g. big data) and their applications in recent years have led to increase on unsupervised solutions, like clustering algorithms, to obtain knowledge from enormous piles of data. Clustering of time-series data has been used in many scientific fields to discover patterns that may lead to valuable information from complex datasets. Often, usage of supervised classification solutions is almost impossible either because of the massive datasets or because patterns are not recognized with bear eye. Unsupervised approaches offer a solution to this problem. In this thesis, unsupervised clustering algorithms are used.

Clustering definition and basic info

Clustering is a data mining technique where similar data are placed into related or homogeneous groups without advanced knowledge of the groups' definitions [31]. In detail, clusters are formed by grouping objects that have maximum similarity with other objects within the group, and minimum similarity with objects in other groups. It is a useful approach for exploratory data analysis as it identifies structure(s) in an unlabeled dataset by objectively organizing data into similar groups. Moreover, clustering is used for exploratory data analysis for summary generation and as a pre-processing step for other data mining tasks or as a part of a complex system.

With increasing power of data storages and processors, real-world applications have found the chance to store and keep data for a long time. Hence, data in many applications is being stored in the form of time-series data, for stock prices in finance, weather data, etc. This amount of time-series data has provided the opportunity of analyzing time-series for many researchers in data mining communities in the last decade. Consequently, many researches and projects relevant to analyzing timeseries have been performed in various areas for different purposes such as: subsequent matching, anomaly detection, motif discovery, indexing, clustering, classification, visualization, segmentation, identifying patterns, trend analysis, summarization, and forecasting. Moreover, there are many on-going research projects aimed to improve the existing techniques. In the recent decade, there has been a considerable amount of changes and developments in time-series clustering area that are caused by emerging concepts such as big data and cloud computing which increased size of datasets exponentially.

Time-series clustering

A special type of clustering is time-series clustering. [31] A sequence composed of a series of nominal symbols from a particular alphabet is usually called a temporal sequence, and a sequence of continuous, real-valued elements, is known as a time-series. A time-series is essentially classified as dynamic data because its feature values change as a function of time, which means that the value(s) of each point of a time-series is/are one or more observations that are made chronologically. Time-series data is a type of temporal data which is naturally high dimensional and large in data size, but can also be seen as a single object. Clustering such complex objects is particularly advantageous because it leads to discovery of interesting patterns in time-series datasets. As these patterns can be either frequent or rare patterns, several research challenges have arisen such as: developing methods to recognize dynamic changes in time-series, anomaly and intrusion detection, process control, and character recognition. To highlight the importance and need for clustering time-series datasets, objectives for clustering of time-series data are given as follows:

- Time-series databases contain valuable information that can be obtained through pattern discovery. Clustering is a common solution performed to uncover these patterns on time-series datasets.
- Time-series databases are very large and cannot be handled well by human inspectors. Hence, many users prefer to deal with structured datasets rather than very large datasets. As a result, time-series data are represented as a set of groups of similar time-series by aggregation of data in non-overlapping clusters or by a taxonomy as a hierarchy of abstract concepts.
- Time-series clustering is the most-used approach as an exploratory technique, and also as a subroutine in more complex data mining algorithms, such as rule discovery, indexing, classification, and anomaly detection.
- Representing time-series cluster structures as visual images (visualization of time-series data) can help users quickly understand the structure of data, clusters, anomalies, and other regularities in datasets.

The problem of clustering of time-series data is formally defined as follows:

Given a dataset of n time-series D = F1,F2,...,Fn, the process of unsupervised partitioning of D into C = C1, C2,..., C3, in such a way that homogenous time-series are grouped together based on a certain similarity measure, is called time-series clustering.

A great challenge addresses the similarity measures that are used to make the clusters. Similar time-series should be found, which needs time-series similarity matching that is the process of calculating the similarity among the whole time-series using a similarity measure. This process is also known as “whole sequence matching” where whole lengths of time-series are considered during distance calculation. However, the process is complicated, because time-series data are naturally noisy and include outliers and shift.

Time series clustering is separated in three categories:

- Whole time-series clustering is considered as clustering of a set of individual time-series with respect to their similarity. Time series are considered as discrete objects and then are clustered.
- Subsequence clustering means clustering on a set of subsequences of a time-series that are extracted via a sliding window, that is, clustering of segments from a single long time-series.

- Time point clustering is clustering of time points based on a combination of their temporal proximity of time points and the similarity of the corresponding values. This approach is similar to subsequence clustering.

Unsupervised vs supervised clustering

In supervised learning techniques, learning models use data where the answer was already known — the class labels were already available in the training data [33]. On the other hand, cluster analysis, a category of unsupervised learning techniques allows to discover hidden structures in data where the right answer is not known upfront. The goal of clustering is to find a natural grouping in data so that items in the same cluster are more similar to each other than to those from different clusters.

2.2 Hierarchical clustering of time-series

Hierarchical clustering is an approach of cluster analysis which makes a hierarchy of clusters using agglomerative or divisive algorithms [33], [31]. The two main approaches to hierarchical clustering are agglomerative and divisive hierarchical clustering. In divisive hierarchical clustering, in the beginning, one cluster encompasses all samples, and iteratively the cluster is split into smaller clusters until each cluster only contains one sample. This thesis will focus on agglomerative clustering, which takes the opposite approach. Each sample starts as an individual cluster and the closest pairs of clusters merge until only one cluster remains (bottom-up approach).

In general, hierarchical algorithms have the disadvantage that they cannot adjust the clusters after splitting a cluster in divisive method, or after merging in agglomerative method. One advantage of hierarchical clustering is that it allows to plot dendrograms (visualizations of a binary hierarchical clustering), which can help with the interpretation of the results by creating meaningful taxonomies. Another useful advantage of this hierarchical approach is that it is not required to specify the number of clusters up front. In contrast to most algorithms, hierarchy clustering does not require the number of clusters as an initial parameter, which is a well-known and outstanding feature of this algorithm. It is also a strength point in timeseries clustering, because usually it is hard to define beforehand the number of clusters in real world problems.

Hierarchical clustering linkage methods

Two common algorithms for agglomerative hierarchical clustering are single linkage and complete linkage [33]. Using single linkage, the distances between the most similar members for each pair of clusters are computed and the two clusters for which the distance between the most similar members is the smallest are merged. The complete linkage approach is similar to single linkage, but, instead of comparing the most similar members in each pair of clusters, the comparison is done on the most dissimilar members to perform the merge. This is shown in the following figure (figure 2.1):

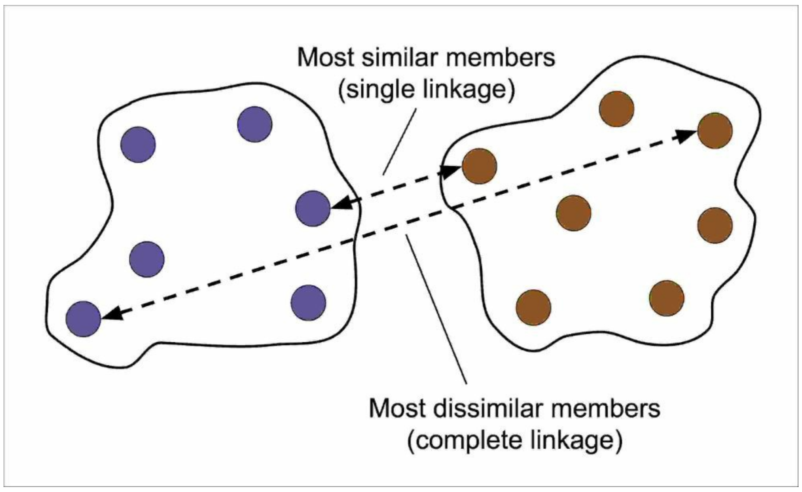


Figure 2.1: Comparison of single and complete linkage (from [33])

Other commonly used algorithms for agglomerative hierarchical clustering include average linkage and Ward's linkage. In average linkage, the cluster pairs are merged based on the minimum average distances between all group members in the two clusters. In Ward's linkage, the two clusters that lead to the minimum increase of the total within-cluster SSE (sum of squared errors) are merged and this is the method used in this thesis. Specifically, hierarchical Ward's linkage clustering is an iterative procedure that can be summarized by the following steps:

1. Compute the distance matrix of all samples.
2. Represent each data point as a singleton cluster.
3. Merge the pair of clusters that leads to minimum increase in total within-cluster variance after merging
4. Update the similarity matrix.
5. Repeat steps 2-4 until one single cluster remains.

From Sci Py Library[1], the following linkage methods for calculating the distance between the newly formed clusters for time series hierarchical clustering are derived and are further with more details explained as follows:

$d(s, t)$ is the distance between two clusters s and t. When two clusters s and t from this forest are combined into a single cluster u, s and t are removed from the initial data forest, and u is added to the forest until only one cluster remains in the forest. A distance matrix is maintained at each iteration. The d[i,j] corresponds to the distance between cluster i and j in the original forest. At each iteration, the distance matrix is updated to reflect the distance of the newly formed cluster u with the remaining clusters in the forest. Suppose there are $|u|$ original observations $u[0], \ldots, u[|u|-1]$ in cluster u and $|v|$ original objects $v[0], \ldots, v[|v|-1]$ in cluster v. Recall s and t are combined to form cluster u. Let v be any remaining cluster in the forest that is not u. The following methods are used for calculating the distance between the newly formed cluster u and each v.

- method='single' assigns

$$d(u, v) = \min(dist(u[i], v[j]))$$

[1]https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

for all points i in cluster u and j in cluster v. This is also known as the Nearest Point Algorithm.

- method='complete' assigns

$$d(u, v) = \max(dist(u[i], v[j]))$$

for all points i in cluster u and j in cluster v. This is also known by the Farthest Point Algorithm or Voor Hees Algorithm.

- method='average' assigns

$$d(u, v) = \sum_{ij} \frac{d(u[i], v[j])}{(|u| * |v|)}$$

for all points i and j where $|u|$ and $|v|$ are the cardinalities of clusters u and v, respectively. This is also called the UPGMA algorithm.

- method='weighted' assigns

$$d(u, v) = (dist(s, v) + dist(t, v))/2$$

where cluster u was formed with cluster s and t and v is a remaining cluster in the forest. (also called WPGMA)

- method='centroid' assigns

$$dist(s, t) = ||c_s - c_t||_2$$

where c_s and c_t are the centroids of clusters s and t, respectively. When two clusters s and t are combined into a new cluster u, the new centroid is computed over all the original objects in clusters s and t. The distance then becomes the Euclidean distance between the centroid of u and the centroid of a remaining cluster v in the forest. This is also known as the UPGMC algorithm.

- method='median' assigns $d(s, t)$ like the centroid method. When two clusters s and t are combined into a new cluster u, the average of centroids s and t give the new centroid u. This is also known as the WPGMC algorithm.

- method='ward' uses the Ward variance minimization algorithm. The new entry $d(u, v)$ is computed as follows,

$$d(u, v) = \sqrt{\frac{|v| + |s|}{T} d(v, s)^2 + \frac{|v| + |t|}{T} d(v, t)^2 + \frac{|v|}{T} d(s, t)^2}$$

where u is the newly joined cluster consisting of clusters s and t, v is an unused cluster in the forest, $T = |v| + |s| + |t|$, and $| * |$ is the cardinality of its argument. This is also known as the incremental algorithm.

2.3 Time-series clustering evaluation measures (or Clustering validity)

In general, evaluating of extracted clusters is not easy in the absence of data labels and is an open problem. The definition of clusters depends on the user, the domain, and it is subjective. Specifically, the number of clusters, the size of clusters, definition for outliers, and definition of the similarity among the time-series in a problem are all concepts which depend on the task at hand and should be declared subjectively. These have made the time-series clustering a big challenge in the data mining domain. However, owing to the classified data labeled by human judge or by their generator (in synthetic datasets), the result can be evaluated by using some measures. The label of human judge is not perfect in terms of clustering raw data, but in practice it captures the strengths and shortcomings of the algorithms as ground truth.

Visualization and scalar measurements are the major technique for evaluation of clustering quality, which is also known as clustering validity [31]. There are many internal indices such as : Sum of Squared Error, Silhouette index, Davies-Bouldin, Calinski-Harabasz, Dunn index, R-squared index, Hubert-Levin (C-index), Krzanowski-Lai index, Hartigan index, Root-Mean-Square Stan- dard Deviation (RMSSTD) index, Semi-Partial R-squared (SPR) index, Distance between two clusters (CD) index, Weighted inter-intra index, Homogenity index, and Separation index.

In this thesis, the average silhouette score is employed.

Average silhouette method

The average silhouette approach briefly, measures the quality of a clustering [2]. That is, it determines how well each object lies within its cluster. A high average silhouette width indicates a good clustering.

Specifically, average silhouette method computes the average silhouette of observations for different values of number of clusters k. The optimal number of clusters k is the one that maximize the average silhouette over a range of possible values for k.

The Silhouette Coefficient in Scikit Learn Library [3] is defined for each sample and is composed of two scores:

- a The mean distance between a sample and all other points in the same class.
- b The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient s for a single sample is then given as:

$$s = \frac{b-a}{max(a,b)}$$

The algorithm is similar, but diverse to the elbow method that is also widely known, and can be computed as follows:

1. Compute clustering algorithm (e.g.hierarchical clustering, k-means clustering) for different values of k. For instance, for varying k from 2 to 70 clusters.
2. For each k, calculate the average silhouette of observations (average silhouette).
3. Plot the curve of average silhouette according to the number of clusters k.
4. The location of the maximum is considered as the appropriate number of clusters.

[2]https://www.datanovia.com/en/lessons/cluster-validation-statistics-must-know-methods/
[3]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

2.4 Cluster templates creation

Distance measure metric

There are many methods that can be implemented towards the creation of representative and accurate templates for given clusters. Some examples include [35] :

- Euclidean distance
- Dynamic Time Warping
- Threshold based distance measures
- Longest Common Subsequence model

In this thesis, the main focus will be on Euclidean distance and Dynamic Time Warping (DTW).

Euclidean distance

The Euclidean distance between two time series Q and C of length n is defined as

$$d(Q,C) = \sqrt{\sum_{i=1}^{n}[Q(i) - C(i)]^2}$$

Typical characteristics:

+ Fast to compute
- Requires equal length of timeseries
- Poor performance when there are too many outliers
- Fails to capture changes in the time axis (i.e. compression and stretching)

In a few words, it calculates the Euclidean distance in respective same points of the time series, as seen in figure 2.2:

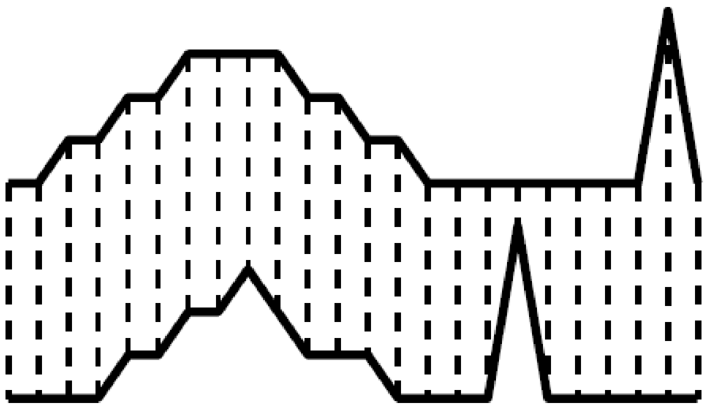


Figure 2.2: Euclidean distance calculation visualisation (from [35])

Dynamic Time Warping (DTW)

Dynamic time warping finds the optimal non-linear alignment between two time series [4] [32]. The Euclidean distances between alignments are then much less susceptible to pessimistic similarity measurements due to distortion in the time axis. However, there is a price to pay for this, because dynamic time warping is quadratic in the length of the timeseries used.

Dynamic time warping works in the following way. Consider two time series Q and C of the same length n where

$$Q = q_1, q_2, ..., q_n$$

and

$$C = c_1, c_2, ..., c_n$$

. The first thing is to construct an $n \times n$ matrix whose i, j^{th} element is the Euclidean distance between q_i and c_j. One wants to find a path through this matrix that minimizes the cumulative distance. This path then determines the optimal alignment between the two time series. It should be noted that it is possible for one point in a timeseries to be mapped to multiple points in the other timeseries.

Let the path be called W where

$$W = w_1, w_2, ..., w_K$$

where each element of W represents the distance between a point i in Q and a point j in C i.e. $w_k = (q_i - c_j)^2$

So one wants to find the path with the minimum Euclidean distance

$$W^* = argmin_W(\sqrt{\sum_{k=1}^{K} w_k})$$

. The optimal path is found via dynamic programming, specifically the following recursive function:

$$\gamma(i,j) = d(q_i, c_j) + min(\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1))$$

A warping path w is a contiguous set of matrix elements which defines a mapping between C and Q that satisfies the following conditions in figure 2.3:

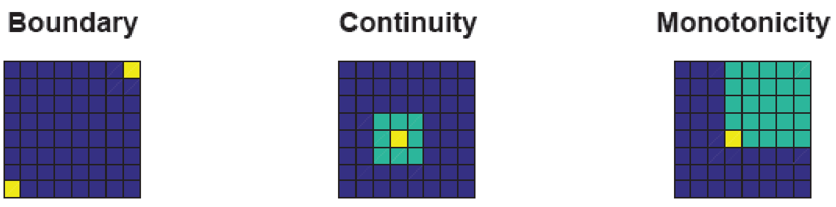


Figure 2.3: Conditions that must be satisfied by DTW (from [35])

Main property of DTW is that it is very slow to compute.

A graphical representation of DTW computation in following figure 2.4:

[4]http://alexminnaar.com/2014/04/16/Time-Series-Classification-and-Clustering-with-Python.html

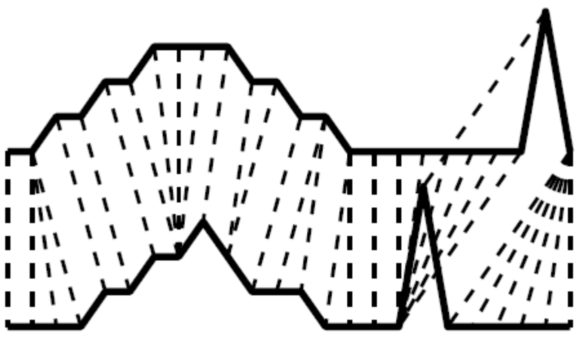


Figure 2.4: DTW method calculation visualisation (from [35])

Although the number of possible alignments is exponential in the length of the two time series, DTW can be computed in only quadractic time using dynamic programming.

Soft Dynamic Time Warping

Many authors have investigated smoothed modifications of the recursive function to define smoothed dynamic programming distances [38], [39]. When applied to the DTW discrepancy, that regularization results in a soft-DTW score, which considers the soft-minimum of the distribution of all costs spanned by all possible alignments between two time series. In short, Soft-DTW proposes to replace the minimum in DTW by a soft minimum. Despite considering all alignments and not just the optimal one, soft-DTW can be computed with a minor modification of recursive function, in which all (min, $+$) operations are replaced with ($+$, $\times$) [40]. As a result, both DTW and soft-DTW have quadratic in time and linear in space complexity with respect to the sequences' lengths. Because soft-DTW can be used with kernel machines, one typically observes an increase in performance when using soft-DTW over DTW for classification [41]. The main advantage of soft-DTW comes from the fact that it is differentiable everywhere and that its gradient can also be computed in quadratic time. The variable gamma γ of recursive function is a positive regularization parameter, with lower values resulting in less smoothing.

Fast Dynamic Time Warping

In short, it is a faster version of DTW [35]. The saving in time comes with accuracy cost as it does not always find the optimal solution. A graphical representation of Fast DTW is presented in figure 2.5:

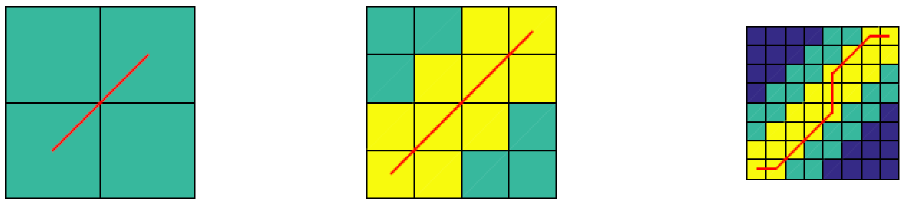


Figure 2.5: Fast DTW method calculation visualization (from [35])

Comparison example of Euclidean distance and DTW

In order to understand how easily the two distance metrics can provide completely different and conflicting outcomes, a simple arithmetic example is presented.

Euclidean distance

At first glance, it seems like simply calculating the Euclidean distance between two time series would give a good idea of the similarity between them. After all, the Euclidean distance between identical time series is zero and the Euclidean distance between very different time series is large. However, before settling on Euclidean distance as a similarity measure the desired criteria for determining the similarity between two time series should be stated.

With a good similarity measure, small changes in two time series should result in small changes in their similarity. With respect to Euclidean distance this is true for changes in the y-axis, but it is not true for changes in the time axis (i.e. compression and stretching). Consider the following example in figure 2.6:

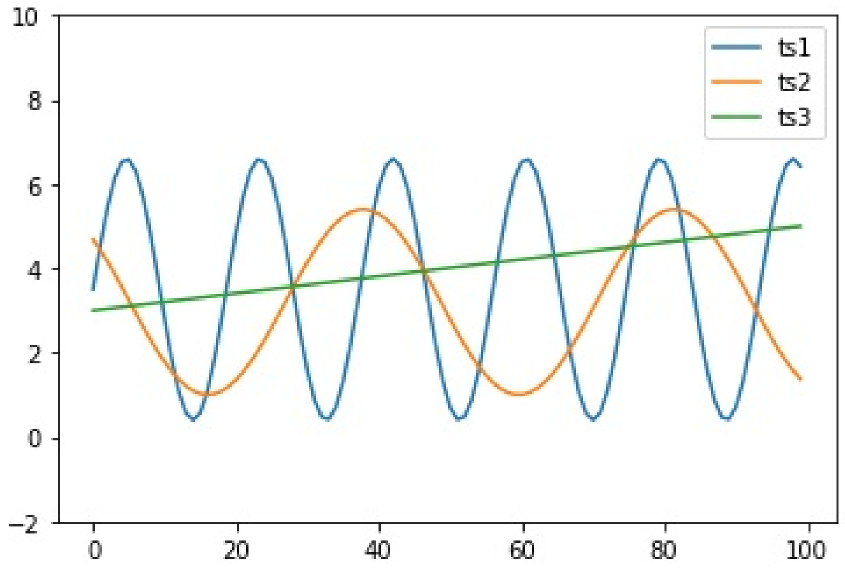


Figure 2.6: Euclidean time series example

In the above example, it is clear that $ts1$ and $ts2$ are most similar (they are both sin functions under different transformations). $ts3$ is clearly the most different. The computation of the Euclidean distance $d(ts1, ts2)$ and $d(ts1, ts3)$ will show if the Euclidean distance measure agrees with the observer's intuition.

The Euclidean distance between $ts1$ and $ts2$ is equal to 26.95 and the Euclidean distance between $ts1$ and $ts3$ 23.19. This result is not appropriate because according to the Euclidean distance measure, $ts1$ is more similar to $ts3$ than to $ts2$ which contradicts human's intuition. This is the problem with using the Euclidean distance measure. It often produces pessimistic similarity measures when it encounters distortion in the time axis. The way to deal with this is to use dynamic time warping.

Dynamic Time Warping

Computation of the Euclidean distance between $ts1$ and $ts2$ using dynamic time warping, will result to 17.92 and between $ts1$ and $ts3$ using dynamic time warping will result to

21.55. It is obvious that the results have changed compared to use of the Euclidean distance measure. Now, in agreement with the intuition, $ts2$ is shown to be more similar to $ts1$ than $ts3$ is.

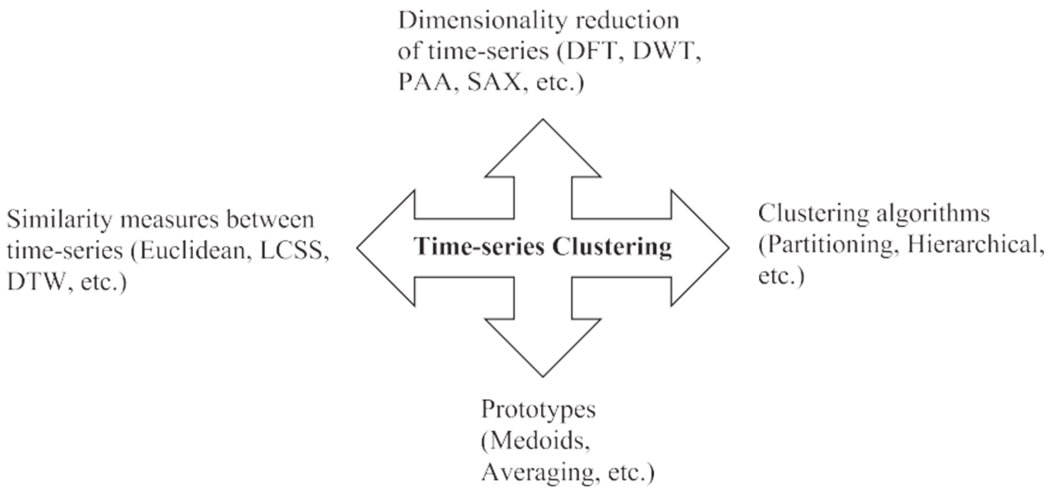

Dimensionality reduction of time-series (DFT, DWT, PAA, SAX, etc.)
Similarity measures between time-series (Euclidean, LCSS, DTW, etc.)
Time-series Clustering
Clustering algorithms (Partitioning, Hierarchical, etc.)
Prototypes (Medoids, Averaging, etc.)

Figure 2.7: The four aspects of studying time series clustering (from [31])

2.5 Time series classification

The clustering results can be used to train, test and validate a classification model. Classification in general can be described by the following workflow (figure 2.8):

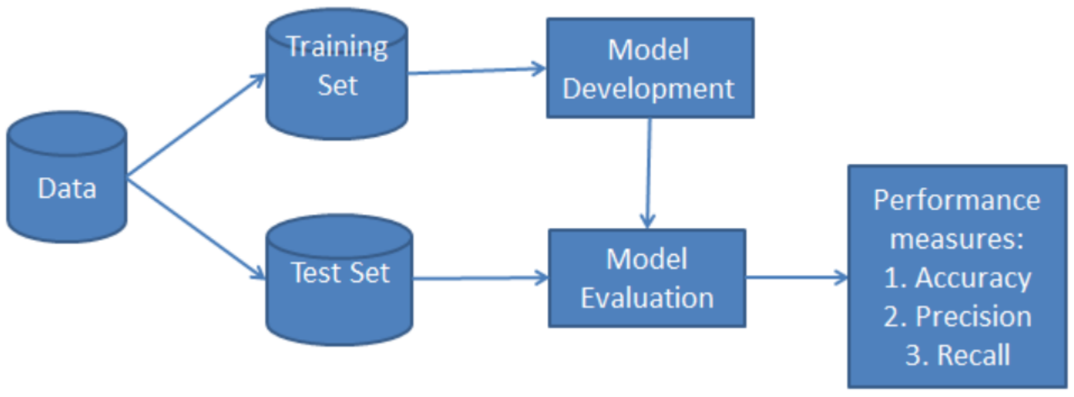


Figure 2.8: Classification workflow diagram (from [33])

K-nearest neighbors – a lazy learning algorithm

K-nearest neighbor (KNN) classifier is a supervised learning algorithm and is a typical example of a lazy learner [33]. It is called lazy not because of its apparent simplicity, but because it doesn't learn a discriminative function from the training data, but memorizes the training dataset instead.

Parametric versus nonparametric models

Machine learning algorithms can be grouped into parametric and nonparametric models [33]. Using parametric models, parameters from the training dataset are estimated to learn a function that can classify new data points without requiring the original training dataset anymore. Typical examples of parametric models are the perceptron, logistic regression, and the linear SVM. In contrast, nonparametric models can't be characterized by a fixed set of parameters, and the number of parameters grows with the training data. KNN belongs to a subcategory of nonparametric models that is described as instance-based learning. Models based on instance-based learning are characterized by memorizing the training dataset, and lazy learning is a special case of instance-based learning that is associated with no (zero) cost during the learning process.

The KNN algorithm itself is fairly straightforward and can be summarized by the following steps:

1. Choose the number of k and a distance metric.
2. Find the k-nearest neighbors of the sample that will be classified
3. Assign the class label by majority vote.

The following figure 2.9 illustrates how a new data point (?) is assigned the triangle class label based on majority voting among its five nearest neighbors.

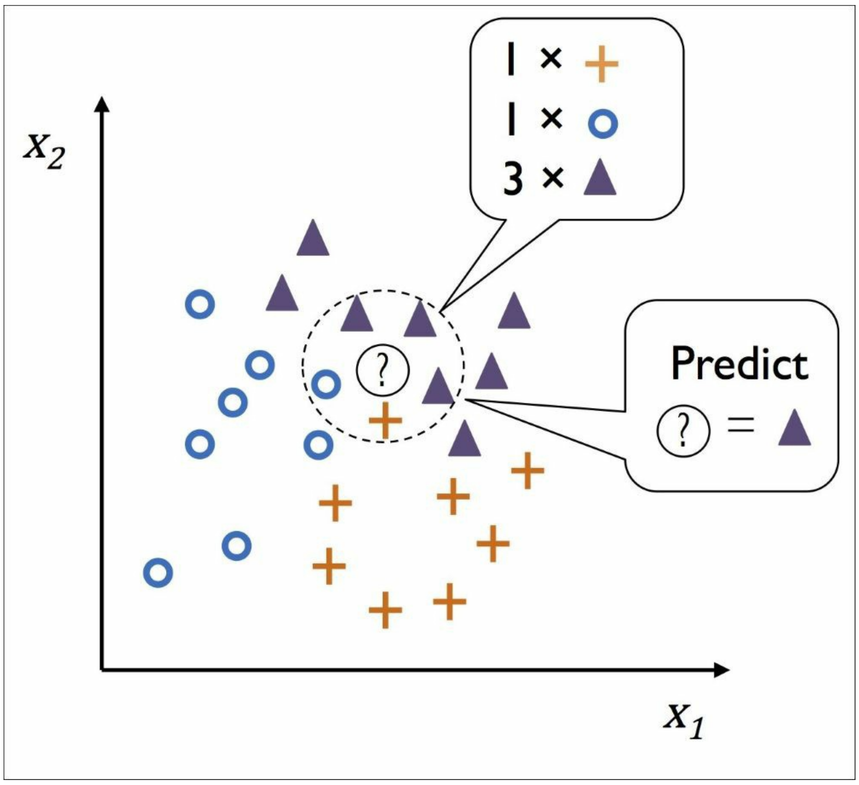


Figure 2.9: Classification example using KNN (from [33])

Based on the chosen distance metric, the KNN algorithm finds the k samples in the training dataset that are closest (most similar) to the point that that is to be classified. The class label of the new data point is then determined by a majority vote among its k nearest neighbors.

In the case of a tie, the scikit-learn implementation of the KNN algorithm [33] in this thesis, will prefer the neighbors with a closer distance to the sample. If the neighbors have similar distances, the algorithm will choose the class label that comes first in the training dataset. The right choice of k is crucial to find a good balance between overfitting and underfitting.

Advantages and disadvantages of KNN

The main advantage of such a memory-based approach is that the classifier immediately adapts as new training data are collected. However, the downside is that the computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario—unless the dataset has very few dimensions (features) and the algorithm has been implemented using efficient data structures such as KD-trees [36]. Furthermore, training samples cannot be discarded since no training step is involved. Thus, storage space can become a challenge when working with large datasets.

Overall, the KNN classifier offers an approach to classification via lazy learning that can make predictions without any model training, but with a more computationally expensive prediction step. For the clustering and classification algorithms, a crucial part of their operation is the available data in the training and test dataset and how they will be processed before feeding to the algorithms.

2.6 Data Preprocessing

The quality of the data and the amount of useful information that it contains are key factors that determine how well a machine learning algorithm can work. Therefore, it is absolutely critical to examine and preprocess a dataset before feeding it to a learning algorithm [33].

Dealing with missing data

It is very common in real-world applications for samples to be missing one or more values for various reasons. There could have been an error in the data collection process, certain measurements may not be applicable, or particular fields could have been simply left blank in a survey, for example. Missing values are typically seen as the blank spaces in data table or as placeholder strings such as NaN, which stands for not a number, or NULL (a commonly used indicator of unknown values in relational databases). In this thesis, the main reason for NaN or NULL values in the dataset is the fact that the sensor during measurement was reseting in certain time intervals and, therefore, for some seconds no measurements were taken and fields were left blank.

Most computational tools are unable to handle such missing values, or produce unpredictable results if they are simply ignored. Therefore, it is crucial that care is taken for those missing values before proceeding with further analyses. There are several practical techniques for dealing with missing values, including removing entries from dataset or imputing missing values with several methods [33]. In this thesis, the former method is occasionally used, while the latter method is mostly used.

Eliminating missing values

One of the easiest ways to deal with missing data is to simply remove the corresponding features (columns) or samples (rows) from the dataset entirely [33]. Although the removal of missing data seems to be a convenient approach, it also comes with certain disadvantages; for example, too many samples may be removed, which will make a reliable analysis impossible. Or, if too many feature columns exist, there is the risk of losing valuable information that the classifier needs to discriminate between classes. Other commonly used alternatives for dealing with missing values implement interpolation techniques.

Inputting missing values (ffil method)

Often, the removal of samples or dropping of entire feature columns is simply not feasible, because of too much loss of valuable data. One of the most common inputting techniques is ffil method, where the missing value is simply replaced by the exactly previous value in the same sample (in the same timeseries). A very convenient way to achieve this is by using the ffil method from pandas library [5], which is also used in practical part.

The above theoretical background aims to define the concepts that are applied in the second part, the practical part, which constitutes the main product of the thesis.

[5]https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html

Chapter 3

Loading Cycles Analytics Framework

3.1 Main concept

Below the practical part framework of this thesis' is presented. This framework aims to tackle challenges as presented in chapter 1, with the data analytics and machine learning concepts thoroughly discussed in chapter 2. As described in the summary, the main scope of the analytics section of the thesis is to provide a tool that creates representative load cycles of vessels and a classification model. The main steps of the practcial part are presented below in figure 3.1:

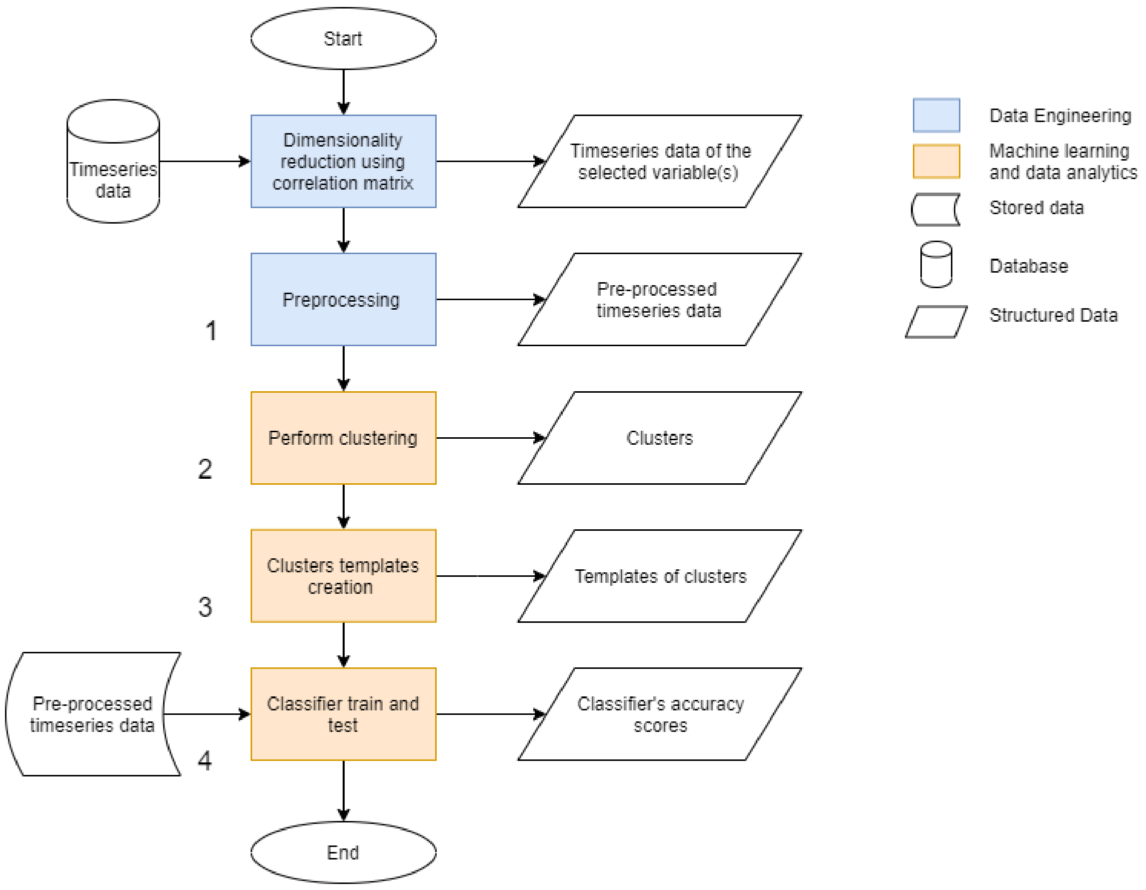

Start
Timeseries data
Dimensionality reduction using correlation matrix
Timeseries data of the selected variable(s)
Preprocessing
Pre-processed timeseries data
1
Perform clustering
Clusters
2
Clusters templates creation
Templates of clusters
3
Pre-processed timeseries data
Classifier train and test
Classifier's accuracy scores
4
End
Data Engineering
Machine learning and data analytics
Stored data
Database
Structured Data

Figure 3.1: Thesis overview flow diagram

The first step of the analysis, is the preprocessing of the available data. As explained in section 1.4, the available data are raw data and contain many variables of load cycles as measured in respect to time. The analysis should consider all available variables. However, by dimensionality reduction, highly correlated variables can be excluded if their correlation with one or more of the remaining variables is high enough, as will be explained further in chapter 4. Therefore, the first step of the preprocessing is to calculate the correlation matrix of the variables, find the relationship between them and, then, proceed to dimensionality reduction. Since dimensionality reduction is considered very important, it is presented as an individual process in figure 3.1. Nevertheless, it is part of preprocessing. Dimensionality reduction can be described by the following flow diagram in figure 3.2:

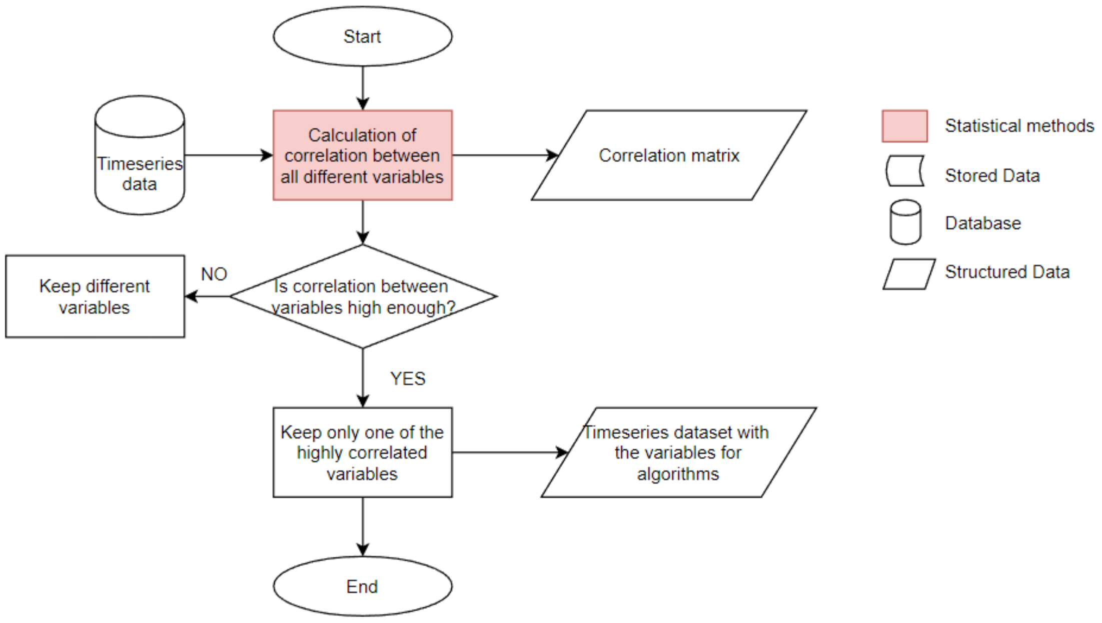

Start
Timeseries data
Calculation of correlation between all different variables
Correlation matrix
Keep different variables
NO
Is correlation between variables high enough?
YES
Keep only one of the highly correlated variables
Timeseries dataset with the variables for algorithms
End
Statistical methods
Stored Data
Database
Structured Data

Figure 3.2: Dimensionality reduction flow diagram

The data after dimensionality reduction contains the main variables of the load cycles in relation to time. As explained in theoretical background in section 2.6, the NaN and NULL values in dataset must be properly handled before the dataset is fed into the algorithm. In this thesis, the main reason for NaN or NULL values is the fact that the sensors, during measurement, were reseting in certain time intervals and, therefore, for some seconds no measurements were taken and fields were left blank. It can safely be assumed that if a sample timeseries contains many NaN values, then, the information that can be provided from this might not be reliable. Therefore, all timeseries that will be selected to participate in the analysis' dataset should contain up to a certain percentage number of NaN values that is defined by the author. In this thesis, this percentage is selected to be 2%. In the remaining timeseries with less than 2% of NaN values, the NaN values are handled with two ways:

- Elimination: delete all NaN values
- Input based on ffil method: NaN values are replaced by the exactly previous value in timeseries

Then, the timeseries that contain only zero values are excluded, because they offer no extra information as they describe a situation where the vessel is stopped and the engine is shut down. In the next step of preprocessing, basic statistical metrics are calculated for the qualified dataset and, then, based on the average timeseries length, the timeseries are cut-off at average length to produce a dataset of timeseries with equal length. The described procedure can be described by the following diagram in figure 3.3:

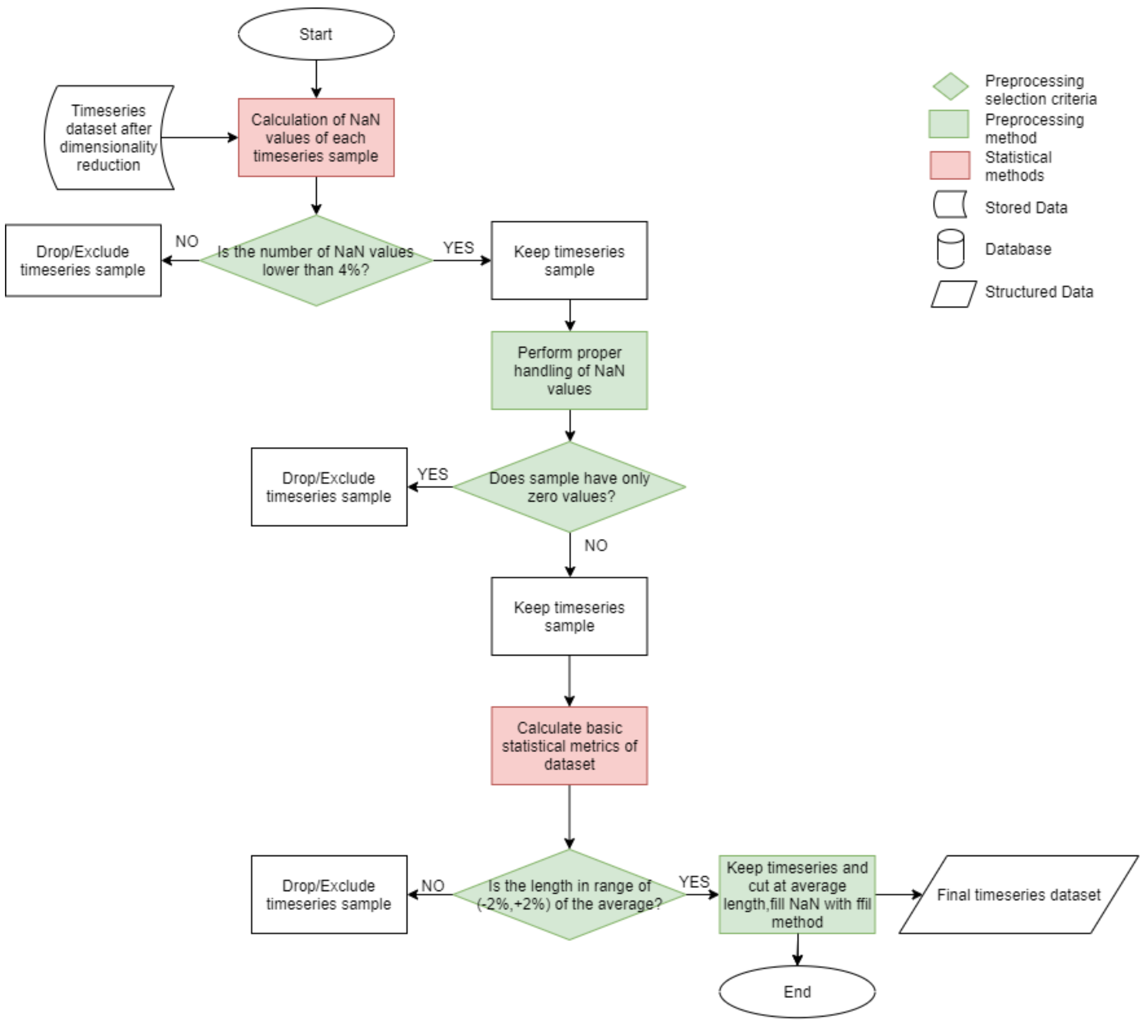

Start
Timeseries dataset after dimensionality reduction
Calculation of NaN values of each timeseries sample
Drop/Exclude timeseries sample
NO
Is the number of NaN values lower than 4%?
YES
Keep timeseries sample
Perform proper handling of NaN values
Drop/Exclude timeseries sample
YES
Does sample have only zero values?
NO
Keep timeseries sample
Calculate basic statistical metrics of dataset
Drop/Exclude timeseries sample
NO
Is the length in range of (-2%,+2%) of the average?
YES
Keep timeseries and cut at average length,fill NaN with ffil method
Final timeseries dataset
End
Preprocessing selection criteria
Preprocessing method
Statistical methods
Stored Data
Database
Structured Data

Figure 3.3: Main part of preprocessing flow diagram

The final part of preprocessing, includes filtering and resampling of timeseries dataset. This part will be further described with more details in sections 4.6-4.7. In short, timeseries are filtered using a low-pass filter that leads to significant reduction of noise and much easier pattern recognision. Further, to reduce computational time and cost, all timeseries are resampled, which leads to decrease of their length without losing any valuable information, as will be seen the relative sections.

Time series clustering and classification is the main focus of this thesis. The main concept followed after the preprocessing of the dataset can be summarized in figure 3.4:

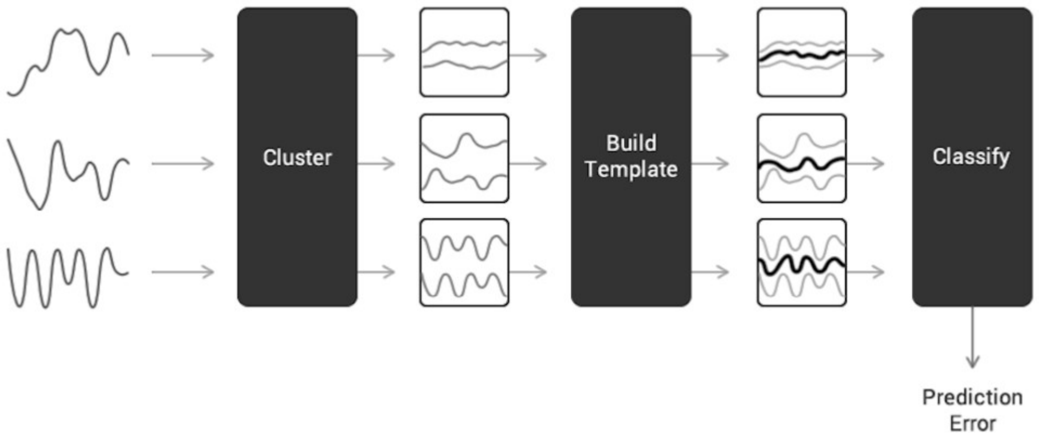


Figure 3.4: Main process of practical part after preprocessing (from [27])

Specifically, firstly, the preprocessed timeseries are clustered with hierarchical clustering method and a hierarchical tree is created. This method makes it possible to select any number of clusters and hierarchical tree provides a useful tool to visualise the clustering result. In addition, as described in theoretical background in section 2.3, a clustering validity method is employed to assess the optimal number of clusters. This method is average silhouette score and provides a reliable indication of the numbers of clusters that lead to best clustering results. Finally, the clustering is performed for the selected number of clusters and clusters are formed. This way, the measured loading cycles are devided in groups based on their simillarity. Every timeseries sample is more simillar with all other samples in the same cluster and less simillar with samples in other clusters.

The next step of the practical part, is the creation of templates for every cluster created. More precisely, a reprsentative load cycle for the timeseries of each of the clusters is created based on machine learning avaraging-distance methods. These methods are three, as described in section 2.4:

- Euclidean distance method
- Dynamic Time Warping Barycenter Averaging (DTW BA or DBA)
- Soft Dynamic Time Warping (Soft DTW)

The results produced are discussed and a conclusion is made about the best results for the representation of load cycles. In the final step of the practical part, a model that can classify load cycles based on machine learning is created. The K-nearest neighbors algorithm is used, which was described on detail in theoretical section 2.5. The timeseries dataset after the preprocessing is split into train and test sets and these sets are fed into the KNN classifier to train and test it, respectively. Then, cross-validation of the classification results and the clustering resuts is performed and the accuracy of the KNN algorithm is calculated. The goal is to achieve an accuracy higher than 80%, which can be considered satisfying.

In the next chapters, the analytics framework presented in this chapter will be used to perform the three methodological steps (namely: the preprocessing of the available data, the clustering of the preprocessed dataset, the creation of templates for all clusters and the classification model production) of the practical part of this thesis.

Chapter 4

Pre-processing of data

As described in section 2.6, the quality of the data and the amount of useful information that it contains are key factors that determine how well a machine learning algorithm can work. Therefore, it is absolutely critical to examine and preprocess the dataset before feeding it to the learning algorithms. This is the goal of this chapter.

4.1 Input data description

In this thesis, measurements from marine diesel engines are used as input. Specifically, before the stop of the vessels' engines, for different vessels with different engines, several variables were measured. The total size of data used is 6.58 giga bytes containing 638 timeseries files in total with different time lengths. These data are raw data as measured by sensors and, also, normalised, which means they do not provide the absolute value of respective variable, but relative values of the variable (for example relative to the maximum or average value). For example, a diesel engine with 15,000 kW MCR operating at 10,000 kW is operating at a normalised power of 10,000/15,000 = 0.66. In the next figures (figure 4.1 - figure 4.3), a sample signal from the timeseries dataset is presented graphically. The x axis represents the time and the y axis the respective variable's signal:

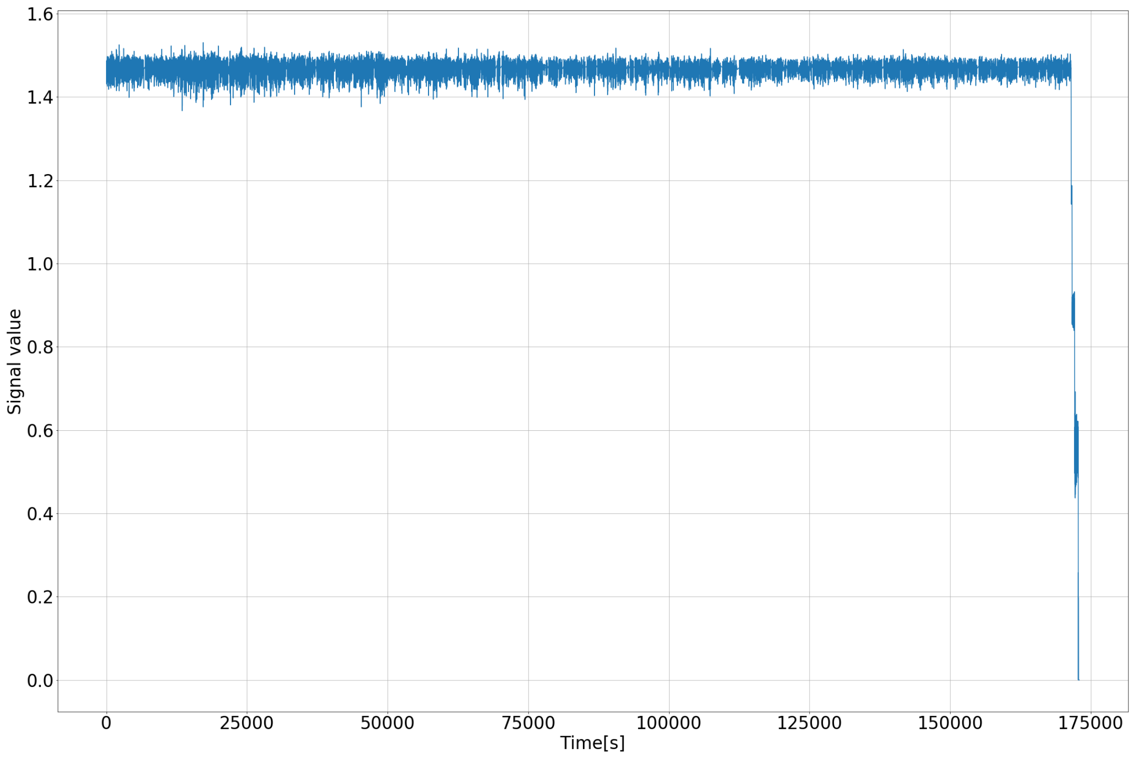


Figure 4.1: Variable 1 signal sample

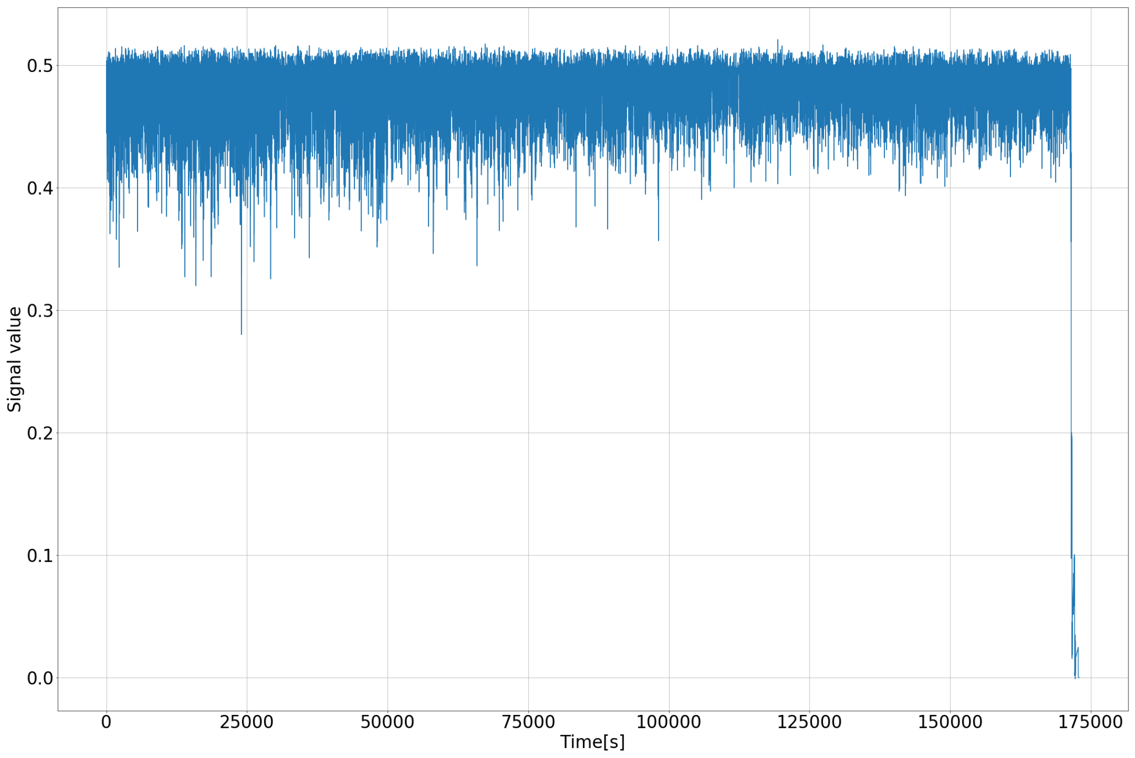
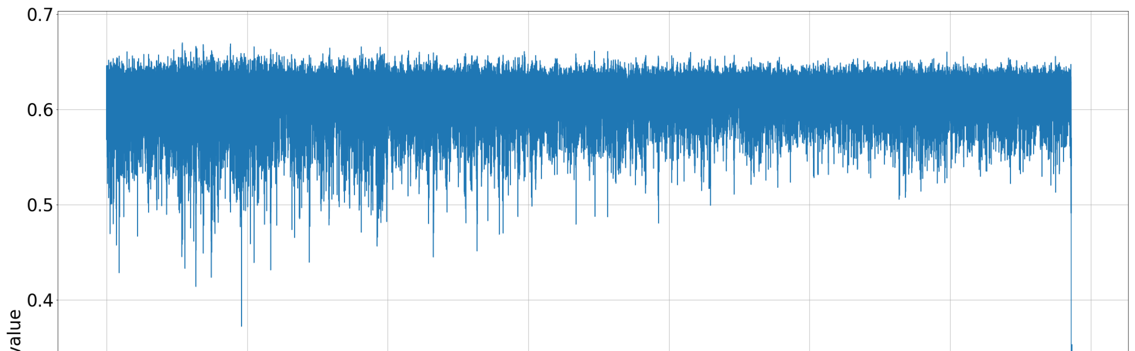


Figure 4.2: Variable 2 signal sample



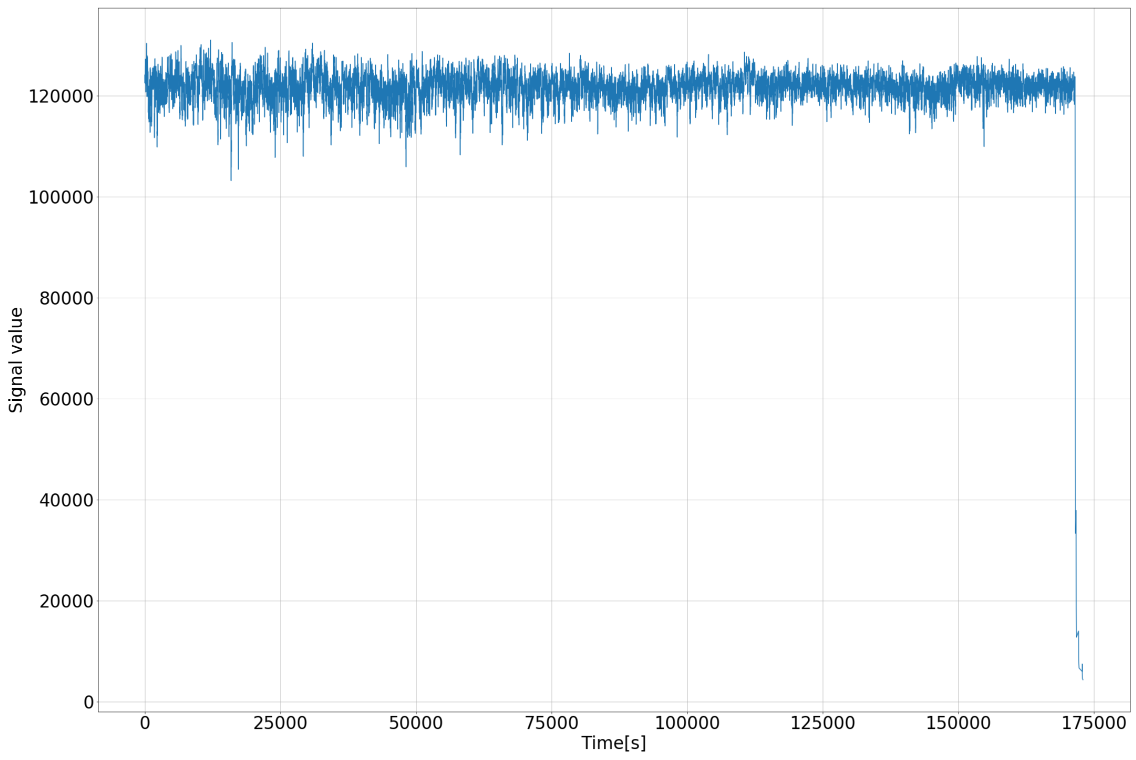


Figure 4.4: Variable 4 signal sample

4.2 Dimensionality reduction using correlation matrix

The first step of the preprocessing is dimensionality reduction. This procedure is crucial when examining many different variables that describe a specific problem (in this thesis the loading cycle of a vessel). Specifically, time and calculation cost for the analysis can be significantly reduced by using the minimum required number of variables through dimensionality reduction. This is achieved by constructing the correlation matrix between the variables.

Simply stated, the R^2 value is simply the square of the correlation coefficient R[1]. The correlation coefficient R of a model (say with variables x and y) takes values between -1 and 1. It describes how x and y are correlated.

- If x and y are in perfect unison, then this value will be positive 1
- If x increases while y decreases in exactly the opposite manner, then this value will be -1
- 0 would be a situation where there is no correlation between x and y
- The closer it is to +1 or -1, the more closely the two variables are related. The positive sign signifies the direction of the correlation i.e. if one of the variables increases, the other variable is also supposed to increase.

However, this R value is only useful for a simple linear model (just an x and y). Once considering more than one independent variable (say x_1, x_2, ...), it is very hard to understand what the correlation coefficient means. Tracking which variable contributes what to the correlation is not so clear. This is where the R^2 value comes into play. It

[1]https://socratic.org/, http://blog.uwgb.edu/bansalg/

takes values between 0 and 1, where values close to 1 imply more correlation (whether positively or negatively correlated) and 0 implies no correlation. Another way to think of it is as the fractional variation in the dependent variable that is the result of all of the independent variables. If the dependent variable is highly dependent on all of its independent variables, the value will be close to 1. So R^2 is much more useful as it can be used to describe multivariate models as well.

The R^2-statistic is a measure of how well the estimated regression equation fits the observed data. The value of R represents how much of the variations in the response variable y can be explained by the regression model [42].

If R^2 is large (>0.9), the regression model explains most of the variations in the response variable [43]. R^2 is useful when a stepwise regression is performed. A limit can be set and the removal of explanatory variables can be stopped when the model no longer shows a large enough fit (when R^2 becomes smaller than a predefined limit). A property of R^2 is that it always grows if more explanatory variables are added to the model. This fact in combination with a small sample size can cause overfitting of the data, and more variables than necessary can be included in the model [43].

Summarizing, based on the above, if two variables have high correlation, close to 1, one can be eliminated from the analysis, as the behavior of either of the two can be approximated by the other, as presented above. In other words, the changes of the one in relation to time are similar to the other's. Therefore, the correlation between each pair of variables presented in section 4.1 is calculated and presented in the following table 4.1:

	Variable 1	Variable 2	Variable 3	Variable 4
Variable 1	1.0000	0.8706	0.8997	0.8162
Variable 2	0.8706	1.0000	0.9759	0.9443
Variable 3	0.8997	0.9759	1.0000	0.9047
Variable 4	0.8162	0.9443	0.9047	1.0000

Table 4.1: Correlation matrix for each pair of variables

In table 4.1, the dimension of the correlation matrix is 4×4. Looking at the individual elements of the correlation matrix, the main diagonal all comprises of 1. This indicates that the correlation of an element with itself is 1, or the highest value possible. This makes complete sense logically and intuitively. The other elements is the Pearson's correlation coefficient between the two respective values.

In table 4.1, it can be seen that the correlation between each pair of variables is very high as all correlation values are higher than 0.8. As a result, it is assumed that all variables are highly correlated with each other and all can be safely represented by one of them, so that study of one variable provides reliable information for all variables. Nevertheless, a researcher would easily make the same assumption, even without proceeding to the aforementioned calculation of the correlation table, since many of the engine's variables are highly correlated with each other.

Based on the above, the dimensionality reduction leads to selecting only one variable for further analysis. This variable is selected to be Variable 2, which is the engine speed reference.

4.3 Handling of NaN and zero values

The data used as input in the algorithms was collected by sensors that were automatically resetting at certain intervals of time. Therefore, the data contained NaN values that were saved during reset procedure. At each timeseries the percentage of NaN values was calculated. All timeseries with percentage of NaN values > 4 % or the ones that have only zero values are excluded from selected dataset.

After these timeseries are collected, all NaN values are replaced based on the $ffil$ method. When $ffil$ [2] is applied across the index then any missing value is filled based on the corresponding value in the previous column. In other words, all NaN values are replaced by the exactly previous value in the timeseries.

In addition, all timeseries with only zero values are excluded as they offer no valuable information for the clustering procedure. The result is a number of timeseries with different lengths, no NaN values and with average value greater than zero.

4.4 Statistical metrics of data

The next step of the pre-processing is to calculate some basic statistical metrics. The most important metrics for a preliminary understanding of the data are assumed to be the arithmetic mean, minimum, maximum and standard deviation of all timeseries. In order to obtain some first insight on the timeseries that will be fed to the algorithms, the minimums, maximums, averages and standard deviations of the aforementioned statistical metrics are calculated, as presented in table 4.2:

	Eng Speed Ref minimums	Eng Speed Ref maximums	Eng Speed Ref averages	Time (Length of timeseries)
Minimum	−0.043	0.11	0.0001	168 341
Maximum	0.466	1.07	0.8900	173 100
Average	−0.007	0.67	0.2743	173 053
Std	0.021	0.19	0.2178	227

Table 4.2: Dataset statistics metric table

It should be noted that the negative values in engine speed reference values correspond to errors of the sensors. Therefore, they were considered as zero values. In table 4.2, it can be seen that the average engine speed reference value of all timeseries is 0.2743 with a standard deviation of 0.2178 or 79.4%. This shows that on average the engines operate at low to medium engine speed values. Another important value from the above table is the average time length, which is equal to 173,053 with 227 or 0.13% standard deviation and 168,341 minimum value. This minimum value is 2.73% lower than the average value. Based on this, a logical conclusion can be derived that the timeseries that corresponds to this minimum length should be excluded from the final dataset, since the deviation in time length compared to the average might be considered too high. This conclusion led to adding one more requirement for qualified dataset that is presented in the next section.

4.5 Cut - off at average length

A requirement in order to proceed to the clustering, was to input timeseries with the same length for algorithms to handle them and results to be comparable. In order to achieve this, timeseries with relative difference from the average length in table 4.2 greater than 2% are excluded. Then, remaining timeseries are cut-off at average length. This step involves no risk for any loss of valuable information, because most timeseries are zero valued for

[2]https://pandas.pydata.org/pandas-docs/version/0.13.1/generated/pandas.DataFrame.fillna.html

some time before the end. After this cut-off, if empty values exist, they are filed with ffil method as described previously.

After the above procedure that is the last step of data selection, the resulted dataset contains 584 timeseries out of 638 initially before preprocessing. Therefore, the 8.46% of the initial timeseries data is excluded from analysis.

As mentioned in chapter 3, a significant part of the preprocessing is filtering and resampling of the timeseries. Specifically, filtering is important in digital signals to reduce noise and reveal the important information of the signals that are in many cases hided by useless noise. Furthermore, a good resampling method reduces the length of timeseries without affecting the information that can be extracted from them. These processes are both discussed with their application results in the next sections of this chapter and lead to the creation of the final preprocessed timeseries dataset.

4.6 Filtering

The available timeseries after cut-off as described previously, contain 584 timeseries with length of 173,053. This means that there are 173,053 points in time in every sample of the dataset. This length is extraordinarily high and a graphical representation proves that changes are very steep from each point to the next. As a result useless information and noise might lead to losing important information. For this reason, a filter is applied in all timeseries in order to reduce the digital noise. The type of filter that is selected is low-pass filter. The main values of variables of the low pass filter that is used, are:

- Sampling frequency: 1 Hz
- Cut-off frequency: 0.001 Hz
- Normalized freq based on: w = fc / (fs/2)

A butterworth signal filter [3] based on above variable values is applied and the digital noise is greatly reduced. The result of a random sample timeseries is graphically presented in comparison to the initial timeseries before filtering in following figure 4.4:

[3]https://gist.github.com/junzis/e06eca03747fc194e322

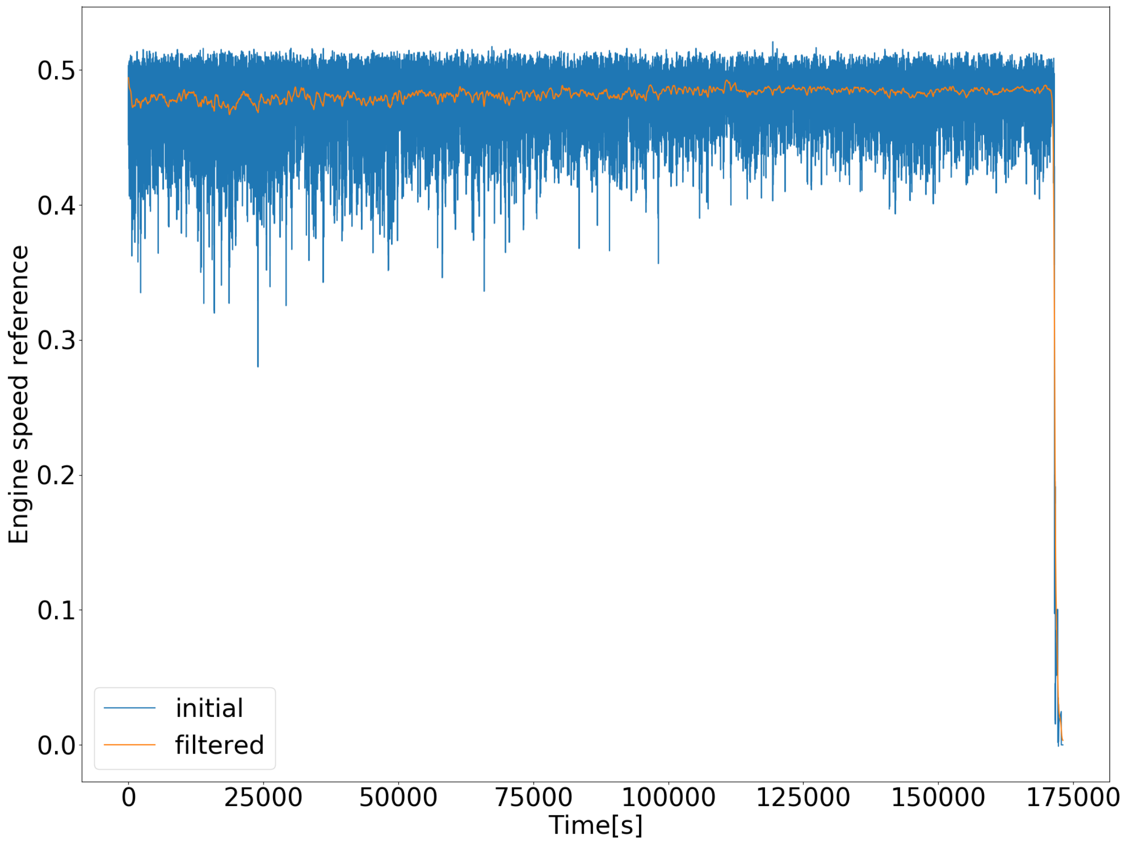


Figure 4.5: Comparison between unfiltered and filtered timeries

As can be seen in figure 4.4, the width of fluctuations in y-axis are greatly decreased, while changes in values are not eliminated. Therefore, the shape and changes in chart remain the same, while the noise is greatly reduced. It is evident from the same figure that pattern recognision is much easier in the filtered graph compared to the initial one and the behavior of the time graph is much more comprehensible. Therefore, the filtering is considered successful and essential since pattern recognision is enhanced.

4.7 Resampling

As described above, the timeseries after the first part of preprocessing had length of 173,053 time units. Most algorithms in literature and in relative websites (e.g. gisthub) need extreme processing time to handle such enormous datasets. Therefore, decreasing the length of the timeseries, while keeping the same information, which means without changing their behaviour and movement in relation to time, is crucial for further analysis to take place. In addition, resampling would be beneficial for the results of the algorithms, since they would be much more comprehensible and easier to handle during evaluation procedure.

The resampling is done with a selection rate of 100, which means 1 point per 100 points was selected. The result are timeseries of 173,053/100 = 1,731 length approximately. A graphical representation in following figure 4.5 before and after the resampling, is presented:

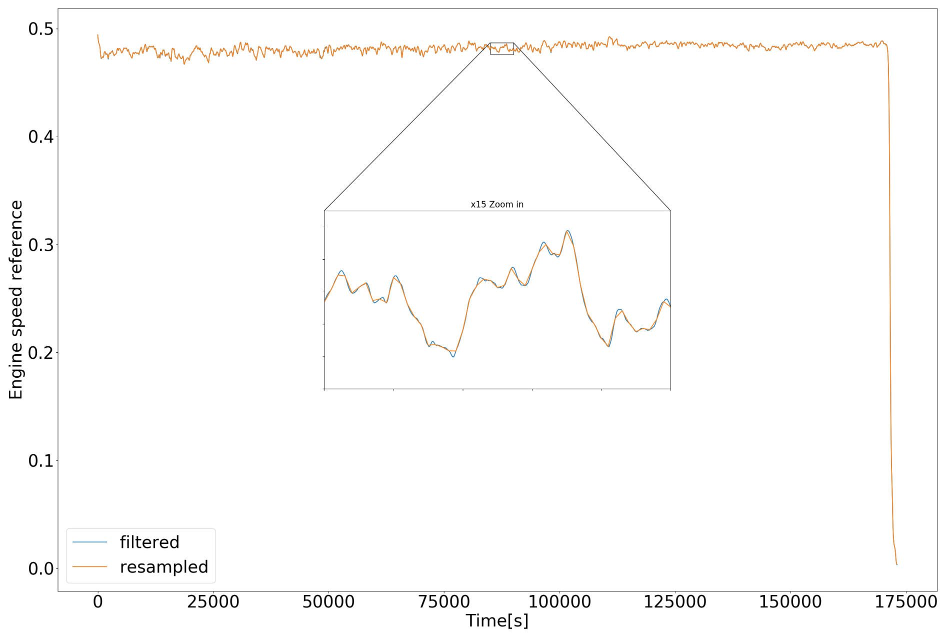


Figure 4.6: Comparison between filtered and resampled timeries

Overall, from figure 4.5 it is clear that there are too slight differences between the two timeseries before and after resampling, since the two timeseries are overlapping almost perfectly. Therefore, it proves that the representation of the timeseries remains the same, while the length is etremely reduced and the resampling is considered satisfying and successful.

In this chapter, the preprocessing of the initial 6.58 gigabytes of data containing 638 timeseries was utilized. From the four different variables that describe a loading cycle, only one representative was selected, due to their high pairwise correlation. This representative was Engine Speed Reference. Before preprocessing, most initial timeseries were over 170,000 time units long, may have contained NaN values and also presented too high unacceptable noise. The preprocessing resulted to 584 timeseries of 1,731 time units long, without any NaN value and with greatly reduced noise. This dataset is the qualified preprocessed dataset that will be used to feed the clustering algorithm of the next chapters.

Chapter 5

Clustering

5.1 Hierarchical Clustering

The clustering method selected is hierarchical clustering. The main advantages, as described in theoretical part, are the following:

- Small computational overhead
- Graphic visual representation of the formation of the clusters, ideal for unsupervised clustering evaluation
- Simple and straightforward setting of the clustering variables

Method

In section 2.2, commonly used algorithms for agglomerative hierarchical clustering are presented. The selected one for this thesis' clustering is Ward's linkage method, which uses the Ward variance minimization algorithm. As already discussed, in Ward's linkage, in each algorithms iteration, the two clusters that lead to the minimum increase of the total within-cluster SSE (sum of squared errors) are merged.

Hierarchical tree

The results of clustering are presented using a hierarchical tree in figure 5.1. The x axis represents the number of clusters and y axis the distance between timeseries of each cluster. As can be observed from the aforementioned graph, there is an evident change in regards to the in-cluster distance between the first and the last clusters created. Evidentely, the distance in the clusters created in the first iteration (hereon after the first clusters)is smaller than 3 and the distance in the last cluster reaches 250. This change represents a percentage increase of around 8,300%. As is clear from chapter 2, the in-cluster distance is a measure of the clustering output quality. Therefore, using the above observations derived from the hierarchical tree, it is safe to conclude that as the number of clusters decreases, the clustering quality is degrading as well.

On the other side, the number of clusters chosen should not be too high as this will result in the formation of clusters having only a few observations. Given the complexity of the timeseries that needs to be captured, the algorithm can be prone to overfitting, meaning that the results of the algorithm may respond closely to the train data, failing however to capture patterns in additonal data. Consequently, the numbers of clusters chosen should be neither too high nor too low so as to avoid overfitting and poor clustering quality, respectively.

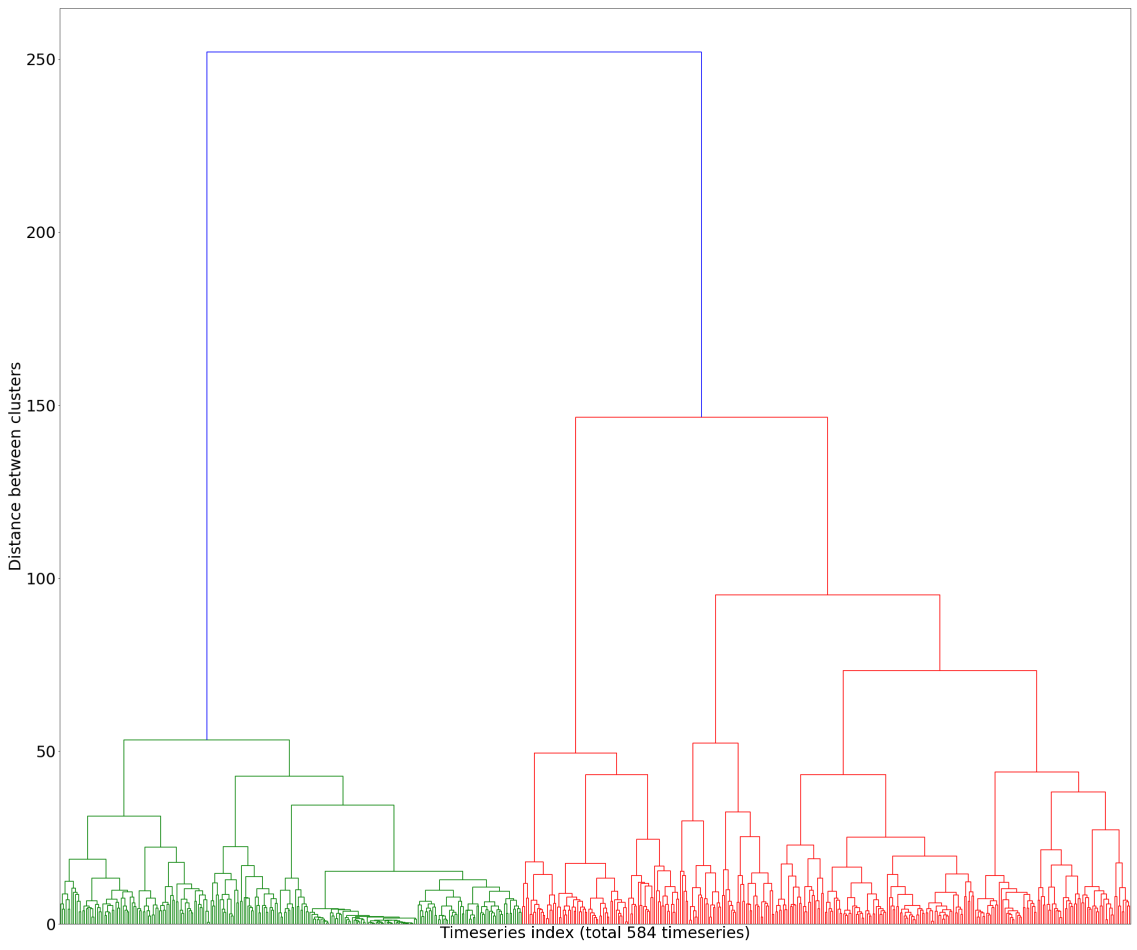


Figure 5.1: Hierarchical clustering dendrogram

One of the advantages of the hierarchical clustering method is that the user can select the number of clusters, by ”cuting” the tree in any suitable distance. Figures 5.2 - 5.3 show how the user can ”cut” the tree in many different distances. At the first example (figure 5.2) it is selected to cut-off at distance equal to 60 that leads to the creation of 5 clusters, or vice versa, and at the second example (figure 5.3) the distance and clusters are, respectively, 120 and 3. It must be noted that this method will be followed to produce final results and is crucial for the outcome of the practical part.

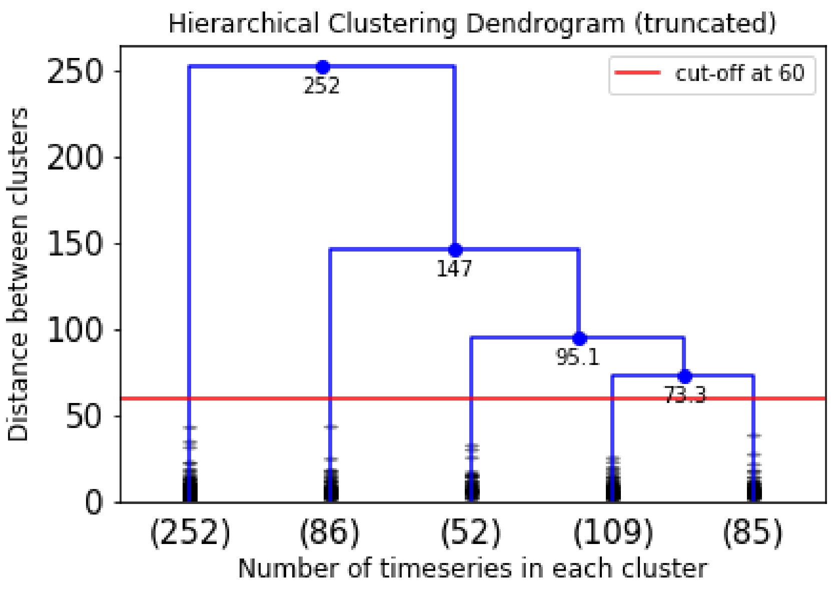


Figure 5.2: Dendrogram cut-off at distance=60 to form 5 clusters

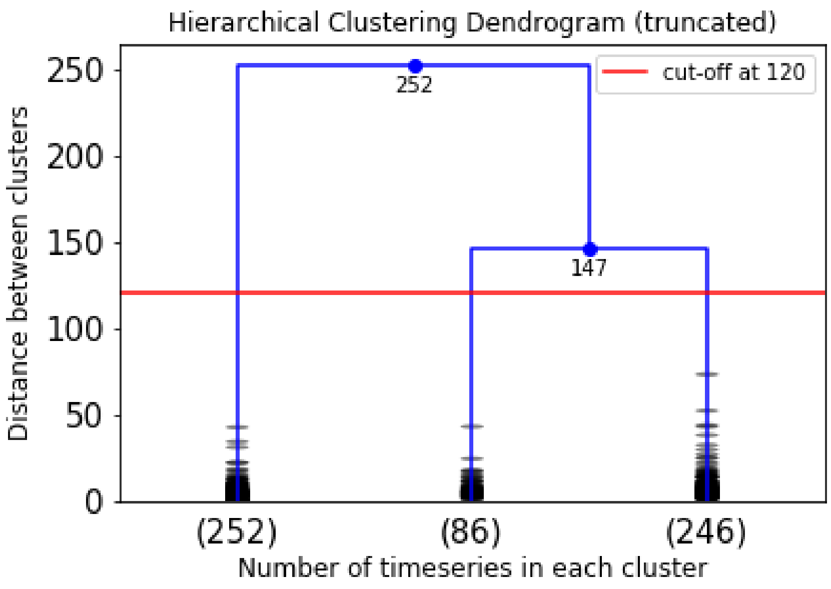


Figure 5.3: Dendrogram cut-off at distance=120 to form 3 clusters

At figure 5.2 the tree was cut at distance=60 and 5 clusters were created. The first cluster contains 252, the second 86, the third 52, the fourth 109 and the fifth 85 timeseries. In addition, at figure 5.3 the tree was cut at distance=120 and 3 clusters were created. The first cluster contained 252 timeseries, the second 86 and the third 246. As can be seen, the first cluster in both cases is the same (the one containing 252 tiemseries).

Since the way of obtaining the desired number of clustering has been determined, the next vital and crucial step for the outcome of the clustering analysis is to decide the number of clusters that is optimal. If this number is determined, then, coming back to the hierarchical tree and cuting-off as described above will create the clusters. This decision is done at next section 5.2.

5.2 Selection of number of clusters and clustering performance

In unsupervised hierarchical clustering, one of the main challenges is to determine the number of clusters. This task becomes more arduous as the test data are non existant and, therefore, there is no way to evaluate results. One possible way to overcome this difficulty, is to graphically present the results for different numbers of clusters and use visual observation to determine the best choice. Nevertheless, this method incorporates the human mistake factor and should not be accepted without the proper support by a mathematical model.

In the theoretical part (section 2.3), a number of methods to evaluate clustering results were stated (e.g. Sum of Squared Error, Silhouette index, Davies-Bouldin). Of these, silhouette score is used as it is one of the most widely used methods in data handling algorithms.

Silhouette score

The silhouette score is calculated for number of clusters in the range from 2 to 60 and is presented in figure 5.4:

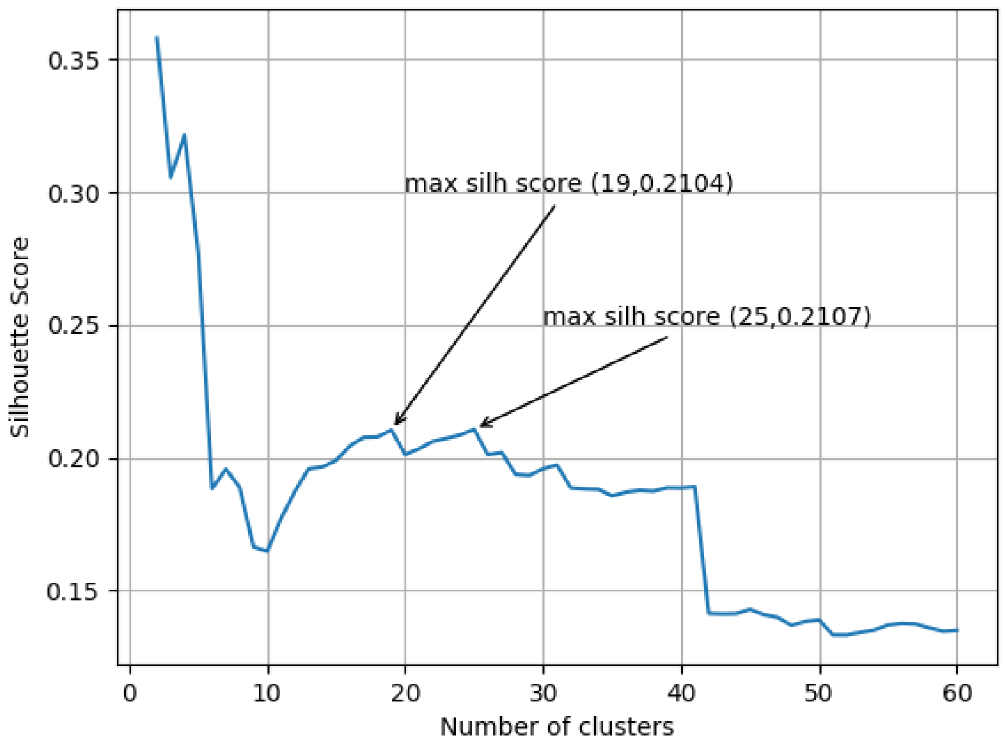


Figure 5.4: Silhouette score graph

The optimal numbers of clusters may be more than one and appear at the points where the Silhouette score is maximized (as explained in section 2.3). Based on the figure 5.4, the optimal number of clusters is 19 or 25. The maximums in the range from 2 to 10 are considered inappropriate, because of the following reasons:

- It is obvious that fewer than 10 clusters is extremely low number for classifying 584 driving/loading cycles of vessels, since the clustering quality shall be degraded as explained above

- The steep declining curve in range from x = 0 to x = 10 is an indication that the local maximisation points at x = 4 and x = 6 might be algorithm's outliers and should not be taken into considerations

- The actual graphical visualisation of the clustering results for this range of clusters made it clear that they are not acceptable

Therefore, taking into consideration all the above, it is concluded that the timeseries should be clustered in 19 and 25 clusters. The cut-off for the creation of 19 clusters is done at figure 5.1 at distance equal to 25.2 and for the creation of 25 clusters at distance equal to 21. This is presented in the following figures, figures 5.5 , 5.6, respectively.

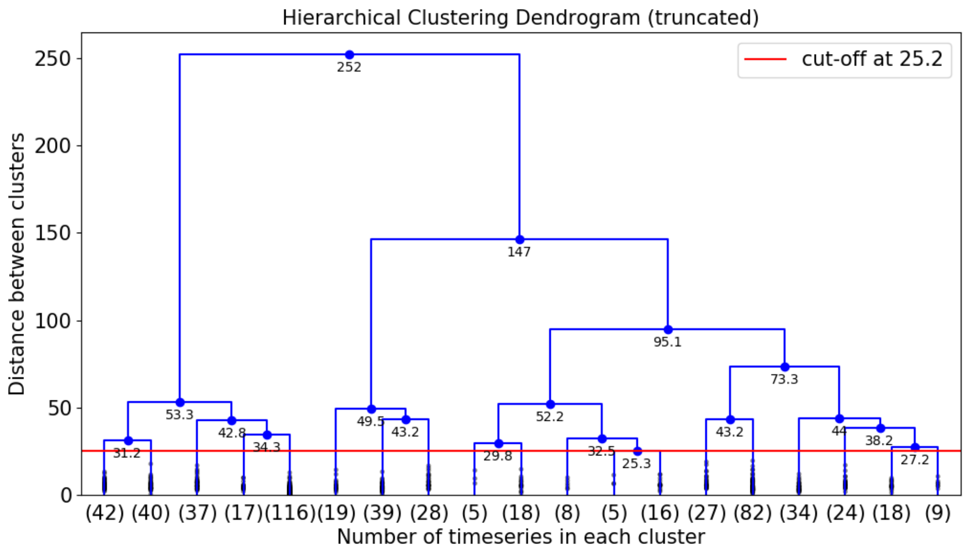


Figure 5.5: Dendrogram cut-off at distance=25.2 to form 19 clusters

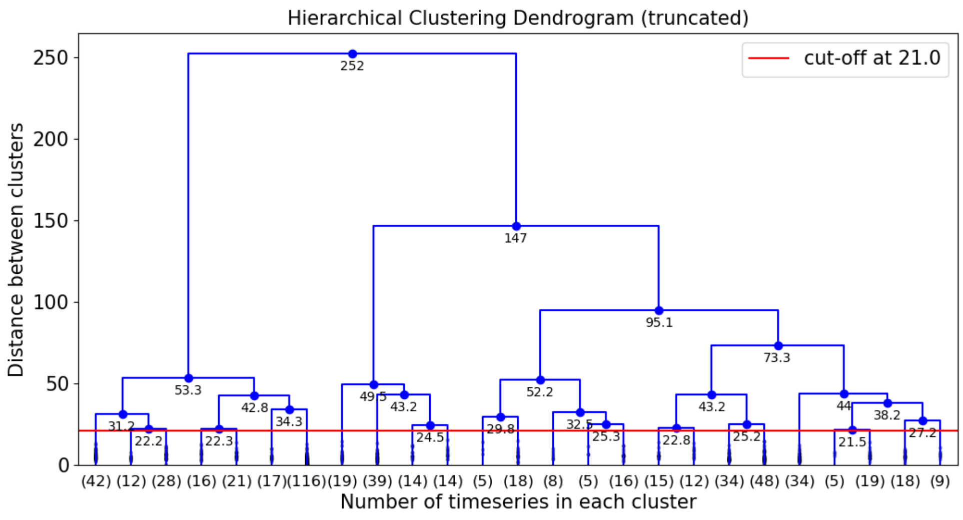


Figure 5.6: Dendrogram cut-off at distance=21 to form 25 clusters

In these figures, the incluster distance of every cluster created is annotated at the point of merging, above the respective cluster. For example, for the creation of the two clusters, the in-cluster distance would be 252. These distances also justify the previous decision of the optimal number of clusters. Specifically, it can be seen that for two clusters, the in-cluster distance (252) is almost 1,200% or twelve times higher than an indicative in-cluster distance for number of clusters equal to two (21). In addition, from figures 5.5-5.6 it can be seen that the clustering results for 19 clusters are very close to the ones for 25 clusters. This is proved both from the close cut-off distances (25.2 and 21 respectively) and from the fact that, as can be observed, some clusters remain the same in both cases.

Summarizing, in this chapter hierarchical clustering was performed on the preprocessed dataset and it was visualized with a hierarchical dendrogram. This dendrogram allows cut-off at any specific in-cluster distance or at any number of clusters that is decided by the user. Then, the average silhouette score metric was used to define the optimal numbers of clusters and the results were 19 and 25. Then, the clustering in 19 and 25 clusters was performed, while the cut-off of the hierarchical tree was presented in both cases. In the next chapter, the clustering results will be presented and visualized and representative load cycles (or templates) for each cluster will be created with several different distance methods.

Chapter 6

Clustering and averaging results

The next step of the practical part is the performance of the clustering and the templates creation. Specifically, the timeseries are clustered in groups based on their similarity. This is achieved by cutting the hierarchical tree of figure 5.1 in such a distance that leads to creation of the desired number of clusters. The desired numbers of clusters were chosen based on the maximization points of the silhouette as described in theoretical section 5.2 and figure 5.4. The cut-off was presented in figures 5.5-5.6. In addition, a representative load cycle for each cluster is created with three different methods, which were in detail described in section 2.4. These three methods are the following:

- Euclidean Distance Method
- Dynamic Time Warping Barycenter Averaging (DTW BA or DBA)
- Soft Dynamic Time Warping Averaging (Soft DTW)

Specifically, the Soft DTW method is used for many different values of the variable gamma γ in the range (0,1). As explained in section 2.4, the variable gamma γ of recursive function is a positive regularization parameter, with lower values resulting in less smoothing. The closer the value to zero, for example 0.01, the closer are the method and the results to those of DTW BA method.

6.1 19 clusters

The clustering results for 19 clusters are presented in figure 6.1. As it will be explained later on, the best results for templates creation are considered the ones produced by Soft DTW method for $\gamma = 0.5$. Therefore, for clearer visualisation, only these templates, among all templates, are selected to be ploted with red color in the same graphs with the clustering results in figure 6.1. It has to be noted that since the number of timeseries in most clusters is too high to be plotted clearly in a 2-D graph, it was selected to plot only the half of the timeseries of each cluster in figure 6.1. In addition, above each cluster's figure, the number of cluster and the number of timeseries (entries) contained in each cluster are stated. Finally the x axis represents the time and the y axis the Engine Speed Reference normalized value.

Furthermore, the templates of each cluster created by all different mentioned methods are presented in figure 6.2. Specifically, with green color are plotted the templates of Euclidean distance, with blue color the ones created with Dynamic Time Warping Barycenter Averaging (DTW BA or DBA) and with red color the Soft DTW results for gamma pameter equal to γ=0.5.

Finally, for better visualization of the overall results of the clustering and templates creation processes, figures of indicative clusters are presented in the figure 6.3. These figures visualize the timeseries of the respective cluster together with the templates created by euclidean method (green color), DBA (blue color) and Soft DTW for $\gamma = 0.5$ (red color). Detailed figures with separate results for each averaging method are presented in figures A.1-A.7 in Appendices.

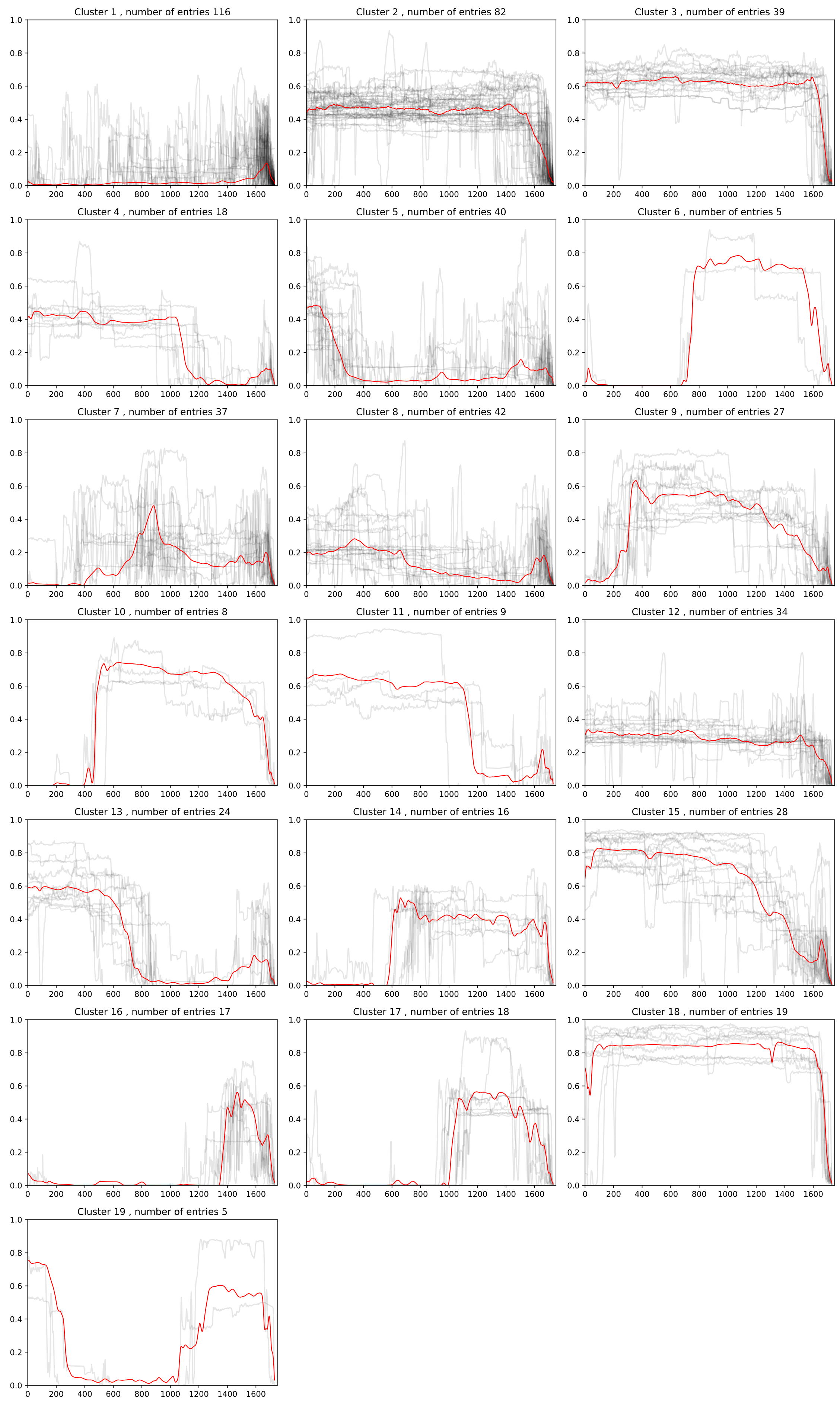


Figure 6.1: 19 Clusters, half of each cluster's timeseries presented (faded black color), Soft DTW templates, $\gamma = 0.5$ (red color)

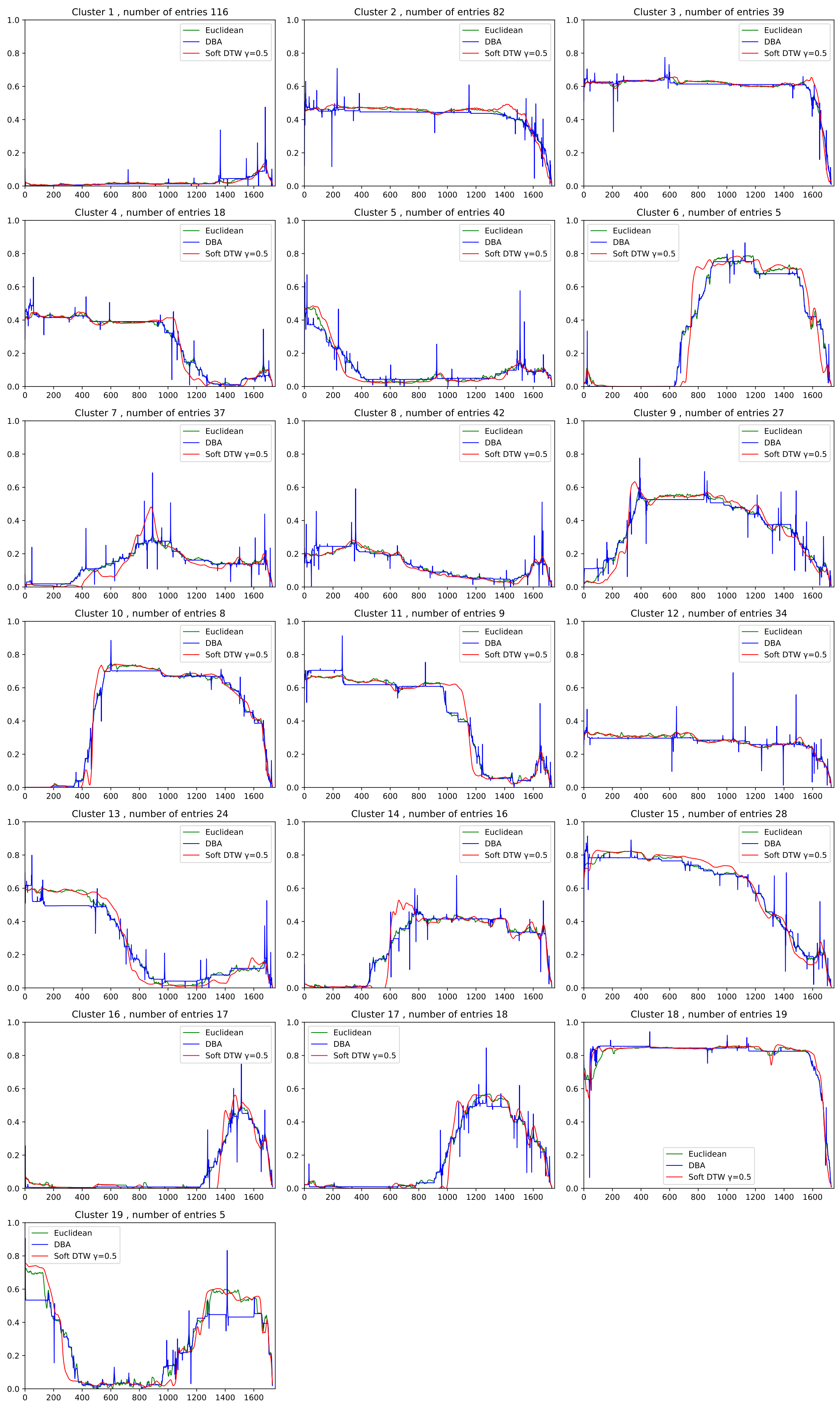


Figure 6.2: 19 Clusters, All methods' templates, $\gamma = 0.5$

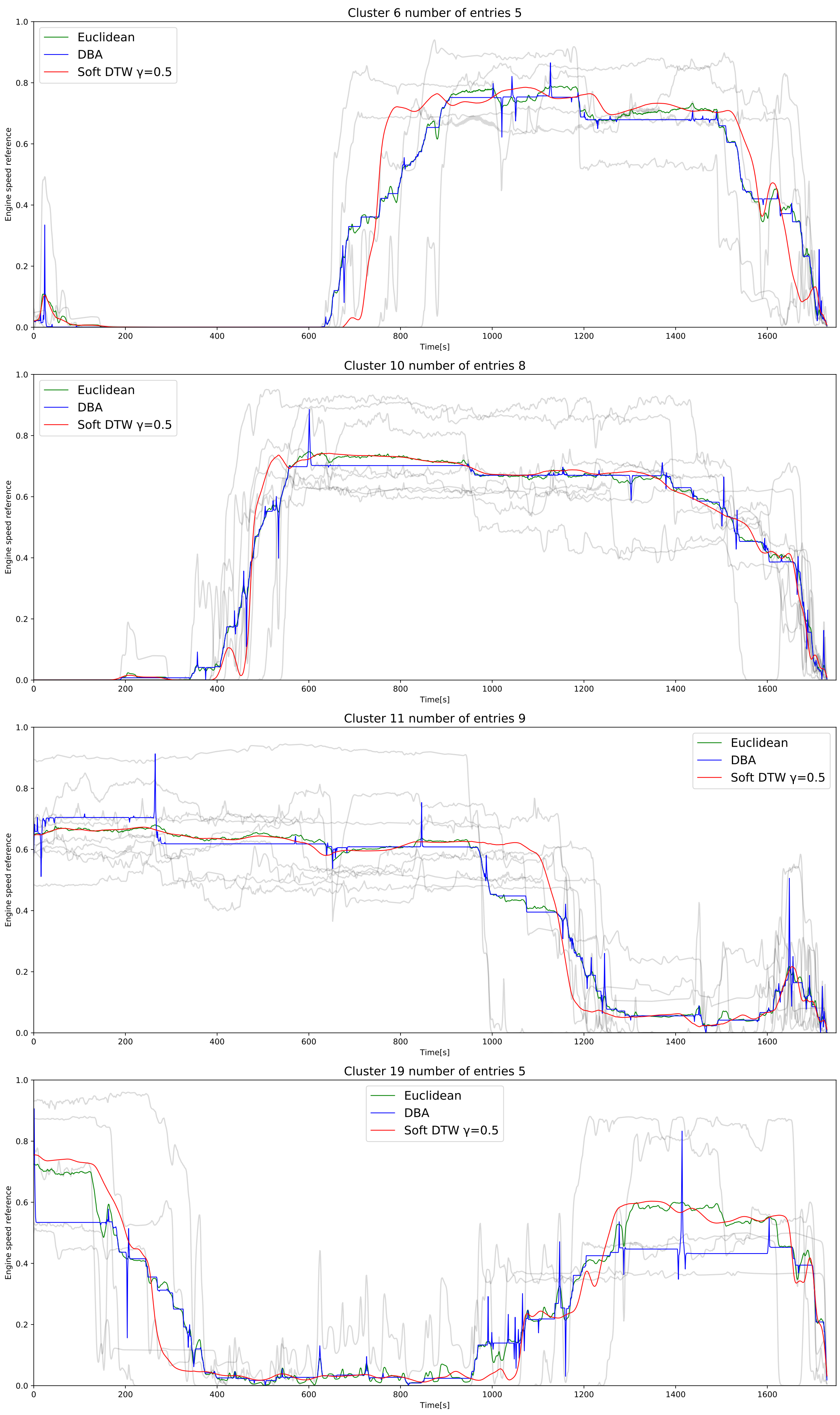


Figure 6.3: Clusters no. 6, 10, 11 and 19 with templates, $\gamma = 0.5$

Clustering results analysis

Based on observation of clusters results' visualization in figure 6.1, the clustering results in 19 clusters overall can be considered as satisfying. It is evident from figure 6.1 that the number of timeseries that each cluster contains is variable. As a result, it is a logical decision to sort the clusters in categories based on their containing number of timeseries. Specifically, the clusters can be categorized in the following groups:

- 4 clusters (21% of total clusters) contain from 5 to 9 timeseries each.
  Clusters number 6, 10, 11, 19 contain 5, 8, 9, 5 timeseries, respectively, which in total represent 4.6% of all timeseries. The timeseries in each of these clusters seem to be very similar with each other and, therefore, one can consider the clustering outcome very successful. However, as explained in theoretical background (see section 2.5), clusters with low number of timeseries are prone to the risk of overfitting. As a result, these cluster must be handled cautiously, because the number of containing timeseries is not sufficient enough to reject with confidence concerns for overfitting.

- 8 clusters (42% of total clusters) contain 16-28 timeseries each.
  Clusters number 4, 9, 13, 14, 15, 16, 17, 18 contain 18, 27, 24, 16, 28, 17, 18, 19 timeseries, respectively, which in total represent 28.6% of all timeseries. The timeseries in each of these clusters seem to be very similar with each other, same as in first category, and, therefore, the clustering result can be considered very satisfying. In addition, as the number of containing timeseries in each cluster is higher than the previous category of clusters, the risk of the clusters being prone to overfitting is reduced. The amount of timeseries in each cluster and the clustering results are in this category the best among all categories and, therefore, it is considered that clustering is performed in the best way compared to all other three categories (the one discussed previously and the other two that follow). The above conclusions are very positive for the whole analysis. Specifically, if one considers that the clustering results in this category are satisfying, since these clusters represent 42% of the total clusters and 29% of total input dataset, which are large percentages, then this conclusion is also very encouraging for the whole analysis.

- 5 clusters (26% of total clusters) contain 34-42 timeseries each.
  Clusters number 3, 5, 7, 8, 12 contain 39, 40, 37, 42, 34 timeseries, respectively, which in total represent 32.9% of all timeseries. As can be seen by these clusters' visualization, the in cluster similarity of timeseries is still good and can be considered as adequate. However, the number of timeseries in each one of these clusters is much higher than in the previous category. This might result to including in cluster timeseries that might have variations in several parts compared to the majority of timeseries in the same cluster. In other words, there might exist timeseries in a cluster that are much different with all the others in the same cluster. However, this is not considered negative and does not imply at any case that the clustering is unsuccessful. In contrast, it is more than expected, because vessels during maneuvering face many unexpected situations. Therefore, even when they face almost the same conditions and aim to follow the same loading "path" in different maneuvering situations, they might face unpredictable situations that will be mirrored in the measured loading cycles in way of variations of load in respect to time.

- 2 clusters (11% of total clusters) contain 80 to 120 timeseries.
  Clusters number 1 and 2 contain 116 and 82 timeseries, respectively, which in total represent 33.9% of all timeseries. In these clusters, the visualization of clustering

offers little to no understanding of the similarity of the contained timeseries, due to their high number of containing timeseries. A possible explanation of the clustering algorithm behaviour in these cases is that there are two common groups of load cycles that are presented in the one third of the maneuvering samples in the given dataset. Specifically, these loading cycles might represent vessels approaching the same ports for the majority of their voyages in certain standard routes. However, since there are no other extra available information about the vessels, their voyages, their type, etc. this is just an assumption and the results of these two clusters should be used cautiously.

Clusters' templates results analysis

Euclidean distance method results discussion

The templates created based on the euclidean distance method are presented in figure 6.2 and separately in figure A.1 in Appendices and in indicative clusters together with other templates in figure 6.3. As described in theoretical background (see section 2.4), the euclidean distance method calculates the euclidean distance between all timeseries in cluster and creates the respective template. This method does calculate differences of the timeseries in their values (y-axis), but has the disdvantage that does not take into consideration differences in the time axis (i.e compression and stretching) (see section 2.4). An example of how missleading euclidean distance can be, was presented at section 2.4. Based on the above, euclidean distance templates outcome can be considered satisfying and acceptable only for a basic analysis of the timeseries clustering. One can obtain a basic insight of the results by using this distance method. For further, more holistic analysis other methods should be used, e.g. DTW.

DBA and Soft DTW results discussion

The templates created with DTW BA or DBA method are presented in figure 6.2 and separately in figure A.2 in Appendices and in indicative clusters together with other templates in figure 6.3. These results are the same with the templates created with the Soft DTW for value of variable $\gamma = 0.01$ presented in figure A.3 in Appendices. This is as expected and the reasons for this are the following. The γ parameter of the Soft DTW method was selected in many different values [0.01, 0.2, 0.5, 0.7, 1]. As explained in theoretical section 2.4, lower values of gamma variable result in less smoothing. The closer the value to zero, for example 0.01, the closer the method and results to those of DBA method. For $\gamma = 0.01$ (which is close to zero), the results are the same with DBA. From figures A.3-A.7 in Appendices, it can be seen that as the variable gamma becomes higher, the templates are becoming smoother. Lower gamma produces the least smooth templates that present very steep changes in y axis (engine speed reference values). As the variable gamma gets a higher value, the templates are becoming smoother and smoother as can be seen in figures A.4-A.6 in Appendices and the fluctuations in y axis are reduced. For the highest value of gamma ($\gamma = 1$), the templates curves are the smoothest (figure A.7 in Appendices). As already explained in chapter 1, load cycles during maneuvering are a complex field of study. Engine's load during maneuvering faces high volatility and changes in a high rate. It is evident that this must also be depicted in the templates that are qualified as most suitable for this thesis' analysis. In addition, the templates should not be too volatile, because valuable information and changes in respect to time will be missed due to high deviations. Based on these two facts, in order to obtain a neither too smooth nor too noisy and volatile result, the value 0.5 for γ variable (figure A.5 in Appendices and red color line in figures 6.1 - 6.3) was selected as the most suitable and the one that produces the optimal results among DBA and all Soft DTW methods. This is also the reason that only this template, among all Soft DTW and DBA tempate results presented in Appendices, is selected to be plotted in figures 6.1 - 6.3.

Optimal results estimation

In order to provide an estimation of the best results, the disadvantages of the euclidean distance method for timeseries analysis discussed above and advantages of the DTW and Soft DTW methods, also discussed above and in section 2.4, are taken into account. In addition. the example in section 2.4 shows how DTW distance method can provide the correct answer to a timeseries analysis problem, where euclidean distance method fails to reach the correct result, because of its diasbility to find differences in the time axis (e.g. compressions and stretching). Finally, considering all the forementioned and the visualization of the clustering and templates of figures 6.1-6.3, it is determined that the optimal templates are created by the Soft DTW method and, specifically, when the variable gamma γ is equal to 0.5 (figure A.5 in Appendices). However, it must be noted that this is the author's opinion and conclusion. One might conclude that results of Soft DTW with $\gamma = 0.2$ might be the optimal, etc. This varies from researcher to researcher.

Pattern recognition in templates

In the created representative loading cycles (templates) loading patterns (or loading profiles or loading phases) can be identified. In addition, the templates can also be considered that follow one of these loading profiles. The templates examined are the ones created with Soft DTW method for γ=0.5 in figure 6.2 with red color or at figure A.5 in Appendices. Three different types of loading profiles are identified:

1. The first one contains more or less constant engine load speed.
   In this loading profile, the fluctuations that are identified in templates are small and the load can be considered constant in the majority of the cycle's length. In this category belong templates of clusters numbers 1, 2, 3, 12 and 18. Template 1 has very low engine load and operates at extremely low engine speed, close to zero. Before the stoppage, the engine load increases for a short duration of time, then falls steeply and the the engine stops. Templates 2, 3 and 12 are almost the same in way of shape and variations in relation to time, but they differ in the value of engine load. Specifically, all three templates operate at constant engine load for almost the whole duration of time with few very short flactuations, until the load drops quickly in the end and the engine stops. The difference is that template 2 engine speed is close to the region of 0.45, template 3 engine speed is close to region of 0.62 and template 12 engine speed is in the region close to 0.33. Finally, template of cluster number 18 is very similar in shape with templates 2,3 and 12 described above, but it can be seen that it has two significant variations in engine speed, even if they have very short time duration. Specifically, a steep decline and then increase in the beginning of the cycle from 0.75 to 0.55 and then up to 0.8 that is the constant engine speed observed in the majority of the cycle and then a steep, but very short-term, decrease towards the end of the cycle. The constant engine speed that is observed, as mentioned, in this cycle is close to 0.8.

2. The second type consists of varying engine load and speed.
   In this type of load profiles, a constant load that can represent the whole profile cannot be identified. This is because engine speed changes in an unexpected way and follows no certain pattern. In these loading cycles, it can be assumed that the

vessels face several unexpected conditions during maneuvering and, therefore, have to change their load and engine speed at high frequency. In this type of load cycles belong the templates of clusters numbers 7, 8, 9, 10, 11, 14, 15, 19. As can be seen in all these templates, they are much different than the first loading profile type as they do not move at a close range around a certain value in a constant way. In contrast, their values vary from low to high engine speeds in an unpredicted way.

3. The third type of loading profiles contains phases of both previous types.
   In this type of load profiles, belong all remaining templates of clusters. Specifically, templates 4, 5, 6, 13, 16, 17. These templates contain a combination of both constant and varying engine speed profiles phases. More precisely, this loading profile type consists of many different parts, which can be considered that belong to one of the two others loading profile types described above. In other words, the thirt type is like a sequence of discrete parts from the other two profile types.

Based on the observations and the above recognised patterns, it can be concluded that a realistic group of representative loading cycles should contain a combination of constant load, varying load and also a phase with starts and stops of the engine. What differs from template to template should be the duration of each phase in relation to the total load cycle, the sequence of the load phases and, also, the load/engine speed values in which the cycle mostly takes place. In a large pool of representative templates, there should also exist some cycles with mostly constant load and engine speed that represent the first type of profiles and others with no significant constant part, but only varying, which represent the second type.

Summarizing, in this section, templates for each of the 19 clusters were created. Then, the clusters were assessed based on the number of containing timeseries and they were categorized in four groups. Finally, in the created templates, three different types of loading profiles or phases were identified and all clusters templates were categorised based on them. One can now use the results and conclusions stated above to select groups of timeseries or their aggregating templates, depending on the desired characteristics and properties that they should have.

6.2 25 clusters

Since the silhouette score (figure 5.4) was maximized for number of clusters equal to 19 and 25, the exactly same clustering proceedure as in section 6.1 was followed for 25 clusters. The respective results are presented in the next figures (figure 6.4 - 6.6) and in figures A.8 - A.14 in Appendices.

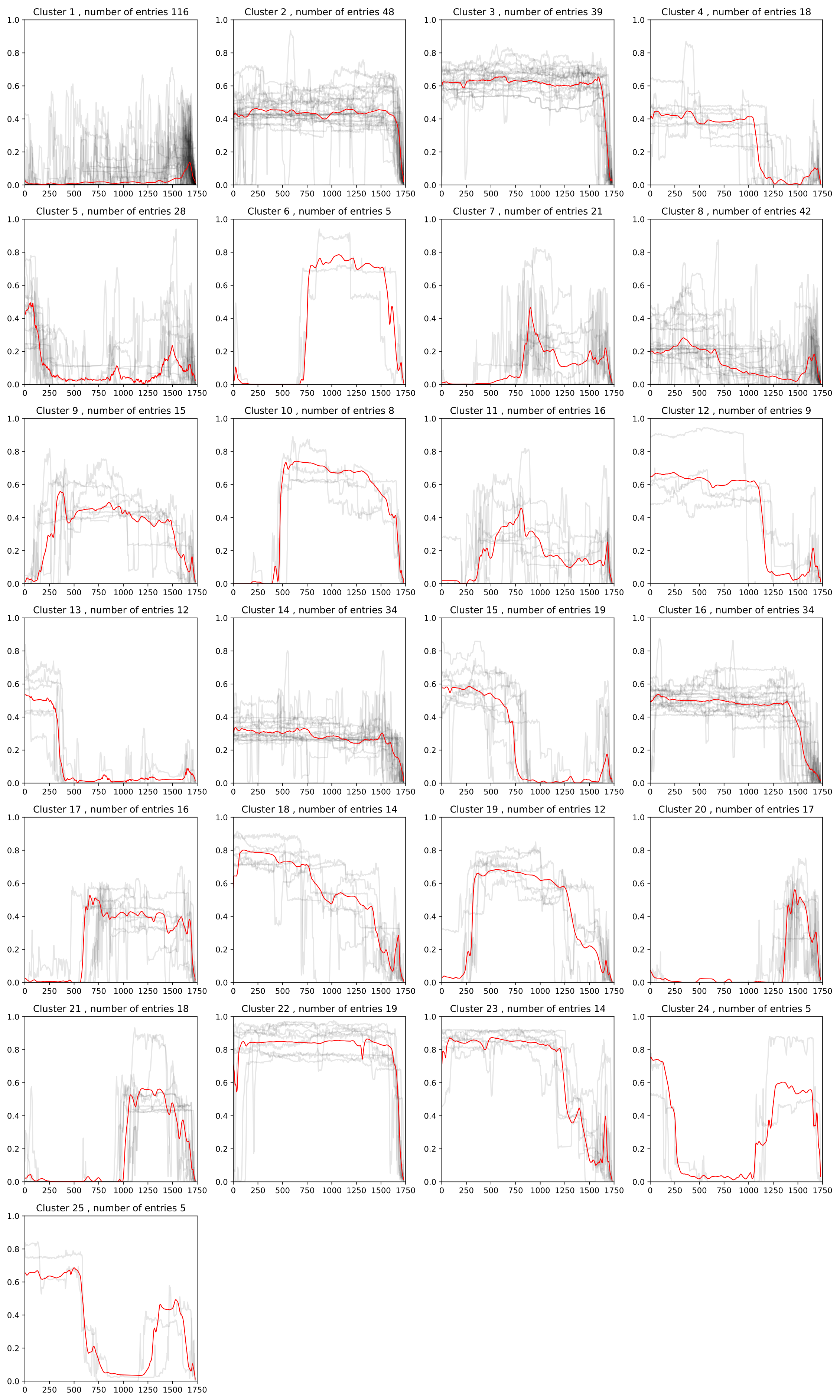


Figure 6.4: 25 Clusters, half of each cluster's timeseries presented (faded black color), Soft DTW templates, $\gamma = 0.5$ (red color)

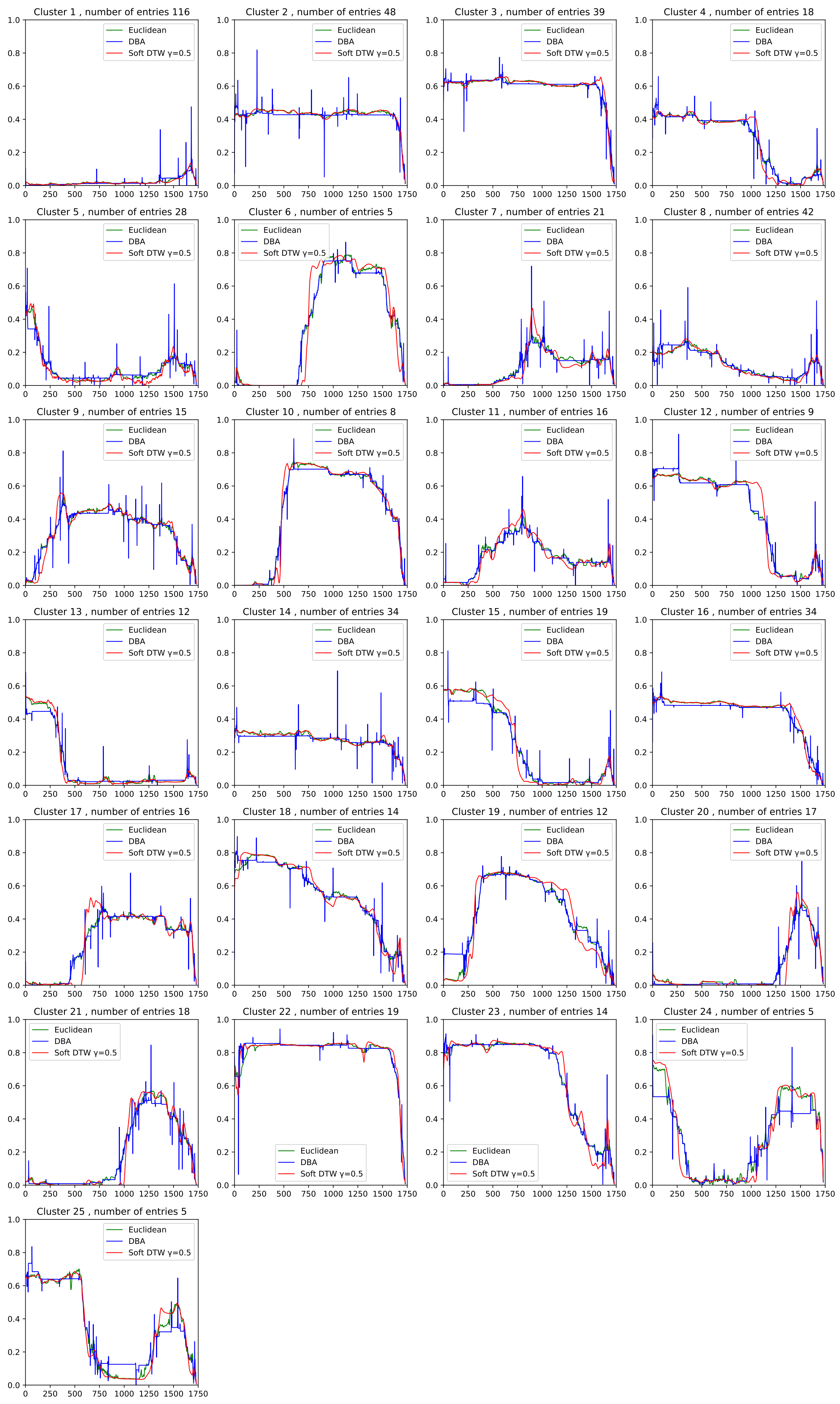


Figure 6.5: 25 Clusters, All methods templates, $\gamma = 0.5$

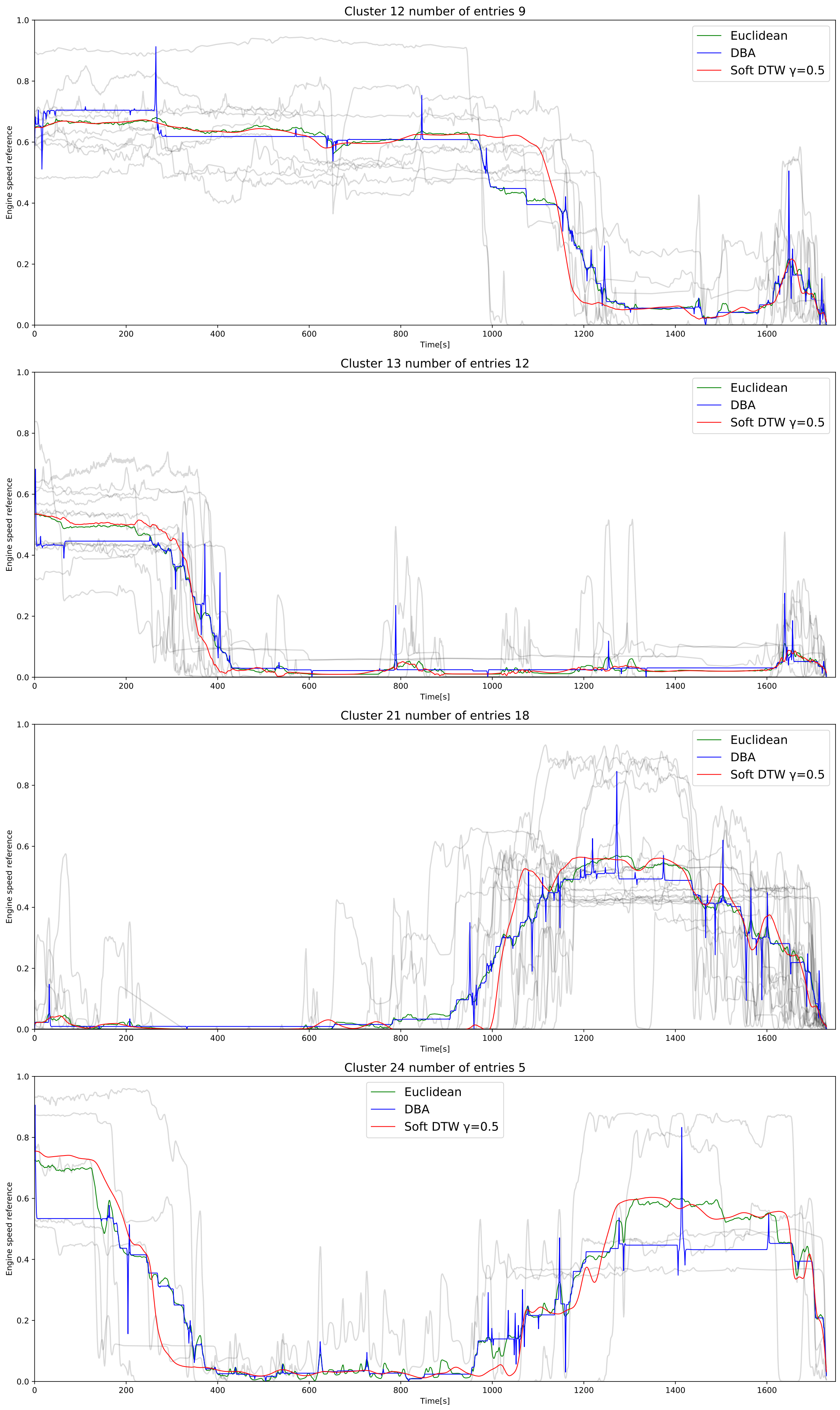


Figure 6.6: Clusters no. 12, 13, 21 and 24 with templates, $\gamma = 0.5$

The analysis of results of clustering and template creation for 25 clusters are presented the same way as the ones for 19 clusters previously. First, the clusters are categorized based on the number of containing timeseries in four types, same as for 19 clusters. Then, the clustering templates are assessed and the optimal are selected. Finally, the three loading patterns identified in section 6.1 are also identified in this case (namely: loading profiles with mostly constant load, varying load and combination of both).

The first thing that is noted is that a large part of the total clusters is the same for both clustering cases (19 and 25 clusters). Specifically, as moving downwards at hierarchical tree in figure 5.1 to cut at 25 clusters, some clusters created at 19 clustering remain unaffected. This can also be seen in figures 5.5 - 5.6. Specifically, between results for 19 and 25 clusters respectively, the following clusters are the same: 1 – 1 clusters with 116 entries, 3 – 3 clusters with 39 entries, 4 – 4 clusters with 18 entries, 6 – 6 clusters with 5 entries, 8 – 8 clusters with 42 entries, 10 – 10 clusters with 8 entries, 11 – 12 clusters with 9 entries, 12 – 14 clusters with 34 entries, 14 – 17 clusters with 16 entries, 16 – 20 clusters with 17 entries, 17 – 21 clusters with 18 entries, 18 – 22 clusters with 19 entries, 19 – 24 clusters with 5 entries. Therefore, 13/19 = 68.4% of the 19 clusters' results and 13/25 = 52% of the 25 clusters' results are the same between these two cases. Based on the above, the clusters are categorized in the following categories.

Clustering results analysis

First, the clusters are split in categories based on their containing number of timeseries. Specifically, the clusters in figure 6.4, with respect to the same categorization of the results of 19 clusters, can be categorized in the following groups:

- 5 clusters (20% of total clusters) contain from 5 to 9 timeseries each.
  Clusters number 6, 10, 12, 24, 25 contain 5, 8, 9, 5, 5 timeseries, respectively, which in total represent 5.5% of all timeseries. As is the case of the 19 clusters, one can consider the clustering outcome very successful, but these clusters must be handled cautiously, because the number of containing timeseries is not sufficient enough to reject with confidence concerns for overfitting.

- 14 clusters (56% of total clusters) contain 12-28 timeseries each.
  Clusters number 4, 5, 7, 9, 11, 13, 15, 17, 18, 19, 20, 21, 22, 23 contain 18, 28, 21, 15, 16, 12, 19, 16, 14, 12, 17, 18, 19, 14 timeseries, respectively, which in total represent 40.9% of all timeseries. As in previous respective case for 19 clusters results, it is considered that clustering is performed in the best way compared to all other three categories (the one discussed previously and the other two that follow).

- 5 clusters (20% of total clusters) contain 34-48 timeseries each.
  Clusters number 2, 3, 8, 14, 16 contain 48, 39, 42, 34, 34 timeseries, respectively, which in total represent 33.7% of all timeseries. As can be seen by these clusters' visualization, the clustering output is considered as adequate.

- 1 cluster (4% of total clusters) contains 80 to 120 timeseries.
  Cluster number 1 contains 116 timeseries, which represents 19.9% of all timeseries. In this cluster, the visualization of clustering offers little to no understanding of the similarity of the contained timeseries, due to high number of containing timeseries and should be handled cautiously.

Clusters' templates results analysis

Clusters' templates are presented in figures 6.5-6.6 and in figures A.8-A.14 in Appendices.

Optimal results estimation

Based on the same reasons as in the case of 19 clusters, it is determined that the optimal templates are created by the Soft DTW method and, specifically, when the variable gamma γ is equal to 0.5 (figure A.12 in Appendices). These templates are also plotted in figures 6.4-6.6 with red color.

Pattern recognition in templates

In the created representative loading cycles (templates) loading patterns (or loading profiles or loading phases) can be identified. In addition, each template can also be considered that follows one of these loading profiles. The templates examined are the ones created with Soft DTW method for γ=0.5 in figure 6.5 or figure A.12 in Appendices. Three different types of loading profiles are identified:

1. The first one contains more or less constant engine load speed.
   In this loading profile, the fluctuations that are identified in templates are small and the load can be considered constant in the majority of the cycle's length. In this category belong templates of clusters numbers 1, 2, 3, 14, 16 and 22. Same as discussed in section 6.1, all these templates operate at constant engine load for almost the whole duration of time with few very short fluctuations, until the load drops quickly in the end and the engine stops. The difference between them is the range in which they operate and the average value of the engine speed.

2. The second type consists of varying engine load and speed.
   In this type of load profiles, a constant load that can represent the whole profile cannot be identified. This is because engine speed changes in an unexpected way and follows no certain pattern. In these loading cycles, it can be assumed that the vessels face many unexpected conditions and, therefore, have to change their load and engine speed at high frequency. In this type of load cycles belong the templates of clusters numbers 5, 7, 8, 9, 10, 11, 15, 18, 23, 25.

3. The third type of loading profiles contains phases of both previous types.
   In this type of load profiles, belong all remaining templates of clusters. Specifically, templates 4, 6, 12, 13, 17, 19, 20, 21 and 24. These templates contain a combination of both constant and varying engine speed profiles phases. In other words, the third type is like a sequence of discrete parts from the other two profile types.

Based on the observations and the above recognised patterns, it can be concluded that a realistic group of representative loading cycles should contain a combination of constant load, varying load and also a phase with starts and stops of the engine. A group of representative templates should contain templates from each type described above.

Summarizing, in this section, templates for each of the 25 clusters were created. Then, the clusters were assessed based on the number of containing timeseries and they were categorized in four groups. Finally, in the created templates, three different types of loading profiles or phases were identified and all clusters templates were categorised based on them. One can now use the results and conclusions stated above to select groups of timeseries or their aggregating templates, depending on the desired characteristics and properties that they should have.

It is evident from the above analysis, certain patterns and simillarities can be identified in the templates created. For this reason, the question is arisen about what would be the result of clustering of these templates, and then creation of new templates for the produced clusters. In addition, it would be of interest to examine if the final templates after several iterations of such a procedure would contain the loading phases identified above. This was the reason that led to one extra clustering performance named ”Split” that follows in the next section.

6.3 Split method

Based on the results of the clustering in 19 and 25 clusters, the question was arised, what would be the result of clustering the templates. In sections 6.1-6.2, some patterns observed in the results were recognized and written down. What should be investigated is if clustering and averaging using the same methods as above on the created templates would lead to templates with the distinct patterns identified. For this reason, the following sequence of clustering is determined to be performed:

1. The available preprocessed 584 timeseries are clustered in 40 clusters and templates for each of these clusters are produced with Euclidean, DBA and Soft DTW methods.
2. The 40 produced templates will be clustered in 20 clusters and templates for every cluster will also be created, resulting to 20 templates.
3. These 20 templated will also be clustered in 10 clusters and new templates will be created, and this proceedure will be repeated for 5 and 2 clusters.
4. So, the result will be 5 different clusters’ templates (namely for 40, 20, 10, 5, 2 clusters respectively) produced by the above procedure.

The above method was named ”Split method” since it incorporates continuous splitting of the number of clusters and templates until their number becomes 2. It must be noted that the selection of the initial clustering in 40 clusters is not random. The selected number of initial clusters should satisfy the following two requirements:

1. The number should be high enough to provide a sufficient margin for further clustering
2. The number should be in accordance with the silhouette score method and the requirements for the selection of the number of clusters, set in section 5.2. Based on the silhouette score calculation in figure 5.4, number of clusters equal to 40 corresponds to a value of silhouette score near the maximum values and before the steep decline at 42 clusters. Therefore, this number of clusters is assumed that will provide satisfying clustering results.

Therefore, the initial clusters produced are 40. Specifically, as in chapters 5-6, hierarchical clustering is performed and, then, templates are created based on the euclidean distance method, DBA (or DTW BA) and Soft DTW. For Soft DTW the clustering is performed for value of variable γ=0.5. As discussed extensively in sections 6.1-6.2, from all template results, the best ones were considered the ones created with Soft DTW method for value of variable gamma γ equal to 0.5 and this the reason why this value for γ is chosen. Based on the above, the clustering and averaging algorithms are run and the results of clusters’ templates creation are presented in figures 6.7- 6.11 and figures A.15-A.29 in Appendices.

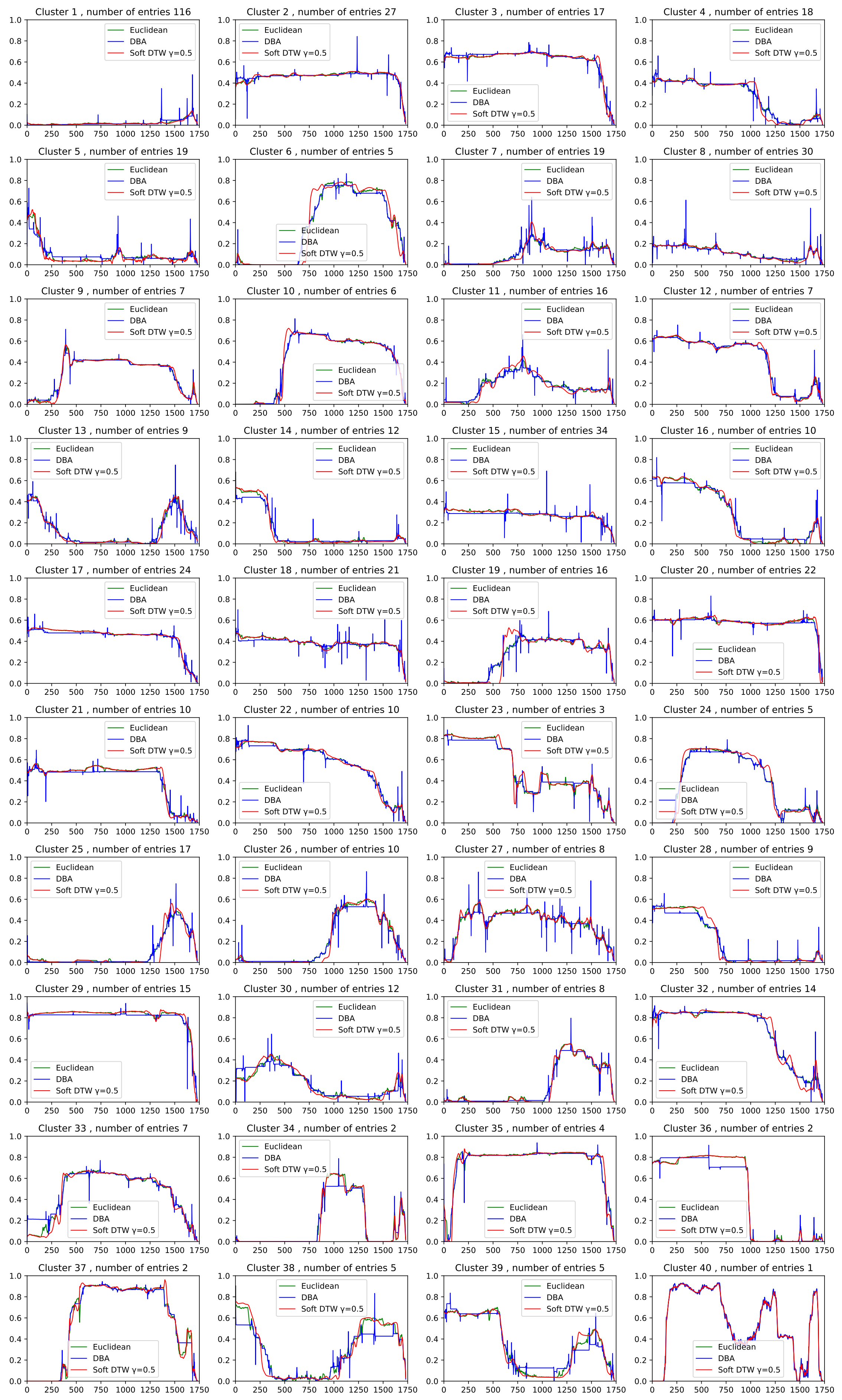


Figure 6.7: 40 Clusters templates, Split method, All averaging methods, γ=0.5

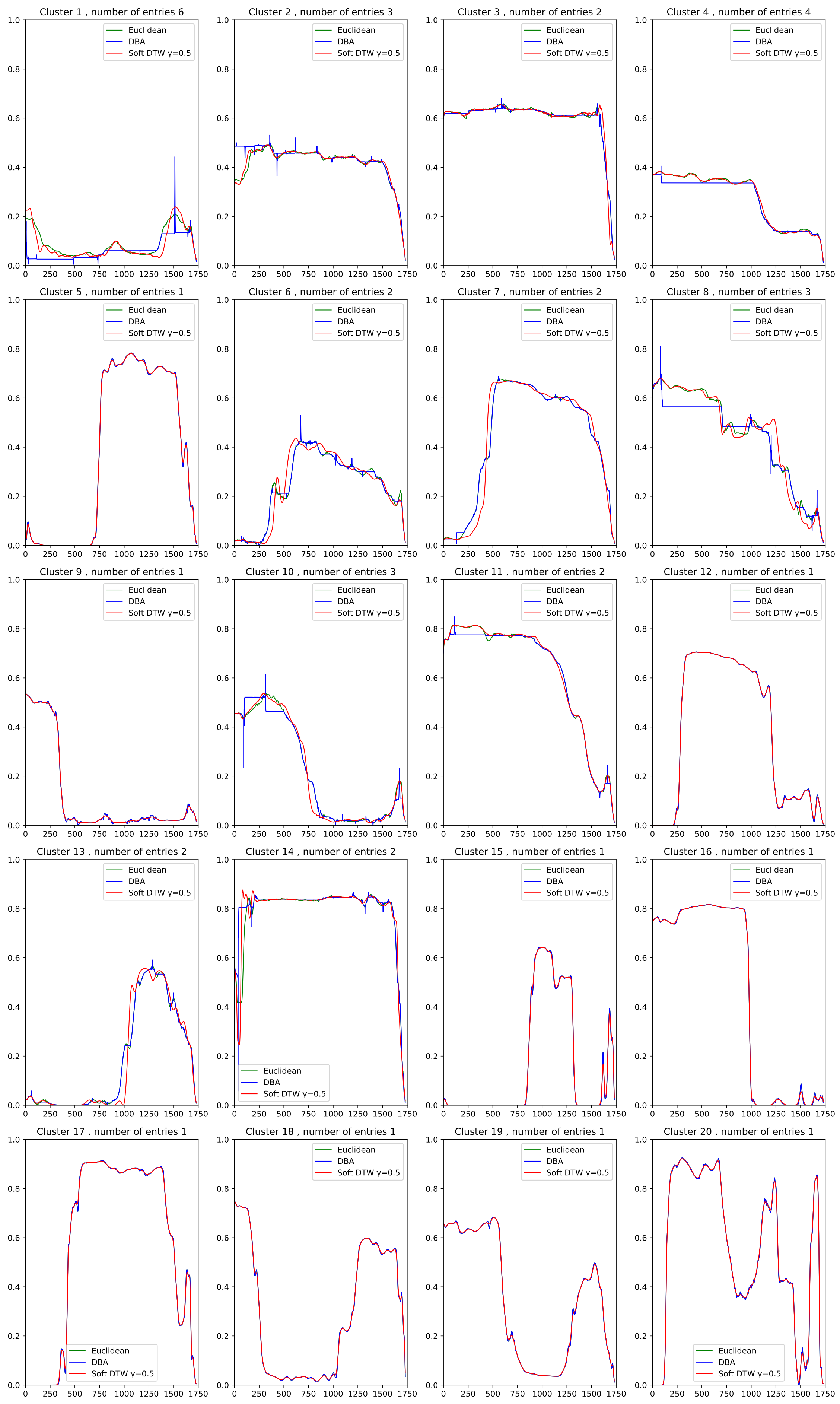


Figure 6.8: 20 Clusters templates, Split method, All averaging methods, γ=0.5

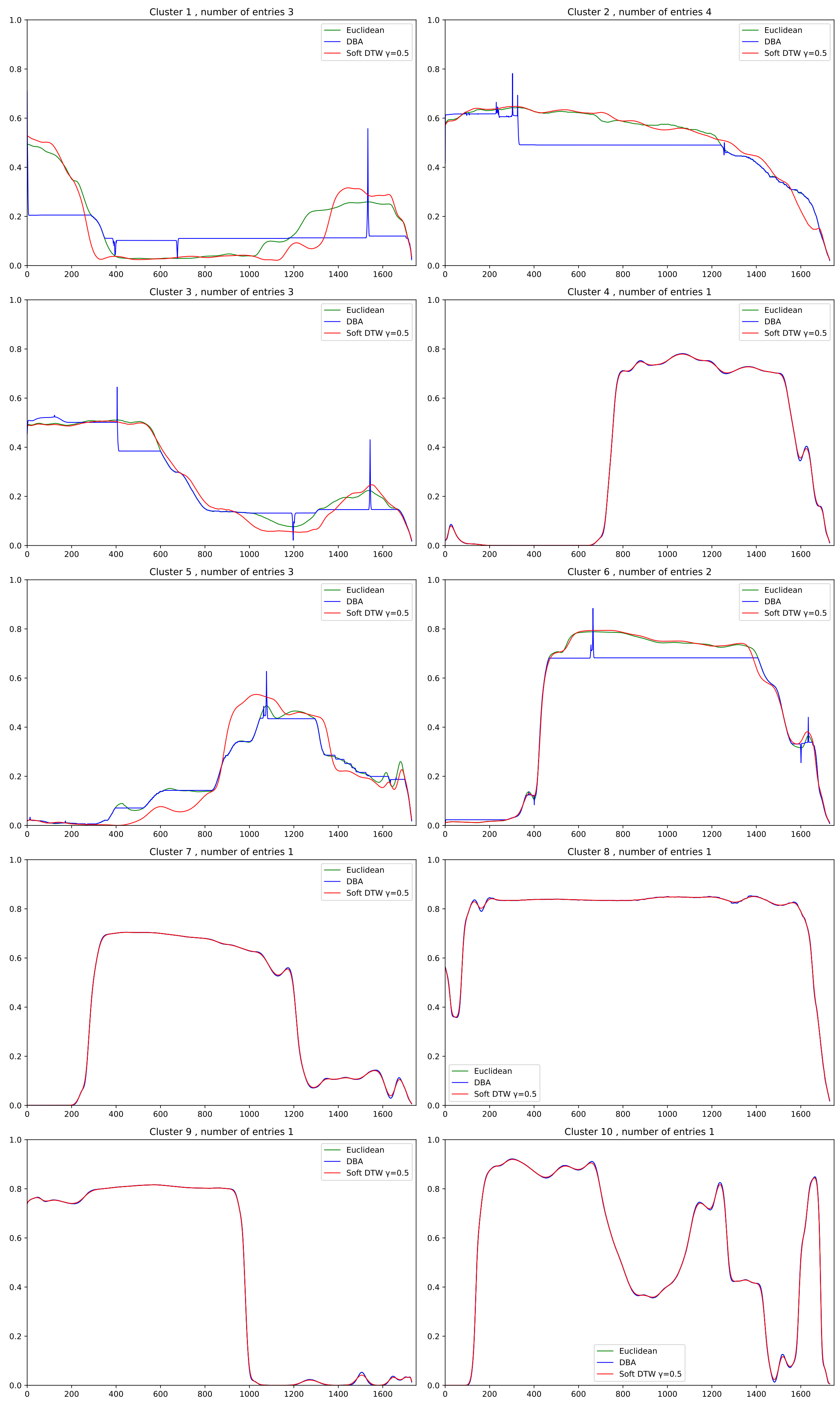


Figure 6.9: 10 Clusters templates, Split method, All averaging methods, γ=0.5

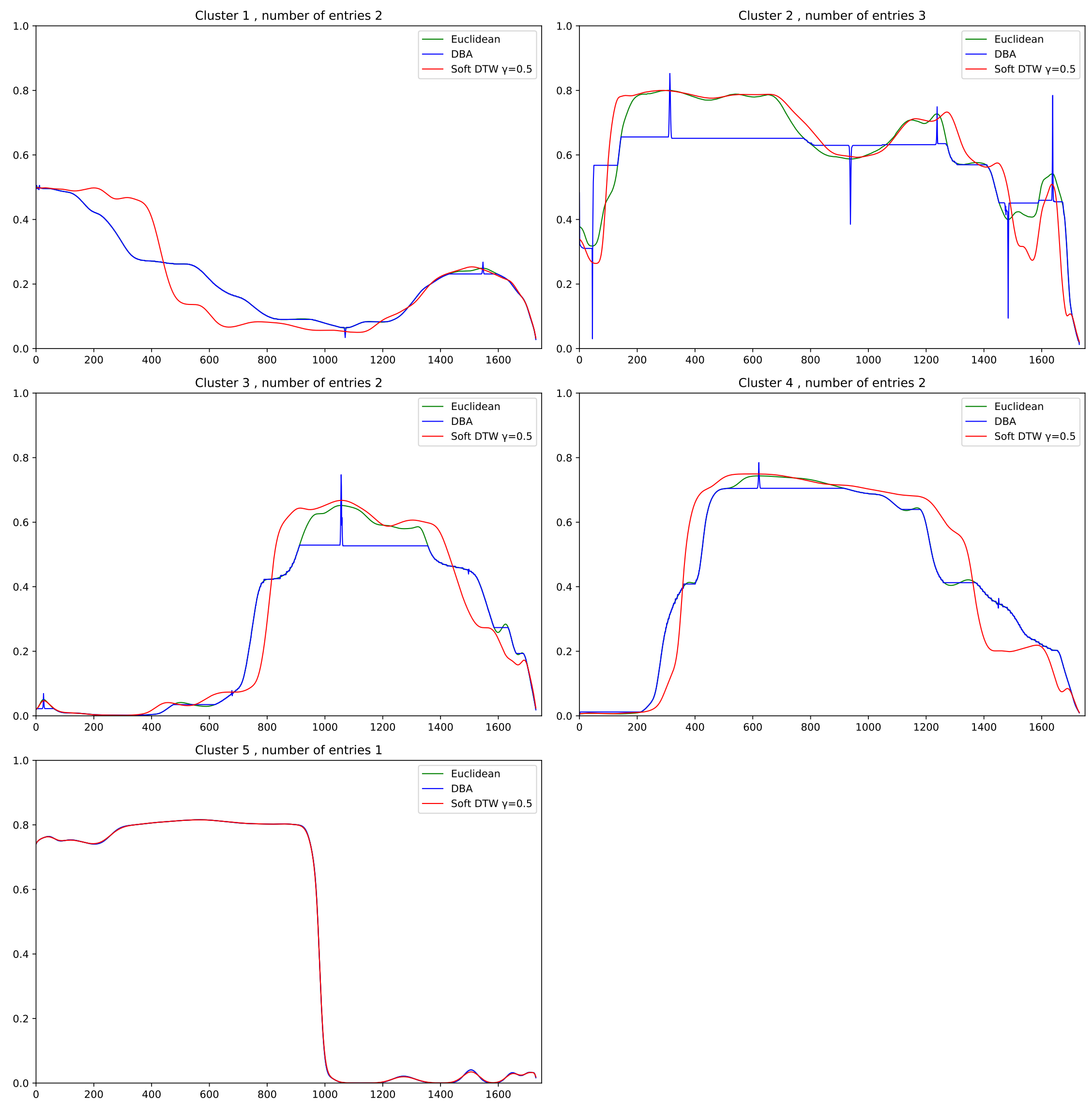


Figure 6.10: 5 Clusters templates, Split method, All averaging methods, γ=0.5

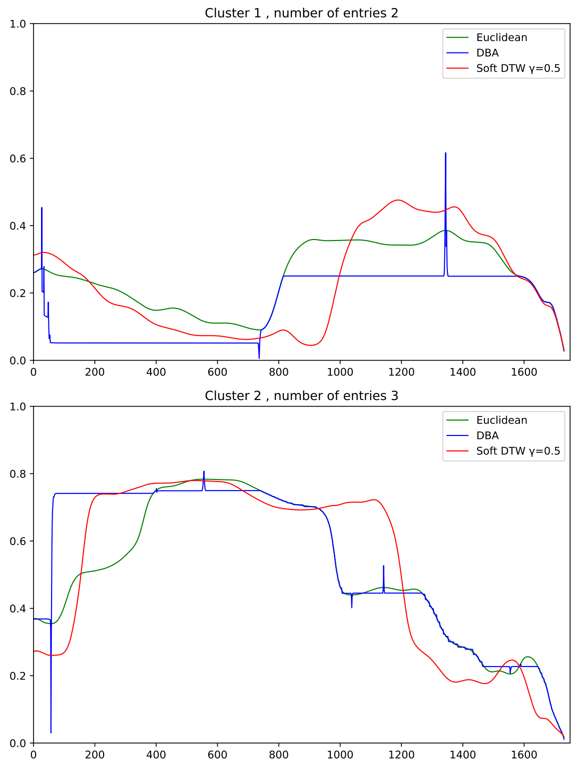


Figure 6.11: 2 Clusters templates, Split method, All averaging methods, γ=0.5

Split results analysis

In this section, the outputs of greatest interest for Split method are assessed. Therefore, the analysis of the results begins with the templates of the second clustering, which means the 20 templates in step 2 (figure 6.8 and figures A.18-A.20 in Appendices). Since the idea of performing this clustering procedure was created after observing the patterns that appear in sections 6.1-6.2 results, the main goal of this section is to assess if the patterns observed in the mentioned sections are also identified in the templates of split method. Therefore, the patterns discussed in sections 6.1-6.2 will be identified in the results of this section and relative conclusions will be made.

- 20 clusters results analysis
  In figure 6.8 and figures A.18-A.20 in Appendices, the templates created for the 20 clusters are presented. From the loading phases-profiles presented in sections 6.1-6.2, the following are identified:

  - Templates of clusters numbers 2 and 3 containing 3 and 2 entries respectively, can be estimated that belong to type 1 loading profile, which corresponds to constant load and engine speed in the biggest part of the loading cycle. These templates represent $5/40 = 12.5\%$ of the total entries in this step of clustering, which is a substantial percentage.
  - Templates numbers 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 17, 18, 19, 20 is estimated that belong to type 2 loading profile, which contains loading phases of varying engine speed and load where the load of the engine is unexpected. These templates represent $32/40 = 80\%$ of the total entries. This percentage also proves that this type of loading profiles is the most common in maneuvering.
  - Templates numbers 14, 16 containing 2, 1 entries respectively, belong to type 3 of loading profiles stated in sections 6.1-6.2, which contains phases of both constant load and variable load of the engine. These entries represent $3/40 = 7.5\%$ of the total entries.

  The above results are in accordance with the results from clustering in 19 and 25 clusters, since type 2 loading profiles represent the majority of the observed loading profiles in all these cases. It should be investigated if this pattern will be the main pattern appered in the rest of the split method's results.

- 10 clusters results analysis
  The 20 clusters templates created with Soft DTW for $\gamma = 0.5$ are then clustered in 10 clusters. In figure 6.9 and figures A.21-A.23 in Appendices, the templates created for these 10 clusters are presented. From the loading phases-profiles presented in sections 6.1-6.2, the following are identified in the mentioned templates:

  - Template of cluster number 8 only, containing 1 entry belongs to type 1 loading profile, which corresponds to constant load and engine speed in the biggest part of the loading cycle. This entry represents the $1/20 = 5\%$ of the total entries in this step of clustering. It can be seen that the percentage of type 1 loading profiles has been reduced more that half (from 12.5% to 5%).
  - Templates numbers 2, 3, 5, 7, 10 containing 4, 3, 3, 1, 1 entries respectively, belong to type 2 loading profile, which contains loading phases of varying engine speed and load, where the load of the engine changes unpredictably. These templates represent $12/20 = 60\%$ of the total entries. This percentage is still high but reduced compared to the respective one for number of clusters equal to 20, which was equal to 80%.

  - Templates numbers 1, 4, 6, 9 containing 3, 1, 2, 1 entries respectively, belong to type 3 of loading profiles stated in sections 6.1-6.2, which contain phases of both constant load and variable load of the engine. These entries represent 7/20 = 35% of the total entries. It can be seen that this percentage has been extremely increased compared to the respective for the 20 templates (7.5%).

  As can be seen from the above, it is evident that the percentage of the type 3 loading profiles has increased very much. This was expected because the algorithm includes in the same clusters, templates of different profile types (namely types 1, 2 and 3) and, therefore, the template result of those clusters is a mix of all of them, which corresponds to type 3 loading cycles.

- 5 clusters results analysis
  The 10 clusters templates created with Soft DTW for gamma = 0.5 are then clustered to 5 clusters. In figure 6.10 and figures A.24-A.26 in Appendices, the templates created for these 5 clusters are presented. From the loading phases-profiles presented in sections 6.1-6.2, the following are identified in the mentioned templates:

  - No template of cluster belongs to type 1 loading profile, which corresponds to constant load and engine speed in the biggest part of the loading cycle. Therefore this type of loading profile represents 0% of total templates in this case and has been eliminated.
  - Templates numbers 2, 3 containing 3, 2 entries respectively, belong to type 2 loading profile, which contains loading phases of varying engine speed and load, where the load of the engine is unexpected. These templates represent 5/10 = 50% of the total entries. This percentage is high since varying load and engine speed are very common during maneuvering.
  - Templates numbers 1, 4, 5 containing 2, 2, 1 entries respectively, belong to type 3 of loading profiles stated in sections 6.1-6.2, which contains phases of both constant load and variable load of the engine. These entries represent 5/10 = 50% of the total entries. It can be seen that this percentage has been further increased compared to the respective percentage for the 20 and 10 templates (7.5% and 35% respectively).

  As can be seen from the above, it is evident that the percentage of the type 3 loading profiles increases gradually as the templates are clustered to fewer clusters. The algorithm includes in the same clusters, templates of all different profile types (namely types 1, 2 and 3) and, therefore, the template result is a combination of parts with constant and varying engine speed in discrete parts, which corresponds to type 3 loading cycles.

- 2 clusters results analysis
  The 5 clusters templates created with Soft DTW for $\gamma = 0.5$ are, finally, clustered to 2 clusters. In figure 6.11 and figures A.27-A.29 in Appendices, the templates created for these 2 clusters are presented. From the loading phases-profiles presented in section 6.2, the following are identified in the mentioned templates:

  - No template of cluster belongs to type 1 and type 2 loading profiles, which correspond to constant and varying engine speed, respectively, in the biggest part of the loading cycle. Therefore these types of loading profiles represent 0% of total templates in this final case.

  - Both templates numbers 1, 2 containing 2, 3 entries respectively, belong to type 3 of loading profiles stated in sections 6.1-6.2, which contains phases of both constant and variable load of the engine. These entries represent 5/5 = 100% of the total entries. It can be seen that this percentage has been greatly increased compared to the respective for the 20, 10 and 5 templates (7.5%, 35% and 50% respectively) and is the only profile type in both clusters in this final step of split method.

  Based on the above analysis, it is evident that the percentage of the type 3 increases as the Split method is further performed, until it is the only loading profile identified in the final step of clustering and template creation.

Summarizing, in this section the Split method was performed. In the results, pattern recognision showed that the patterns identified in the templates of 19 and 25 clusters in sections 6.1 - 6.2 are also identified by the clustering and aggregating algorithms as the number of clusters is gradually reduced. In addition, the results of split method prove the conclusion made previously that if some representative loading cycles should be selected, then they should contain discrete phases of both contsant and varying engine load. All results in this chapter can be used as input in experimental facilities to provide a representative load cycle of real world vessel's operation. In the next chapter, the results of this chapter will be used to cross-check the accuracy of a classification algorithm that will be produced.

Chapter 7

Classification

In this chapter, a classification model is created that can classify given loading cycles accurately. Specifically, as further described in chapter 1 and theoretical background in chapter 2, being able to classify sample loading cycles into groups without any prior knowledge of the clusters is of great importance and interest in experimental research. More precisely, by training properly the classification model on already available data, the classifier learns to allocate additionally given load cycles and provide accurate results of unsupervised clustering. The classifier employed is the K-Neighbors Time Series Classifier or K-Nearest Neighbors (KNN), which was on detail described in theoretical section 2.5.

7.1 K-Neighbors Time Series Classifier

The preprocessed dataset contains 584 timeseries which was clustered as presented in chapters 5-6. The same dataset is also used for the classification model creation. Specifically, the following steps towards the creation of this model are followed:

- The dataset is specified. Specifically, as mentioned previously, the dataset consists of 584 timeseries with equal length after the preprocessing
- The dataset is shuffled to achieve random allocation of timeseries data samples
- The dataset is split into training and testing datasets based on selected percentages. From relative discussion and normal practices among researchers, it can be said that percentages of 70-30% and 80-20% respectively are considered the optimal. It is author's opinion that if the number of available data is high, even a combination of 90-10% can also be considered optimal
- The KNN algorithm's variables are selected. The main variable's value that must be set is the number of nearest neighbors n based on which the classifier will classify the input data. The variable n is further described in theoretical section 2.5.
- The classification is performed. Specifically, the classifier is trained using the train set and, then, performs classification in the test set. The results of the testing are then assessed in the following step.
- The results of the classification of the test set are compared to the clustering results of the test set. More precisely, cross-validation (in the form of labels cross-validating) is performed between the clustering and classification results of the test set. The classification is considered satisfying and accurate only if it provides the same label (same cluster's number) with the clustering procedure of section 6

- The accuracy of the classifier is calculated The number of correct/accurate classification cases is divided with the total performed cases. The result will be a number in the range of [0,1]. This number in the form of a percentage (%) is the accuracy score of the classifier. An accuracy score of more than 80% is considered satisfying.

- The above procedure is repeated for number of nearest neighbors n in values [3, 4, 5, 8, 10], for:

    – 300 different shuffles of the preprocessed dataset

    – Different percentages of training-test set from 40%-60% to 90%-10%, respectively

- The average accuracy and its standard deviation for every different combination of training-testing sets and value of variable n is calculated and presented in following figures 7.1-7.5.

Results

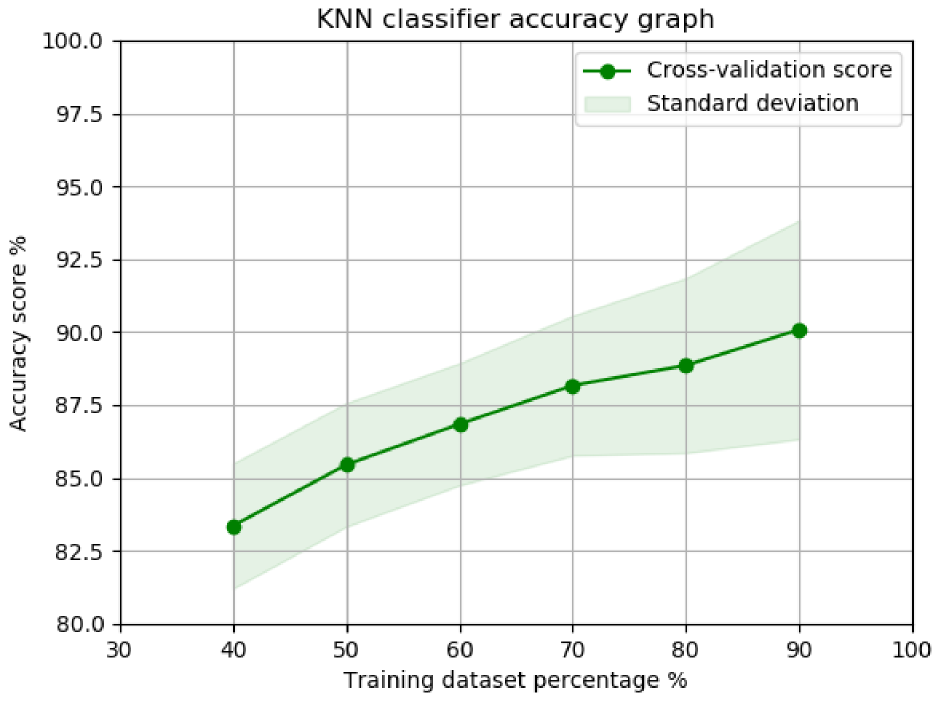


Figure 7.1: KNN classifier results for n = 3

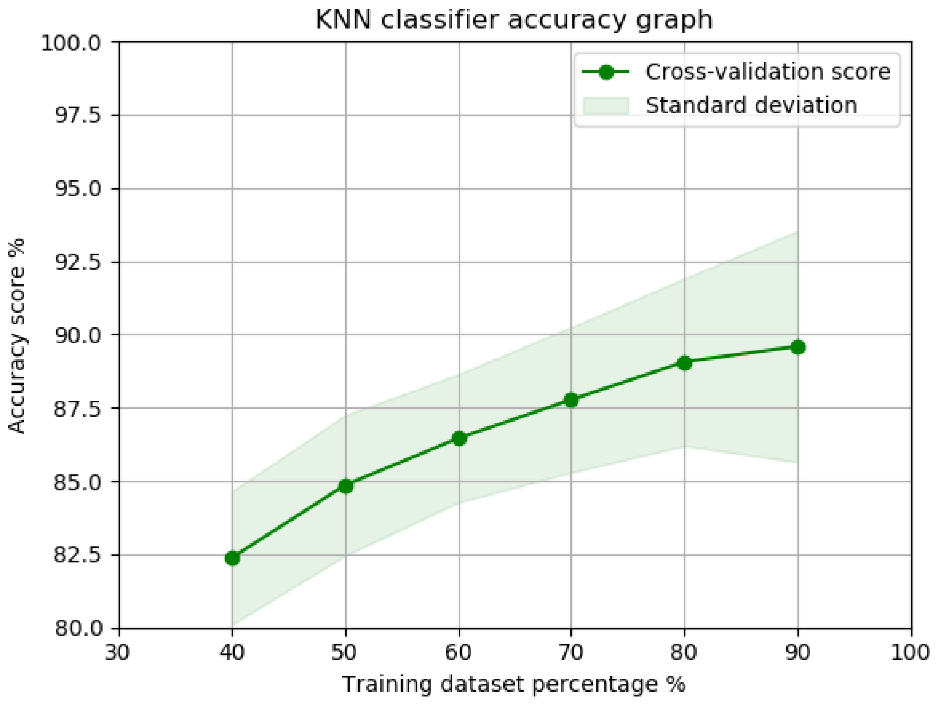


Figure 7.2: KNN classifier results for n = 4

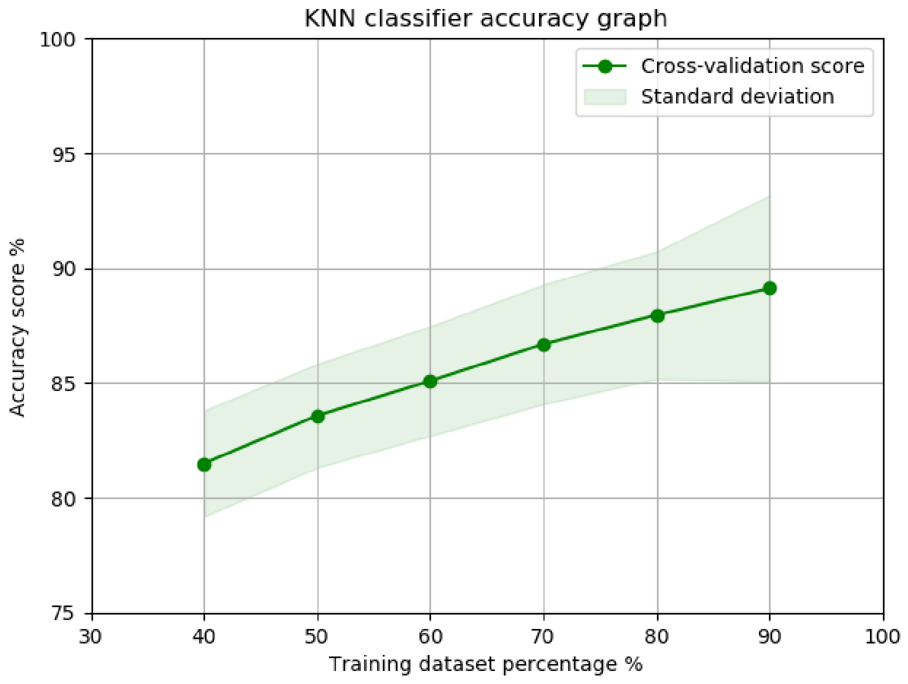


Figure 7.3: KNN classifier results for n = 5

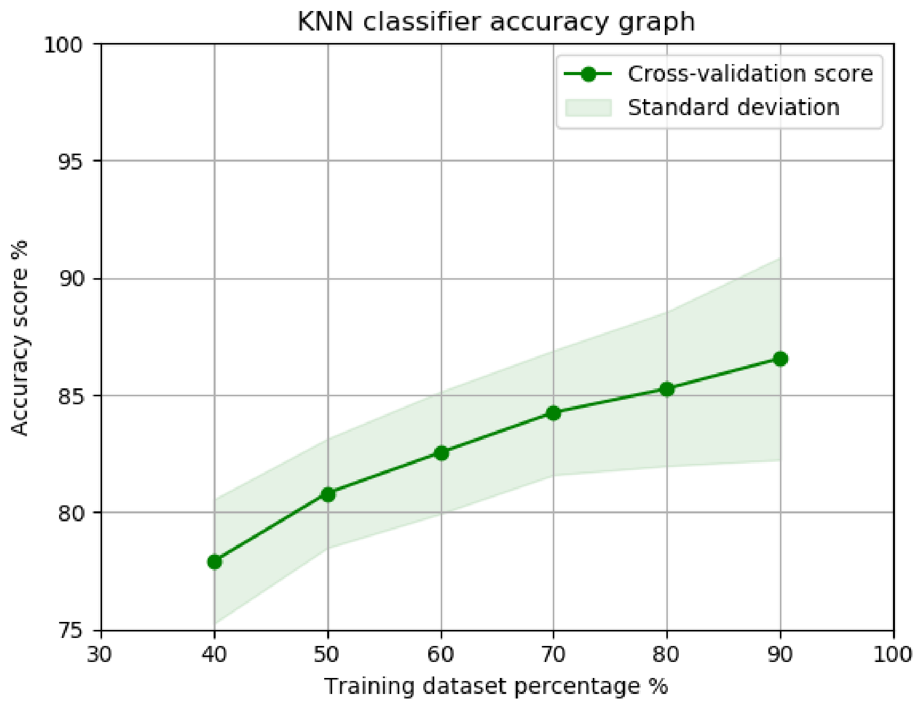


Figure 7.4: KNN classifier results for n = 8

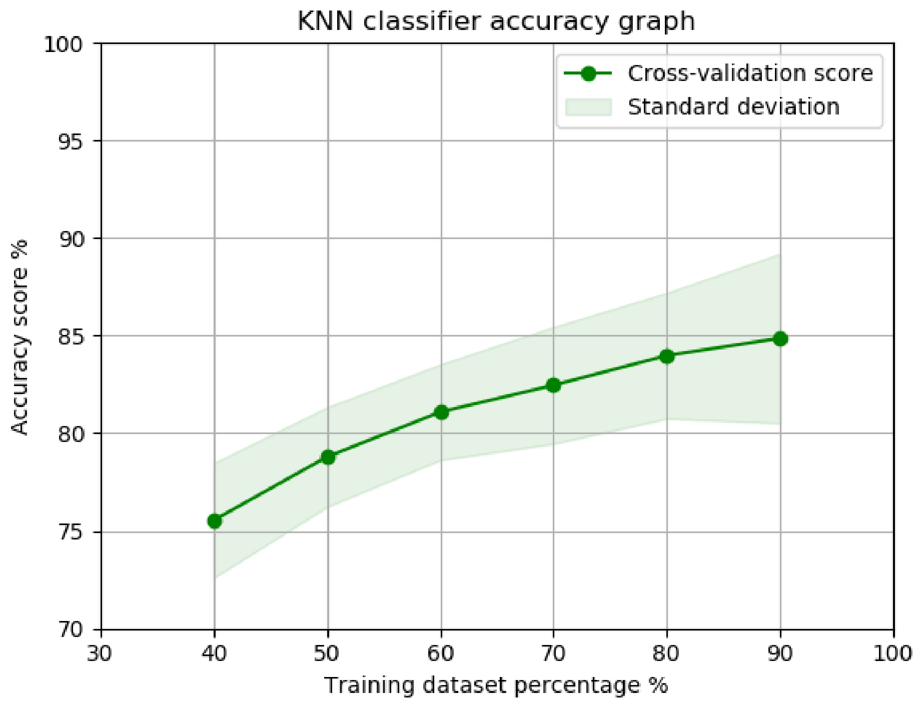


Figure 7.5: KNN classifier results for n = 10

Results' discussion

Based on figures 7.1-7.5 above, it can be observed that the accuracy of the classification results increases as the size of the training set increases as well. This happens for all different numbers of neighbors, denoted by n.

For train-test sets percentages 40-60% respectively, the classifier is more accurate for n = 3 (figure 7.1, accuracy approximately 83%), and constantly becomes less accurate as n becomes higher (for n = 10 , accuracy 75.5% approximately at figure 7.5).

For train-test set 90-10% respectively, the classifier's accuracy is between 89-90% for n = 3, 4, 5 (figures 7.1-7.3), but decreases to around 86.5% for n = 8 (figure 7.4) and around 85% for n = 10 (figure 7.5).

In conclusion, it is evident that the classifier's performance is ameliorating as the relative and absolute size of the training set increases. The performance is optimal for number of neighbors equal to three, n=3, and size of training dataset 90%.

Summarizing, the classification model produced using data analytics and machine learns concepts is capable of classifying with great accuracy given loading cycles. As the results in section 7.1 prove, a requirement to achieve minimum accuracy error is to choose the right variables for the algorithm and a large training dataset, both in terms of absolute data and relative to test set's size.This classification model, is now available and prepared classify additionally provided loading cycles and contribute to better understanding of the behavior of vessels' engine during maneuvering.

Chapter 8

Summary of key findings and further research

The original research objective as set out in the summary was to create a comprehensive tool that would classify given loading cycles during meneuvering, cluster them in groups and create representative templates of each different group. The available data were measured values of engine's variables in relation to time.

This would be achieved by firstly reducing the base of analysis by identifying the main variables that represent a loading cycle using correlation matrix and preprocess the available data. The next step would be to create a clustering algorithm using data analytics and machine learning that groups most similar loading cycles together and, also, provides a base indication of the number of clusters to be created. The third step involves templates creation for every group using different distance metrics, whose results are compared and assessed. Lastly, a classification model is created which is trained and tested with the clustering results and can classify given load cycles very accurately.

The main conclusions of the above are:

- Dimensionality reduction of engine's variables using correlation matrix led to the selection of only one variable that represents the engine's operation
- Hierarchical clustering was performed. Silhouette coefficient score is a clustering validation metric, whose result suggested optimal clustering of available data is achieved for 19 and 25 clusters. Templates were created for each cluster using Euclidean distance, DBA and Soft DTW methods and the optimal ones were concluded to be those created by Soft DTW method for $\gamma = 0.5$
- The templates were handpicked and categorized in four groups based on the number of their containing timeseries and their quality was assessed
- Pattern recognition resulted in identifying three different loading patterns or loading profiles in the created templates: type one with mostly constant engine speed and load for the majority of time, type two with varying engine speed and load and type three that is a combination of both types one and two and which contains discrete parts of those two profiles
- The pattern recognition procedure instrumented the performance of split method clustering, which involved consecutive clustering and averaging iterations of the templates created and, finally, patterns were also recognized in the results. In the final step of split method, the templates contained only type three loading profile with discrete parts of constant and varying load.

- Finally, preprocessed timeseries were used to train and test KNN algorithm classifier. Cross-validation of classification and clustering results resulted to the accuracy calculation of the classifier for several combinations of classification variables. The classifier was assessed as very accurate.

In addition, throughout the course of this thesis the below areas were identified as the ones most deserving of further research:

- Re-evaluation of the entire algorithmic framework when more data and information for the vessels have been amassed. More specifically, with an even more sizeable mass of data that are not limited to maneuvering of the vessels, but also include load conditions during normal high seas operations, machine learning algorithms could be performed and assessed to produce results with a more global application.
- Add new dimensions (e.g. vessel's speed, deadweight, weather conditions, vessel's chartering information, etc.) in order to tackle the high complexity of a vessel's behavior and operation that is not limited to the load of the engine. Including these dimensions may lead to more accurate clustering and templates creation. However, significant modification will have to be done to the algorithms used to become suitable for multi-dimensional clustering.
- Acquiring access to vessel database with positioning data will be crucial at vessel load cycle analysis.
- Further investigate clustering options including both the clustering method used and the distance metric. Specifically, in this thesis hierarchical clustering method is employed, which by default uses Euclidean distance metric, because of the reasons described in chapter 2. In future work, other clustering methods should be applied and also other distance methods used for clustering with best candidate the dynamic time warping (DTW). The main reason this method was not applied in this thesis is that it requires extreme computation time and processing power, which can only be achieved by parallel processing. This is out of the scope of a diploma thesis and requires extensive knowledge of computer science and coding.
- Finally, other classifiers that use neural networks and not lazy algorithms should be used in future work to classify load cycles. In addition, KNN can also be used in the future with DTW as distance metric, which requires extreme processing power and computational cost.

Chapter 9

Conclusions

The findings of this thesis confirm that there is merit in applying advanced analytics concepts and machine learning algorithms in attempt to create representative loading cycles of marine engines and tackle present challenges as discussed in the introduction. Although limitations in the type of available data (e.g. no available information about vessel's speed, deadweight, weather conditions, etc.) allowed no further analysis, the results of this thesis can be considered accurate and may be used as a starting point for further investigating vessel's operations and as input to experimental marine power plants facilities.

A systematic methodology has been developed to generate several loading cycles that are representative compared to a given database of real world vessel's loading data. A crucial part in the generation of loading cycles is to take into account the fluctuations in time axis and not only in values using data analytics methods, like DTW. The computation time is tractable so that many cycles can be generated if so desired, and it is a good way to make use of available large datasets. The feasibility of the approach was successfully demonstrated in a number of examples, and thus the method can be a sound engineering tool for development and classification of standard loading cycles.

As described in the introduction, the analysis of such a complex field, like loading cycles of vessels, depends on many variables. An overall project that includes multiple data measurement and collection of several variables of vessels' operation should be initiated by a company or organization that has the capability and size to undertake this.

In this thesis, limitations and obstacles were come across when working towards the completion of this thesis. The primary restriction was dataset dimensions: advanced analytics and machine learning need a highly dimensional dataset to take into consideration most cases of vessel's movements and engine loads. Including more variables that are not highly correlated with each other to the clustering procedure, would extend the results in other dimensions and would make the creation of representative loading cycles multi-variable function and not explicitly function of engine's speed. Since there was only one variable included in clustering algorithms, the patterns and loading phases that exist in the results were hand-picked to ensure their structure was not prone to overfitting due to lack of extensive data to distinct them. An other restriction was extreme computing power for the processing of such enormous piles of data. To apply more complex and sophisticated algorithms in larger datasets, parallel processing and very high computational power are required.

This thesis proved that with sufficient size of available data useful models towards optimization of vessels engines operation efficiency can be created, including but not limited to the optimization of engine's loading cycle through pattern recognition and categorization. That was the main contribution of this diploma thesis.

Appendices

Appendix A

Analytical template results

A.1 19 clusters templates results

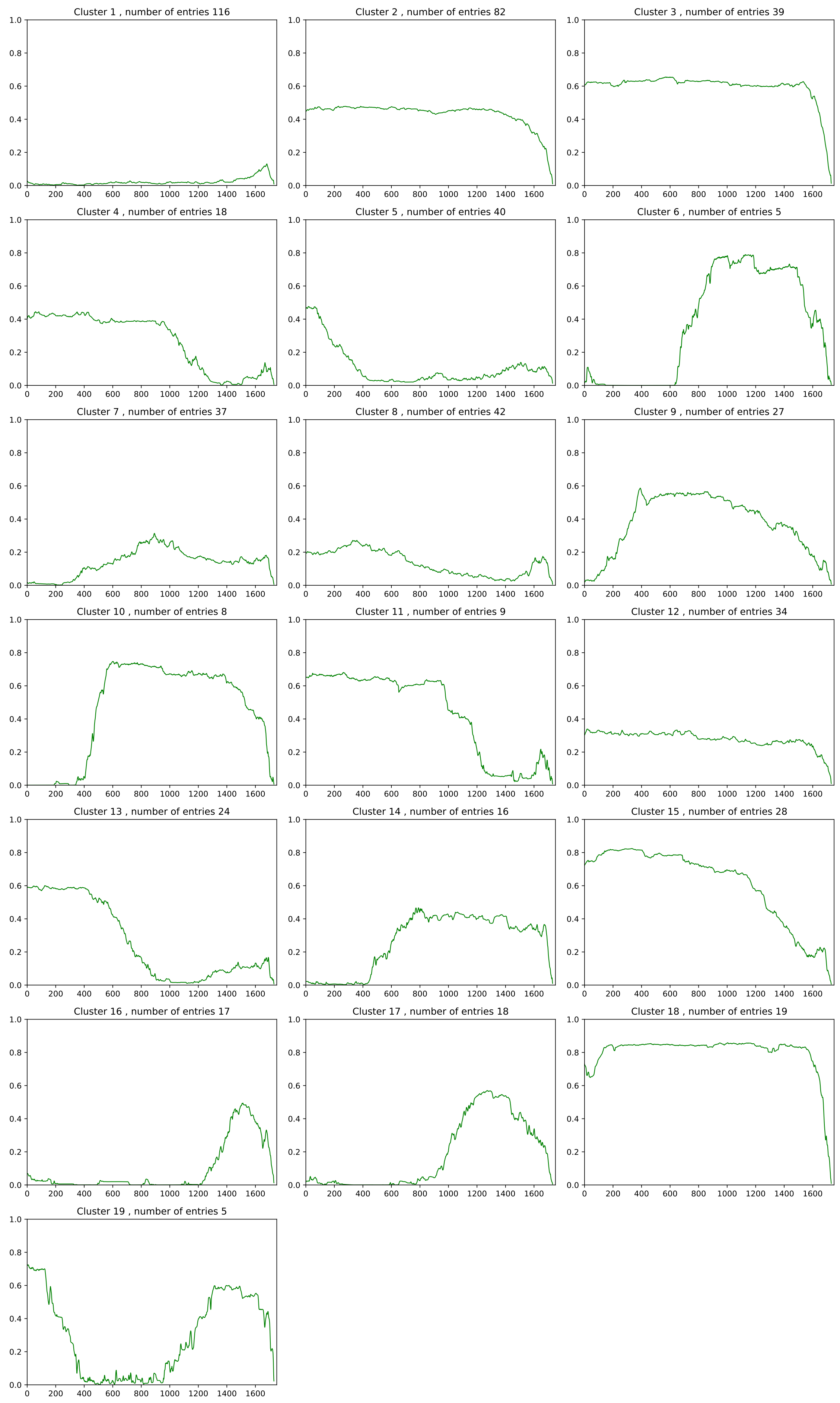


Figure A.1: 19 Cluster templates, Euclidean distance method

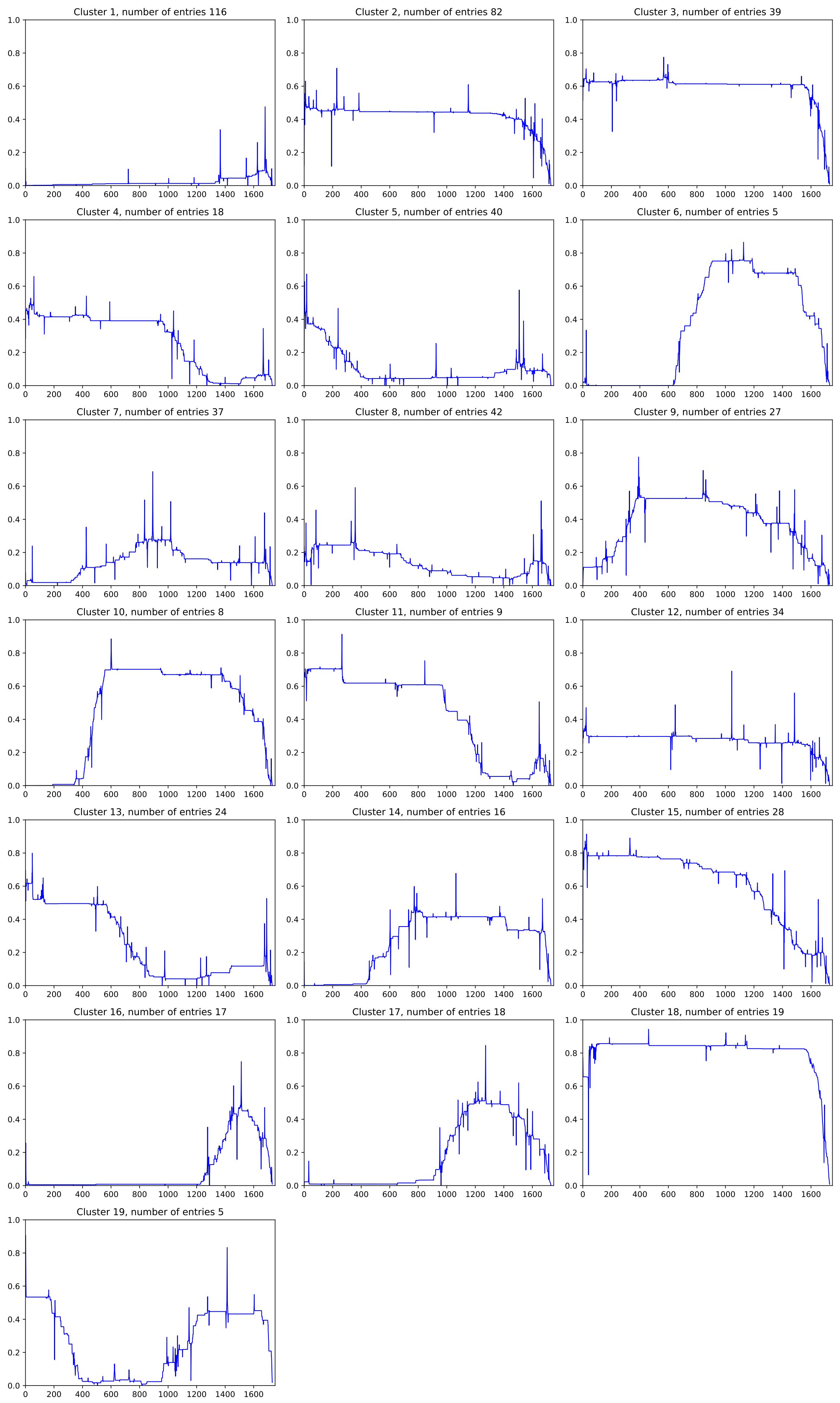


Figure A.2: 19 Cluster templates, DBA distance method

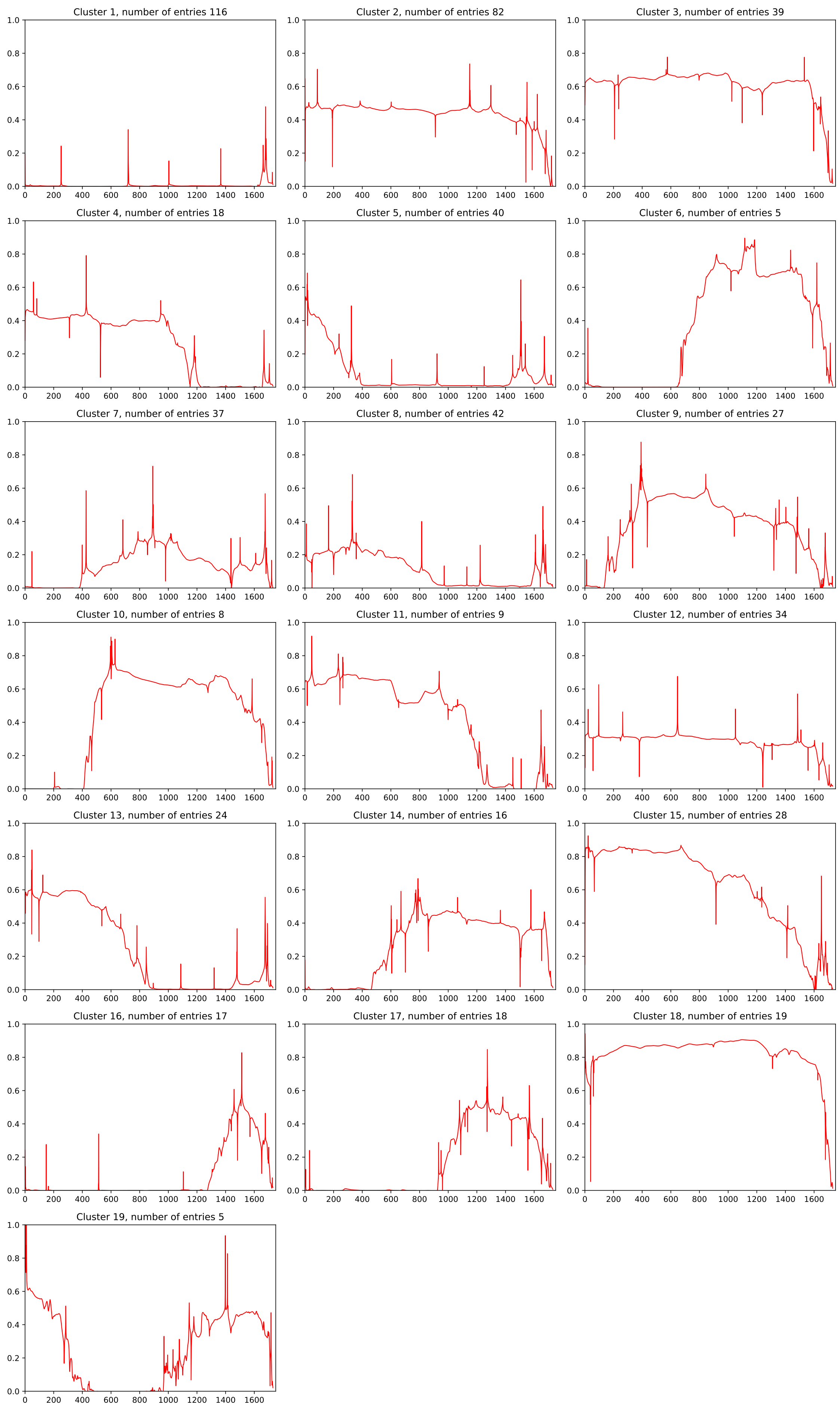


Figure A.3: 19 Cluster templates, Soft DTW method, gamma=0.01

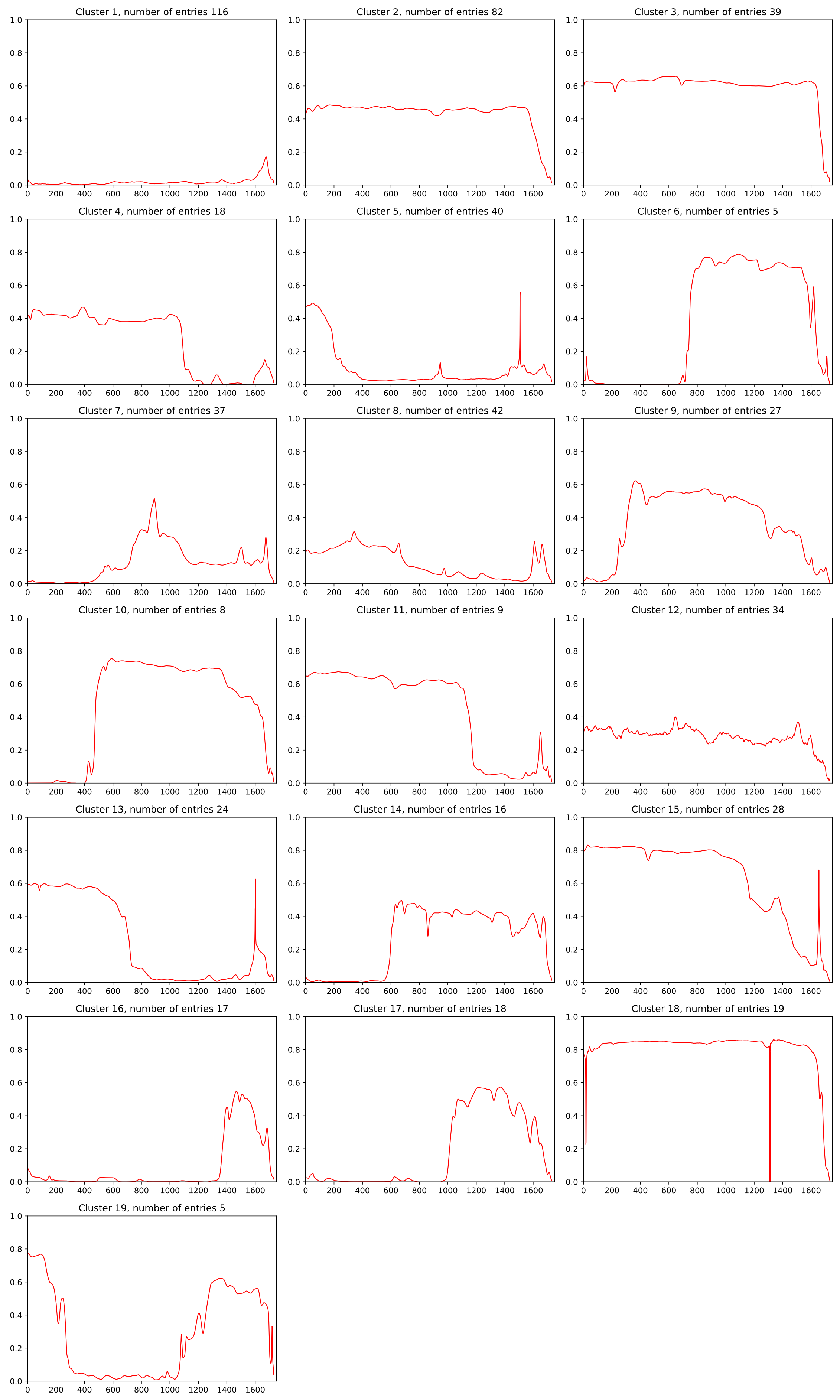


Figure A.4: 19 Cluster templates, Soft DTW method, gamma=0.2

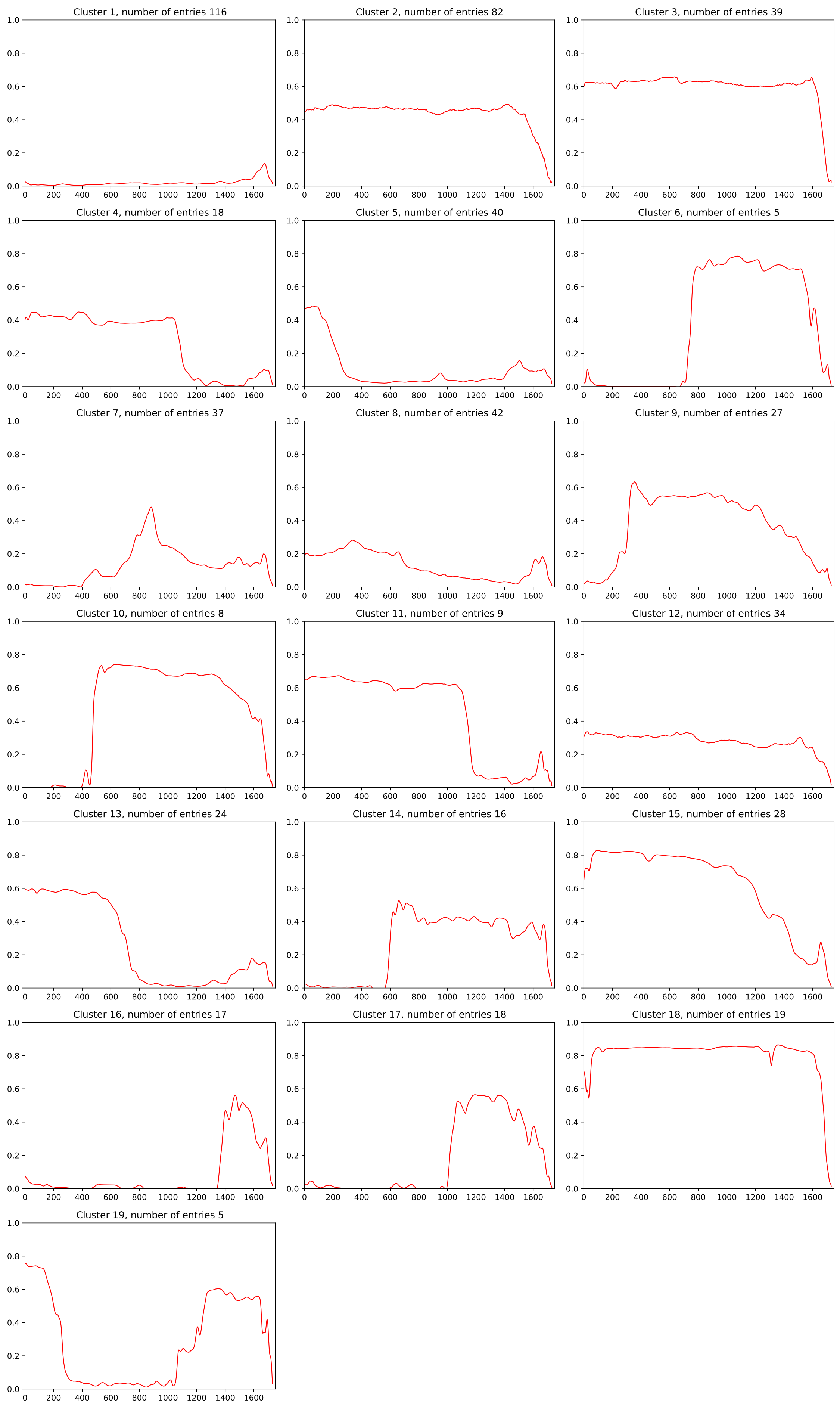


Figure A.5: 19 Cluster templates, Soft DTW method, gamma=0.5

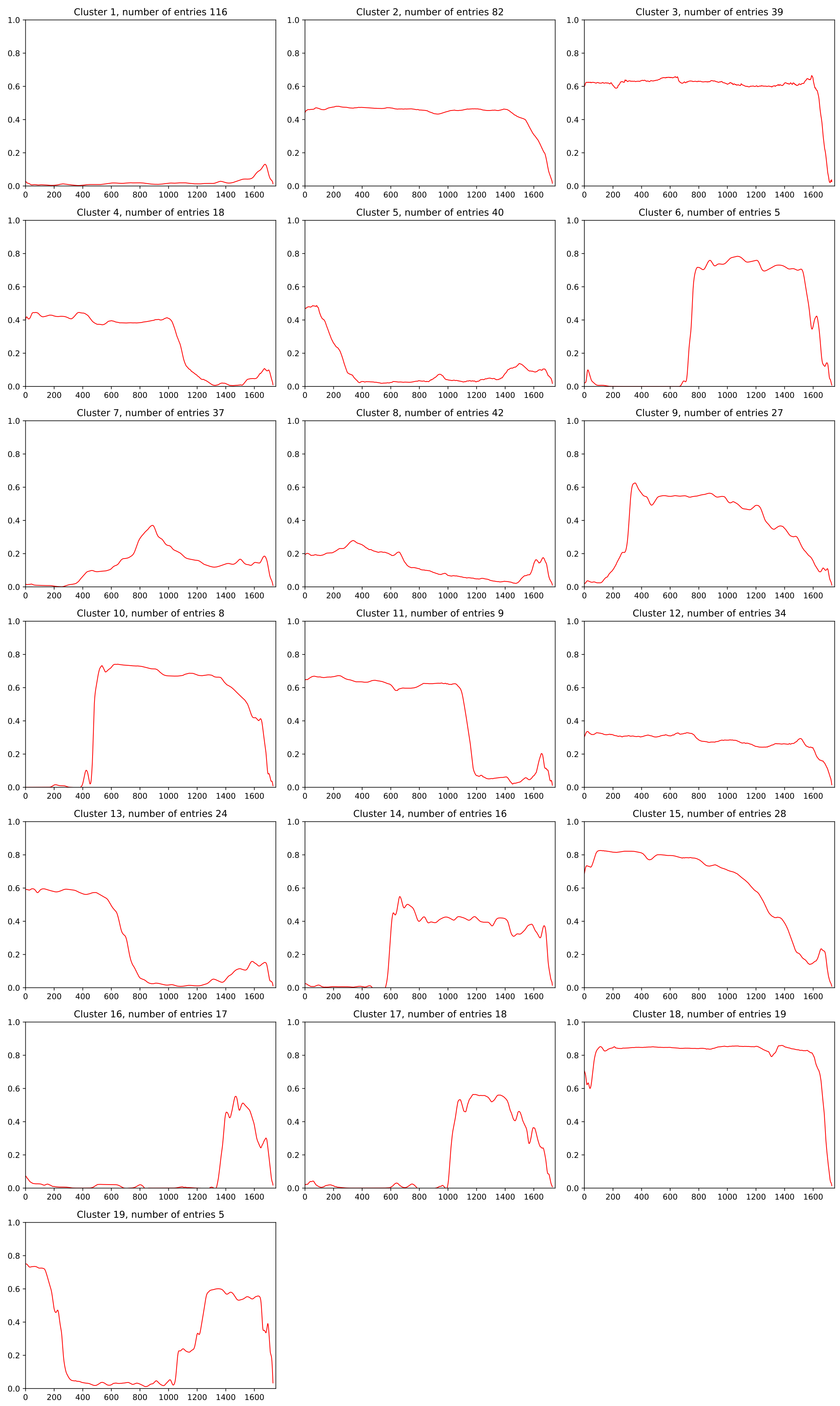


Figure A.6: 19 Cluster templates, Soft DTW method, gamma=0.7

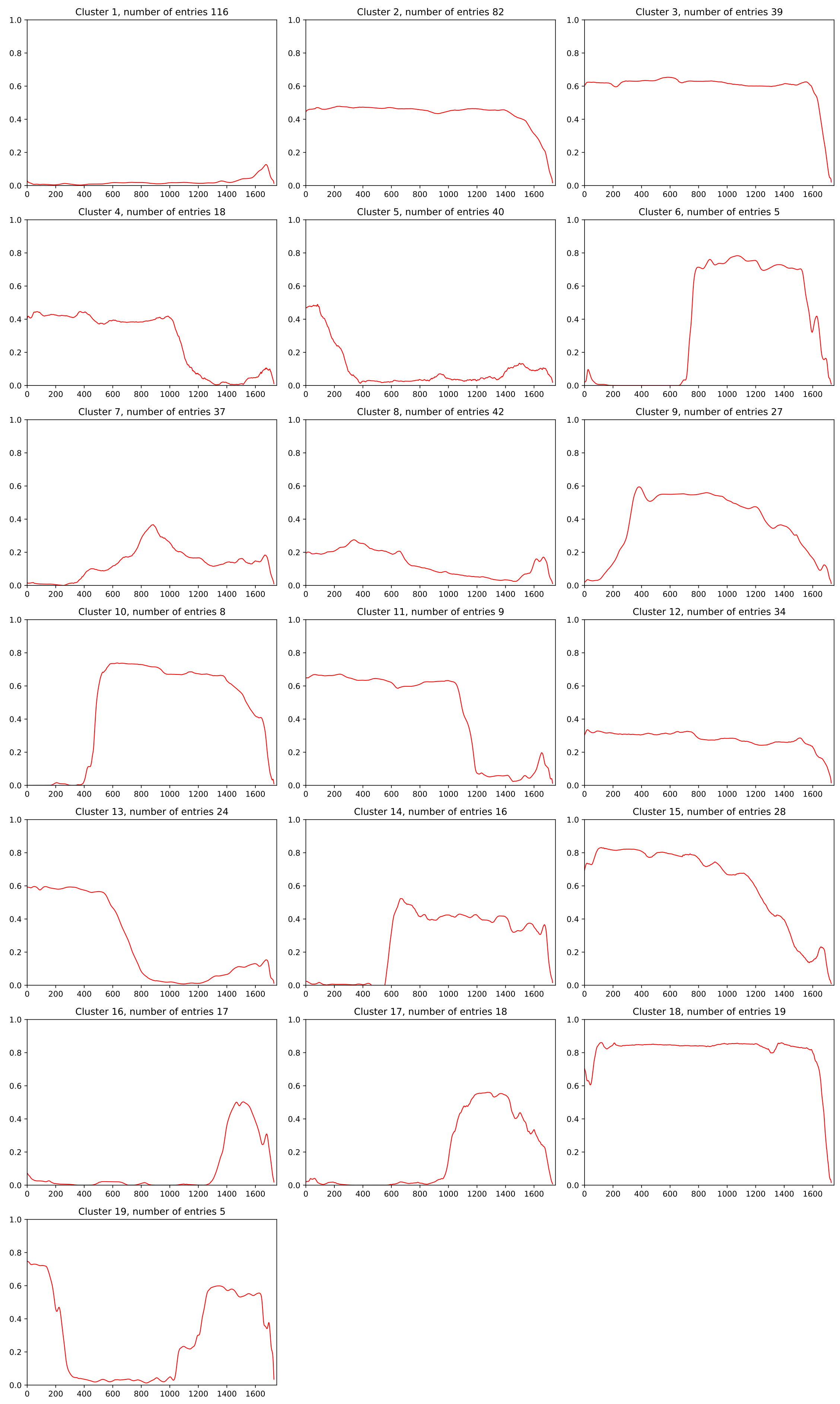


Figure A.7: 19 Cluster templates, Soft DTW method, gamma=1

A.2 25 clusters templates results

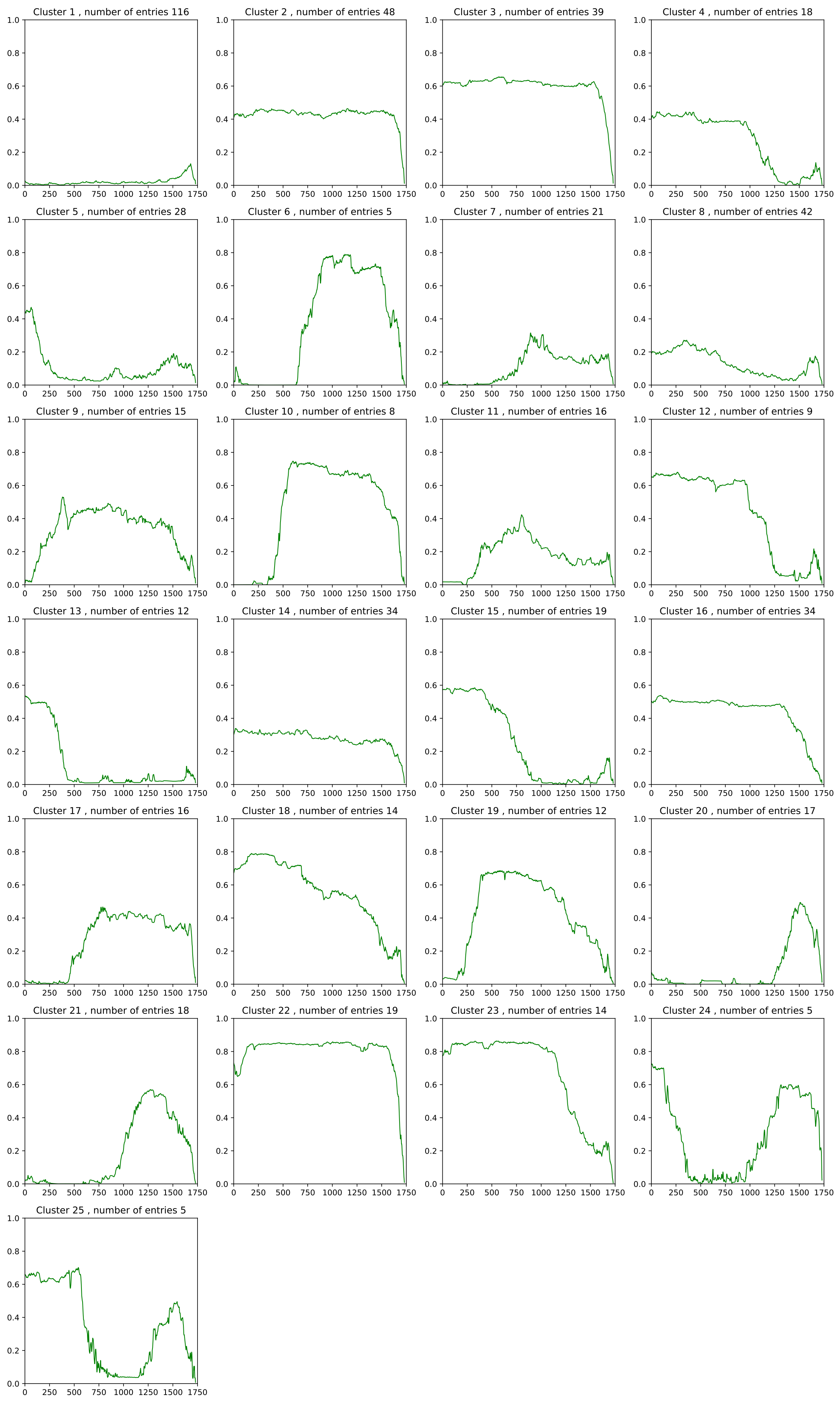


Figure A.8: 25 Cluster templates, Euclidean distance method

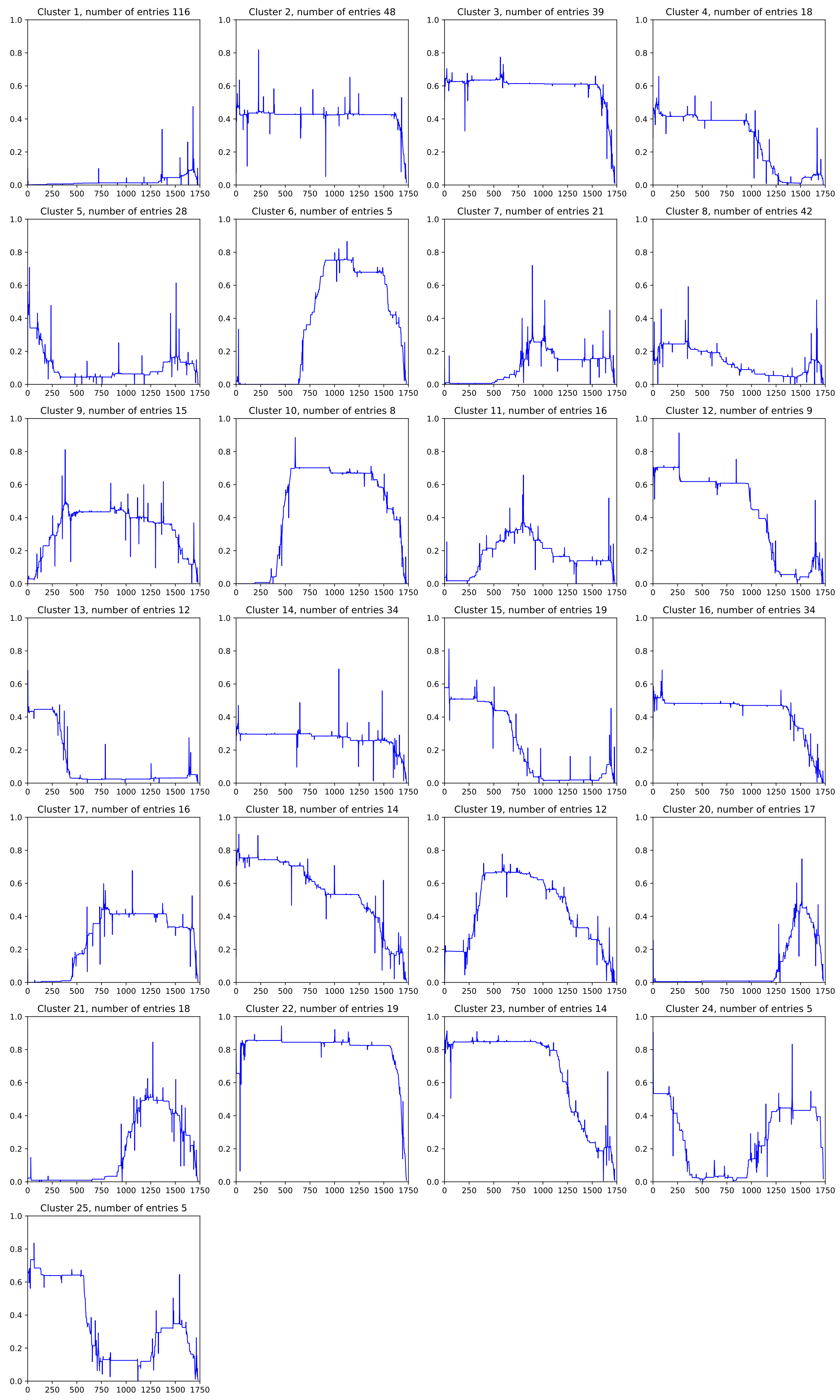


Figure A.9: 25 Cluster templates, DBA distance method

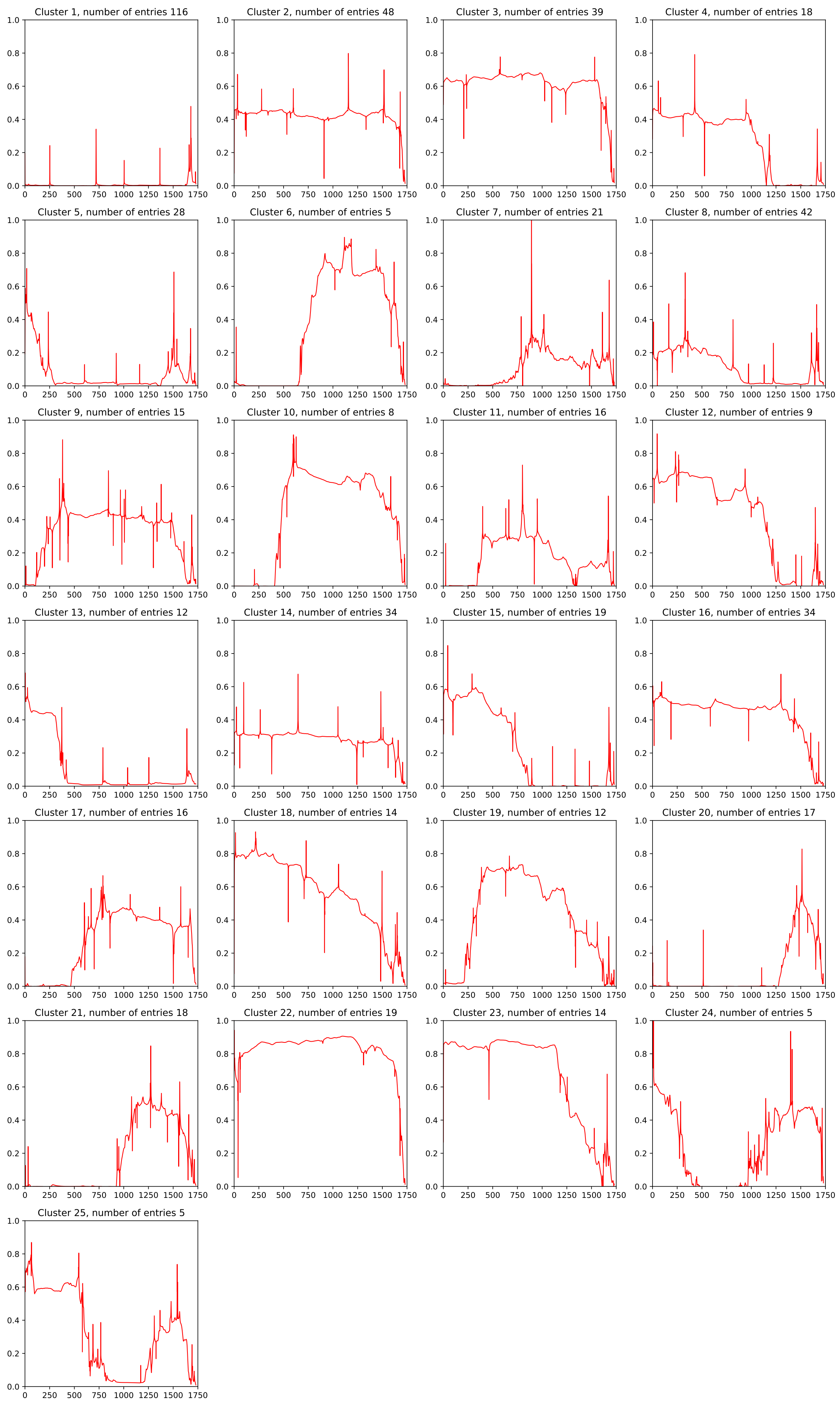


Figure A.10: 25 Cluster templates, Soft DTW method, gamma=0.01

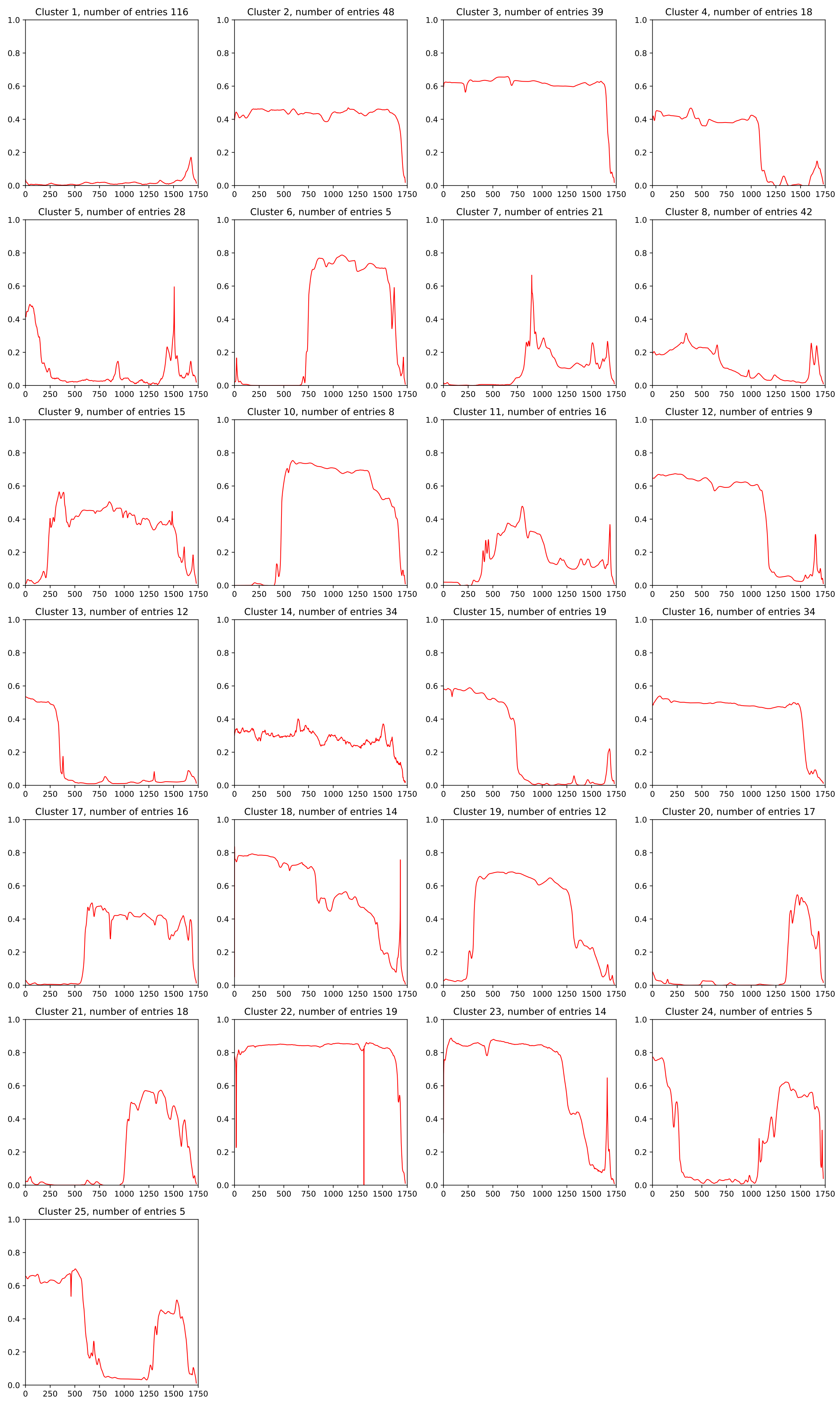


Figure A.11: 25 Cluster templates, Soft DTW method, gamma=0.2

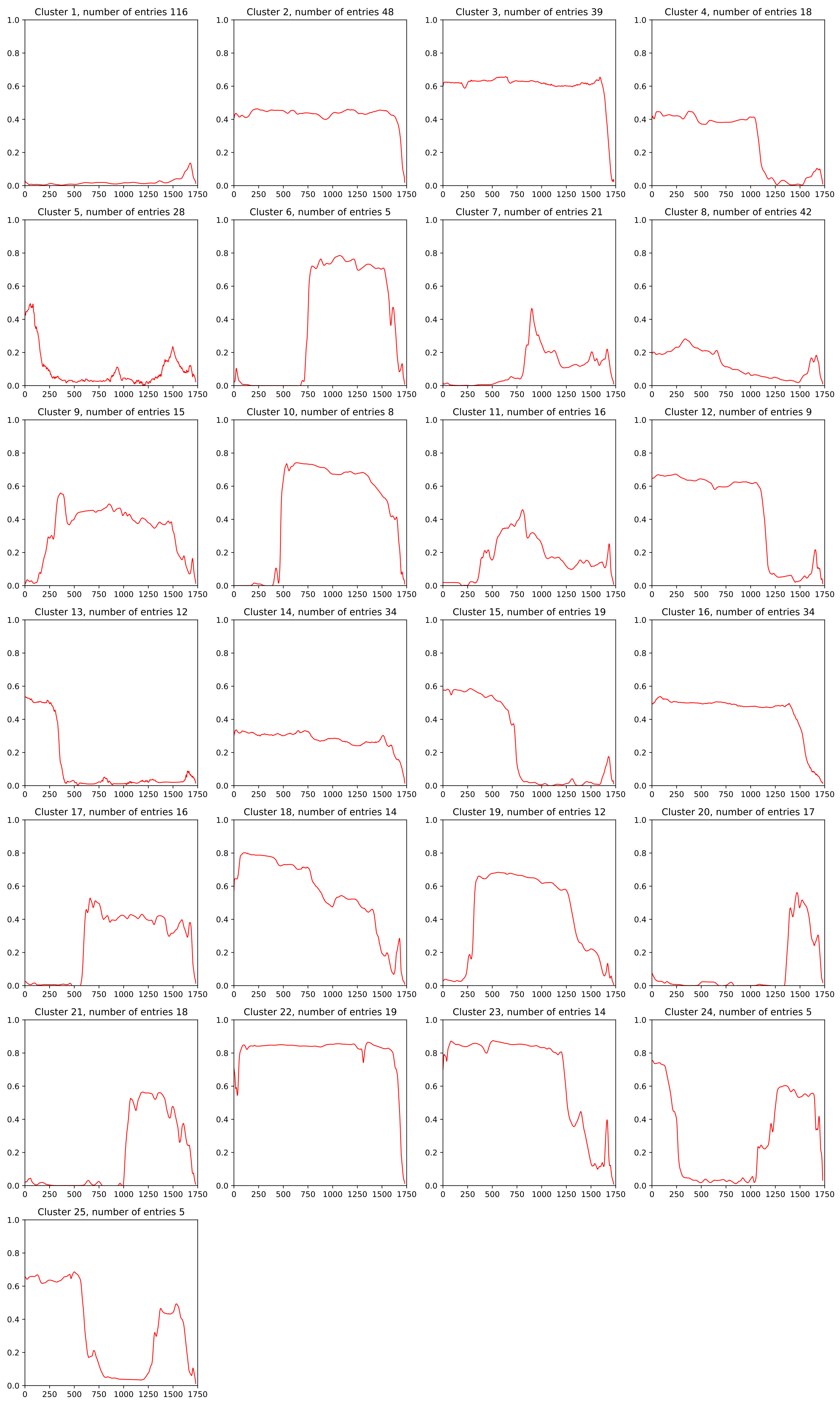


Figure A.12: 25 Cluster templates, Soft DTW method, gamma=0.5

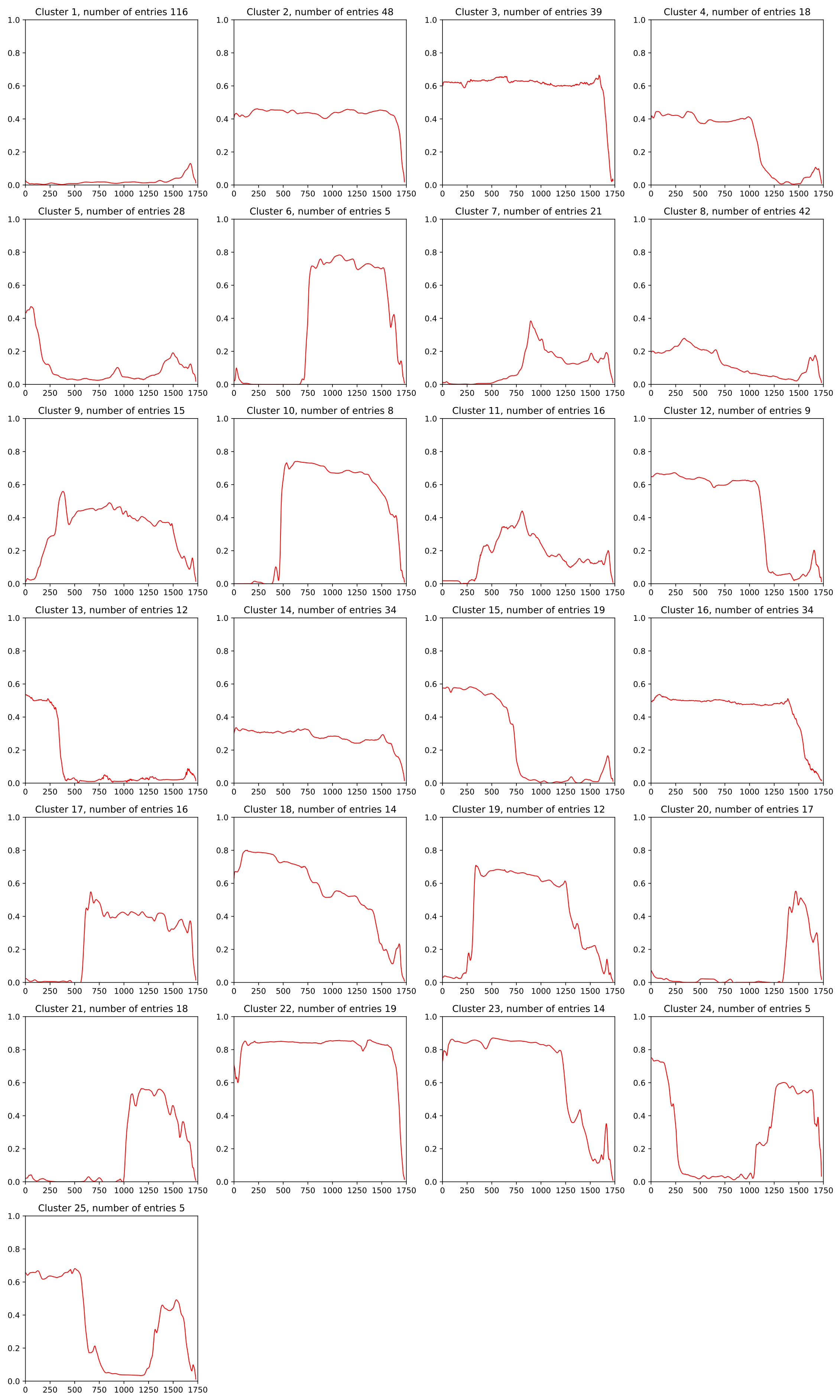


Figure A.13: 25 Cluster templates, Soft DTW method, gamma=0.7

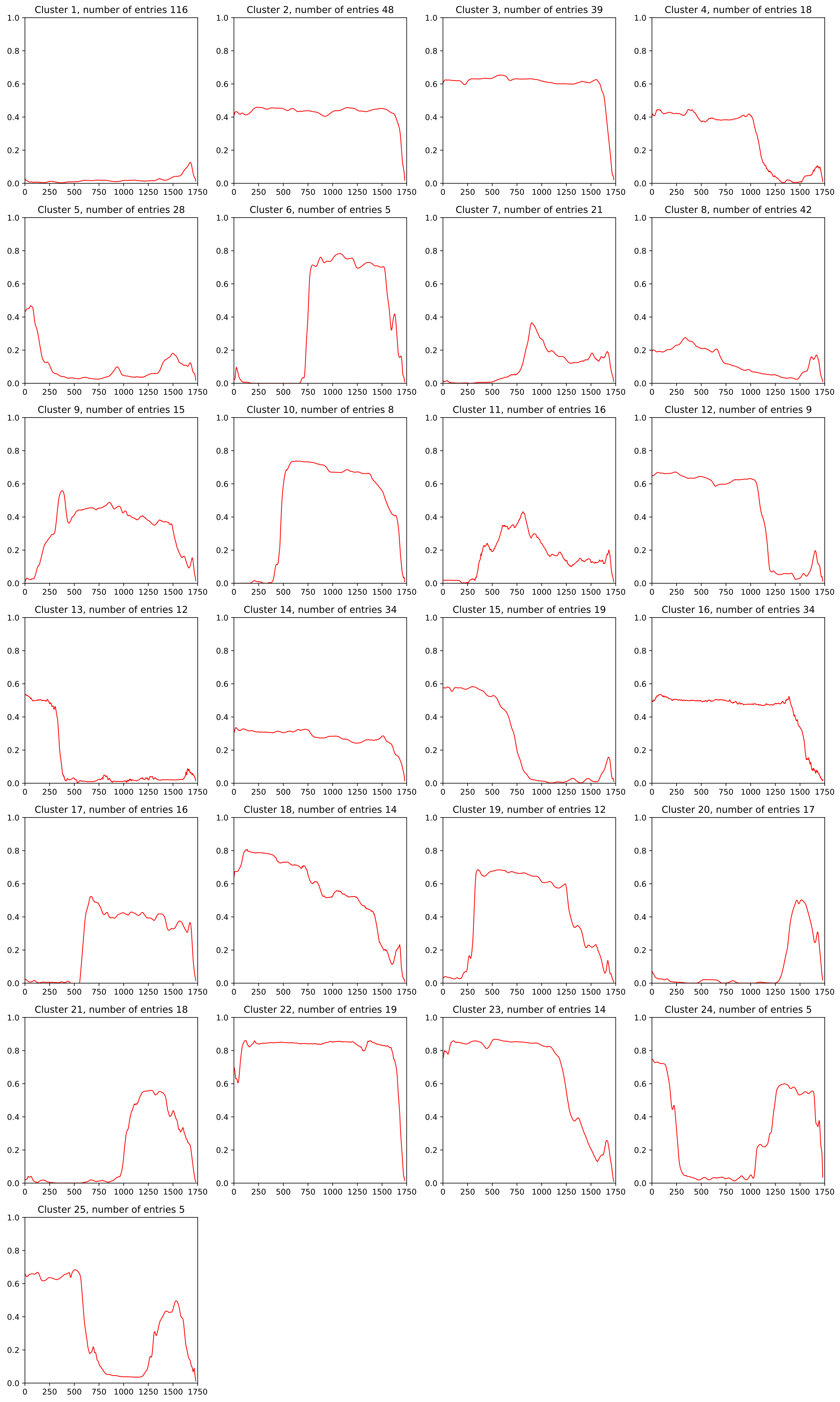


Figure A.14: 25 Cluster templates, Soft DTW method, gamma=1

A.3 Split method templates results

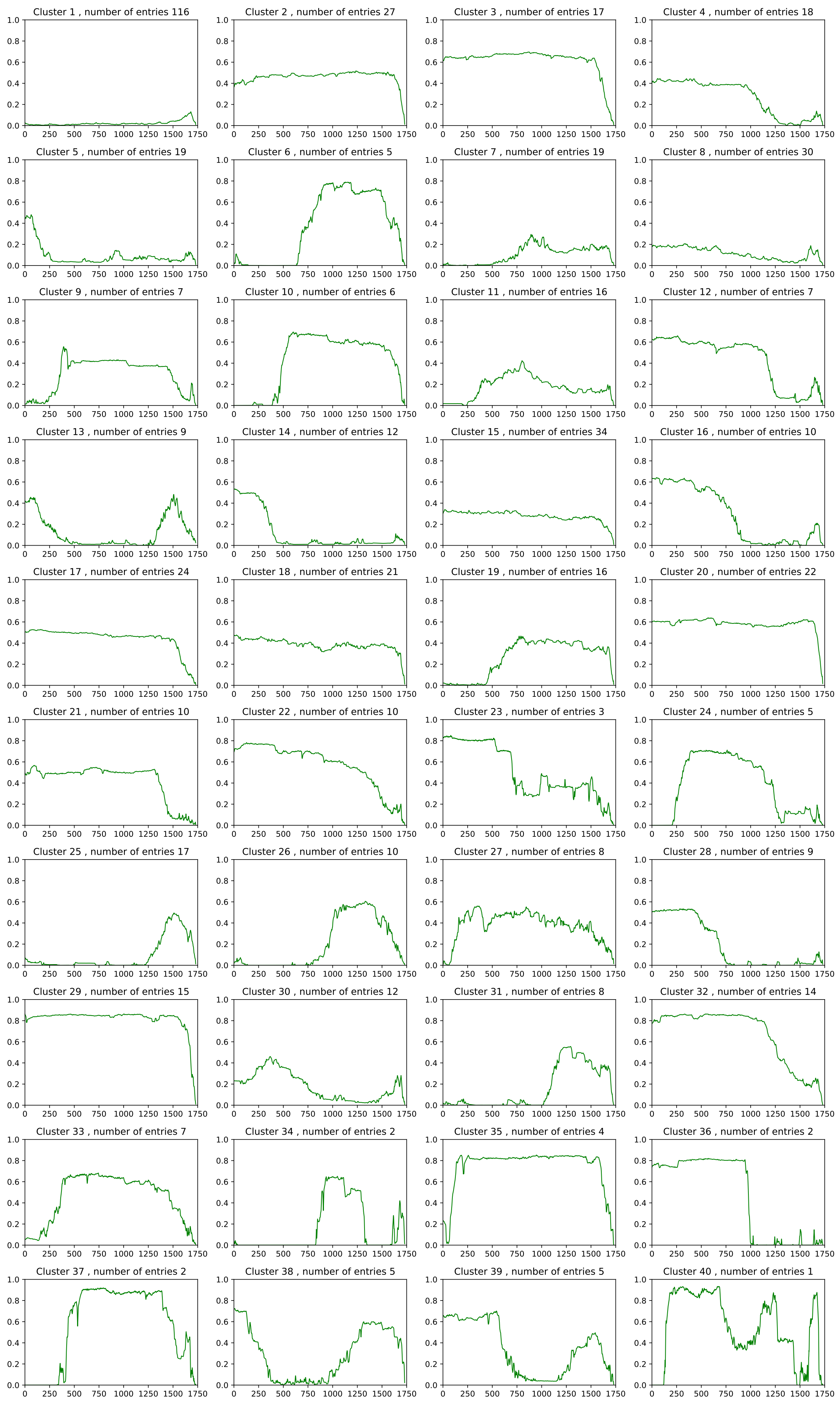


Figure A.15: 40 Clusters templates, Split method, Euclidean distance method

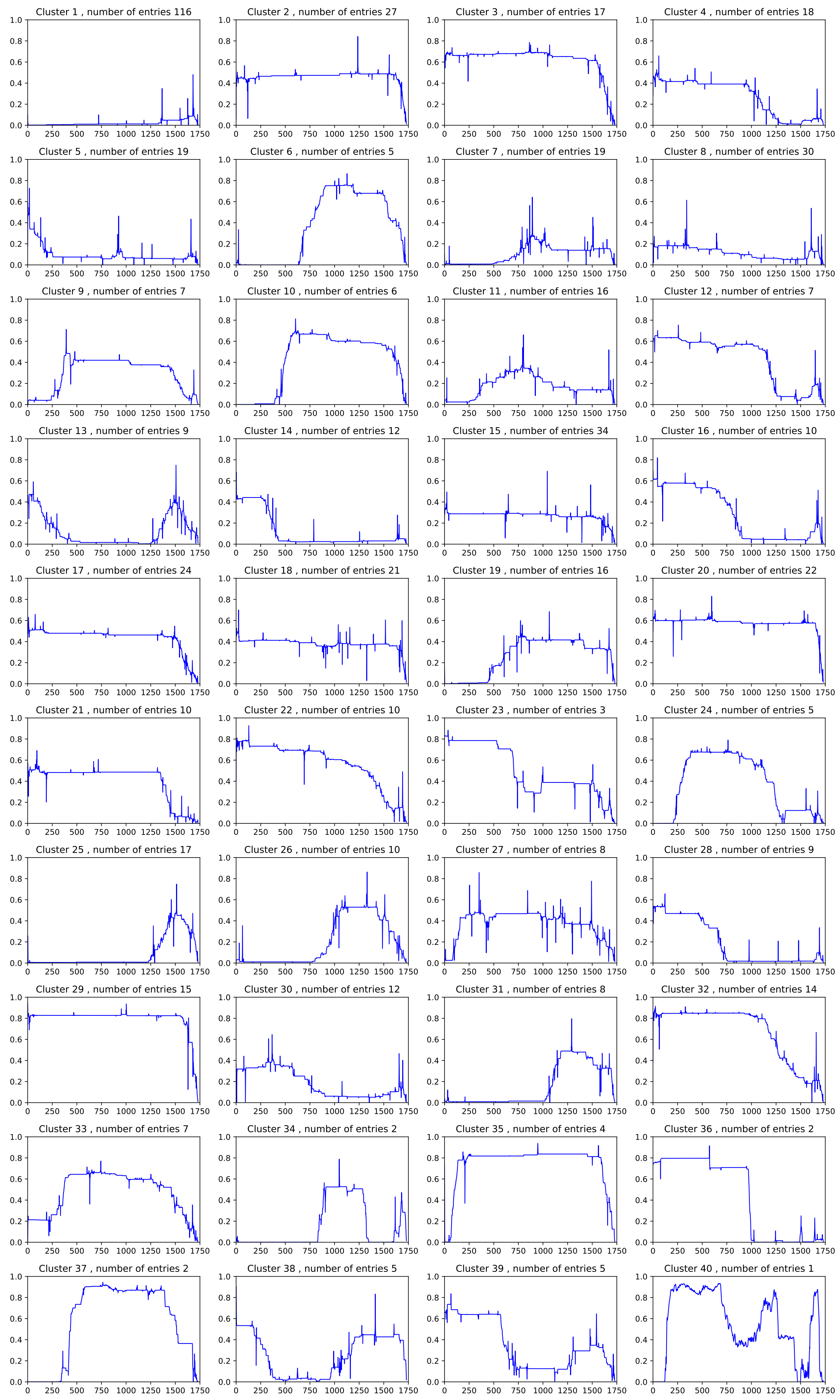


Figure A.16: 40 Cluster templates, Split method, DBA distance method

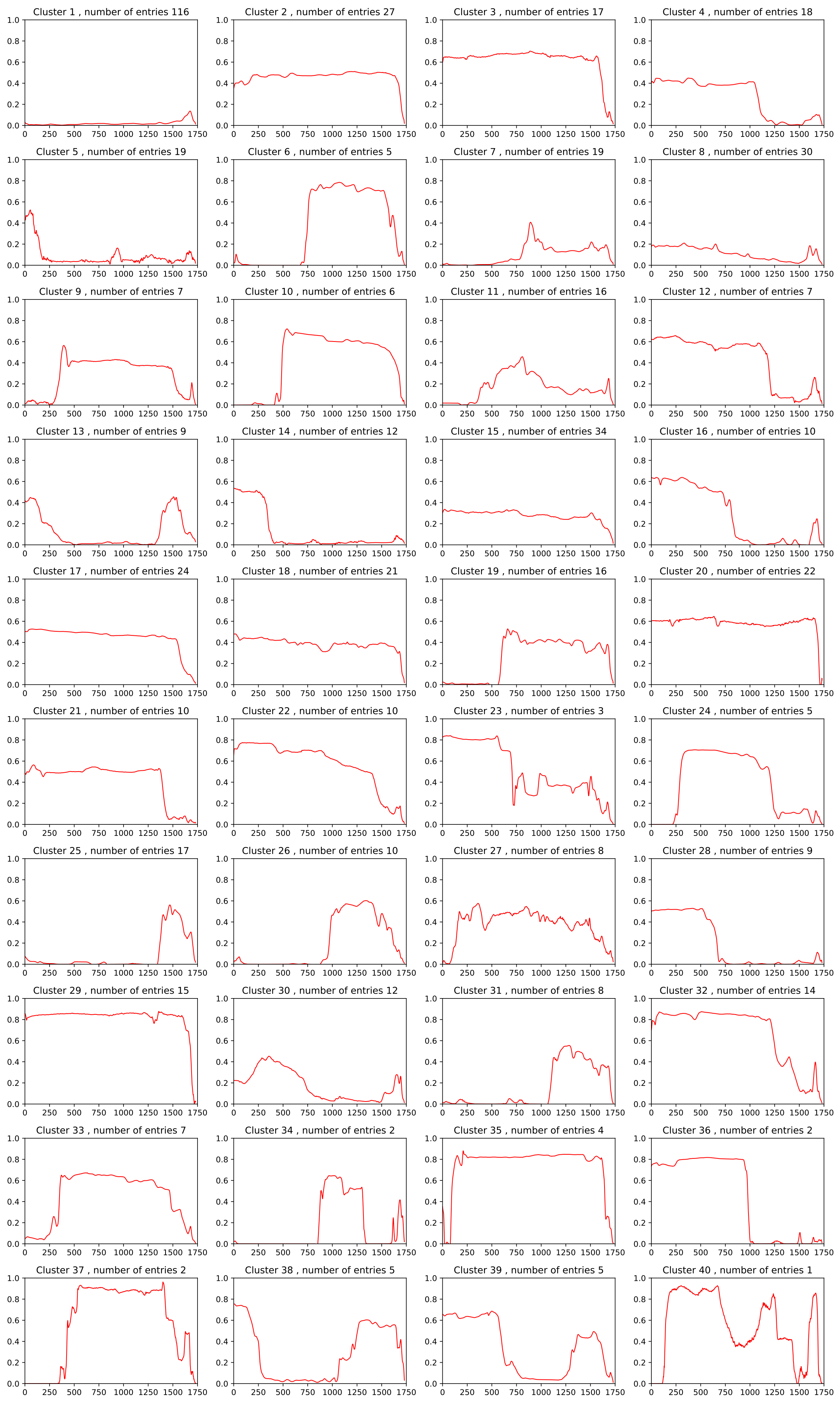


Figure A.17: 40 Cluster templates, Split method, Soft DTW method, gamma=0.5

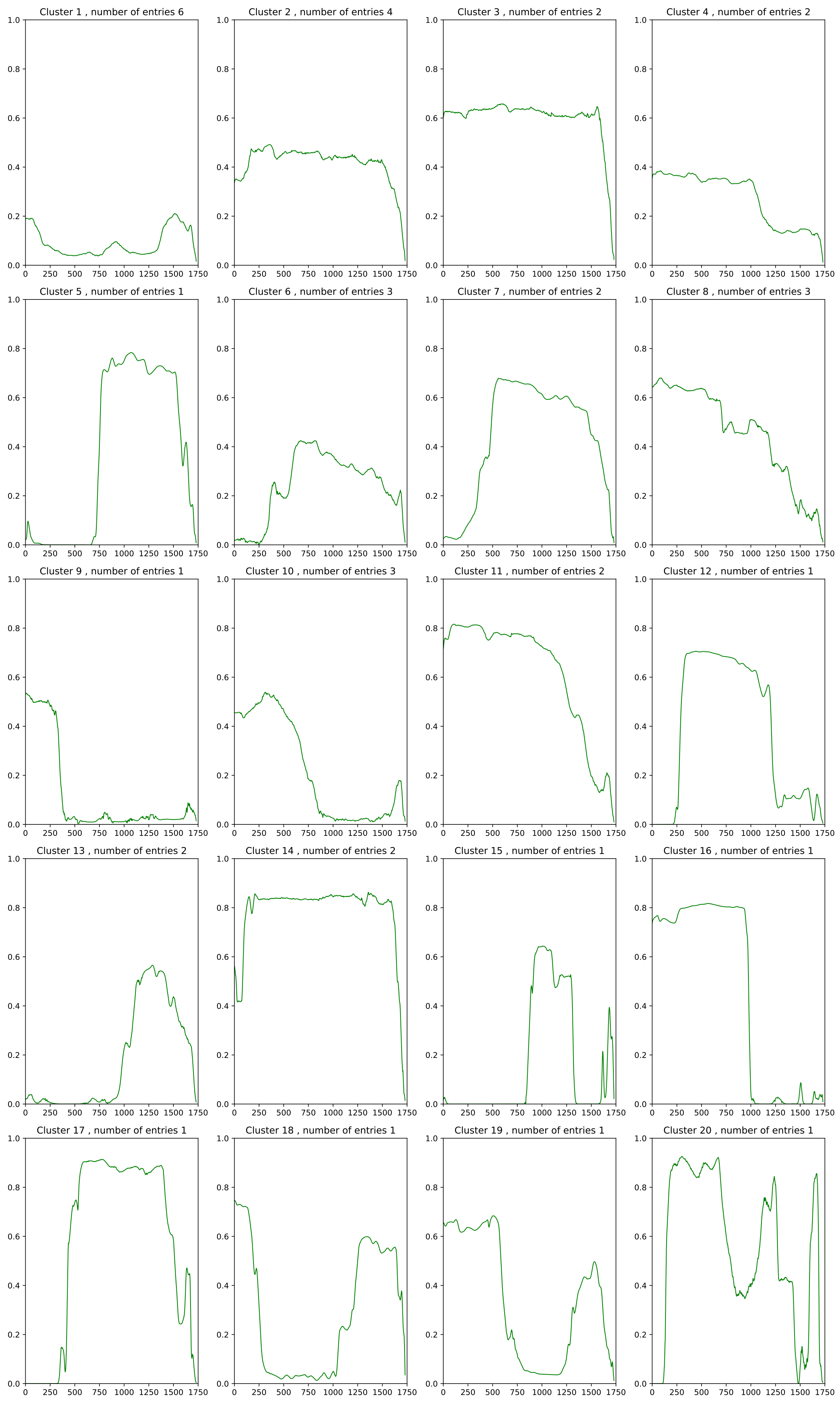


Figure A.18: 20 Clusters templates, Split method, Euclidean distance method

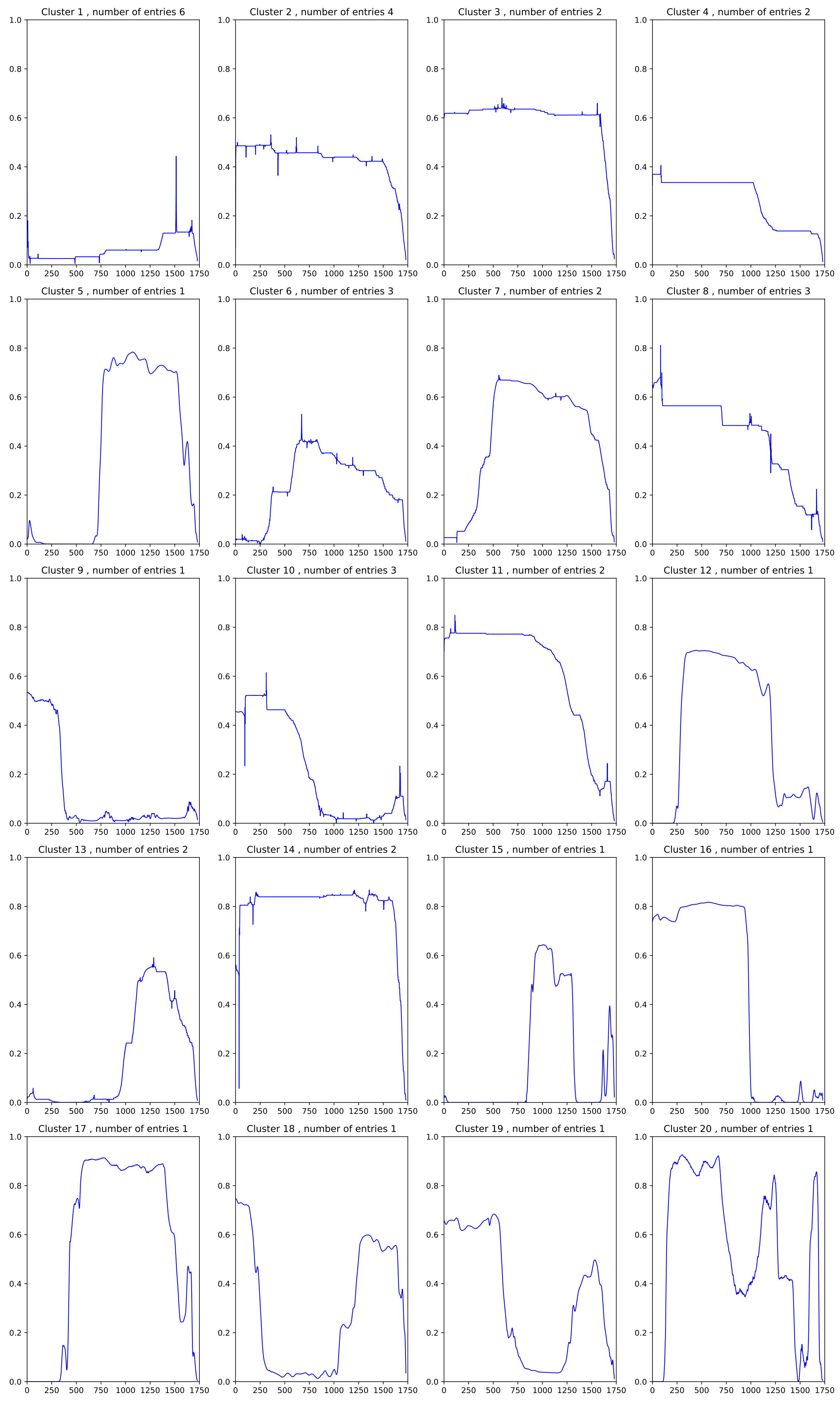


Figure A.19: 20 Cluster templates, Split method, DBA distance method

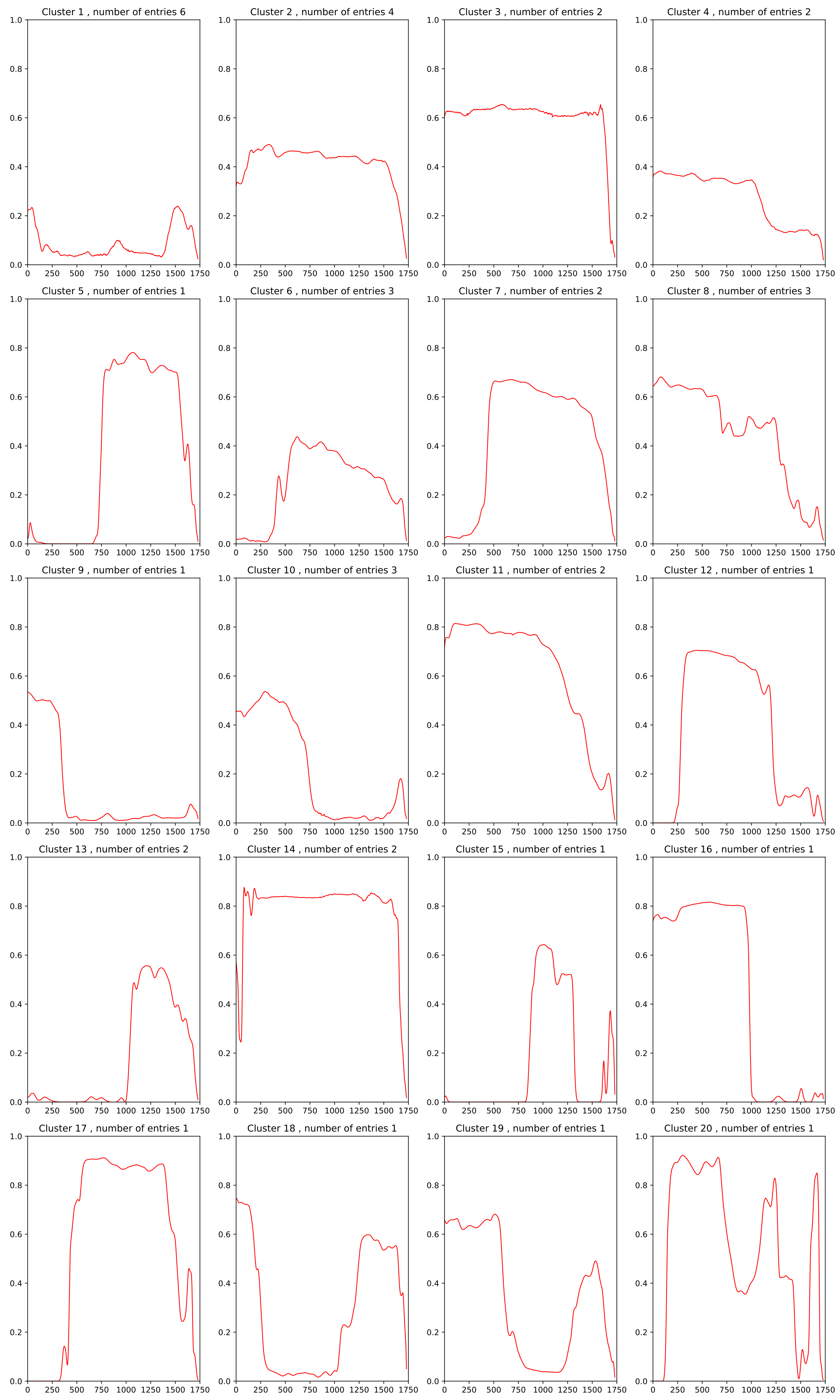


Figure A.20: 20 Cluster templates, Split method, Soft DTW method, gamma=0.5

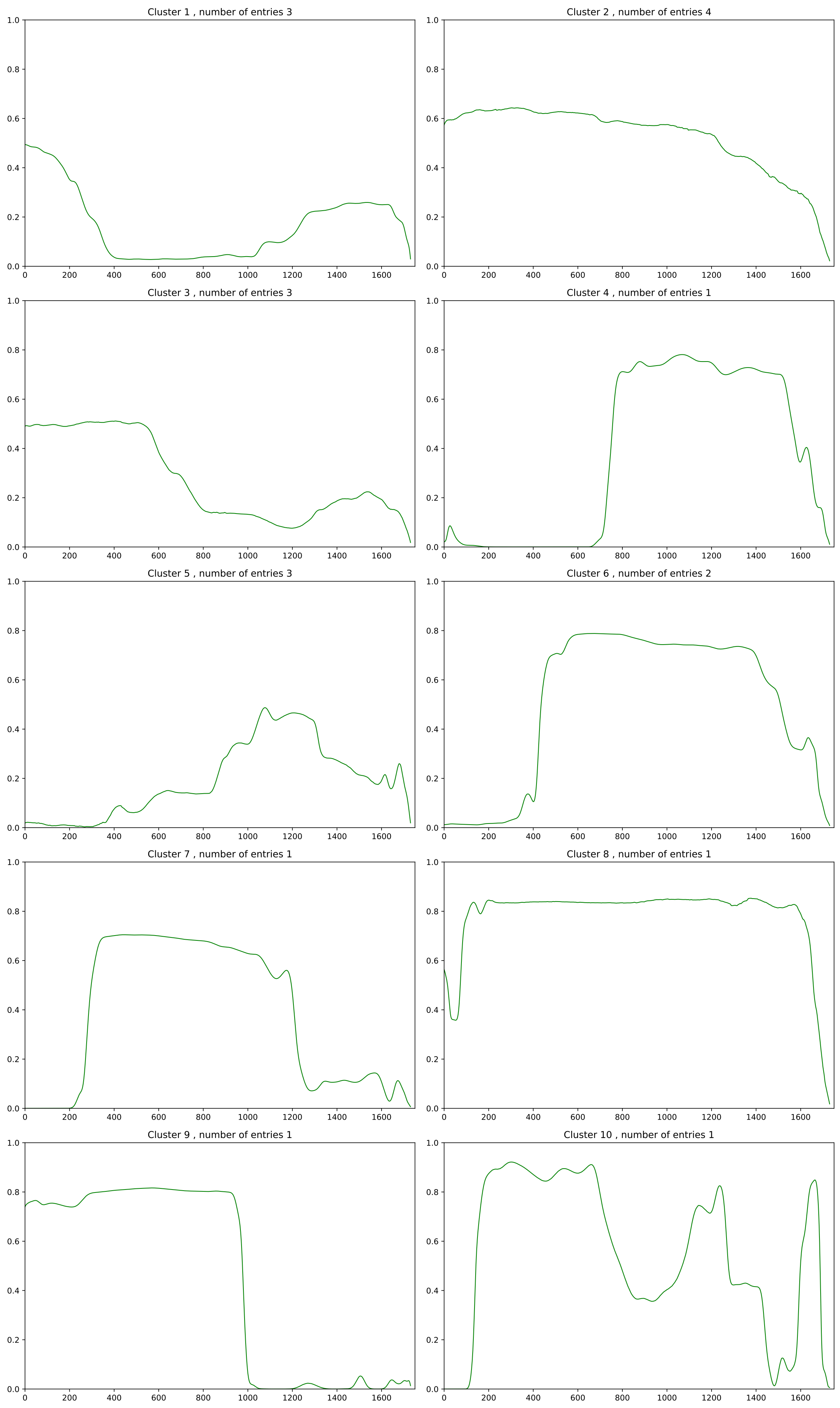


Figure A.21: 10 Clusters templates, Split method, Euclidean distance method

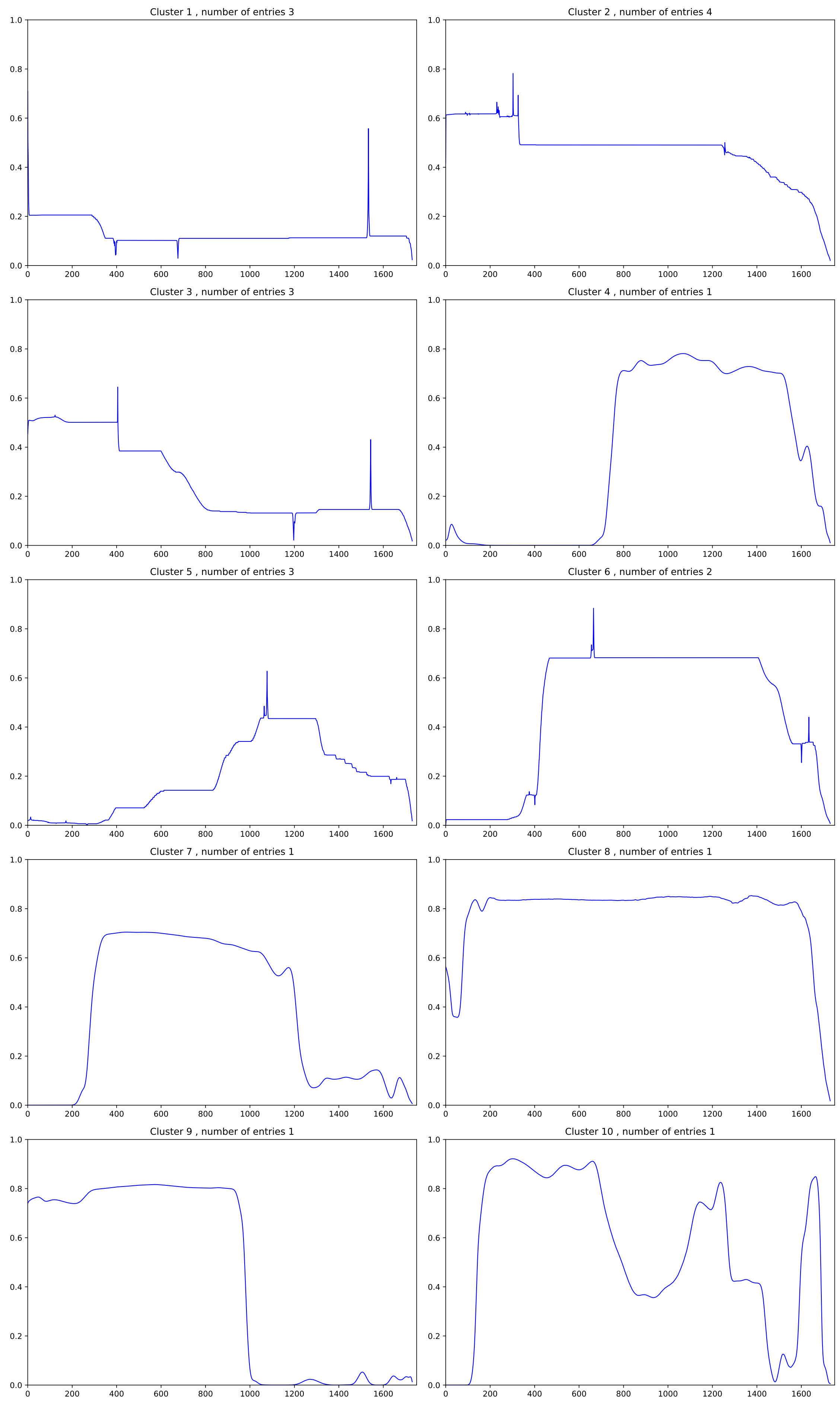


Figure A.22: 10 Cluster templates, Split method, DBA distance method

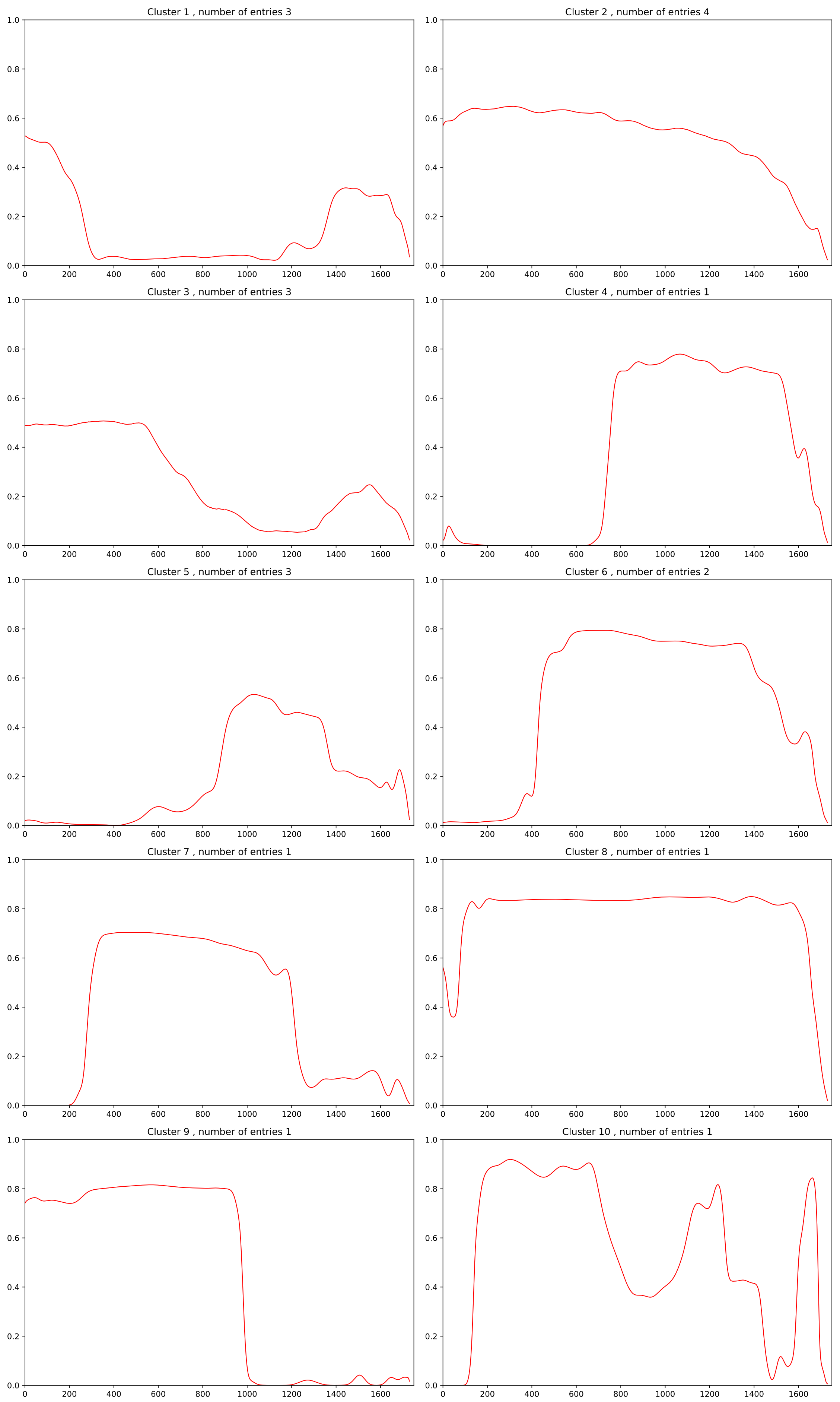


Figure A.23: 10 Cluster templates, Split method, Soft DTW method, gamma=0.5

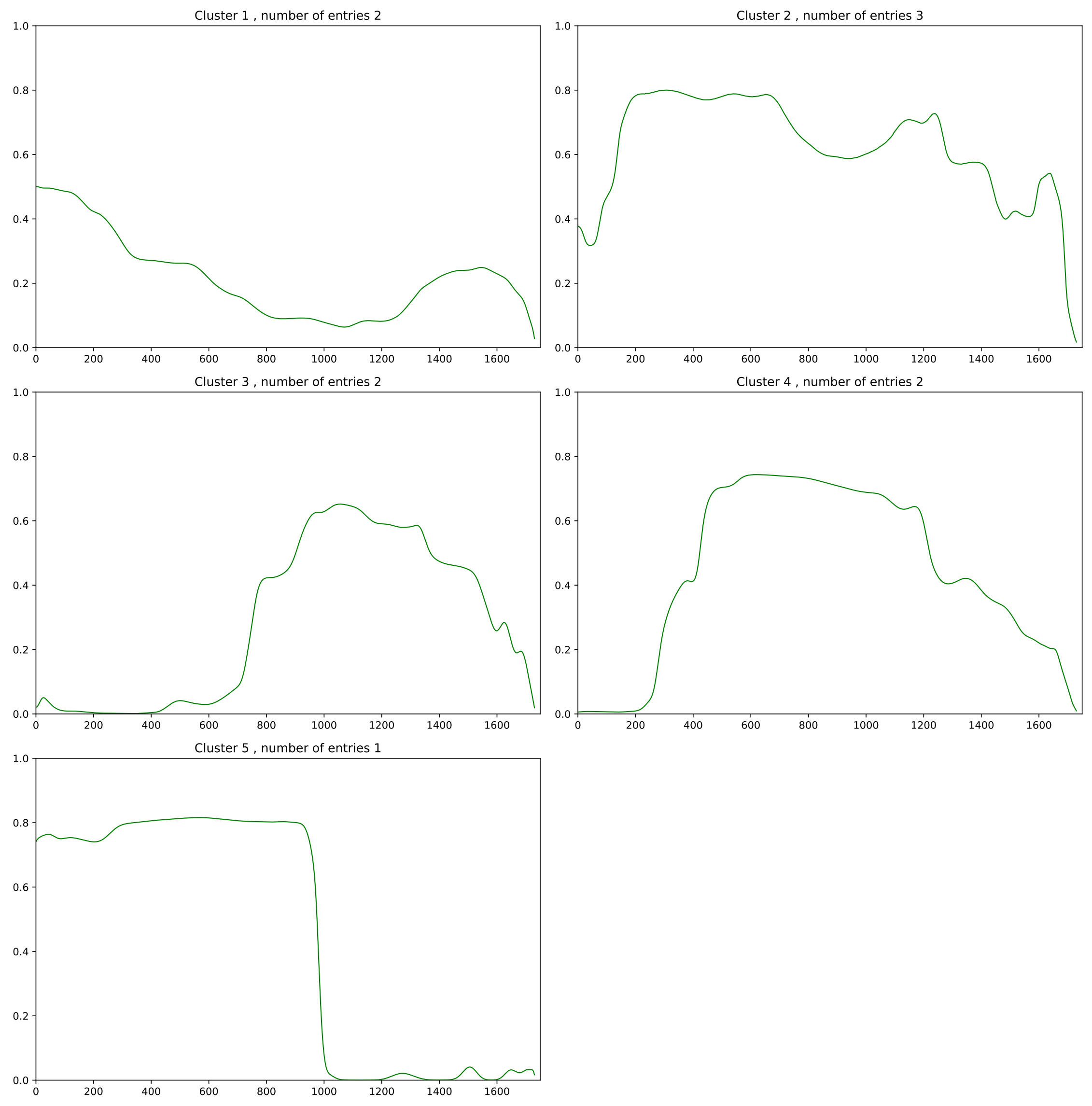


Figure A.24: 5 Clusters templates, Split method, Euclidean distance method

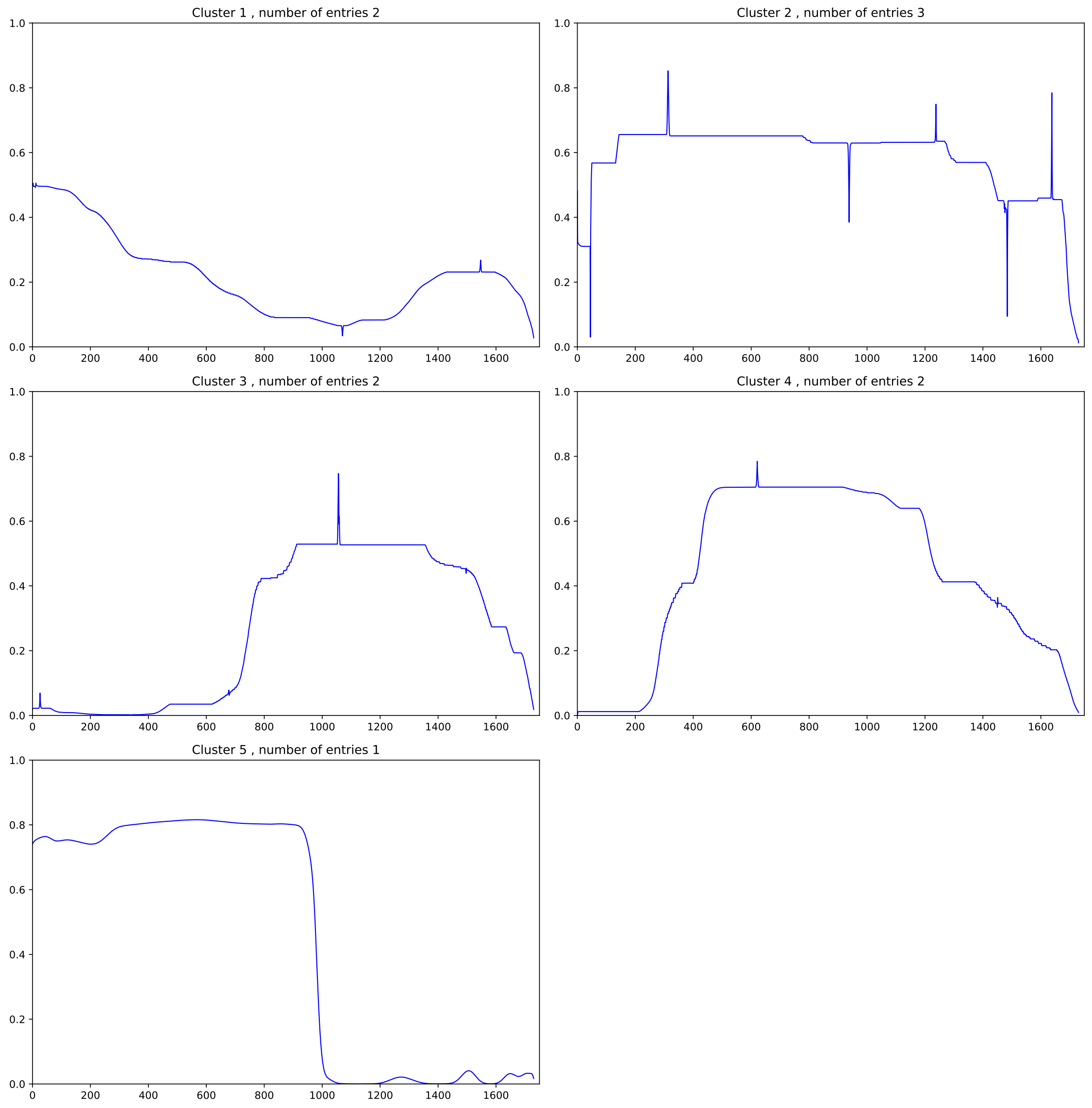


Figure A.25: 5 Cluster templates, Split method, DBA distance method

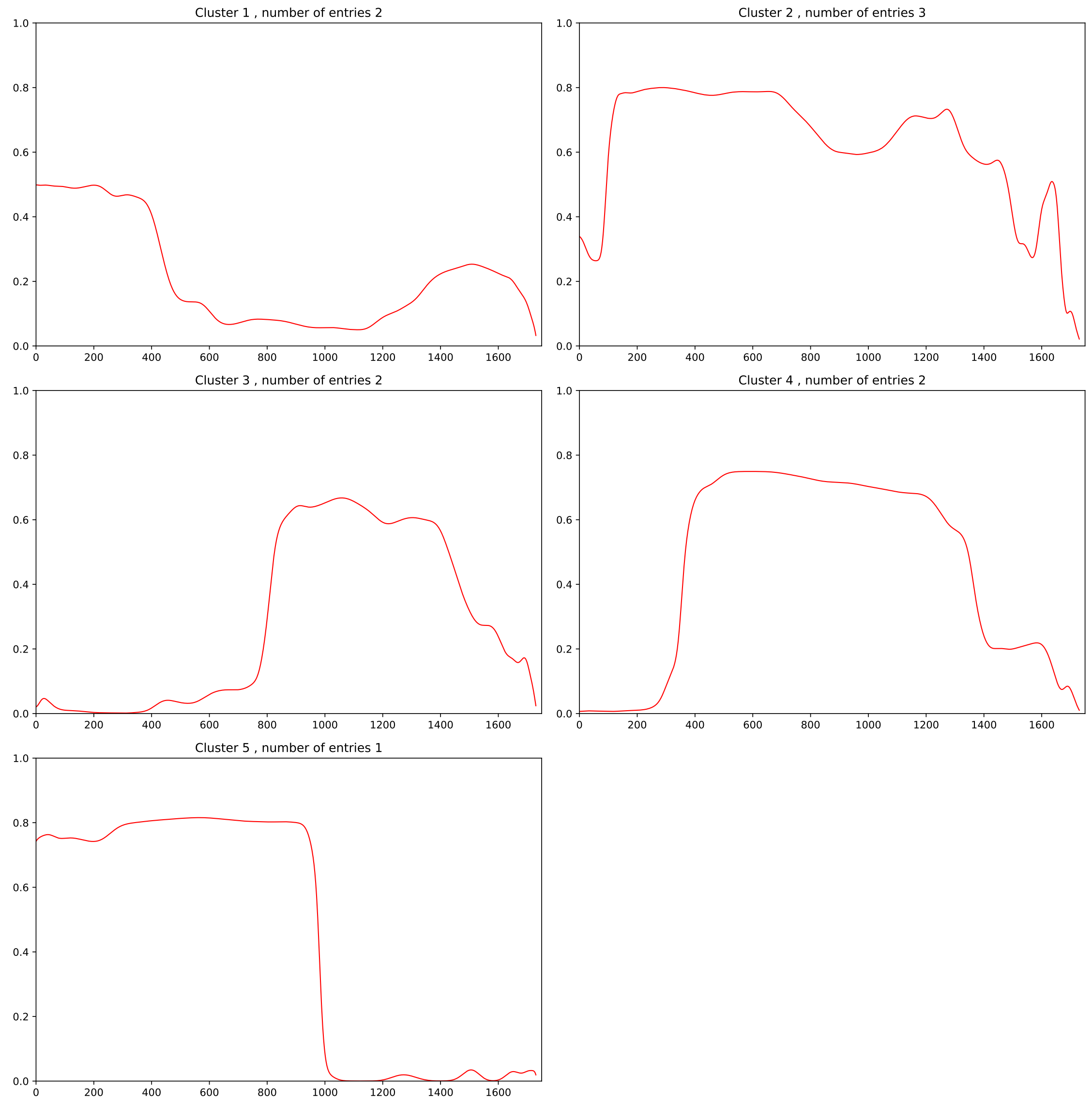


Figure A.26: 5 Cluster templates, Split method, Soft DTW method, gamma=0.5

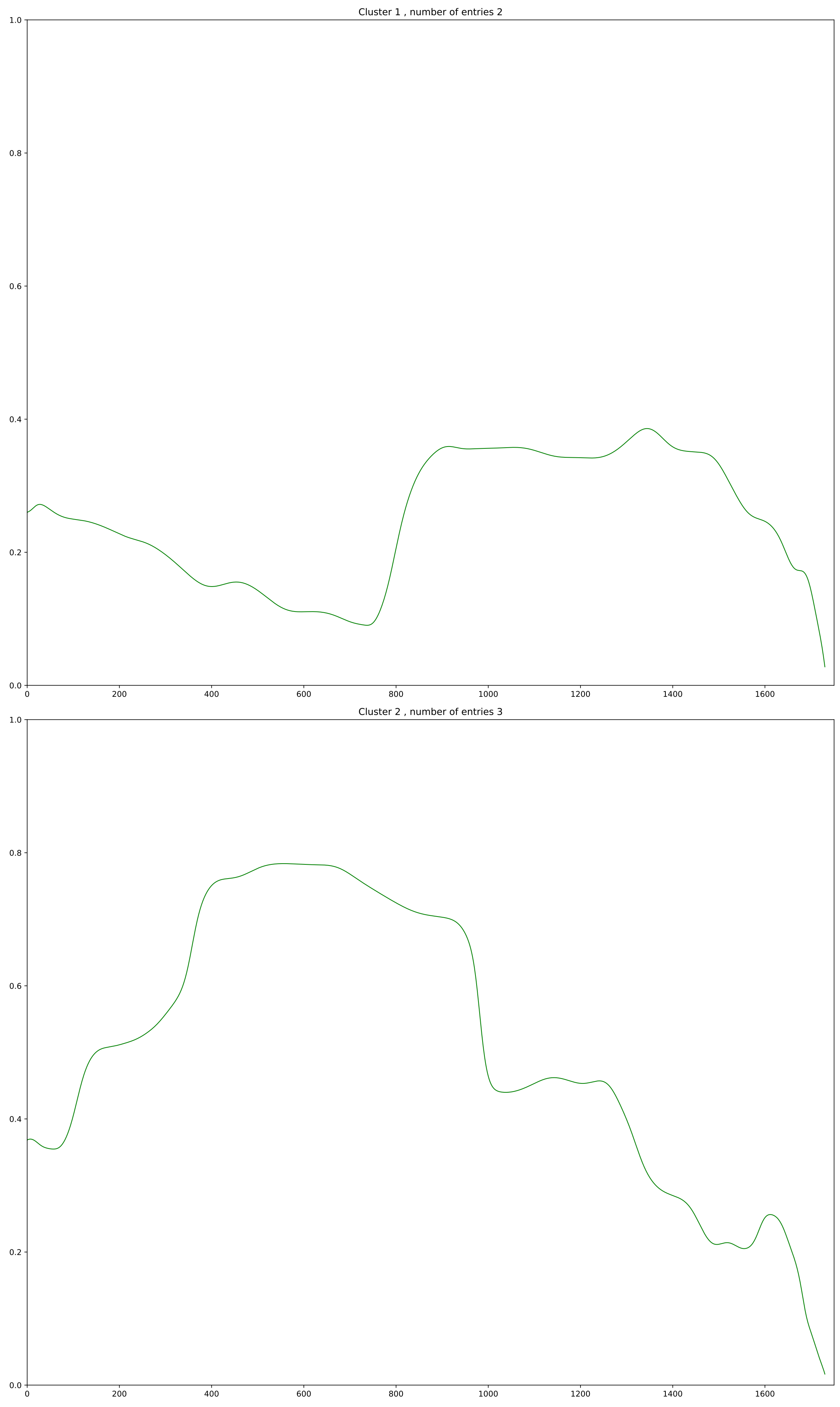


Figure A.27: 2 Clusters templates, Split method, Euclidean distance method

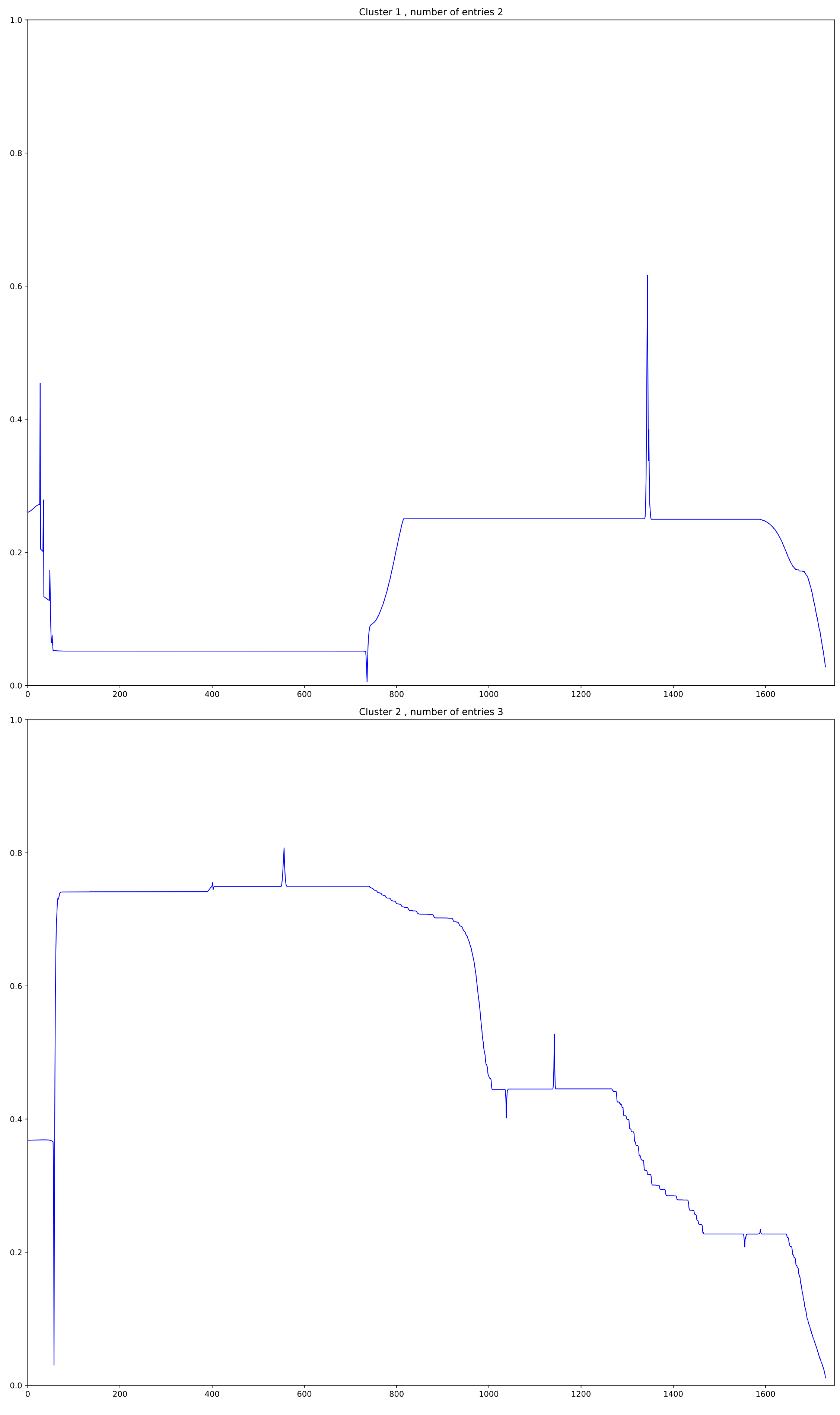


Figure A.28: 2 Cluster templates, Split method, DBA distance method

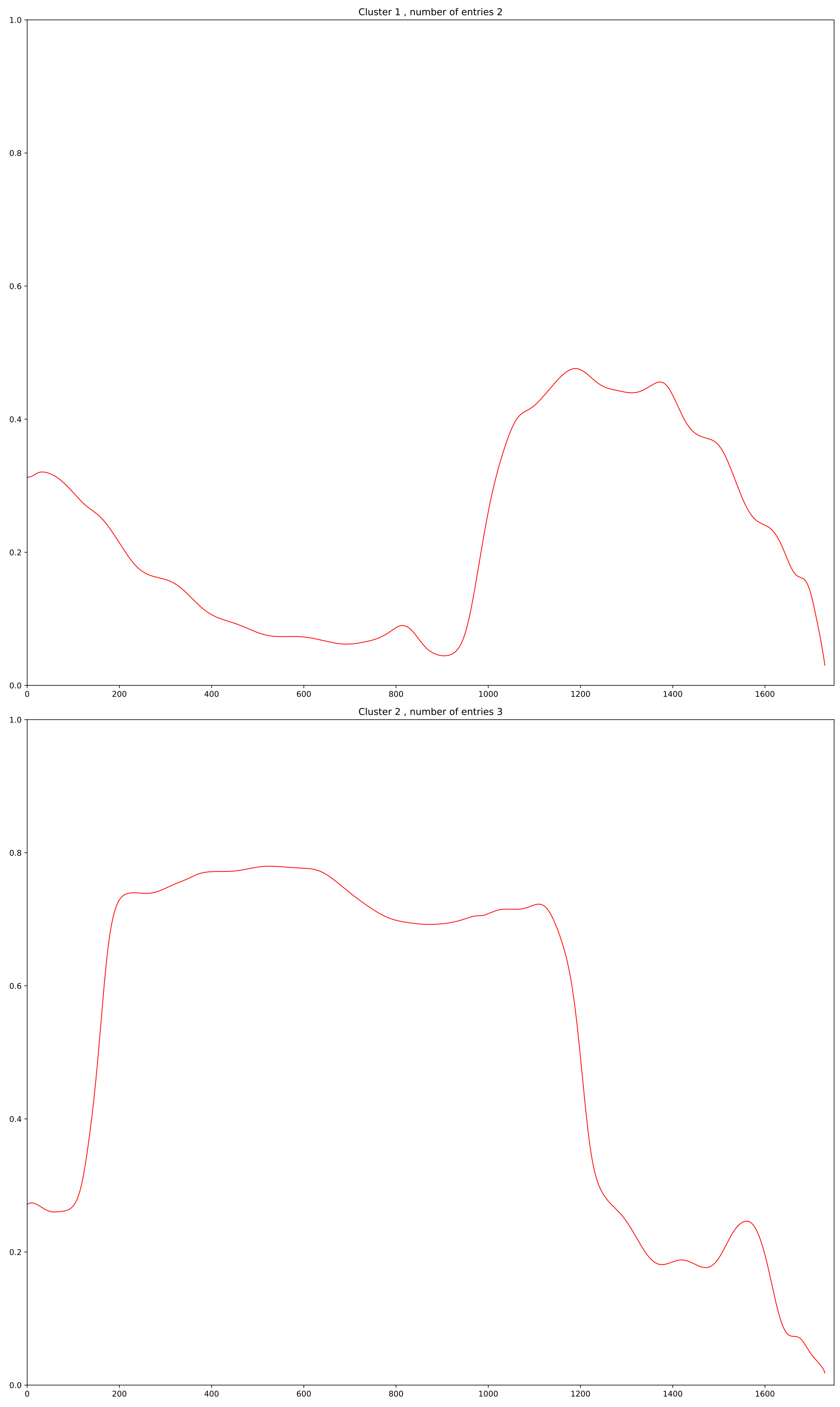


Figure A.29: 2 Cluster templates, Split method, Soft DTW method, gamma=0.5

Bibliography

[1] E. Torp and P. Önnegren. Driving cycle generation using statistical analysis and markov chains. Master's thesis, Linköping University, SE-581 83 Linköping, 2013.

[2] Peter Nyberg. Evaluation, Generation, and Transformation of Driving Cycles. Linköping Studies in Science and Technology Dissertations, No. 1669. 2015.

[3] B. Liaw and M. Dubarry. From driving cycle analysis to understanding battery performance in real-life electric hybrid vehicle operation. Journal of Power Sources, 174(1):76–88, 2007.

[4] M. André and M. Rapone. Analysis and modelling of the pollutant emis- sions from European cars regarding the driving characteristics and test cycles. Atmospheric Environment, 43(5):986–995, 2009.

[5] E. Ericsson. Variability in urban driving patterns. Transportation Research Part D: Transport and Environment, 5(5):337–354, 2000.

[6] M. André. The ARTEMIS European driving cycles for measuring car pollutant emissions. Science of The Total Environment, 334-335(0):73–84, 2004.

[7] M. André, R. Joumard, R. Vidon, P. Tassel, and P. Perret. Real-world European driving cycles, for measuring pollutant emissions from high- and low-powered cars. Atmospheric Environment, 40(31):5944–5953, 2006. 13th International Symposium on Transport and Air Pollution (TAP-2004).

[8] E. Tazelaar, J. Bruinsma, B. Veenhuizen, and P. van den Bosch. Driving cycle characterization and generation, for design and control of fuel cell buses. World Electric Vehicle Journal, 3(1), 2009.

[9] V. Schwarzer and R. Ghorbani. Drive cycle generation for design optimization of electric vehicles. IEEE Transactions on Vehicular Technology, 62(1):89–97, 2013.

[10] Q. Wang, H. Huo, K. He, Z. Yao, and Q. Zhang. Characterization of vehicle driv- ing patterns and development of driving cycles in Chinese cities. Transportation Research Part D: Transport and Environment, 13(5):289–297, 2008.

[11] G. Fontaras, G. Karavalakis, M. Kousoulidou, T. Tzamkiozis, L. Ntziachristos, E. Bakeas, S. Stournas, and Z. Samaras. Effects of biodiesel on passenger car fuel consumption, regulated and non-regulated pollutant emissions over legislated and real-world driving cycles. Fuel, 88(9):1608–1617, 2009.

[12] J.-P. Gao, G.-M. G. Zhu, E. G. Strangas, and F.-C. Sun. Equivalent fuel consumption optimal control of a series hybrid electric vehicle. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 223(8):1003–1018, 2009.

[13] S. Stockar, V. Marano, M. Canova, G. Rizzoni, and L. Guzzella. Energy-optimal control of plug-in hybrid electric vehicles for real-world driving cycles. IEEE Transactions on Vehicular Technology, 60(7):2949–2962, 2011.

[14] A. Jaafar, B. Sareni, and X. Roboam. A systemic approach integrating driving cycles for the design of hybrid locomotives. IEEE Transactions on Vehicular Technology, 62(8):3541–3550, 2013.

[15] X. Hu, N. Murgovski, L. Johannesson, and B. Egardt. Comparison of three electrochemical energy buffers applied to a hybrid bus powertrain with simultaneous optimal sizing and energy management. IEEE Transactions on Intelligent Transportation Systems, 15(3):1193–1205, 2014.

[16] J.-M. Zaccardi and F. Le Berr. Analysis and choice of representative drive cycles for light duty vehicles - Case study for electric vehicles. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 2012.

[17] G. Souffran, L. Miegeville, and P. Guerin. Simulation of real-world vehicle missions using a stochastic markov model for optimal powertrain sizing. IEEE Transactions on Vehicular Technology, 61(8):3454–3465, 2012.

[18] H. Tong, H. Tung, W. Hung, and H. Nguyen. Development of driving cycles for motorcycles and light-duty vehicles in Vietnam. Atmospheric Environment, 45(29):5191–5199, 2011.

[19] T.-K. Lee and Z. Filipi. Synthesis of real-world driving cycles using stochastic process and statistical methodology. International Journal of Vehicle Design, 57(1):17–36, 2011.

[20] Q. Gong, S. Midlam-Mohler, V. Marano, and G. Rizzoni. An iterative markov chain approach for generating vehicle driving cycles. SAE International Journal of Engines, 4(1):1035–1045, 2011.

[21] J. Lin and D. Niemeier. An exploratory analysis comparing a stochastic driving cycle to California's regulatory cycle. Atmospheric Environment, 36 (38):5759–5770, 2002.

[22] S. Kamble, T. Mathew, and G. Sharma. Development of real-world driving cycle: Case study of Pune, India. Trans. Research Part D, 14(2):132–140, 2009.

[23] S. Shahidinejad, E. Bibeau, and S. Filizadeh. Statistical development of a duty cycle for plug-in vehicles in a North American urban setting using fleet information. IEEE Transactions on Vehicular Technology, 59(8):3710–3719, 2010.

[24] P. Nyberg, E. Frisk, and L. Nielsen. Using real-world driving databases to generate driving cycles with equivalence properties. 2015

[25] P. Nyberg, E. Frisk, and L. Nielsen. Driving cycle adaption and design based on mean tractive force. In Proceedings of 7th IFAC Symposium on Advances in Automotive Control, volume 7, pages 689–694, Tokyo, Japan, 2013

[26] P. Nyberg, E. Frisk, and L. Nielsen. Generation of equivalent driving cycles using markov chains and mean tractive force components. In Proceedings of 19th IFAC World Congress, volume 19, pages 8787–8792, Cape Town, South Africa, 2014

[27] Skyler Seto, Wenyu Zhang, Yichen Zhou. Multivariate Time Series Classification Using Dynamic Time Warping Template Selection for Human Activity Recognition. Dec 2015

[28] John Paparrizos, Luis Gravano. K-Shape: Efficient and Accurate Clustering of Time Series. *ACM SIGMOD Record 45(1):69-76 . June 2016*

[29] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In AAAI Workshop on KDD, pages 359–370, 1994.

[30] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. Tung. Spade: On shape-based pattern detection in streaming time series. In ICDE, pages 786–795, 2007.

[31] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, Teh Ying Wah. Time-series clustering - A decade review *Information Systems, vol. 53, p. 16-38, 2015.*

[32] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Eamonn Keogh. The UCR Time Series Classification Archive. 2018

[33] Sebastian Raschka, Vahid Mirjalili. 2017. Python Machine Learing Second Edition *EXPERT INSIGHT, p. 174-186, 500-529.*

[34] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In AAAI Workshop on KDD, 1994.

[35] Steven Elsworth. 2017. Dynamic Time Warping. *Lecture notes School of Mathematics, University of Manchester, p. 1-45.*

[36] J. H. Friedman, J. L. Bentley, and R.A. Finkel. 1977. An Algorithm for Finding Best Matches in Logarithmic Expected Time *ACM transactions on mathematical software (TOMS), 3(3): 209–226*

[37] Scikit-learn: Machine Learning in Python, Pedregosa et al. *JMLR 12, pp. 2825-2830, 2011*

[38] Bahl, L and Jelinek, Frederick. Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *IEEE Transactions on Information Theory, 21(4):404–411, 1975.*

[39] Ristad, Eric Sven and Yianilos. Peter N. Learning stringedit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(5):522–532, 1998.*

[40] Marco Cuturi, Mathieu Blondel. Soft-DTW: a Differentiable Loss Function for Time-Series. ICML 2017

[41] Cuturi, Marco. Fast global alignment kernels. *In Proceedings of the 28th international conference on machine learning (ICML-11), pp. 929–936, 2011.*

[42] Olivier Renaud, Maria-Pia Victoria-Feser. A Robust Coefficient of Determination for Regression. *Journal of Statistical Planning and Inference, Vol. 140, No. 7, pp. 1852-1862, 2010*

[43] Harrell, F.E.. Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis. *Springer-Verlag, New York, 2001*