Politecnico di Torino

Master Degree in Computer Engineering

Master Thesis

A Big Data Solution for Silhouette Computation

Advisor
Paolo Garza
Co-Advisor
Eliana Pastor

Candidate
Sara Prone

Academic Year 2018/2019

Abstract

For data analysis, the partitioning into groups based on data characteristics is crucial. This process is called *clustering* and its result is a set of groups containing all the original objects, where objects in the same group are more similar to each other than to objects in other groups. The clustering process only partitions data into clusters, so at the end of the process the number of records is the same as the original, with the additional information about their division in groups. Since in real world the data sets are likely to contain huge amounts of data, a way to reduce these quantities maintaining the most important features of original objects is proposed in this work. The idea is simply to summarize the already clustered data, dividing them into cells with a certain size and computing a representative object for each cell. This object represents all the original records contained in the cell and it has a weight equal to the number of represented records. In this way, clusters of weighted objects are generated and a resultant so-called *weighted clustering* is obtained. The *weighted clustering* is a representation of the original partition into clusters, with a reduced cardinality (i.e. an amount of records lower than the starting one is used to represent all the original records).

The reduction of cardinality is crucial because operations on lower amounts of data are faster and easier. In many cases, the representation of the original records with a lower number of representative objects allows to perform operations that would have been not feasible on original data. To evaluate the quality of the clusters of weighted objects representation, the *silhouette index* has been used. It is an index used to evaluate the quality of a clustering (i.e. a partition into clusters). A modification of this index that considers weights of objects has been created in this thesis. This version, called *weighted silhouette*, is particularly important because the *silhouette index* time complexity is quadratic in the number of considered data. For this reason, the application of the normal *silhouette index* on large data sets is not feasible. Using the weighted version proposed in this work, the index can be computed for high amounts of clustered objects. Before the computation of the *weighted silhouette*, the proposed *weighted clustering* process must be applied in order to generate a representation of the original objects using a lower number of weighted objects.

Contents

Chapter 1

Introduction

1.1 About clustering

We are living in a world where everyone and everything produces data. Everyday a larger amount of information that can be represented as data is produced. The organization of this enormous amount of information is crucial: understanding information and learning from that is not possible without the knowledge of how to analyze, manage and classify it. In particular classify data and divide them into groups plays an essential role in data science. These groups take the name of clusters and the process of grouping them into different clusters is the *data clustering* or *cluster analysis*. It is also called *unsupervised learning* or *unsupervised classification* [1], since data are partitioned into clusters without any knowledge about their *natural* partitions.

Cluster analysis or clustering is the organization of a number of objects, usually very large, into different clusters, trying to discover their *natural* grouping. A definition of clustering can be the following: given N objects, divide them into K groups based on a measure of similarity, such that two objects in the same group are more similar to each other than to objects in other groups, while objects in different groups are less similar to each other than to objects in the same group [2]. The grouping is normally done based on similarity, in a way to achieve internal homogeneity and external separation [3]. The following figures show an example of clustering: points in Figure 1.1 are the original two dimensional objects that have to be clustered, while in Figure 1.2 the desired clusters are identified with different colors. The diversity of clusters in terms of size, shape and density can be noticed. While a human eye easily recognize the division of points in Figure 1.1 into eight groups, the automatic partition of them is not so trivial. Algorithms that aim to find a good division in clusters starting from a set of data are called *clustering algorithms*.

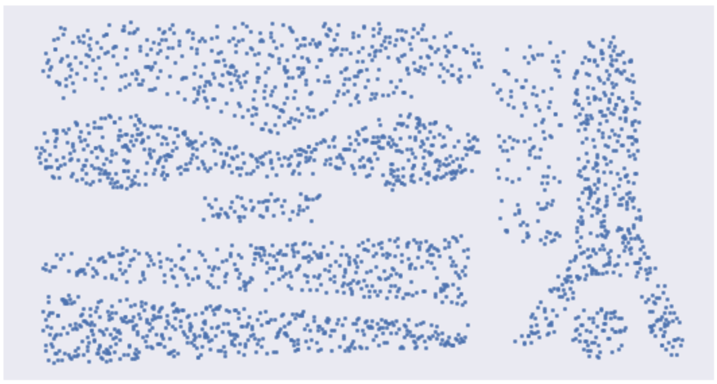

Figure 1.1: Original data

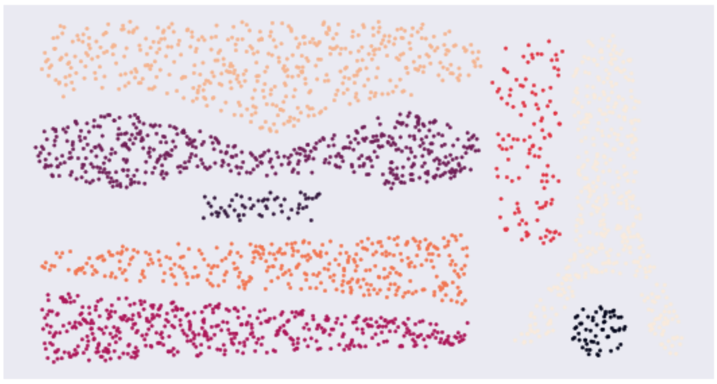

Figure 1.2: Desired clusters

1.2 About clustering evaluations

Clustering algorithms organize objects into clusters. A typical clustering process includes the following steps [4]:

- objects representation, optionally including feature extraction and/or selection
- definition of a similarity criterion appropriate to the type of data
- clustering (grouping)
- data abstraction, if needed
- production of output, if needed

An ideal cluster is a compact and isolated set of objects. In reality the partition into clusters is not trivial and the obtained clusters may not be the desired ones.

In some cases, with the current available clustering algorithms it is not possible to obtain the desired partitions. Moreover, the result of a clustering process varies depending not only on the used clustering algorithm but also on the set parameters and on the used criterion of similarity. There is the need of metrics to evaluate the quality of the clustering. These metrics indirectly evaluate the similarity measures used in the clustering process, since these similarity criteria combined with the mechanism are responsible for the obtained partitions [5].

Clustering validation indices, also called *CVIs* [6], are used to measure the quality of a clustering (i.e. a partition in clusters). These indices can be divided into three categories: external, internal and relative approaches [1]. External CVIs use external information to evaluate the clustering: if some prior knowledge such as the exact (or desired) partition exists, external CVIs can be used to evaluate how much the evaluated partition is similar to the prior partition. When such prior knowledge is absent, such as in unsupervised clustering, internal CVIs are used. In internal indices the quality of clustering is usually measured based on two aspects: the intra-cluster compactness and the inter-cluster separation, also known as isolation [6]. In relative methods different data partitioning results obtained by a clustering algorithm setting different values of input parameters are considered; they are compared with other partitioning schemes in order to find the best results [1]. In this work an internal clustering validation index has been studied and used: the *silhouette index* proposed by Rousseeuw in 1987. Some other examples of internal indices are the *Dunn index*, the *I index*, and the *Calinsky-Harabasz index*. They all use geometric information to evaluate the clustering [6]. The silhouette index is based on a measure of distance between the clustered objects: it uses the comparison of a measure of closeness of each object to the cluster where it has been allocated and a measure of distance from the closest alternative cluster [7].

1.3 The main idea

Clustering algorithms assign each object of a starting data set to a cluster, adding important information to the objects themselves: after clustering process, data can be seen as part of particular group where all the components share some features or have similar characteristics. This task adds information to data, without changing the cardinality of the data set.

Almost always, in real life, objects to be clustered are in very large number. Think for example to the huge quantity of web pages, e-mails, blogs, transactions, that every day create terabytes of new data. The quantity of available images and videos grows continuously, and also their quality and size increase thanks to the spread of inexpensive cameras. The increase in the quantity of data requires improvements in techniques to automatically understand, process, and summarize the available data [2]. Given the huge amount of data in data sets, also after the clustering when they are assigned to groups based on similarity, their analysis and the extraction of relevant information from them can be slow and complex because of the high cardinality.

In this work we propose a simple way to represent clustered data with a lower number of representative objects, in order to reduce cardinality. The idea is simply to divide clustered data into cells, where each cell has a number of dimensions equal to the number of dimensions of each original object (i.e. the considered characteristics of the original data). Then an object that represents the whole set of data contained in a cell is computed for each cell. To each representative object is finally assigned a weight, corresponding to the number of original records contained in the represented cell. This process can be applied after the partitioning in clusters. Each representative object keeps the cluster of the objects in the represented cell, in this way clusters of weighted representative objects are obtained. The final cardinality can be decided by setting a parameter: the size of the cells. These clusters of weighted objects will be called *weighted clusters* and the idea is that they can be used instead of the original clusters when operations on the originals are too slow because of their high cardinality. In the second part of the work, a modification of the silhouette index to evaluate the quality of weighted cluster is proposed. It expands the silhouette index, adding the consideration of the weights when computing closeness of each object to objects of the same cluster and distance of each object from objects of other clusters.

1.4 Next chapters

Section 2 contains a brief summary of background researches on computation of clustering validation indices and on assignment of weights to data. While a lot of material can be found about the silhouette index, studies about weighted data are not so common in the research community. In particular, no solutions have been found similar to the one proposed in this work, that aims to reduce the cardinality of already clustered data using weights.

In Section 3 the proposed solution is described. It starts illustrating the initial idea of weighted clustering algorithm: the division in cells, the computation of representative points, the assignment of weights. Then the proposed modification of silhouette index that considers the weights is described. Finally, a modification of the weighted clustering algorithm that handles better situations with scattered data is presented. For each process there are an initial description, the pseudo-code with comments and an example.

The following part (Section 4) is about the performed experiments and the obtained results. It starts with a description of the input data set used in the experiments and of the competitors used to evaluate our results. Then the detailed illustration of the main experiments done with a particular data set is reported and the results are fully commented. Finally, most important results and special cases are described for all the other considered data set.

The conclusive section contains an overview on all the obtained results and some hints on possible future works in this field.

Chapter 2

Background

2.1 Clustering validation indices

The clustering process is an unsupervised process, this means that there are no predefined classes and no examples to show how to group data. Clustering algorithms compute a division in clusters that most of the times is not known a priori, so the obtained clustering requires to be evaluated [8]. When instead the desired partition in clusters is known, evaluation metrics are a way to compare the obtained clustering to the desired one.

The clustering validation indices are used to measure the quality of a division in clusters obtained with a certain clustering algorithm. Among them, silhouette index has been studied. This index can be calculated for each object of the starting data set, and it allows to know if the object has been assigned to the best fit cluster or if there was a better choice. The silhouette index can also be computed for the whole clustered data set: it is the average of all the indexes computed for each single object of the data set. In this case it gives a measure of the quality of the clustering.

2.1.1 The silhouette index

The silhouette index, also called *SIL*, was proposed by Rousseeuw in 1987. Unlike most of clustering validation indices, the silhouette can be used for clusters of arbitrary shapes [1]. The author specifies that in order to calculate this index two premises are required [5]:

- a clustering of the original objects obtained using some clustering algorithm
- a structure for the storage of proximity values between objects

The silhouette index is defined as follows. Assuming the data of the data set D have been clustered into K clusters, let us denote these clusters with $\{C_1, C_2, ..., C_k, ..., C_K\}$, where C_k indicates the k^{th} cluster, with $k = 1, 2, ..., K$.
For each object X_i assigned to the cluster C_k, let $a(X_i)$ be the average distance between X_i and all the other objects X_k belonging to the same cluster C_k. The value of $a(X_i)$, being the mean of within-cluster distances, is a measure of how well the object X_i is assigned to its cluster C_k. The smaller is $a(X_i)$, the better is the assignment of X_i to cluster C_k.

$$a(X_i) = \frac{1}{N_k - 1} \sum_{X_k \in C_k} d(X_i, X_k) \tag{2.1}$$

where N_k is the number of objects in cluster C_k and $d(X_i,\ X_k)$ is a measure of distance between X_i and X_k.

The average dissimilarity between X_i and other objects X_l belonging to the cluster C_l, where $l = 1, 2, ..., K$ and $l \neq k$ is defined as:

$$\delta(X_i, X_l) = \frac{1}{N_l} \sum_{X_l \in C_l} d(X_i, X_l) \tag{2.2}$$

where N_l is the number of objects in cluster C_l. The cluster with the smallest average dissimilarity is called the *neighbouring cluster* of X_i because, excluding cluster C_k, it is the best cluster for this object.

Let $b(X_i)$ be the smallest average distance $\delta(X_i, X_l)$ computed for object X_i. This value indicates the distance between X_i and its neighbouring cluster.

$$b(X_i) = \min_{l=1, l \neq k}^{K} \delta(X_i, X_l) \tag{2.3}$$

The silhouette index of object X_i is finally defined as:

$$S(X_i) = \frac{b(X_i) - a(X_i)}{\max(a(X_i), b(X_i))} \tag{2.4}$$

Which can also be written as:

$$S(X_i) = \begin{cases} 1 - a(X_i)/b(X_i), & \text{if } a(X_i) < b(X_i) \\ 0, & \text{if } a(X_i) = b(X_i) \\ 1 - b(X_i)/a(X_i), & \text{if } a(X_i) > b(X_i) \end{cases} \tag{2.5}$$

While the silhouette index *SIL* of the whole clustered data set is defined as:

$$SIL = \begin{cases} \frac{1}{N_D} \sum_{X_i \in D} S(X_i), & \text{if } N_D \neq 1 \\ 0, & \text{if } N_D 1 \end{cases} \tag{2.6}$$

where N_D is the number of objects in the original data set D. Note that the *SIL* is 0 if the data set D contains only one point.

From these definitions is clear that $-1 \leq S(X_i) \leq 1$ and so also $-1 \leq SIL \leq 1$. The element X_i is assigned to the right cluster when $S(X_i)$ is close to 1. When this value is near to -1, this means that the neighbouring cluster of X_i is a better choice than the chosen cluster, so the object X_i is located in the wrong cluster. A value of the silhouette index *SIL* close to 1 indicates that the partition in clusters has been done well. Given two clustering obtained from the same data set using different clustering algorithms or different parameters, the one with higher value of *SIL* is the best one. If an optimal partitions in clusters is available for the considered data set, the evaluation of the quality of the clustering obtained with a certain clustering algorithm can be done by comparing the *SIL* of the obtained clustering and the *SIL* of the given optimal clustering.

The silhouette index is based on two components: $a(X_i)$ and $b(X_i)$. The first component is the average distance between X_i and other objects belonging to the same cluster, so it can also be interpreted as a measure of cluster compactness. The other component is the smallest of the average distances between X_i and the objects belonging to other clusters. The difference between $b(X_i)$ and $a(X_i)$, can be considered as a measure of cluster separability [1].

2.1.2 Implementations of the silhouette index

Apache Spark offers the library *org.apache.spark.ml.evaluation.ClusteringEvaluator*, available since Spark 2.3.0 for Scala, Java, Python and R, which computes the silhouette index using the *squared Euclidean distance* [9].

Deriving the Euclidean distance between two data involves computing the square root of the sum of the squares of the differences between corresponding values. The equation for the calculation of this distance between a point $P(p_1, p_2, ..., p_N)$ and a point $Q(q_1, q_2, ..., q_N)$ is reported in Equation 2.7, where N is the number of dimensions of the considered points.

$$\delta(P, Q) = \sqrt{\sum_{i=1}^{N} (p_i - q_i)^2} \tag{2.7}$$

The squared Euclidean distance metric uses the same equation as the Euclidean distance, but does not take the square root:

$$\delta_{\text{squared}}(X, Y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{2.8}$$

As a result, clustering processes using the Euclidean distance metric are slower than the ones using the squared Euclidean distance. Even if this second metric is faster, in some cases the use of the Euclidean distance may be preferred. For example the output of Jarvis-Patrick and K-Means clusterings is not affected if Euclidean distance is replaced with its squared version, but the output of hierarchical clustering is likely to change [10].

In this work it has been decided to provide a new implementation of the computation of the silhouette index that uses the Euclidean distance, missing in Spark.

2.2 Weighted data

2.2.1 Weighted clustering

In traditional clustering each cluster is simply a group containing a subset of the initial data, i.e. each initial object is assigned to a cluster and the result of this operation is the clustering. In this work, some transformations are applied to the clustering in order to reduce the cardinality of each cluster. At the end of the process a so-called *weighted clustering* is obtained, where clusters are composed by *weighted objects*. Each cluster is converted to a *weighted cluster* which contains some representative objects, each one of them represents a sub set of the original objects in the considered cluster. The *weighted* adjective is used because these representative objects have a weight, which corresponds to the number of represented objects.

The issue of assigning a weight to data has been addressed a lot in the research community. Most clustering algorithms treat data as entities with equal weights, but it is clear that in most cases not all objects in a data set have the same importance in cluster analysis [11]. For example, outliers should have less impact than other objects on clustering results. So it can be useful to compute objects weights before applying a clustering algorithm to a data set. Implementations of weighted clustering can be found, that assign a weight to each object before the clustering process, and then consider the weights during the clustering. In some solutions the sample weights in clustering algorithms need to be provided by users or heuristic methods. Attempts to compute sample weights are Li et al. [12], which determine the sample weight for an object X_i as the number of objects around X_i less than a threshold. Zhang et al. [13] calculate sample weights for documents by computing the so-called *PageRank* value of each document based on the citing relationship between them. Jian Yu, Miin-Shen Yang, E. Stanley Lee [11] consider a probability distribution over a data set to represent its sample weights, then apply the maximum entropy principle to automatically compute sample weights for clustering.

All these solutions assign the weights before the clustering process and use them during the partition in clusters. In this work we describe a way to summarize already clustered data using some representative points with weights, in order to make easier and faster the after clustering operations such as the silhouette calculation.

2.2.2 Weighted silhouette index

While many researches have been made on cluster analysis considering weights, just little material can be found about evaluation of weighted clustering.

The silhouette index previously described (Section 2.1.1) works for traditional clustering (i.e. each cluster is simply a group containing a subset of the initial data). In our work, a cluster is not a subset of the initial objects but it is a set of representative objects, where each one of them represents a subset of the original data and has a weight equal to the quantity of represented data. Each representative objects clearly belongs to a cluster: the cluster that was assigned to the objects that it represents. For the evaluation of the quality of this new type of clusters, the standard silhouette index can not be applied since it does not consider the weights.

In the *WeightedCluster* library for *R* [14], an implementation for the computation of a weighted version of the silhouette index is provided. The clustering methods available in the WeightedCluster library are based on a measure of dissimilarity between the objects, used to compare them. Two main stages of cluster analysis are described: the algorithms for partitioning the objects into clusters and the measure of the quality of the obtained clustering. The WeightedCluster library considers the weight of the objects in both these main phases of the analysis [14].

For the computation of an indicator for the quality of the weighted clusters (i.e. clusters of weighted objects), the following modification of the silhouette index is proposed. The original formulation of this index supposes that one weighting unit is equivalent to one observation (i.e. the data are not weighted). For weighted data, a variant of the silhouette index called *weighted silhouette* is available in this library. As the original silhouette index, this modification measures the coherence of the assignment of an objects to its cluster [14].

The clusters to be evaluated are clusters of weighted objects: each object has the features of the original data and a weight. Let D be a set of weighted objects. A single weighted object is denoted with X_i, and its weight is w_i. Using some weighted clustering algorithm, the weighted objects of D have been clustered into K clusters of weighted objects, denoted with $\{C_1, C_2, ..., C_k, ..., C_K\}$, where C_k indicates the k^{th} cluster, with $k = 1, 2, ..., K$. Considering a weighted object X_i with weight w_i belonging to the cluster C_k, W_k is the sum of all the weights of objects assigned to the same cluster C_k:

$$W_k = \sum_{X_k \in C_k} w_k \tag{2.9}$$

where w_k is the weight assigned to weighted object X_k belonging to cluster C_k.

For the weighted object X_i, $a(X_i)$ is defined as the average weighted distance between X_i and all the other objects X_k belonging to the same cluster C_k:

$$a(X_i) = \frac{1}{W_k - 1} \sum_{X_k \in C_k} w_k \cdot d(X_i, X_k) \tag{2.10}$$

where W_k is defined in Equation 2.9, w_k is the weight of X_k and $d(X_i,\ X_k)$ is a measure of distance between X_i and X_k.

For the weighted object X_i, let $\delta(X_i, X_l)$ be the average weighted distance between X_i and any other object X_l belonging to another cluster C_l, where l = 1, 2, ..., K and $l \neq k$:

$$\delta(X_i, X_l) = \frac{1}{W_l} \sum_{X_l \in C_l} w_l \cdot d(X_i, X_l) \tag{2.11}$$

where W_l is the sum of the weights of objects belonging to cluster C_l.

For the weighted object X_i belonging to cluster C_k, $b(X_i)$ is defined as the smallest average weighted distance $\delta(X_i, X_l)$ computed between X_i and any other object X_l belonging to another cluster C_l with l = 1, 2, ..., K and $l \neq k$:

$$b(X_i) = \min_{l=1, l\neq k}^{K} \delta(X_i, X_l) \tag{2.12}$$

The weighted silhouette of X_i is computed as:

$$S(X_i) = \frac{b(X_i) - a(X_i)}{\max(a(X_i), b(X_i))} \tag{2.13}$$

Which can also be written as:

$$S(X_i) = \begin{cases} 1 - a(X_i)/b(X_i), & \text{if } a(X_i) < b(X_i) \\ 0, & \text{if } a(X_i) = b(X_i) \\ 1 - b(x_i)/a(X_i), & \text{if } a(x_i) > b(X_i) \end{cases} \tag{2.14}$$

The weighted silhouette index of the whole weighted clustering is defined as:

$$wSIL = \frac{1}{N_D} \sum_{X_i \in D} S(X_i) \tag{2.15}$$

where D is the starting set of weighted objects, and N_D is the number of weighted objects in D.

Also for this modified index is clear that $-1 \leq S(X_i) \leq 1$ and so also $-1 \leq wSIL \leq 1$. A value $S(X_i)$ can be computed for each object, but more attention is paid to the average silhouette $wSIL$. If this is weak, it means that the groups are not clearly separated or that the homogeneity of the groups is low [14].

In Equation 2.10 the weighted sum of distances is divided by W_k-1. If some of the w_i or some of the W_k are lower than 1, the $a(X_i)$ are undefined and the silhouette can not be computed. The assumption for dividing by W_k-1 is that the weighting unit is equivalent to one observation: this happens when objects are not weighted or when the weights are computed from an aggregation of identical objects. There may be other cases, for example when the weights are used for representing the importance of an object of the original data set, where weights may be lower than 1. To solve this issue, in the WeightedClustering library the use of $aw(X_i)$ instead of $a(X_i)$ is proposed [14]:

$$aw(X_i) = \frac{1}{W_k} \sum_{X_k \in C_k} w_k \cdot d(X_i, x_k) \tag{2.16}$$

The $aw(X_i)$ value can be interpreted as the distance between the object X_i and its own cluster, considering all the objects of the cluster. In the original formulation, X_i is removed from the set before computing the $a(X_i)$ value. Another interpretation for the $aw(X_i)$ can be the $a(X_i)$ value when the weighting unit is as small as possible, i.e. when it tends to zero [14].

Chapter 3

Proposed solution

3.1 Weighted clustering

Let us assume the data of a data set have been partitioned into clusters using a certain clustering algorithm. The result is a number of clusters, each one of them contains a subset of the original data set. The traditional clustering is considered: each data after the clustering process has been assigned to one and only one cluster, differently from the *Fuzzy* clustering in which an object can be classified to more than one clusters [8].

The process to obtain the weighted clusters consists in the partition of the clustered data into cells. A representative point called *weighted macropoint* will be computed for each cell, and it will represent all objects in the cell. It will have a weight equal to the number of represented objects. A value called *step* is received as only input parameter and it defines the size of cells on each dimension. Each cell will have a number of dimensions equals to the number of dimensions of initial objects, i.e. the number of features considered for initial data, and on each dimension the size of the cell will be equal to the *step* value.

The procedure for obtaining the cells and their representative points (weighted macropoints) is the following. Starting from the already clustered data, one cluster at a time is considered and it is divided into cells using the *step* value. Let us assume the cluster *Ck* is the considered one. A generic point belonging to cluster *Ck* is identified by $P(p_1, p_2, ..., p_j, ..., p_N)$, where N is the number of dimensions of the objects and $\mathrm{p_j}$ is the value of object P on the j^{th} dimension. In order to divide each cluster in cells of dimension *step*, first the minimum value of the cluster is computed on each dimension:

$$min_j = \min_{P \in C_k} \; p_j \tag{3.1}$$

where C_k is the cluster considered to be divided into cells, $P \in C_k$ indicates a point of cluster C_k, p_j is the value of point P on the j^{th} coordinate, and the result min_j is the value of the minimum on the j^{th} dimension among all points of cluster C_k.

The cluster of points is then divided into a number of cells with size *step* on each coordinate, starting from the minimum of the set computed with Equation 3.1. A point $P(p_1, p_2, ..., p_j, ..., p_N)$ is assigned to the cell identified by the point $C(c_1, c_2, ..., c_j, ..., c_N)$, where c_j is computed for each dimension j = 1, 2, ..., N as:

$$c_j = Floor\left(\frac{p_j - min_j}{step}\right) \cdot step + min_j \tag{3.2}$$

where p_j is the j^{th} coordinate of point P, min_j is the minimum value computed on the j^{th} coordinate among all the points of the considered cluster with Equation 3.1, and the result c_j is the value of the point that identifies the cell on its j^{th} coordinate. In this way the cell to which the point P has been assigned is identified by a point C which contains on each coordinate the minimum value of the cellâĂŹs points computed on that coordinate.

After assigning each point P to a cell, a representative point called macropoint M is computed for each cell as the average value on each dimension computed on all the points belonging to cell C. Let N_C be the number of points in cell C. All these points are represented by the macropoint $M(m_1, m_2, ..., m_j, ..., m_N)$, where m_j is computed on each dimension as:

$$m_j = \frac{1}{N_C} \sum_{P \in C} p_j \tag{3.3}$$

where N_C is the number of points in the cell identified by the point C, $P \in C$ indicates a point belonging to cell C, and p_j is the j^{th} coordinate of point P. Being N_C the number of points in cell C represented by the macropoint M, the weight of M is set to N_C. All points belonging to cell C are represented by the macropoint M with weight $w_M = N_C$. In the resulting weighted clustering the macropoint M is assigned to the cluster C_k, which was the cluster of all the original points represented by M.

In this way, starting from the original cluster C_k containing N_k points, a new weighted cluster wC_k is obtained. The weighted cluster wC_k contains a number of weighted points (i.e. the computed macropoints) that depends on the *step* value, so the cardinality of the resulting representation can be set by setting this parameter.

3.1.1 Weighted clustering pseudocode

Algorithm 1 Weighted Clustering

Input: the original clustering: the set *clusters* which contains clusters of points; the value of *step*

Output: the result of weighted clustering process: the set *weightedClusters* which contains clusters of weighted macropoints

```
1: clusters;                                              ▷ Clusters of points
2: weightedClusters = {};                   ▷ Set of computed weighted clusters
3: For each cluster, compute the cells, the weighted macropoints, and the weighted
   cluster corresponding to the considered cluster:
4: for cluster in clusters do
5:     weightedCluster = {};        ▷ Weighted cluster corresponding to the cluster
6:     clusterPoints = cluster.points;
7:     For each dimension, compute the minimum value between cluster points:
8:     min = computeMin(clusterPoints);
9:     step = args.step;
10:    For each point of the cluster, compute the cell:
11:    for point in clusterPoints do
12:        Compute the value of the cell for the point on each coordinate:
13:        cell = computeCell(point, min, step);
14:        Add the point to the set of points belonging to the computed cell:
15:        cell.points.add(point)
16:    cells;                                               ▷ All computed cells
17:    For each cell, retrieve the value of the macropoint on each dimension:
18:    for cell in cells do
19:        cellPoints = cell.points;
20:        macropoint = computeAverage(cellPoints);
21:        macropoint.weight = cellPoints.size;
22:    Add the computed macropoints to the weighted cluster:
23:    for cell in cells do
24:        weightedCluster.add(cell.macropoint);
25:    Add the computed weighted cluster to the set of weighted clusters:
26:    weightedClusters.add(weightedCluster);
27: return weightedClusters;
```

Algorithm 2 computeMin

Input: the original points of a cluster or a cell, referred with *points*

Output: minimum value *min* among all *points* computed on each dimension

```
1: N = number of dimensions;
2: for i ← 0 to N-1 do
3:     $min_i$ = MAXVALUE;
4: for point in points do
5:     for i ← 0 to N-1 do
6:         if $point_i < min_i$ then
7:             $min_i = point_i$
8: return min;
```

Algorithm 3 computeCell

Input: *point* to be assigned to a cell, *min* of the cluster of the point, *step*

Output: identifier of the *cell* the *point* has been assigned to, computed on each dimension

```
1: N = number of dimensions;
2: for i ← 0 to N-1 do
3:     $cell_i$ = Floor(($point_i$-$min_i$)/step)*step+$min_i$;
4: return cell;
```

Algorithm 4 computeAverage

Input: original points of a cell *cellPoints*

Output: values of the *macropoint* representing all the *cellPoints* on each dimension

```
1: N = number of dimensions;
2: for i ← 0 to N-1 do
3:     $sum_i$ = 0;
4:     $count_i$ = 0;
5:     for point in cellPoints do
```

```
6:          sum_i += point_i;
7:          count_i++;
8:      for i ← 0 to N-1 do
9:          macropoint_i = sum_i/count_i;
10:     return macropoint;
```

Weighted Clustering algorithm

The *Weighted Clustering* process starts from a set of points already partitioned into clusters: *clusters* (line 1). For each cluster, the correspondent weighted cluster *weightedCluster* is retrieved as follows. The original points of the considered cluster are extracted (line 6), and the minimum among all these points is computed on each dimension (line 8), using *computeMin* (Algorithm 2). In line 9 the *step* value is retrieved from command line arguments. For each point of the considered cluster, the identifier of the cell the point has to be assigned to, is computed on each dimension (line 13), using the logic described in *computeCell* (Algorithm 3). Given the values of the identifier of the cell on each coordinate, the considered point is added to the set of points belonging to that cell (line 15). At the end of this process, a set of cells has been obtained, each cell containing a sub set of original points of the considered cluster (line 16). For each one of these cells, the coordinates of the correspondent macropoint are computed as the average of the values of all the points belonging to the cell on that coordinate (line 20), using *computeAvg* (Algorithm 4). The weight of the macropoint is set to the number of points belonging to the cell represented by the macropoint (line 21). Finally, the macropoints retrieved for the current *cluster* are added to the correspondent *weightedCluster* (line 24), and the *weightedCluster* is added to the set of *weightedClusters* (line 26). The set of *weightedClusters* is the result of the *Weighted Clustering algorithm.*

3.1.2 Weighted clustering example

In the following figures, a graphical explanation can be found abouSt the assignment of points to cells, the computation of macropoints for the cells, the retrieving of weighted clusters. The original points considered for the example are displayed in Table 3.1 and in Figure 3.1, already divided into two clusters.

	p_x	p_y	*cluster*
P_1	2	1	1
P_2	1	2	1
P_3	2	2	1
P_4	2	3	1
P_5	3	3	1
P_6	3	5	1
P_7	5	2	2
P_8	5	5	2
P_9	6	1	2
P_{10}	6	2	2
P_{11}	6	3	2
P_{12}	6	5	2

Table 3.1: Original points partitioned into cluster 1 and cluster 2: point (p_x, p_y) is assigned to cluster *cluster*

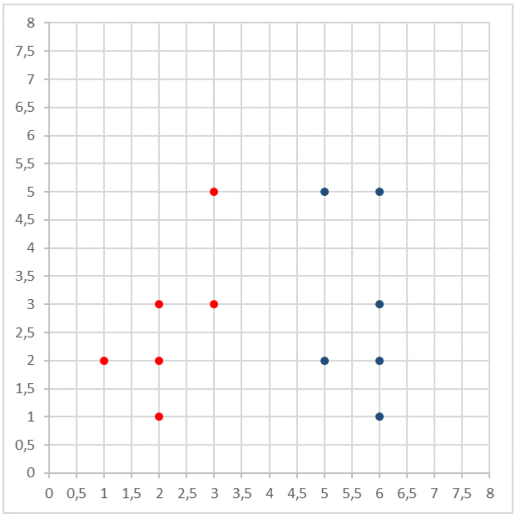

Figure 3.1: Original points partitioned into cluster 1 (red points) and cluster 2 (blue points)

At the beginning, the minimum values of the clusters are computed. Results are displayed in Table 3.2 and in Figure 3.2.

An example of computation of the minimum for a cluster is reported in the following lines. Let us consider cluster 2 (indicated with C_2). It contains points P_7, P_8, P_9, P_{10}, P_{11} and P_{12}. Among all these points the minimum $MIN_2(min_x, min_y)$ is computed on each dimension x and y as:

$$min_x = \min_{P_i \in C_2} p_x = min\{5.0, 5.0, 6.0, 6.0, 6.0, 6.0\} = 5.0$$
$$min_y = \min_{P_i \in C_2} p_y = min\{2.0, 5.0, 1.0, 2.0, 3.0, 5.0\} = 1.0$$

So the minimum for cluster 2 is $MIN_2(min_x, min_y) = (5.0, 1.0)$.

	min_x	min_y	*cluster*
MIN_1	1	1	1
MIN_2	5	1	2

Table 3.2: Minimums on each dimension among points of cluster 1 and 2

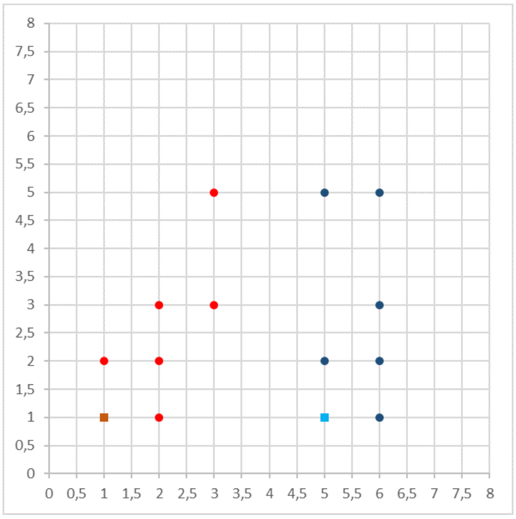

Figure 3.2: Minimums among points of cluster 1 (the brown point) and 2 (the light blue point)

The partition of points into cells is performed as illustrated in the next example. The assignments of points to the computed cells are shown in Table 3.3 and in Figure 3.3. The *step* value is set to 1.5.

Example of assignment of a point to a cell:
Let us consider the point $P_{12}(p_x,\ p_y) = (6.0,\ 5.0)$. It belongs to cluster C_2, minimum of cluster 2 is $MIN_2(min_x,\ min_y) = (5.0,\ 1.0)$. Point P_{12} is assigned to cell $C(c_x,\ c_y)$, computed as:

$$c_x = Floor\left(\left(p_x - min_x\right)/step\right) \cdot step + min_x = Floor\left(\left(6.0 - 5.0\right)/1.5\right) \cdot 1.5 + 5.0 = 5.0$$
$$c_y = Floor\left(\left(p_y - min_y\right)/step\right) \cdot step + min_y = Floor\left(\left(5.0 - 1.0\right)/1.5\right) \cdot 1.5 + 1.0 = 4.0$$

So point P_{12} is assigned to cell $C(c_x,\ c_y) = (5.0,\ 4.0)$.

	p_x	p_y	*cluster*	c_x	c_x
P_1	2	1	1	1	1
P_2	1	2	1	1	1
P_3	2	2	1	1	1
P_4	2	3	1	1	2.5
P_5	3	3	1	2.5	2.5
P_6	3	5	1	2.5	4
P_7	5	2	2	5	1
P_8	5	5	2	5	4
P_9	6	1	2	5	1
P_{10}	6	2	2	5	1
P_{11}	6	3	2	5	2.5
P_{12}	6	5	2	5	4

Table 3.3: Assignement of points to cells: point $(p_x,\ p_y)$ is assigned to the cell identified by the point $(c_x,\ c_y)$

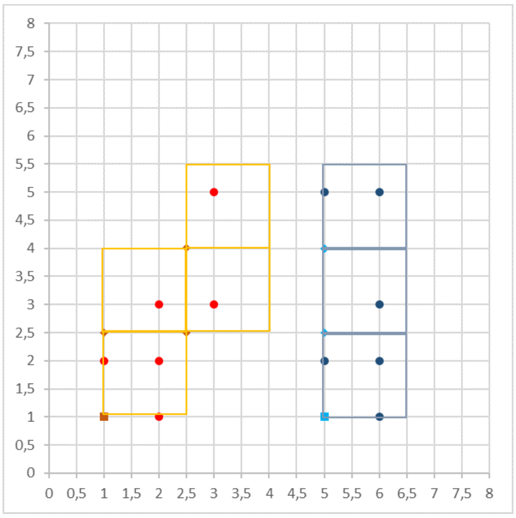

Figure 3.3: Assignement of points to cells (yellow and light blue sqares)

For each cell, the macropoint that represents all its points is computed as described in the next example. Macropoints representing the cells can be found in Table 3.5 and in Figure 3.4.

Example of macropoint computation:

Let us consider again point $P_{12}(p_x,\ p_y) = (6.0,\ 5.0)$, assigned to cell $C(c_x,\ c_y) = (5.0,\ 4.0)$. Also point $P_8 = (5.0,\ 5.0)$ has been assigned to cell C, so the macropoint associated to cell C must represent both point P_8 and point P_{12}. The macropoint $M(m_x,\ m_y)$ computed for cell C is defined on each dimension as:

$$m_x = \frac{1}{N_C} \sum_{P \in C} p_x = \frac{1}{2}(5.0 + 6.0) = 5.5$$
$$m_y = \frac{1}{N_C} \sum_{P \in C} p_y = \frac{1}{2}(5.0 + 5.0) = 5.0$$

So point $P_8 = (5.0,\ 5.0)$ and point $P_{12} = (5.0,\ 6.0)$, both assigned to cell $C = (5.0,\ 4.0)$, are represented by macropoint $M(m_x,\ m_y) = (5.5,\ 5.0)$. Since the represented points are two, the weight of M is set to 2.0.

	p_x	p_y	*cluster*	c_x	c_x	m_x	m_y	*weight*
P_1	2	1	1	1	1	1.67	1.67	3
P_2	1	2	1	1	1	1.67	1.67	3
P_3	2	2	1	1	1	1.67	1.67	3
P_4	2	3	1	1	2.5	2	3	1
P_5	3	3	1	2.5	2.5	3	3	1
P_6	3	5	1	2.5	4	3	5	1
P_7	5	2	2	5	1	5.67	1.67	3
P_8	5	5	2	5	4	5.5	5	2
P_9	6	1	2	5	1	5.67	1.67	3
P_{10}	6	2	2	5	1	5.67	1.67	3
P_{11}	6	3	2	5	2.5	6	3	1
P_{12}	6	5	2	5	4	5.5	5	2

Table 3.4: Cells and macropoints for each point: point $(p_x,\ p_y)$ is assigned to the cell identified by $(c_x,\ c_y)$, and identified by the macropoint $(m_x,\ m_y)$ with weight *weight*

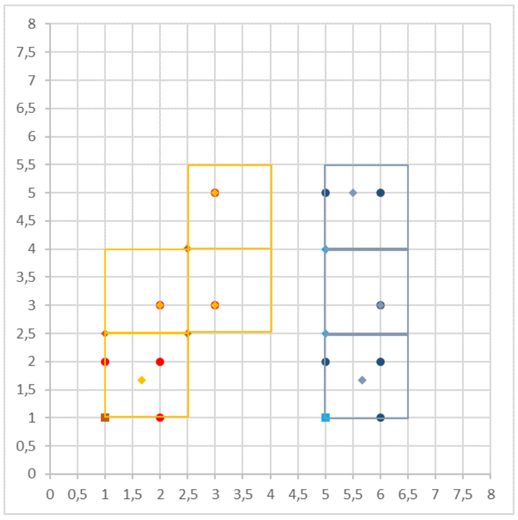

Figure 3.4: Macropoints for cells in cluster 1 (yellow points) and cluster 2 (light blue points)

The original two clusters were composed by twelve points, six in the first cluster and six in the second one. Using a *step* of 1.5, cluster 1 has been divided into four cells and cluster 2 into three cells. In this way, a representation of the original clusters with a reduced cardinality is obtained: the weighted cluster 1 contains four weighted macropoints, and the weighted cluster 2 contains three weighted macropoints, so the total cardinality has been reduced from twelve to seven.

Result of the weighted clustering process is shown in the following table.

	m_x	m_y	*weight*	*weighted cluster*
M_1	1.67	1.67	3	1
M_2	2	3	1	1
M_3	3	3	1	1
M_4	3	5	1	1
M_5	5.67	1.67	3	2
M_6	5.5	5	2	2
M_7	6	3	1	2

Table 3.5: Resultant weighted clusters composed by weighted macropoints: macropoint $(m_x,\ m_y)$ with weight with weight *weight* is assigned to the weighted cluster *weighted cluster*

3.2 Weighted silhouette

Most of the times clustering processes are applied without any knowledge about the truth natural partitioning of the data set, so validation indices have a lot of importance at the end of the process. After the division of the original data set in clusters, these indices can be used to obtain information about the quality of the clustering. Even when an optimal division in clusters is available, validation indices can be useful to compare the quality of the obtained clustering with the quality of the optimal one. Among all indices, the silhouette has been studied.

The time complexity of the silhouette index computation is quadratic in the number of considered points [15]. It is time consuming and it does not scale well when the number of data involved in the clustering is high. For Big Data the use of the original version of the silhouette is not feasible. To solve this issue the weighted clustering method previously described can be applied. Then a weighted version of the silhouette index will operate on a reduced number of data, since the weighted clustering reduced the cardinality.

The so-called *weighted silhouette* operates on a quantity of weighted data obtained with the weighted clustering algorithm that is lower than the quantity of original data. The aim is to solve the scalability issue.

This implementation of the weighted silhouette starts from an idea similar to the one described in Section 2.2.2. Using the weighted clustering algorithm previously presented (Section 3.1), a number of weighted clusters equal to the number of original clusters is retrieved. The weighted clusters contain weighted macropoints that represent the original objects and have a weight equal to the number of represented objects. The number of macropoints in each cluster depends on the value of the *step* but is generally lower than the original number of points in the cluster. These clusters of weighted macropoints need to be evaluated.

Let us denote a single weighted macropoint with M_i, and its weight with w_i. The number of weighted clusters (i.e. clusters of weighted macropoints) is K, and the set of all clusters is $\{C_1, C_2, ..., C_k, ..., C_K\}$ where C_k indicates the k^{th} cluster, with k = 1, 2, ..., K. Considering a weighted macropoint M_i with weight w_i assigned to the weighted cluster C_k, let W_k be the sum of all the weights of the macropoints belonging to the same weighted cluster C_k:

$$W_k = \sum_{M_k \in C_k} w_k \tag{3.4}$$

where w_k is the weight associated to macropoint M_k.

For the considered weighted macropoint M_i, the value of textita(M_i) is defined as the average weighted distance between M_i and all the other macropoints M_k belonging to the same cluster C_k:

$$a(M_i) = \frac{1}{W_k - 1} \sum_{M_k \in C_k} w_k \cdot d\left(M_i, M_k\right) \tag{3.5}$$

where W_k is defined in Equation 3.4 and $d(M_i, M_k)$ is computed as the Euclidean Distance between M_i and M_k.

Let $\delta\left(M_i, M_l\right)$ be the average weighted distance between M_i and any other macropoint M_l belonging to another cluster C_l, with $l = 1, 2, ..., K$ and $l \neq k$:

$$\delta(M_i, M_l) = \frac{1}{W_l} \sum_{M_l \in C_l} w_l \cdot d(M_i, M_l) \tag{3.6}$$

where W_l is the sum of the weights of macropoints belonging to cluster C_l, w_l is the weight of macropoint M_l and $d(M_i, M_l)$ is the Euclidean Distance between M_i and M_l.

For the macropoint M_i, $b(M_i)$ is the smallest average weighted distance computed between M_i and any other macropoint belonging to another cluster $C_l \neq C_k$:

$$b(M_i) = \min_{l=1, l \neq k}^{K} \delta(M_i, M_l) \tag{3.7}$$

The weighted silhouette of the weighted macropoint M_i is finally defined as:

$$S(M_i) = \frac{b(M_i) - a(M_i)}{\max(a(M_i), b(M_i))} \tag{3.8}$$

While the weighted silhouette index which gives information about the quality of a single cluster of weighted macropoints is:

$$SIL_k = \begin{cases} 0, & \text{if } (N_k = 1 \text{ AND weight} = 1) \\ \frac{1}{N_k} \sum\limits_{M_i \in C_k} S(M_i), & \text{otherwise} \end{cases} \tag{3.9}$$

where C_k is the considered weighted cluster and N_k is the number of macropoints in C_k. Note that the SIL_k is 0 if the weighted cluster C_k contains only one macropoint and the weight of this macropoint is 1.

Finally the weighted silhouette index which gives information about the quality of the whole weighted clustering is defined as:

$$SIL = \begin{cases} 0, & \text{if } (N_D = 1 \text{ AND weight} = 1) \\ \frac{1}{N_D} \sum\limits_{M_i \in D} S(M_i), & \text{otherwise} \end{cases} \tag{3.10}$$

where D is the set containing all the macropoints computed during the weighted clustering process and $\mathrm{N_D}$ is the quantity of these macropoints. Note that the *SIL* is 0 if the set D contains only one macropoint and the weight of this macropoint is 1. The case $N_D < 0$ is not considered as it corresponds to an empty set of macropoints.

As described in Section 3.1, the weight assigned to each macropoint is equal to the number of original objects represented by the macropoint. One macropoint is computed for each cell, and a cell is created only if there are objects in the covered area. From these considerations it is clear that $math1 \leq weight \leq N_D$, where N_D is the number of objects in the original data set. So the value of W_k, being obtained as a sum of weight, will never be less than 1. In the case of only one considered macropoint and weight of this macropoint equal to 1, W_k would be equal to 1 but the weighted silhouette is set to 0 by definition without performing any previous operation. It is clear that the division by W_k-1 in Equation 3.5 is always safe.

3.2.1 Weighted silhouette pseudocode

Algorithm 5 Weighted Silhouette

Input: the result of the weighted clustering process: the set of weighted clusters *wClusters*

Output: the silhouettes computed for each single macropoint, for each single weighted cluster and for the whole set of macropoints

```
 1: wClusters;                                   ▷ The collection of weighted clusters
 2: datasetSumWeights = 0;
 3: datasetSumSilhouettes = 0;
 4: For each weighted cluster, compute the weighted silhouette of all its
    macropoints and the average weighted silhouette of the weighted cluster:
 5: for wCluster in wClusters do
 6:     macropoints = wCluster.macropoints;
 7:     clusterSumWeights = 0;
 8:     clusterSumSilhouettes = 0;
 9:     if macropoints.size == 1 AND m.weight == 1 then
10:         datasetSumWeights += 1;
11:         datasetSumSilhouettes += 0;
12:         Average weighted silhouette of the weighted cluster:
13:         S(wCluster) = 0;
14:     else
15:         for m in macropoints do
16:             A(m) = computeA(m, wCluster)
17:             B(m) = computeB(m, wCluster, wClusters);
18:             Weighted silhouette of the macropoint:
19:             S(m) = computeS(A(m), B(m));
20:             clusterSumWeights += m.weight;
21:             clusterSumSilhouettes += S(m) * m.weight;
22:             datasetSumWeights += m.weight;
23:             datasetSumSilhouettes += S(m) * m.weight;
24:         Average weighted silhouette of the weighted cluster:
25:         S(wCluster) = clusterSumSilhouettes / clusterSumWeights;
26: Average weighted silhouette of the whole set of weighted macropoints:
27: SIL = datasetSumSilhouettes / datasetSumWeights;
        =0
```

Algorithm 6 computeA

Input: macropoint m and its weightedCluster *wCluster*
Output: $A(m)$

```
1: macropoints = wCluster.macropoints;
2: sumWeights = 0;
3: sumFactors = 0;
4: for macropoint in macropoints do
5:     sumWeights += macropoint.weight
6:     if NOT(macropoint.equals(m)) then
7:         factor = macropoint.weight*EuclideanDistance(m, macropoint);
8:         sumFactors += factor;
9: A(m) = sumFactors / (sumWeights - 1);
10: return A(m);
```

Algorithm 7 computeB

Input: macropoint m, its weighted cluster *wCluster_m*, all the weighted clusters *wClusters*
Output: $B(m)$

```
1: for wCluster_i in wClusters do
2:     if NOT(wCluster_i.equals(wCluster_m)) then
3:         sumWeights = 0;
4:         sumFactors = 0;
5:         macropoints = wCluster_i.macropoints;
6:         for macropoint_i in macropoints do
7:             sumWeights += macropoint_i.weight;
8:             factor = macropoint_i.weight*EuclideanDistance(m, macropoint_i);
9:             sumFactors += factor;
10:        dissimilarities.add(sumFactors / sumWeights);
11: B(m) = MIN(dissimilarities);
12: return B(m);
```

Algorithm 8 computeS

```
    Input: $A(m)$, $B(m)$
    Output: $S(m)$
1: if A(m) < B(m) then
2:     S(m) = 1 - A(m)/B(m);
3: else if A(m) > B(m) then
4:     S(m) = B(m)/A(m) - 1;
5: else
6:     S(m) = 0;
7: return S(m);
```

Weighted silhouette algorithm

Starting from the set of weighted macropoints already partitioned into weighted clusters *wClusters* (line 1), for each weighted cluster the weighted silhouette of all its macropoints and the average weighted silhouette of the cluster are computed as follows. Variables *datasetSumWeights* and *datasetSumSilhouettes* (lines 2 and 3) are used as accumulators for the sum of the weights of all macropoints and the sum of the weighted silhouettes of all macropoints. They will be used to compute the average weighted silhouette of the whole set as an average of the weighted silhouettes computed for all the macropoints. For each weighted cluster *wCluster*, the macropoints belonging to *wCluster* are retrieved (line 6). Variables *clusterSumWeights* and *clusterSumSilhouettes* (lines 7 and 8) are used as accumulators for the sum of the weights and the sum of the weighted silhouettes of *wCluster*'s macropoints. They will be used to compute the average weighted silhouette of *wCluster* as an average of the weighted silhouettes of all its macropoints. If the number of macropoints in *wCluster* is equal to 1, and the weight of this only macropoint m is equal to 1 (line 9), the weighted silhouette of *wCluster* is set to 0 without computing any other parameter (line 13). Else, for each macropoint m of the considered weighted cluster *wCluster* the values of A and B are retrieved, and the weighted silhouette S is computed (lines 16, 17 and 18, explanation in next algorithms). Given the weighted silhouettes of all the macropoints of the considered weighted cluster *wCluster*, the average weighted silhouette of *wCluster* is computed (line 25). Given the weighted silhouettes of all the macropoints of all clusters, the average weighted silhouette of the whole set is computed (line 27).

ComputeA algorithm Starting from a macropoint m and its weighted cluster *wCluster*, the value of A is computed for the macropoint m. The macropoints belonging to *wCluster* are retrieved (line 1). Variables *sumWeights* and *sumFactors* are used as accumulators for the sum of the weights of all the macropoints in *wCluster* and the sum of the weighted distances between m and all the other macropoints in *wCluster*. For each macropoint in *wCluster* different from the considered macropoint m (line 6) the weighted distance from m is computed using the Euclidean Distance (line 7). Given all these weighted distances the A parameter is computed as their weighted average (line 9).

ComputeB algorithm Starting from a macropoint m, its weighted cluster *wCluster_m* and the list of all weighted clusters *wClusters*, the value of B is computed for m. For each weighted cluster in the list, if it is different from *wCluster_m* (line 2), the weighted distances between m and all the macropoints of the weighted cluster are computed. Variables *sumWeights* and *sumFactors* (lines 3 and 4) are used, for each weighted cluster different from *wCluster_m*, as accumulators for the sum of the weights of all the macropoints in the cluster and the sum of the weighted distances between m and all the macropoints in the other cluster. The macropoints belonging to the cluster are retrieved (line 5), and for each one of them the weighted distance from m is computed using the Euclidean Distance (line 8). The dissimilarity between m and the macropoints of the cluster is retrieved as the weighted average of the computed weighted distances (line 10). Given the dissimilarities between the considered macropoint m and all the weighted clusters different from its cluster, the value of B is the minimum of these dissimilarities (line 11).

ComputeS algorithm Starting from values of A and B for a macropoint m, the value of the silhouette S is computed for m following the definition:

$$S(m) = \begin{cases} 1 - A(m)/B(m), & \text{if } A(m) < B(m) \\ 0, & \text{if } A(m) = B(m) \\ 1 - B(m)/A(m), & \text{if } A(m) > B(m) \end{cases} \tag{3.11}$$

3.2.2 Weighted silhouette example

Let us consider again the clustered objects used for the weighted clustering example, displayed in Figure 3.1. The weighted macropoints that represent the original objects, partitioned in two clusters, are displayed in Table 3.6 and in Figure 3.5.

	m_x	m_y	*weight*	*weighted cluster*
M_1	1.67	1.67	3	1
M_2	2	3	1	1
M_3	3	3	1	1
M_4	3	5	1	1
M_5	5.67	1.67	3	2
M_6	5.5	5	2	2
M_7	6	3	1	2

Table 3.6: Set of weighted macropoints divided in two weighted clusters: macropoint (m_x, m_y) with weight *weight* belongs to the weighted cluster *weighted cluster*

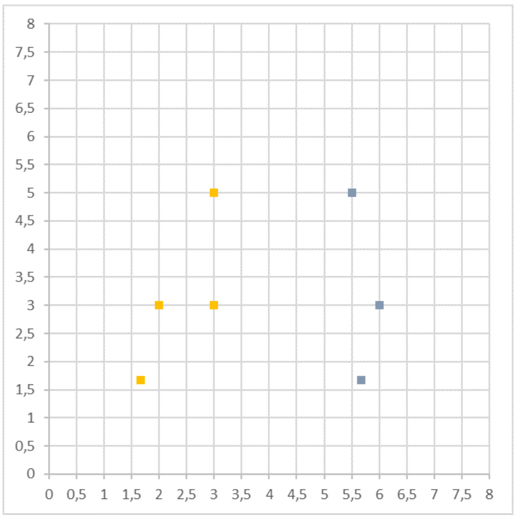

Figure 3.5: Set of weighted macropoints divided in weighted cluster 1 (yellow points) and 2 (light blue points)

The weighted silhouette computed on these weighted clusters should have the same value as the silhouette computed on the clusters of standard objects (i.e. objects without weights) obtained as follows: for each weighted macropoint M_i with coordinates (m_x, m_y) and weight w_i, a number equal to w_i of standard objects should be considered, all of them with coordinates $(m_x,\ m_y)$. Table 3.7 lists these correspondent points.

	p_x	p_y	*cluster*
P_1	1.67	1.67	1
P_2	1.67	1.67	1
P_3	1.67	1.67	1
P_4	2	3	1
P_5	3	3	1
P_6	3	5	1
P_7	5.67	1.67	2
P_8	5.67	1.67	2
P_9	5.67	1.67	2
P_{10}	5.5	5	2
P_{11}	5.5	5	2
P_{12}	6	3	2

Table 3.7: Standard points corresponding to the macropoints

As described in Section 2.1.1, the silhouette on the standard clusters is computed for each point P_i as follows:

$$a\left(P_i\right) = \frac{1}{N_k - 1} \sum_{P_k \in C_k} d\left(P_i, P_k\right)$$

$$b\left(P_i\right) = \min_{l=1, l\neq k}^{K} \frac{1}{N_l} \sum_{P_l \in C_l} d\left(P_i, P_l\right)$$

$$S\left(P_i\right) = \frac{b(P_i) - a(P_i)}{\max(a(P_i), b(P_i))}$$

While the silhouette of the whole data set, which evaluates the quality of the entire clustering is computed as:

$$SIL = \frac{1}{N_D} \sum_{P_i \in D} S(P_i)$$

In Table 3.8 the silhouette values computed for all the standard points are listed.

	p_x	p_y	*cluster*	*silhouette*
P_1	1.67	1.67	1	0.6924893048173986
P_2	1.67	1.67	1	0.6924893048173986
P_3	1.67	1.67	1	0.6924893048173986
P_4	2	3	1	0.629138760543797
P_5	3	3	1	0.4348668651717875
P_6	3	5	1	0.15965091377204932
P_7	5.67	1.67	2	0.5833763013920013
P_8	5.67	1.67	2	0.5833763013920013
P_9	5.67	1.67	2	0.5833763013920013
P_{10}	5.5	5	2	0.41993424213309505
P_{11}	5.5	5	2	0.41993424213309505
P_{12}	6	3	2	0.5916055721594646

Table 3.8: Values of the silhouettes computed for the standard points: point (p_x, p_y) belonging to cluster *cluster* has a silhoutte index of *silhouette*

The resultant silhouettes is the average between all the silhouettes of single points: $SIL = 0.540227284545124$.

The weighted silhouette is computed for each weighted macropoint M_i as described in Section 3.2:

$$a(M_i) = \frac{1}{W_k - 1} \sum_{M_k \in C_k} w_k \cdot d(M_i, M_k)$$
$$b(M_i) = \min_{l=1, l \neq k}^{K} \frac{1}{W_l} \sum_{M_l \in C_l} w_l \cdot d(M_i, M_l)$$
$$S(M_i) = \frac{b(M_i) - a(M_i)}{\max(a(M_i), b(M_i))}$$

The weighted silhouette of the whole set of weighted macropoints, which evaluates the quality of the weighted clustering is:

$$SIL_{weighted} = \frac{1}{N_D} \sum_{M_i \in D} S(M_i)$$

The weighted silhouette values of weighted macropoints are listed in the following table.

	m_x	m_y	*weight*	*weighted cluster*	*weighted silhouette*
M_1	1.67	1.67	3	1	0.6924893048173986
M_2	2	3	1	1	0.629138760543797
M_3	3	3	1	1	0.4348668651717875
M_4	3	5	1	1	0.15965091377204932
M_5	5.67	1.67	3	2	0.5833763013920013
M_6	5.5	5	2	2	0.41993424213309505
M_7	6	3	1	2	0.5916055721594645

Table 3.9: Values of the weighted silhouettes computed for the weighted macropoints: macropoint (m_x, m_y) belonging to weighted cluster *weighted cluster* has a weighted silhoutte index of *weighted silhouette*

The resultant weighted silhouette is the average between all the weighted silhouettes of the macropoints: $wSIL = 0.5402272845451239$.

As desired, $wSIL = SIL$.

3.3 Weighted clustering modification

The weighted silhouette results obtained were in some cases not as good as expected. Observations about the standard deviation of points in cells, that represents their dispersion, lead to the conclusion that these bad results were caused by too dense macropoints. In case of high values of *step*, the obtained cells are large and macropoints weights are high. As shown in the example in Section 3.2.2, a macropoint $M(m_x, m_y)$ with weight w_M corresponds exactly, in the weighted silhouette computation, to a number w_M of standard points (i.e. not weighted points or points with weight $= 1$) with coordinates (m_x, m_y). If a cell is large and contains a lot of scattered points, the representation of all of them in one single point, the macropoint $M(m_x, m_y)$, is too dense. This leads to values of weighted silhouette higher than the silhouette computed on starting points, and for big values of *step* the distance between the weighted silhouette and the desired silhouette is not acceptable. In order to solve this issue, a secondary partition using a lower value of *step* is applied on cells with high standard deviation and high cardinality.

In order to select the cells to be divided a second time, a percentage value of the standard deviation of points in the cell is considered. For each cell, during the computation of its macropoint, the standard deviation of represented points is retrieved on each dimension. An array with the standard deviation values computed on all the coordinates is associated to each macropoint and it represents the amount of variation (dispersion) of the set of points represented by the macropoint. For each macropoint also the minimum and the maximum values on all the coordinates are computed and used to obtain the range of variation. This range is then used to calculate the percentage value of the standard deviation, that is finally used to decide if a cell has to be splitted or not.

The process of selection and split of the cells with an high percentage of standard deviation is the following. Let M be the macropoint that represents a cell C. The cell C contains the points $\{P_1, P_2, ..., P_i, ..., P_{N_C}\}$, where N_C is the number of points in cell C, and P_i is the i^{th} point belonging to cell C. The points represented by the macropoint have N dimensions corresponding to the considered features of initial data, so a generic point is defined as $P_i(p_{i1}, p_{i2}, ..., p_{ij}, ..., p_{iN})$, with $j = 1, 2, ..., N$, where N is the number of coordinates and $\mathrm{p_{ij}}$ is the value of the of point P_i on the j^{th} coordinate. A macropoint M represents a certain number of points and have the same N dimensions of the represented points, with the additional information of the weight. It is defined as $M(m_1, m_2, ..., m_j, ..., m_\mathrm{N})$, with $j = 1, 2, ..., N$, where N is the number of coordinates and $\mathrm{m_j}$ is the value of the macropoint M on the

j^{th} coordinate. The weight of M is w_M and it is set to the number of represented original points.

Three vectors need to be defined for each cell C: one for the minimums, one for the maximums and one for the standard deviations. Then a vector with the ranges of variations on each dimension is derived from the minimums and maximums.
Let $MIN = [min_1, min_2, ..., min_j, ..., min_N]$ be the vector with the minimums for cell C. The value of min_j is computed on each dimension $j = 1, 2, ..., N$ as:

$$min_j = \min_{P_i \in C} \; p_{ij} \tag{3.12}$$

where p_{ij} is the value of point $P_i \in C$ on the j^{th} coordinate, and C is the considered cell. The same is the defined for the vector of the maximums of cell C on each dimensions: $MAX = [max_1, max_2, ..., max_j, ..., max_N]$. The value of max_j is computed for each dimension j as:

$$max_j = \max_{P_i \in C} \; p_{ij} \tag{3.13}$$

The range of variation defined as $RANGE = [range_1, range_2, ..., range_j, ..., range_N]$ is obtained on each dimension j as:

$$range_j = max_j - min_j \tag{3.14}$$

For the standard deviations vector $STDDEV = [stddev_1, stddev_2, ..., stddev_j, ..., stddev_N]$, first the value of the arithmetic average $mean_j$ is computed on each dimension $j = 1, 2, ..., N$:

$$mean_j = \frac{1}{N_C} \sum_{P_i \in C} p_{ij} \tag{3.15}$$

where N_C is the number of points in cell C, p_{ij} is the j^{th} coordinate of point $P_i \in C$, and C is the considered cell.
Then, the variance $variance_j$ is obtained for each dimension j as:

$$variance_j = \frac{1}{N_C} \sum_{P_i \in C} (p_{ij} - mean_j)^2 \tag{3.16}$$

where $mean_j$ is the mean computed for cell C on the j^{th} dimension. Finally, the standard deviations vector for cell C, defined as $STDDEV = [stddev_1, stddev_2, ..., stddev_j, ..., stddev_N]$, is retrieved on each dimension j as:

$$stddev_j = \sqrt{variance_j} \tag{3.17}$$

where $variance_j$ is the variance computed on the j^{th} coordinate for the considered cell C. For each cell, the value of the range of variation $range_j$ and the value of the standard deviation $stddev_j$ have been computed on each dimension $j = 1, 2, ..., N$. A vector PERCSTDDEV = $[percStddev_1, percStddev_2, ..., percStddev_j, ..., percStddev_N]$ is used to contain the percentages of the standard deviation compared to the range of variation. The $percStddev_j$ is defined on each dimension j as:

$$percStddev_j = \frac{stddev_j \cdot 100}{range_j} \tag{3.18}$$

Given this value, the idea is to select all the cells with at least one value of $percStddev_j$ greater than a certain threshold *thPercStddev*. These cells are defined *cells to be splitted*.

The value of *thPercStddev* can be set according to the desired precision. The application of the algorithm with a low value of threshold will lead to a more precise representation of the original clustering (i.e. the weighted silhouette computed on it will be closer to the desired value). A trade-off between the quality of the obtained result and the reduction of the cardinality has to be considered. If the value of the threshold *thPercStddev* is low, a lot of cells will be splitted granting a weighted clustering more similar to the original clustering; in this case a large number of macropoints will be computed and so the representation will be more precise, but the reduction of the cardinality will be lower. On the other side, if the value of *thPercStddev* is high, less cells will be splitted granting a lower number of macropoints, improving the reduction of the cardinality but worsening the precision. A more precise discussion about the choice of the threshold can be found in a dedicated section (4.4.2).

The selection of all the cells with a value of $percStddev_j$ greater than the threshold *thPercStddev* for at least one dimension j, as shown in the next sections, led to the split of some unnecessary cells. As said before, the problematic cells are the ones with a large size and a lot of scattered points inside. The dispersion of points has been measured with the standard deviation. When a cell contains a low number of points, even if the dispersion of its points is high (i.e. the percentage of standard deviation computed for the cell is greater than the threshold), the split is not necessary. To solve this issue another check is performed on each cell in order to decide if it has to be splitted. The percentage of represented points of the cell compared to the number of points of its cluster is computed for each cell (note that thisi is done once for each cell, not for each dimension of the cell). This percentage for a cell C is defined as $percRepresentedPoints_C$:

$$percRepresentedPoints_C = \frac{N_C \cdot 100}{N_k} \tag{3.19}$$

where N_C is the number of points in the cell C, i.e. the number of points represented by the macropoint M associated to cell C, also equal to the *weight* of M; N_k is the number of points contained in the original cluster C_k, which is the clusters the cell belongs to. When $percRepresentedPoints_C$ is greater to a certain threshold *thPercRepresentedPoints*, the cell C is selected. The value of the threshold *thPercRepresentedPoints* has to be set in a way to grant that cells containing a low number of points are not splitted. A more precise discussion about the choice of this threshold will be presented in the section dedicated to the thresholds.

The conditions on *percStddev* and on *percRepresentedPoints* are used together. Given the thresholds *thPercStddev* and *thPercRepresentedPoints* the condition to choose if a cell C has to be splitted or not is the following:

```
1: N = number of dimensions;
2: toBeSplitted_stddev = FALSE;
3: toBeSplitted = FALSE;
4: for j ← 0 to N-1 do
5:     if $percStddev_j$ > thPercStddev then
6:         toBeSplitted_stddev = TRUE;
7:         break;
8: if toBeSplitted_stddev = TRUE then
9:     if $percRepresentedPoints_C$ > thPercRepresentedPoints then
10:        toBeSplitted = TRUE;
```

Once selected the *cells to be splitted*, the points inside them are reassigned to new smaller cells. The original cells had size *step* on each dimension. The new cells need a size proportional to the previous one and to the standard deviation. The standard deviation of a cell represents the amount of variation of the set of points contained in the cell. The more this value is high, the more the new sub-cells need to be small. An array with the standard deviation values computed on all the coordinates is associated to each macropoint and represents the dispersion of the points represented by the macropoint. All the point belonging to a cell selected to be splitted are reassigned to new cells with a size equal to *newStep*, with $newStep < step$. In order to define the *newStep* value, first the maximum value of standard deviation is computed for each cell among all dimensions $j = 1, 2, ..., N$:

$$maxPercStddev_C = \max_{j=1}^{N} \; percStddev_j \tag{3.20}$$

where N is the number of dimensions and $percStddev_j$ is the value of percentage standard deviation computed for the considered cell C on dimension j.
Given this $maxPercStddev_C$ value, a *newStep* is computed for a cell $C \in cellsToBeSplitted$ as $newStep_C$:

$$newStep_C = step - \frac{\text{step} \cdot maxPercStddev_C}{100} \tag{3.21}$$

Note that the value $newStep_C$ is computed for a single cell C and so different cells have different values of *newStep* based on the dispersion of their points, represented by their percentage of standard deviation. In this way cells with a lot of scattered points are divided into smaller new cells (i.e. the higher is the maximum percentage of standard deviation of C, the smaller is the value of $newStep_C$).

Considering one cluster at a time, the points belonging to the *cellsToBeSplitted* selected for the cluster are reassigned to new smaller cells using the *newStep* value. The points belonging to the other cells remain assigned to them. The reassignment of points is performed in the following way. Let C_k be the considered cluster of points. The set of points belonging to C_k and to all the cells of C_k selected as to be splitted is called *points to be remapped*. All points in this set will be partitioned into new smaller cells. The starting point of this partition into new cells is the minimum of the set *points to be remapped* computed on each dimension $j = 1, 2, ..., N$. The minimum value computed on the j^{th} coordinate among all the points to be remapped of the considered cluster C_K is $newMin_j$:

$$newMin_j = \min_{\substack{C \in cellsToBeSplitted \\ P_i \in C}} \; p_{ij} \tag{3.22}$$

where $C \in cellsToBeSplitted$ indicates cells in cluster C_k selected to be splitted; $P_i \in C$ indicates points belonging to cells in cluster C_k selected to be splitted; p_{ij} is the value of the j^{th} coordinate of point P_i.

A point $P_i(p_{i1}, p_{i2}, ..., p_{ij}, ..., p_{iN})$ belonging to the considered cluster C_K and to the cell to be splitted C, is assigned to the new cell identified by the point $newC(newc_1, newc_2, ..., newc_j, ..., newc_N)$, where $newc_j$ is computed for each dimension j = 1, 2, ..., N as:

$$newc_j = Floor\left(\frac{p_{ij} - newMin_j}{newStep_C}\right) \cdot newStep_C + newMin_j \tag{3.23}$$

where p_{ij} is the j^{th} coordinate of point P_i, $newStep_C$ is the value of *newStep* computed for cell C (Equation 4.7), and $newMin_j$ is the minimum value computed on the j^{th} coordinate among all the *points to be remapped* of the considered cluster C_K (Equation 3.22). In this way each new cell, as for old cells, is identified by a point which contains on each coordinate the minimum value of the cell's points computed on that coordinate (Equation 3.23).

After assigning each point to a new cell, the new representative points (macropoints) are computed. For each new cell the new macropoint *newM* is defined, as for previous cells, as the average value on each dimension computed on all the points belonging to cell *newC*. Let N_{newC} be the number of points in cell *newC*. All these points are represented by the macropoint $newM(newm_1, newm_2, ..., newm_j, ..., newm_N)$, where $newm_j$ is computed for each dimension j as:

$$newm_j = \frac{1}{N_{newC}} \sum_{P_i \in newC} p_{ij} \tag{3.24}$$

where $P_i \in newC$ indicates a point P_i belonging to the new cell identified by the point *newC*, and p_{ij} is the j^{th} coordinate of point P_i. Being N_{newC} the number of points in cell *newC* represented by the macropoint *newM*, the weight of *newM* is set to N_{newC}. The macropoint *newM* remains assigned to the cluster C_k.

The new clusters contain all the macropoints of the old cells not selected to be splitted, and all the new macropoints of the new cells obtained from the points to be remapped. In this way, the points belonging to cluster C_k that where assigned to one of the cells selected to be splitted have been divided in a number of new cells greater than the number of old cells. The more the points of the old cells were dispersed, the more the new cells are small. The result is that for critical areas the number of macropoints is higher and the weighted clustering is more precise. It is clear that applying this modification, the cardinality of the new weighted clusters is higher than the cardinality obtained without the secondary split. As demonstrate in the next sections, a little increase in the number of macropoints allows a considerable improvement in the weighted silhouette results for the new weighted clusters.

3.3.1 Weighted clustering modification pseudocode

Algorithm 9 Weighted Clustering Modification

Input: the original clustering: the set *clusters* which contains clusters of points; the value of *step*

Output: the result of weighted clustering process: the set *weightedClusters* which contains clusters of weighted macropoints

```
 1: clusters;                                                  ▷ Clusters of points
 2: weightedClusters = {};                       ▷ Set of computed weighted clusters
 3: For each cluster, compute the cells, the weighted macropoints, and the weighted
    cluster corresponding to the considered cluster:
 4: for cluster in clusters do
 5:     weightedCluster = {};     ▷ Weighted cluster corresponding to the cluster
 6:     clusterPoints = cluster.points;
 7:     Compute the minimum value between cluster points on each dimension:
 8:     min = computeMin(clusterPoints);
 9:     step = args.step;                    ▷ Step from command line arguments
10:     For each point of the cluster, compute the cell:
11:     for point in clusterPoints do
12:         Compute the value of the cell for the point on each coordinate:
13:         cell = computeCell(point, min, step);
14:         Add the point to the set of points belonging to the computed cell:
15:         cell.points.add(point)
16:     cells;                                              ▷ All computed cells
17:     For each cell, retrieve the value of the macropoint on each dimension and
    set its weight:
18:     for cell in cells do
19:         cellPoints = cell.points;
20:         cell.macropoint = computeAverage(cellPoints);
21:         cell.macropoint.weight = cellPoints.size;
22:     For each cell, compute the percStddev on each dimension and the
    percRepresentedPoints:
23:     for cell in cells do
```

```
24:         Compute the minimum and the maximum values on each dimension:
25:         cellPoints = cell.points;
26:         cell.min = computeMin(cellPoints);
27:         cell.max = computeMax(cellPoints);
28:         Retrieve the range of variation:
29:         cell.range = computeRange(cell.min, cell.max);
30:         Compute the standard deviation values on each dimension:
31:         cellStddev = computeStddev(cellPoints);
32:         Compute the percentage of standard deviation on each dimension:
33:         cell.percStddev = computePercStddev(cell.stddev, cell.range);
34:         Compute the percentage of represented points of the cell:
35:         cell.percRepresentedPoints = cellPoints.size*100/clusterPoints.size;
36:     Retrieved the thresholds values from command line arguments:
37:     thPercStddev = args.thPercStddev;
38:     thPercRepresentedPoints = args.thPercRepresentedPoints;
39:     Select the cells to be splitted:
40:     for cell in cells do
41:         if isCellToBeSplitted(cell.percStddev, thPercStddev,
    cell.percRepresentedPoints, thPercRepresentedPoints) == TRUE then
42:             cellsToBeSplitted.add(cell)
43:         else
44:             cellsNotToBeSplitted.add(cell)
45:     Compute the minimum on each dimension among all the points in cells to
    be splitted:
46:     pointsToBeRemapped = cellsToBeSplitted.points;
47:     newMin = computeMin(pointsToBeRemapped);
48:     Compute the newStep for each cell to be splitted:
49:     for cell in cellsToBeSplitted do
50:         cell.newStep = computeNewStep(step, cell.percStddev);
51:     For each point to be remapped, compute the new cell:
52:     for point in pointsToBeRemapped do
53:         oldCell;                ▷ Old cell to which the point previously belonged to
54:         newCell = computeCell(point, newMin, oldCell.newStep);
55:         newCell.points.add(point);
56:     newCells;                                        ▷ All new computed cells
57:     For each newCell, retrieve the value of the new macropoint on each
```

```
    *dimension and set its weight:*
58:     for newCell in newCells do
59:         newCellPoints = newCell.points;
60:         newCell.macropoint = computeAverage(newCellPoints);
61:         newCell.macropoint.weight = newCellPoints.size;
62:     *Add the old and the new macropoints to the weighted cluster:*
63:     for cell in cellsNotToBeSplitted do
64:         weightedCluster.add(cell.macropoint);
65:     for newCell in newCells do
66:         weightedCluster.add(newCell.macropoint);
67:     *Add the computed weighted cluster to the resultant weightedClusters:*
68:     weightedClusters.add(weightedCluster);
69: return weightedClusters;
```

Algorithm 10 computeMax

Input: original points of a cell: *cellPoints*
Output: maximum value *max* among all *cellPoints* computed on each dimension

```
1: N = number of dimensions;
2: for i ← 0 to N-1 do
3:     $max_i$ = MINVALUE;
4: for point in cellPoints do
5:     for i ← 0 to N-1 do
6:         if $point_i > max_i$ then
7:             $max_i = point_i$
8: return max;
```

Algorithm 11 computeRange

Input: *min* and *max* of the cell on each dimension
Output: *range* of variation of the cell on each dimension

```
1: N = number of dimensions;
2: for i ← 0 to N-1 do
```

```
3:     $range_i$ = $max_i$ - $min_i$;
4: return range;
```

Algorithm 12 computeStddev

Input: original points of a cell *cellPoints*

Output: standard deviation value *stddev* between all *cellPoints* computed on each dimension

```
1: N = number of dimensions;
2: Arithmetic mean between all the points of the cell on each dimension:
3: mean = computeAverage(cellPoints);
4: for i ← 0 to N-1 do
5:     for point in cellPoints do
6:         $factor(point)_i = (mean_i - point_i)^2$
7:         $factors_i$.add($factor(point)_i$);
8:     $variance_i$ = average($factors_i$)
9:     $stddev_i$ = sqrt($variance_i$)
10: return stddev;
```

Algorithm 13 computePercStddev

Input: standard deviation *cellStddev* and range *cellRange* on each dimension for a cell

Output: *cellPercStddev* on each dimension for the cell

```
1: N = number of dimensions;
2: for i ← 0 to N-1 do
3:     $cellPercStddev_i$ = $cellStddev_i$ * 100 / $cellRange_i$;
4: return cellPercStddev;
```

Algorithm 14 isCellToBeSplitted

Input: data about a cell: *percStddev* on each dimension and *percRepresentedPoints*; thresholds: *thPercStddev* and *thPercRepresentedPoints*

Output: *toBeSplitted* = TRUE if the cell has to be splitted, FALSE if not

```
1: toBeSplitted_stddev = FALSE;
2: toBeSplitted = FALSE;
3: N = number of dimensions;
4: for i ← 0 to N-1 do
5:     if $\text{percStddev}_j$ > thPercStddev then
6:         toBeSplitted_stddev = TRUE;
7:         break;
8: if toBeSplitted_stddev = TRUE then
9:     if percRepresentedPoints > thPercRepresentedPoints then
10:        toBeSplitted = TRUE;
11: return toBeSplitted;
```

Algorithm 15 computeNewStep

```
    Input: The old step, the cellPercStddev of the cell on each dimension
    Output: newStep value computed for the cell
1: maxCellPercStddev = maximum(cellPercStddev); ▷ Maximum value among all the dimensions
2: newStep = step-(step*maxCellPercStddev/100);
3: return newStep;
```

Weighted Clustering Modification algorithm

The weighted clustering process starts from a set of points already partitioned into clusters: *clusters* (line 1). For each cluster, the correspondent weighted cluster *weightedCluster* is retrieved as follows. The original points of the considered cluster are extracted (line 6), and the minimum among all these points is computed on each dimension (line 8) using the *computeMin* function (Algorithm 2 in Section 3.1.1). In line 9 the *step* value is retrieved from command line arguments. For each point of the considered cluster, the identifier of the cell the point has to be assigned to, is computed on each dimension (line 13) with the logic presented in function *computeCell* (Algorithm 3 in Section 3.1.1). Given the values of the identifier of the cell on all dimensions, the considered point is added to the set of points belonging to that cell (line 15). At the end of this process, cells containing original points have been obtained (line 16). For each one of these cells, the coordinates of the correspondent macropoint are computed as the average of the values of all the points belonging to the cell on that coordinate (line 20) using function *computeAverage* (Algorithm 4 in 3.1.1). The weight of the macropoint is set to the number of points belonging to the cell represented by the macropoint (line 21).

From here, the modification is applied. For each cell, the *percStddev* and the *percRepresentedPoints* are computed. To do that, the minimum, the maximum, the range and the standard deviation are computed among the points of the cell on each dimension (lines 26-31) using the corresponding functions: *computeMin* (Algorithm 2), *computeMax* (10), *computeRange* (11) and *computeStddev* (12). Given these values, the percentage of standard deviation compared to the range of variation of points in the cell *percStddev* is computed on each dimension (line 33, Algorithm 13). The percentage of represented points compared to the number of points in the cluster *percRepresentedPoints* is computed for the cell (line 35). Retrieved these percentages, they are used to decide for each cell if it has to be splitted. The thresholds *thPercStddev* and *thPercRepresentedPoints* are retrieved from command line arguments (lines 37 and 38). Given these values, for each cell it is chosen if it has to be splitted or not (line 41) with the logic presented in Algorithm 14. If yes, it is added to the *cellsToBeSplitted* set, else it is added to the *cellsNotToBeSplitted* set. All the points belonging to all the cells selected as to be splitted are put in the set *pointsToBeRemapped* (line 46). For these points the remapping process is applied as following. The minimum *newMin* among all these points is computed on each dimension (line 47, Algorithm 2). The value of the *newStep* is computed for each cell (line 50, Algorithm 15). Each point selected as point to be remapped is assigned to a *newCell* using the *newStep* of its old cell and starting from the *newMin* (line 54) computed with Algorithm 3. The point is then added to the set of points assigned to the *newCell* (line 55). The set of all the computed new cells is defined *newCells* (line 56). For each one of these cells, the coordinates of the correspondent macropoint are computed as the average of the values of all the points belonging to the cell on that coordinate (line 60). The average is retrieved with Algorithm 4. The weight of the macropoint is set to the number of points belonging to the cell represented by the macropoint (line 61).

Finally the macropoints of the old cells selected as *cellsNotToBeSplitted* and the new macropoints of the *newCells* computed for the current *cluster* are added to the correspondent *weightedCluster* (line 64 and 66), and the *weightedCluster* is added to the list of *weightedClusters* (line 68). The list of *weightedClusters* is the result of the weighted clustering process.

3.3.2 Weighted clustering modification example

A graphical explanation of how points are assigned to cells has been previously presented (in Section 3.1.2). During that process, the points already divided into clusters have been divided into cells. Each cell has its representative macropoint that represents all the points contained in the cell. This resultant condition is shown in Table 3.10 and in Figure 3.6, considering only cluster 1.

	p_x	p_y	*cluster*	c_x	c_x	m_x	m_y	*weight*
P_1	2	1	1	1	1	1.67	1.67	3
P_2	1	2	1	1	1	1.67	1.67	3
P_3	2	2	1	1	1	1.67	1.67	3
P_4	2	3	1	1	2.5	2	3	1
P_5	3	3	1	2.5	2.5	3	3	1
P_6	3	5	1	2.5	4	3	5	1

Table 3.10: Cells and macropoints for cluster 1: point (p_x, p_y) is assigned to the cell identified by (c_x, c_y), and represented by the macropoint (m_x, m_y) with weight *weight*

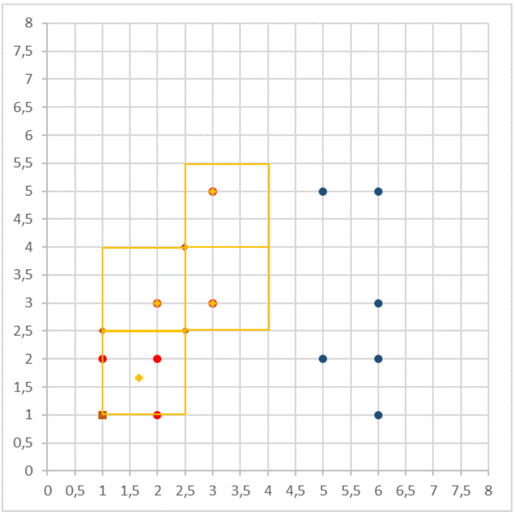

Figure 3.6: Cells and macropoints for cluster 1

In order to apply the modified weighted clustering described in Section 3.3, the minimums, maximums, ranges and standard deviations have to be computed for each cell on each dimension.

In the following lines an example of computation of these values is presented. Let us consider the cell $C_1(c_x,\ c_y) = (1.0,\ 1.0)$. This cell contains the points $P_1 = (2.0,\ 1.0)$, $P_2 = (1.0,\ 2.0)$ and $P_3 = (2.0,\ 2.0)$. The dimensions are x and y, and the minimum is computed on each dimension as:

$$min_x = \min_{P_i \in C_1} p_{ix} = min(2.0,\ 1.0,\ 2.0) = 1.0$$
$$min_y = \min_{P_i \in C_1} p_{iy} = min(1.0,\ 2.0,\ 2.0) = 1.0$$

The maximum is computed on each dimension as:

$$max_x = \max_{P_i \in C_1} p_{ix} = max(2.0,\ 1.0,\ 2.0) = 2.0$$
$$max_y = \max_{P_i \in C_1} p_{iy} = max(1.0,\ 2.0,\ 2.0) = 2.0$$

The range of variation on each coordinate is:

$$range_x = max_x - min_x = 2.0 \text{ - } 1.0 = 1.0$$
$$range_y = max_y - min_y = 2.0 \text{ - } 1.0 = 1.0$$

For the standard deviation, first the arithmetic average is calculated on each dimension:

$$mean_x = \frac{1}{N_{C_1}} \sum_{P_i \in C_1} p_{ix} = \frac{1}{3}\,(2.0 + 1.0 + 2.0) = 1.67$$
$$mean_y = \frac{1}{\mathrm{C}} \sum_{P_i \in C} p_{iy} = \frac{1}{3}\,(1.0 + 2.0 + 2.0) = 1.67$$

Then the variance is computed on each coordinate:

$$variance_x = \frac{1}{N_{C_1}} \sum_{P_i \in C_1} (p_{ix} - mean_x)^2 = \frac{1}{3}\left((2.0 - 1.67)^2 + (1.0 - 1.67)^2 + (2.0 - 1.67)^2\right)$$
$$= 0.22$$
$$variance_y = \frac{1}{N_{C_1}} \sum_{P_i \in C_1} (p_{iy} - mean_y)^2 = \frac{1}{3}\left((1.0 - 1.67)^2 + (2.0 - 1.67)^2 + (2.0 - 1.67)^2\right)$$

$= 0.22$

Finally the standard deviation is retrieved:

$stddev_x = \sqrt{variance_x} = \sqrt{0.22} = 0.47$
$stddev_y = \sqrt{variance_y} = \sqrt{0.22} = 0.47$

Values of minimums, maximums, ranges and standard deviations on each dimensions for cells of cluster 1 are reported in the following table.

	c_x	c_y	*cluster*	min_x	min_y	max_x	max_y	$range_x$	$range_y$	$stddev_x$	$stddev_y$
C_1	1	1	1	1	1	2	2	1	1	0.47	0.47
C_2	1	2.5	1	2	3	2	3	0	0	0	0
C_3	2.5	2.5	1	3	4	3	4	0	0	0	0
C_4	2.5	4	1	3	5	3	5	0	0	0	0

Table 3.11: Minimums, maximums, ranges and standard deviations on each dimension computed for cluster 1 cells

The percentage of standard deviation compared to the range for cell *C₁* is retrieved on each dimension:

$percStddev_x = \frac{stddev_x \cdot 100}{range_x} = \frac{0.47 \cdot 100}{1.0} = 47.0\%$
$percStddev_y = \frac{stddev_y \cdot 100}{range_y} = \frac{0.47 \cdot 100}{1.0} = 47.0\%$

The maximum percentage of standard deviation $maxPercStddev_{C_1}$ for the cell C_1 is 47.0%

Let the threshold *thPercStddev* be equal to 15.0. For simplicity, the *thPercRepresentedPoints* will not be considered in this example. The only cell selected to be splitted is C_1, since the *maxPercStddev* for the other cells are equal to 0. The points to be remapped are all the points in C_1: P_1, P_2 and P_3.

The minimum of the set *pointsToBeRemapped*, which is the starting point for the division in new cells, is retrieved on each coordinate:

$$newmin_x = \min_{P_i \in pointsToBeRemapped} p_{ix} = min(2.0,\ 1.0,\ 2.0) = 1.0$$
$$newmin_y = \min_{P_i \in pointsToBeRemapped} p_{iy} = min(1.0,\ 2.0,\ 2.0) = 1.0$$

Table 3.12 and Figure 3.7 show the minimum computed on each dimension among all the points to be remapped.

	$newmin_x$	$newmin_y$	*cluster*
newMIN	1	1	1

Table 3.12: Minimum on each dimension among points of the set *pointsToBeRemapped* selected for cluster 1

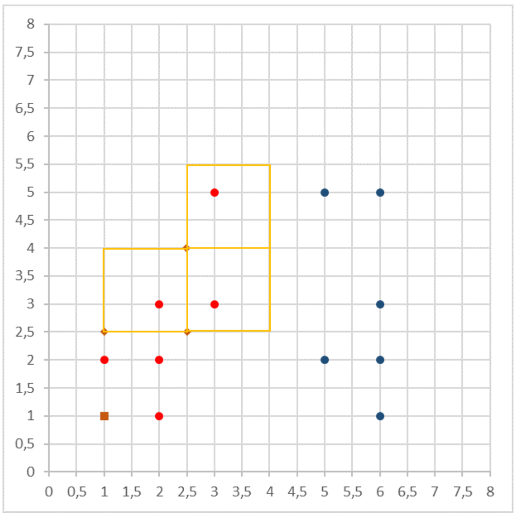

Figure 3.7: Minimum among points of the set *pointsToBeRemapped* selected for cluster 1 (brown point)

The *newStep* value is computed for the cell C_1 as:

$newStep_{C_1} = step - \frac{\text{step}\cdot maxPercStddev_{C1}}{100} = 1.5 - \frac{1.5\cdot 47.0}{100} = 0.79$

Given this value each point to be remapped is assigned to a new cell with the same process used for original weighted clustering.

An example of assignment of a point to a new cell is reported in the following lines. Let us consider the point $P_1(p_{1x},\ p_{1y}) = (2.0,\ 1.0)$. The value of $newStep_{C_1}$ is 0.79. Point P_1 belongs to the set *pointsToBeRemapped*, minimum of this set is $newMIN(newmin_x,\ newmin_y) = (1.0,\ 1.0)$.
Point P_1 is assigned to the new cell $newC(newc_x,\ newc_y)$, computed as:

$newc_x = Floor((p_{1x} - newMin_x)/newStep_{C_1}) \cdot newStep_{C_1} + newMin_x = Floor((2.0 - 1.0)/0.79) \cdot 0.79 + 1.0 = 1.79$
$newc_y = Floor((p_{1y} - newMin_y)/newStep_{C_1}) \cdot newStep_{C_1} + newMin_y = Floor((1.0 - 1.0)/0.79) \cdot 0.79 + 1.0 = 1.0$

So point P_1 is reassigned to cell $newC(newc_x,\ newc_y) = (1.79,\ 1.0)$.

The division in new cells using $newStep_{C_1}$ and the assignment of each point to the new computed cell are shown in Table 3.13 and in Figure 3.8.

	p_x	p_y	*cluster*	$newmin_x$	$newmin_y$	$newc_x$	$newc_x$
P_1	2	1	1	1	1	1.79	1.0
P_2	1	2	1	1	1	1.0	1.79
P_3	2	2	1	1	1	1.79	1.79

Table 3.13: New cells: point $(p_x,\ p_y)$ is reassigned to the cell identified by $(newc_x,\ newc_y)$

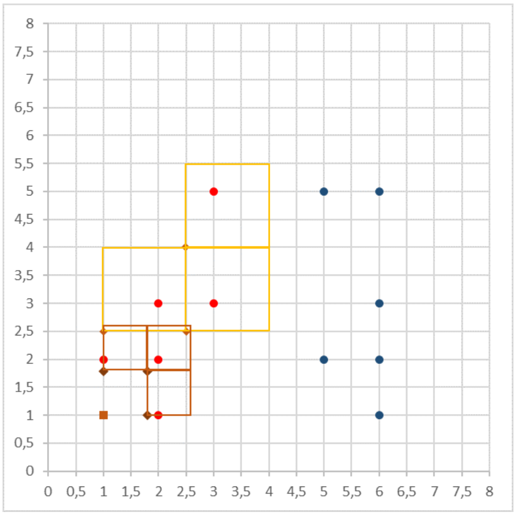

Figure 3.8: Partition of points to be remapped into new cells

Finally, the macropoints are computed for the new cells.

Example of macropoint computation: let us consider the new cell $newC(newc_x, newc_y) = (1.79, 1.0)$. Point $P_1 = (2.0, 1.0)$ is the only one assigned to cell $newC$, so the macropoint associated to $newC$ must represent only point P_1. The new macropoint $newM$ for cell $newC$ is calculated on each dimension as:

$$newm_x = \frac{1}{N_{newC}} \sum_{P_i \in newC} p_{ix} = \frac{1}{1}(2.0) = 2.0$$

$$newm_y = \frac{1}{N_{newC}} \sum_{P_i \in newC} p_{iy} = \frac{1}{1}(1.0) = 1.0$$

Point $P_1 = (2.0, 1.0)$ assigned to cell $newC = (1.79, 1.0)$ is represented by macropoint $newM = (2.0, 1.0)$. Since the represented point is only one, the *weight* of $newM$ is set to $weight_{newM} = 1.0$.

The macropoints representing the new cells are shown in Table 3.14 and in Figure 3.9.

	p_x	p_y	*cluster*	$newc_x$	$newc_x$	$newm_x$	$newm_y$	*weight*
P_1	2	1	1	1.79	1	2	1	1
P_2	1	2	1	1	1.79	1	2	1
P_3	2	2	1	1.79	1.79	2	2	1

Table 3.14: New cells and new macropoints: point (p_x, p_y) is reassigned to the cell identified by ($newc_x$, $newc_y$), and represented by the macropoint ($newm_x$, $newm_y$) with weight *weight*

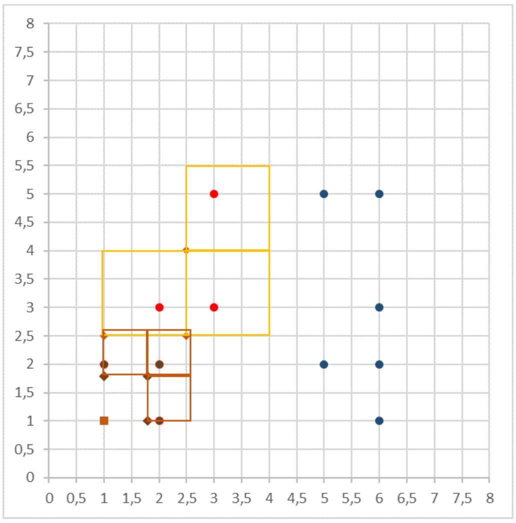

Figure 3.9: New cells (orange square) and new macropoints (brown points)

The original cell C_1 contained three points. Using the modification of weighted clustering, with a computed *newStep* of 0.79, cell C_1 has been divided into three new cells. Cluster 1 was intially divided into four cells and now it is divided into six cells: three new and three old ones. In this way, a more precise representation of the original cluster is obtained.

Chapter 4

Experiments

4.1 Input data sets

For the experiments a number of different data sets with different shapes and cardinalities have been used. All of them contain two-dimensional data: they can be considered as sets of points in a plane, each point identified by two numerical coordinates. Some data sets have been generated using Python library *sklearn.datasets* [16]: functions *make_ blobs* and *make_ moons* have been used to generate data sets with *blobs* and *moons* shapes. Since these data sets were not clustered, the K-means algorithm has been used to partition them into clusters. The K-means belongs to the clustering algorithms based on partition, it is known to be concise, efficient and fast [17]. It requires the user to give a value of K (the number of clusters to be identified), which was not a problem in this work since the data sets were generated by us using K=3 for *Blobs* and K=2 for *Moons*. Examples of data sets with these two shapes and with number of points set to 10000 for both of them can be seen in Figure 4.1 and in Figure 4.2. The data have been clustered using a Spark version of the K-means algorithm (the Java library *org.apache.spark.ml.clustering.KMeans*, [9]). In all the following figures, different colors correspond to different clusters.

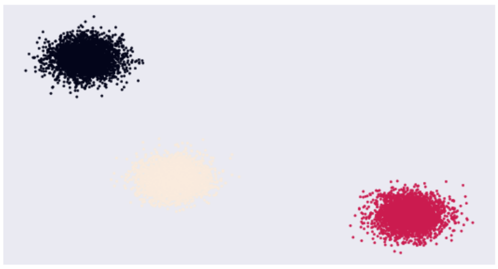

Figure 4.1: *Blobs* dataset with 10000 points

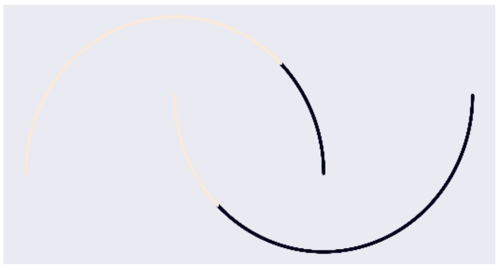

Figure 4.2: *Moons* dataset with 10000 points

For further experiments, some already clustered data sets with more complex shapes have been used, taken from the project *Clustering Benchmarks* (available at *https://github.com/deric/clustering-benchmark/*). This project contains a collection of clustered data sets that can be found in the literature. Most of them are artificially created. In particular, the artificial shapes *Complex8*, *Complex9*, *Compound* and *Cure-t1-2000n-2d* were chosen. They are displayed in the following images.

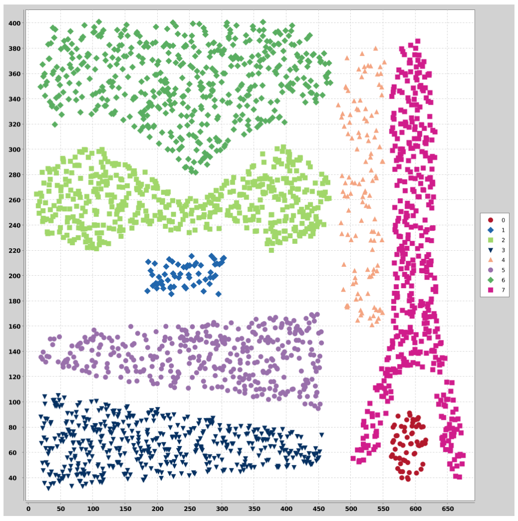

Figure 4.3: *Complex8* data set with 2551 points

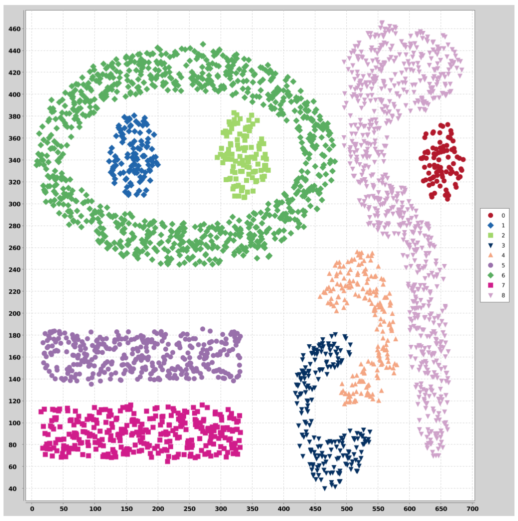

Figure 4.4: *Complex9* data set with 3031 points

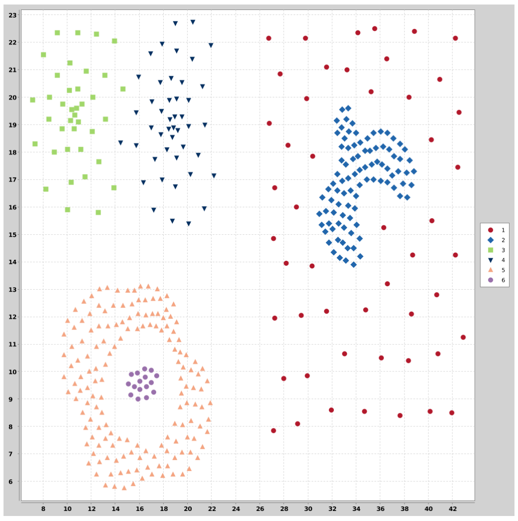

Figure 4.5: *Compound* data set with 399 points

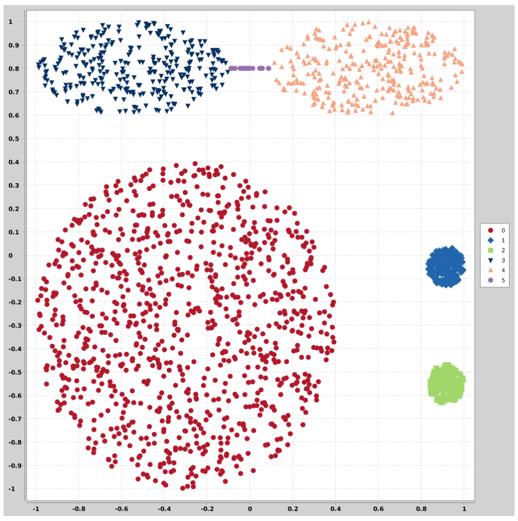

Figure 4.6: *Cure-t1-2000n-2d* data set with 2000 points

For other kinds of experiments some randomly clustered data sets have been used: starting from the data contained in some of the previous presented data sets, a process of random partition into clusters was applied. This process simply consists in the random assignment of each record to a cluster, given the number of clusters in which the data have to be partitioned, as shown in the following code.

Algorithm 16 Random assignment of points to clusters

Input: *points* to be clustered; number K of clusters to be created
Output: the assignment of each point to one of the K *clusters*

```
1: points;                                      ▷ Set of points to be clustered
2: K;                                           ▷ Number of clusters to be created
3: Assign each point to a randomly chosen cluster:
4: for point in points do
5:     Generate a random number between 0 and K-1:
6:     clusterNum = randomNumber(0,K-1);
7:     Assign the point to the randomly chosen cluster:
8:     point.cluster = clusterNum;
```

Example of the obtained random generated clusters are shown in the following figures for the data sets *Blobs* and *Complex8*.

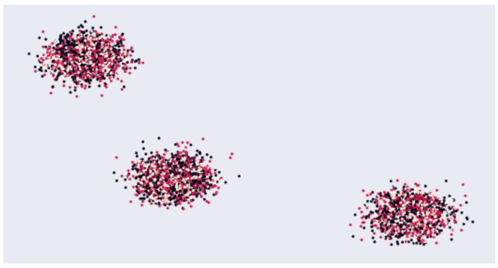

Figure 4.7: An example of *Blobs* data randomly assigned to three clusters

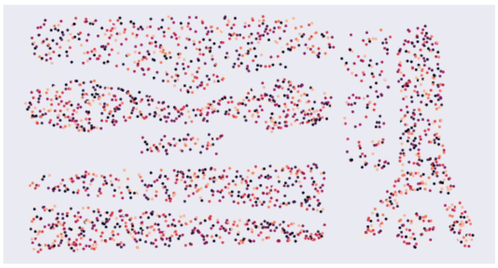

Figure 4.8: An example of *Complex 8* data randomly assigned to eight clusters

4.2 Random competitors

The aim of the weighted clustering algorithm presented in this thesis work is to reduce the cardinality of a set of clustered data without loosing the most important characteristics of the data. This is done through the representation of a number of records using only one representative record, that tries to summarize all their characteristics. This process allows to obtain a new set of clustered data with a reduced cardinality, which makes operations on it simpler and faster.

Since no works with the same aim have been found, some artificial competitors have been created. The results obtained using these competitors have been compared with the results given by our weighted clustering. In order to reduce the cardinality of a set of clustered data, the easiest approach is to randomly select a number of data from each cluster, lower than the number of data originally in the cluster. This has been applied in two different ways. Both the solutions require the knowledge of the number of clusters, which is not a problem because our process starts from already clustered data.

The first approach, called *random strategy 0*, consists in the random selection of a number of points from each cluster, where the number of selected points from a certain cluster depends from the original number of points of the cluster. The information about the number of data contained in each cluster is requested, which again is not a problem since the process operates directly on the clusters of data. The total quantity of points to be randomly selected is given by the user: it has to be set to values lower than the original cardinality of the entire dataset in order to reduce the cardinality. If the cardinality needs to be reduced a lot, a smaller value of this parameter is chosen. Given this value, the number of points to be randomly selected from each cluster is computed proportionally to the number of points in the considered cluster, as shown in Equation 4.1.

$$numRandomPoints0_{C_k} = \frac{totNumRandomPoints \cdot numPoints_{C_k}}{totNumPoints} \tag{4.1}$$

where C_k is the considered cluster; $numRandomPoints0_{C_k}$ is the number of points to be randomly selected from cluster C_k using strategy 0; $totNumRandomPoints$ is the total number of points to be randomly selected, given by the user; $numPoints_{C_k}$ is the original number of points contained in cluster C_k; $totNumPoints$ is the original total number of points of the data set.

When a point is selected from a cluster, it keeps the information about its cluster. In this way the new clusters of randomly selected points with reduced cardinality, called *startegy 0 random clusters*, are the clusters composed by the randomly selected points.

The second solution does not consider the cardinality of the single clusters: the total number of data to be selected is simply divided by the number of clusters; a number of points equal to the result of this division is selected randomly from each cluster, as described in Equation 4.2.

$$numRandomPoints1_{C_k} = \frac{totNumRandomPoints}{numClusters} \tag{4.2}$$

where C_k is the considered cluster; $numRandomPoints1_{C_k}$ is the number of points to be randomly selected from cluster C_k using strategy 1 (this value is the same for each cluster); *totNumRandomPoints* is the total number of points to be randomly selected, given by the user; *numClusters* is the number of clusters.

Each randomly selected point keeps the information about its cluster. So, as for strategy 0, the new clusters with reduced cardinality called *startegy 1 random clusters* are simply the clusters of randomly selected points.

Examples of *random clusters* obtained using strategies 0 and 1 are reported in the following figures. The starting data set was the *Complex 8* (Figure 4.3), the number of points to be randomly selected was set to 500 (while the original data set cardinality was 2551), and the number of clusters was set to 8 (equal to the number of original clusters).

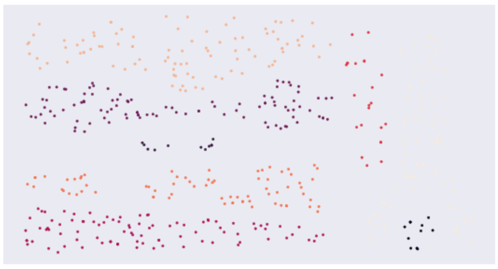
Figure 4.9: Clusters of 500 randomly selected points for data set *Complex 8* obtained using strategy 0

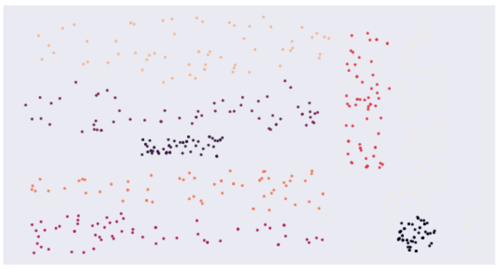

Figure 4.10: Clusters of 500 randomly selected points for data set *Complex 8* obtained using strategy 1

It is evident that using strategy 0, where the number of random points of each cluster depends from the size of the considered cluster, larger numbers are used to represent larger clusters. On the other side, in strategy 1 the same number of points is selected for each cluster. For small clusters the number of randomly selected points is close to the number of original points, while for large clusters the number of representative points is dramatically smaller than the quantity of original points.

4.3 Main experiments

Defined the weighted clustering and the weighted silhouette processes, experiments have been performed to evaluate their results. The silhouette index has been used to compare the quality of the original clustering, measured with the silhouette index applied on the original data, with the quality of the weighted clustering, measured with the weighted silhouette index applied on the weighted data. Also silhouettes indices computed on the set of random selected data obtained with strategies 0 and 1 have been considered. To summarize, the comparison has been performed between silhouettes computed in four different ways:

- the silhouette index calculated on the whole original clustering (i.e. the starting set of points that has been previously clustered with some clustering algorithm); this is the desired silhouette value
- the weighted version of the silhouette index computed on the weighted clustering (i.e. the set of weighted macropoints partitioned in clusters, obtained through the application of the weighted clustering algorithm on the original clusters)
- the average of $N_{iterations}$ silhouette indices computed on the sets of random clusters obtained applying strategy 0 $N_{iterations}$ times on the original clusters (i.e. the sets of points randomly selected from each cluster using strategy 0)
- the average of $N_{iterations}$ silhouette indices computed on the sets of random clusters obtained applying strategy 1 $N_{iterations}$ times on the original clusters (i.e. the sets of points randomly selected from each cluster using strategy 1)

The number of points considered in the first case is simply the total number of points in all the clusters of the original clustering. The number of macropoints considered in the case of weighted silhouette applied on weighted clustering depends on the value of the parameter *step*: the higher the *step*, the lower the number of macropoints. The number of points randomly selected to create random clusters can be set by the user. In order to compare the weighted silhouette with the silhouettes on random clusters, the total number of points to be randomly selected during the application of random strategy 0 or 1 has been set to the number of macropoints obtained with the weighted clustering. In this way, qualities of partitions in clusters with the same cardinality are compared. In weighted clustering the cardinality of the original clustering is reduced dividing into cells and representing each cell with a macropoint. In random cases the cardinality is reduced choosing randomly the requested number of points, using strategy 0 or 1 to choose how many points to select from each cluster. Remember that in strategy 0 the number of points randomly selected from a cluster is a fraction of the total number of points to be selected that is proportional to the number of points in the cluster. In strategy 1 the number of points randomly chosen from a cluster is a fraction of the total number of points equal for all clusters (i.e. the total number of points is simply divided by the number of clusters).

The silhouette index on the original set of clusters is computed and this result is the desired value of silhouette. Considering a set of clusters with reduced cardinality (i.e. obtained applying weighted clustering or random strategies on the original clustering), if its silhouette index is close to the desired value of silhouette it means that the considered clustering is a good representation of the original clustering. Given two sets of clusters with reduced cardinality, the one with the silhouette nearest to the desired value is the one that best represents the original clustered data.

A number of weighted clusterings is retrieved using different values of *step*. In these experiments 12 different weighted clusterings have been considered, obtained with 12 different values of *step*. Each value of *step* corresponds to a certain number of obtained macropoints. Let N_D be the number of points in the original data set. Values of *step* are chosen in order to obtain quantities of macropoints equal to fractions of N_D that cover the interval between $N_D/1.5$ (cardinality reduced by a $1/1.5$ factor) and $N_D/150$ (cardinality reduced by a $1/150$ factor).

The random selection of points from clusters is performed on the original clusters using strategies 0 and 1 with a number of points to be selected equal to the number of macropoints obtained with the weighted clustering. The silhouette index is applied on these sets of random clusters. Since the points are selected randomly, the results are fortuitous and the silhouette index on a set of random clusters can randomly be close the desired value of silhouette or far from it. For this reason, the random selection of points is applied $N_{iterations}$ times for each number of points to be selected. In this way, $N_{iterations}$ sets of random clusters are generated, and the silhouette index is calculated on each one of them. The final value of silhouette that will be compared with the desired and the weighted silhouettes is the arithmetic mean between the $N_{iterations}$ silhouette indices. This is done both for strategies 0 and 1. The $N_{iterations}$ parameter can be set by the user, and in the following experiments will always be set to 10.

To sum up, the results that have been compared are: the silhouette index on the original clustering, the weighted silhouette index on the weighted clustering with a certain *step*, the mean of $N_{iterations}$ silhouette indices computed on $N_{iterations}$ sets of random clusters obtained using strategy 0 with a number of points to be randomly selected equal to the number of macropoints, and the mean of $N_{iterations}$ silhouette indexes computed on $N_{iterations}$ sets of random clusters obtained using strategy 1 with number of points equal to the number of macropoints. This comparison has been done for each data set for different values of *step* in order to consider different factors of cardinality reduction. Clearly the desired silhouette (i.e. the silhouette index calculated on the original partition in clusters) does not depend on the *step* so is a constant.

The quality of the original clustering is evaluated by the silhouette on the original data (i.e. the desired silhouette), while the quality of the weighted clustering is evaluated by the weighted silhouette. The goodness of the weighted clustering as a representation of the original clustering with a reduced cardinality can be measured as the distance between the weighted silhouette and the desired silhouette. The lower is this distance, the better the weighted clustering represents the original.

In the same way, the quality of a single set of random clusters (i.e. clusters of random selected points) is represented by the silhouette computed on it, and the goodness of the random clusters as a representation of the original clusters with a reduced cardinality is indicated by the distance between the silhouette on the set of random clusters and the silhouette on original data. Comparisons have been made both considering one silhouette computed on one set of random clusters and considering the mean of 10 silhouettes computed on 10 different sets of random clusters. Of course the second case, being an average of 10 results obtained on 10 sets of clusters, does not represent an existing set of clusters. So the distance between this average silhouette and the desired silhouette does not indicate the quality of a real representation. So if the goal is to obtain a representation of the clustering with a reduced cardinality, random strategies are not so useful.

Finally, some minor experiments have been performed to study special situations encountered for some data sets. They are described in the next sections, where experiments and results are reported for each data set.

4.4 Blobs data set experiments and results

4.4.1 Silhouettes results

The data set with Blobs displayed in Figure 4.1 of Section 4.1 is considered. It has been generated using a Python library and then clustered with the K-means algorithm.

The original number of points in this data set is $N_D = 10000$. This cardinality has been chosen because it m sil thakcomputaee shof theoution ette index still feasible. Since the time complexity of the silhouette is quadratic in the number of data, for higher cardinalities the index takes too long to be retrieved. Twelve different *step* values has been considered, chosen in order to have numbers of macropoints covering the interval $[N_D/1.5,\ N_D/150]$. For example, the application of weighted clustering on the *Blobs* data set using a *step* of 0.03 gives a number of macropoints equal to 6694, which is the 1/1.49 of 10000. The weighted silhouette computed on weighted clustering obtained with this value of *step* is compared to the silhouette of the original clustering and to the averages of the silhouettes on random clusters obtained with strategies 0 and 1 with number of points set to 6694. The distance between the desired silhouette and the weighted silhouette indicates the quality of the weighted clustering as a representation of the original clustering with a cardinality reduced from 10000 to 6694.

The following values of *step* are used, covering the interval of number of macropoints between 52 and 6694. In Table 4.1 correspondences *step* - obtained number of macropoints are displayed, while in Figure 4.11 comparisons between obtained number of macropoints and original number of points are reported for each *step* value. Of course, the original number of points is a constant.

step	*numMacropoints*
0.03	6694
0.04	5329
0.05	4211
0.07	2822
0.1	1730
0.15	953
0.2	607
0.3	315
0.4	203
0.5	141
0.7	78
0.9	52

Table 4.1: Number of macropoints obtained for *Blobs* data set using the weighted clustering algorithm, depending on the value of *step*

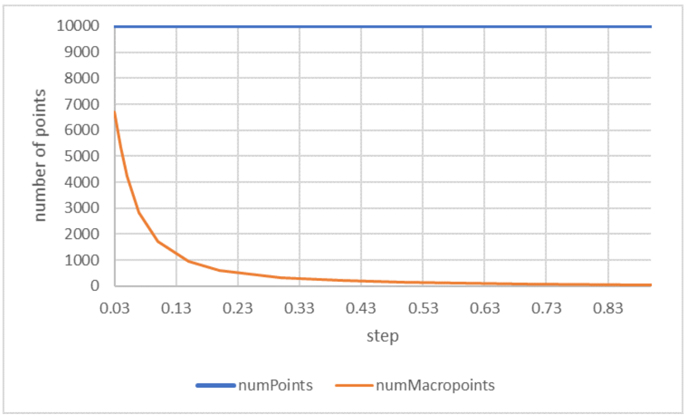

Figure 4.11: Comparison between original number of points (*numPoints*) and number of macropoints (*numMacropoints*) obtained applying weighted clustering on *Blobs* data set with different *steps*

Using these values of *step*, the silhouette on the original data, the weighted silhouette on the result of the weighted clustering process, the average of silhouettes on random clusters obtained with strategy 0, and the average of silhouettes on random clusters obtained with strategy 1 are computed for each *step*. These silhouettes are compared in Figure 4.12. The weighted silhouette is computed on the result of the weighted clustering presented in Section 3.1, without the modifications described in Section 3.3. Remember that average silhouettes on results of both random strategies 0 and 1 are obtained as an average of 10 silhouettes computed on 10 results of random clusters. It is reasonable to compare the weighted silhouette result with only one silhouette calculated on one set of random clusters. In figures 4.13 and 4.14, the worst cases (minimum and maximum) among the computed silhouettes on results of random strategies are shown separately for strategy 0 and strategy 1.

Notice that for this particular data set strategies 0 and 1 are the same, since the number of points is equally distributed between the three clusters of the *Blobs* data set. Silhouettes on these sets of random clusters are slightly different because the selection of points is random.

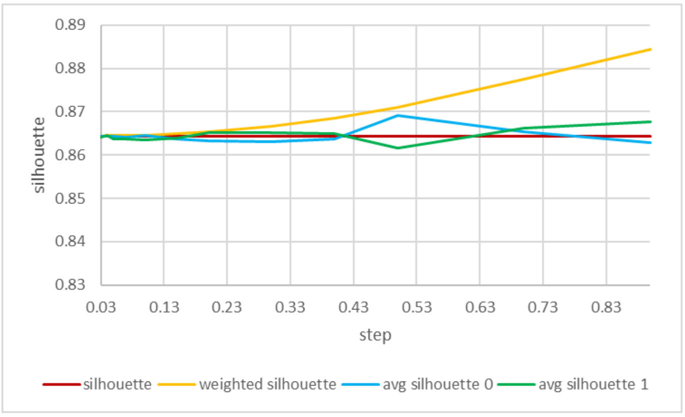

Figure 4.12: Silhouette on the original data (*silhouette*), weighted silhouette on weighted clustering (*weighted silhouette*), average silhouette on random clusters strategy 0 (*avg silhouette 0*), and average silhouette on random clusters strategy 1 (*avg silhouette 1*)

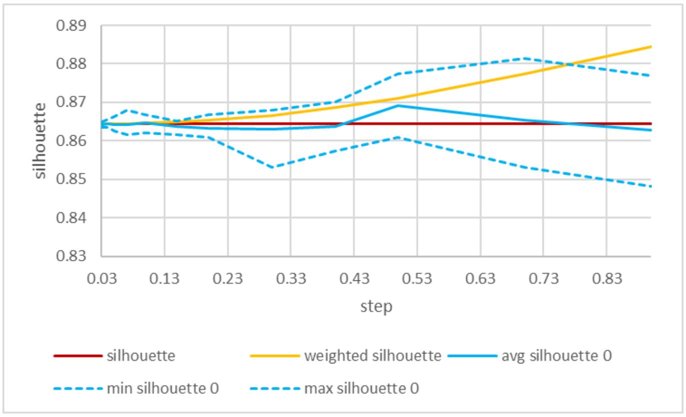

Figure 4.13: Details of the worst cases (*min silhouette 0* and *max silhouette 0*) of silhouettes on random clusters strategy 0

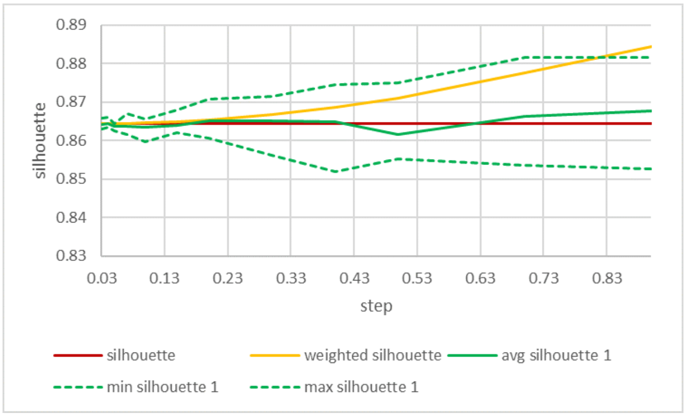

Figure 4.14: Details of the worst cases (*min silhouette 1* and *max silhouette 1*) of silhouettes on random clusters strategy 1

In any case, both for the weighted clustering and for the random strategies, the quality of the representation decreases (i.e. the silhouette results are more distant from the desired value) with the increase of the *step* value. This clearly happens because an increase of the *step* means a reduction of the number of macropoints or random selected points that represent the initial points. The worsening for greater *steps* is terrific for weighted clustering, that for higher values results to be even worse than random strategies.

Comparing the weighted silhouette result with only one silhouette computed on only one set of random clusters, weighted clustering (whose quality is measured by the weighted silhouette) results for most of the *steps* better than random strategies worst cases, as shown in Figure 4.13 and in Figure 4.14. So the quality of a single representation obtained using a random approach is lower than the average silhouette of random strategies results. For higher *steps* the weighted silhouette is even more distant from the desired value than the worst case of silhouette on random clusters. For lower *steps*, the average of the silhouettes computed on results of random strategies may be more similar to the desired silhouette, but this average does not represent the quality of any clustering. If the goal is to produce a representation of the original clustering with a reduced cardinality, the weighted clustering is generally better than the random methods.

For high values of *step* the weighted silhouette rises up, increasing the distance from the desired silhouette. As already said, the increase of this distance is caused by the decrease in the number of macropoints that represent the original points. The rise of the weighted silhouette for high values of *step* has been studied.

The idea was that probably these bad results were caused by too dense macropoints. In case of high values of *step*, the obtained cells are large and if the cell contains a lot of points the weight of its macropoint will be high as well. A macropoint $M(m_x,\ m_y)$ with weight w_M corresponds exactly, in the weighted silhouette computation, to a number w_M of not weighted points with coordinates $(m_x,\ m_y)$. If a cell contains a lot of scattered points, the representation of all of them in the single macropoint $M(m_x,\ m_y)$ is too dense. This leads to an increase of the cohesion, the measure of how similar an object is to its own cluster. As a result, the weighted silhouette takes higher values than the silhouette computed on starting data. For big values of *step* the distance of the weighted silhouette from the desired silhouette is not acceptable, being greater than the distance of the silhouette on random strategies results. In order to solve this issue, a secondary partition of cells with large size and high standard deviation is applied, using a lower value of *step*. This process has been described in the section dedicated to the weighted clustering modification (Section 3.3) and requires the values of two thresholds. In the next section these threshold are studied and possible choices of values are presented.

4.4.2 Thresholds

The modification of weighted clustering process, described in Section 3.3, tries to solve the issues caused by too dense macropoints. In order to apply this process the value of two thresholds need to be decided: *thPercStddev* and *thPercRepresentedPoints*.

Percentage of standard deviation

During the weighted clustering process, after the division of a cluster into cells, the range of variation $range_j$ and the standard deviation $stddev_j$ are computed for each cell on each dimension $j = 1, 2, ..., N$ where N is the number of dimensions (2 in our experiments). The range indicates the maximum variation of the points of the considered cell on the considered coordinate. The standard deviation on a certain dimension represents the variations of points values on that dimension compared to the value of the macropoint on that dimension. Remember that the macropoint represents the points of a cell and it is computed on each coordinate as an average of the points values on that coordinate. Given these values, the percentages of standard deviation compared to the range of variation $percStddev_j$ are defined on each dimension j as:

$$percStddev_j = \frac{stddev_j \cdot 100}{range_j} \tag{4.3}$$

For each j, $percStddev_j$ is compared with the threshold *thPercStddev*, and the cell is selected as *cell to be splitted* if at least one of them is greater than the threshold.

The value of *thPercStddev* has to be set in a way to select critical cells: cells with high number of points and high dispersion of these points. To do that, some data has been collected for different values of the threshold. The *thPercStddev* has been set to $\{15, 20, 25, 30, 40, 50\}$, and the following information have been collected:

- number of cells selected to be splitted
  - for each cluster: *numCellsToBeSplitted*
  - for the entire clustering: *totNumCellsToBeSplitted* (sum of *numCellsToBeSplitted* of all clusters)
- percentage of cells selected to be splitted
  - for each cluster: $percCellsToBeSplitted = numCellsToBeSplitted \cdot 100 / numCells$

- for the entire clustering: *totPercCellsToBeSplitted* = *totNumCellsToBeSplitted* · 100 / totNumCells
- average: *avgPercCellsToBeSplitted* (average of *percCellsToBeSplitted* of all clusters)

Numbers of cells to be splitted for cluster 1 and for the whole data set using *thPercStddev* = {15, 20, 25, 30, 40, 50} are shown in Figure 4.15 for cluster 1 and Figure 4.16 for the data set. Information about other clusters are not reported because in the *Blobs* case the points are equally distributed between three clusters with the exact same shape, so other clusters are totally similar to cluster 1.

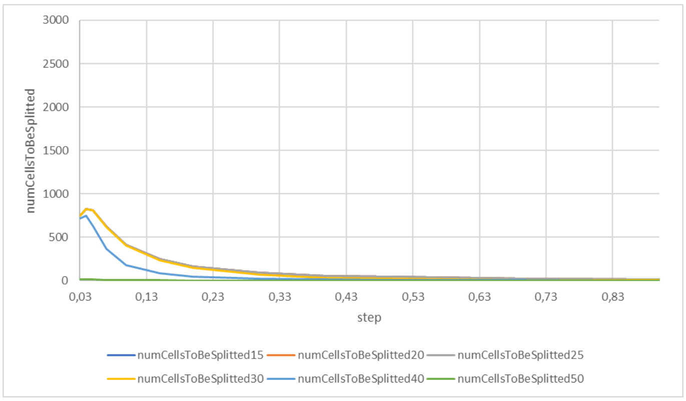

Figure 4.15: Numbers of cells to be splitted obtained for cluster 1 with *thPercStddev* = {15, 20, 25, 30, 40, 50}. *NumCellsToBeSplittedx* indicates the number of cells selected to be splitted using *thPercStddev* = *x*

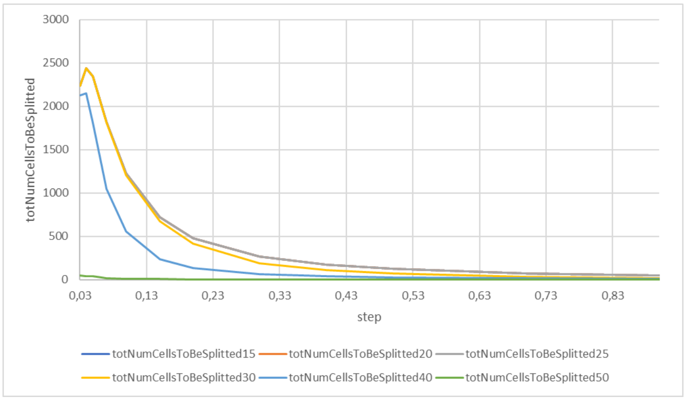

Figure 4.16: Numbers of cells to be splitted for the entire data set using *thPercStddev* = {15, 20, 25, 30, 40, 50}. *TotNumCellsToBeSplittedx* indicates the number of cells selected to be splitted using *thPercStddev* = *x*

Using *thPercStddev* = 50 the quantity of cells to be splitted is almost zero for all *steps*, so this threshold will not be considered. For all threshold values, for lower *steps* the number of cells selected results to be higher, but it is important to consider that for lower *steps* the total number of cells is greater. For this reason the percentage of cells to be splitted has been considered instead of the number of cells to be splitted.

Percentages of cells to be splitted for cluster 1, obtained as $numCellsToBeSplitted_{cluster1} \cdot 100 \ / \ numCells_{cluster1}$, and for the entire clustering, obtained as $totNumCellsToBeSplitted \cdot 100 \ / \ totNumCells$, computed for each value of *step* are shown in the next images (4.17 and 4.18). In Figure 4.19 the average percentage for the whole clustering is displayed, computed as the average of the percentages of all clusters.

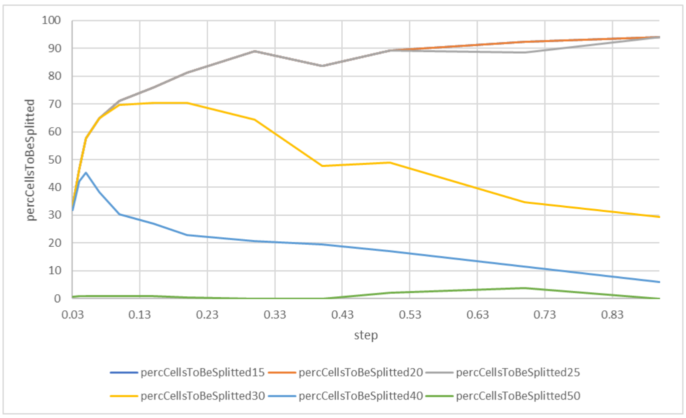

Figure 4.17: Percentage of cells to be splitted for cluster 1: $percCellsToBeSplitted_{cluster1} = numCellsToBeSplitted_{cluster1} \cdot 100 \ / \ \mathrm{numCells}_{\mathrm{cluster1}}$. $PercCellsToBeSplittedx$ indicates the percentage obtained using $thPercStddev = x$

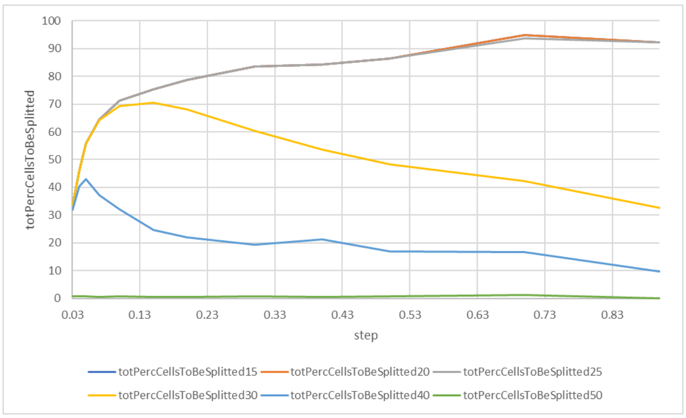

Figure 4.18: Percentage of cells to be splitted for the entire clustering: $totPercCellsToBeSplitted = totNumCellsToBeSplitted \cdot 100 \ / \ \mathrm{totNumCells}$. $TotPercCellsToBeSplittedx$ indicates the percentage obtained using $thPercStddev = x$

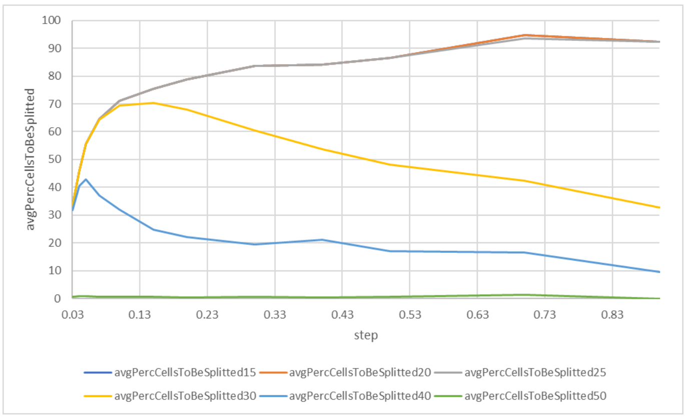

Figure 4.19: Average of *percCellsToBeSplitted* among all clusters: *avgPercCellsToBeSplitted*. *AvgPercCellsToBeSplittedx* indicates the average obtained using *thPercStddev* $= x$

It is clear that *thPercStddev* $= \{30, 40, 50\}$ are not good choices: as already said, the value 50 leads almost always to empty sets of cells to be splitted; 30 and 40 are not good since our purpose is to split more cells for higher values of *step*, when cells are bigger and the situation starts to be critical. So the candidates for *thPercStddev* are values $\{15, 20, 25\}$.

Let us consider the lowest of the possible values: *thPercStddev* $= 15$. In the following figures, some further charts are reported to highlight how do the number of macropoints and the value of the weighted silhouette change using the modified weighted clustering with this threshold. In Figure 4.20 the number of macropoints obtained with the normal weighted clustering and the one obtained with the modification are compared. In the modification only the threshold *thPercStddev* $= 15$ is considered. In a similar way, in Figure 4.21 the value of the weighted silhouette computed on the normal weighted clustering result and the value computed on the modified weighted clustering result are compared.

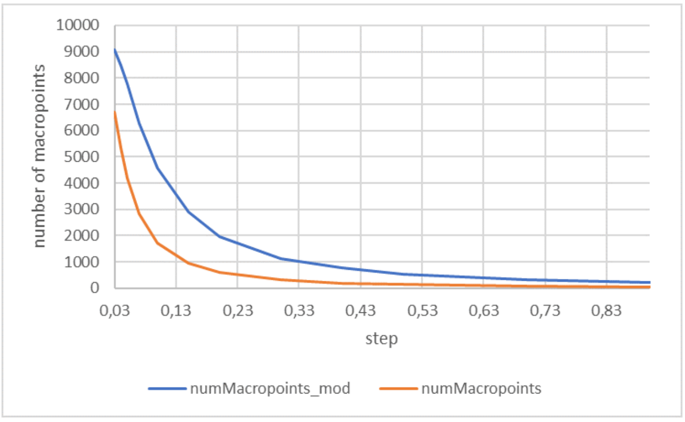

Figure 4.20: Number of macropoints obtained with the normal weighted clustering (*numMacropoints*) and with the modified weighted clustering (*numMacropoints_mod*). In the modification only the threshold *thPercStddev* = 15 is considered.

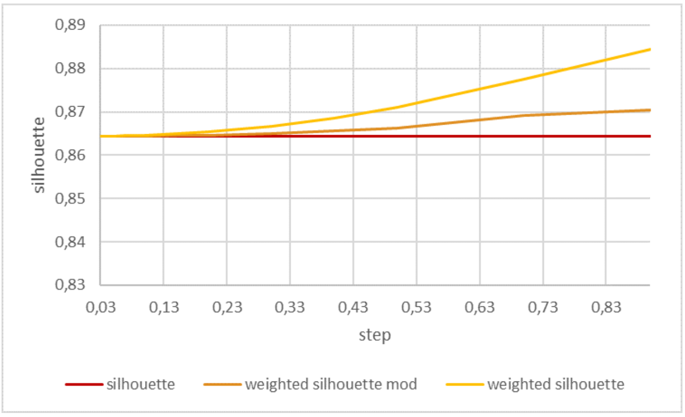

Figure 4.21: Weighted silhouette computed on the normal weighted clustering result (*weighted silhouette*) and on the modified weighted clustering result(*weighted silhouette mod*). In the modification only the threshold *thPercStddev* = 15 is considered.

Looking at the first chart it is evident that the number of macropoints obtained with the modification is too high for low values of *step*, where an increase in this number is not really needed. For these low values the obtained weighted silhouettes are similar enough to the desired one, so the number of macropoints should ideally be the same as the one obtained without the modification. About the second figure it can be noticed that for lower *steps* the weighted silhouettes with and without the modification are almost the same, confirming the fact that the increase in the number of macropoints with the modification is not useful. For higher *steps* the weighted silhouette on the modified weighted clustering is much better than the normal one (i.e. the weighted silhouette on it is much more similar to the desired value of silhouette). Even if for high *step* values the number of macropoints is only slightly higher in the modified version, the improving in the quality is clearly huge. To solve the issue of the useless increase in the number of macropoints for low *steps*, a new threshold is introduced.

The *thPercStddev* threshold is used to select the cells with an high standard deviation. This means that the points in the selected cell are scattered. The selection using only *thPercStddev* does not consider number of points contained in cells. As said before, the worsening of the weighted clustering representation arise when the obtained cells are large and contain a lot of scattered points. The variety of points in a cell is measured with the *percStddev*, and using *thPercStddev* if its points have an high standard deviation the cell is selected. Since a cell is critical if its points are scattered and if it contains a lot of points, the number of points contained in the cell has to be considered as well. For this purpose, the threshold *thPercRepresentedPoints* is introduced.

Percentage of represented points

During the weighted clustering process, as already explained, the percentages of standard deviation compared to the range of variation $percStddev_j$ is computed for each cell on each dimension j. After this computation, the *percRepresentedPoints* is retrieved for each cell. This value is not dimensional, for each cell only one value of *percRepresentedPoints* is computed. The definition of *percRepresentedPoints* for a cell C belonging to cluster K is:

$$percRepresentedPoints_C = \frac{N_C \cdot 100}{N_k} \tag{4.4}$$

where N_C is the number of points in the considered cell C and N_k is the total number of points in cluster K.

Considering a cell, its $percStddev_j$ for each dimension j and its *percRepresentedPoints* have been computed. The $percStddev_j$ on each dimension is compared with the threshold *thPercStddev*, and the cell is selected as *possible cell to be splitted* if at least one of them is greater than the threshold. The *percRepresentedPoints* is compared with the threshold *thPercRepresentedPoints*, and the cell is selected as *cell to be splitted* if the previous condition was satisfied and its *percRepresentedPoints* results greater than *thPercRepresentedPoints*.

Other data has been collected in order to choose thresholds values. The threshold *thPercStddev* has been set to the values selected before: {15, 20, 25}; while *thPercRepresentedPoints* has been set to {2, 4, 5, 10}. The collected data are the same as before:

- number of cells selected to be splitted
  - for each cluster
  - for the entire clustering
- percentage of cells selected to be splitted
  - for each cluster
  - for the entire clustering
  - average among all clusters

Numbers of cells to be splitted for cluster 1 and for the entire data set using *thPercStddev* set to {15, 20, 25} and *thPercRepresentedPoints* set to {2, 4, 5, 10} are represented in the following figures. As before, information about other clusters are not reported because they are totally similar to cluster 1.

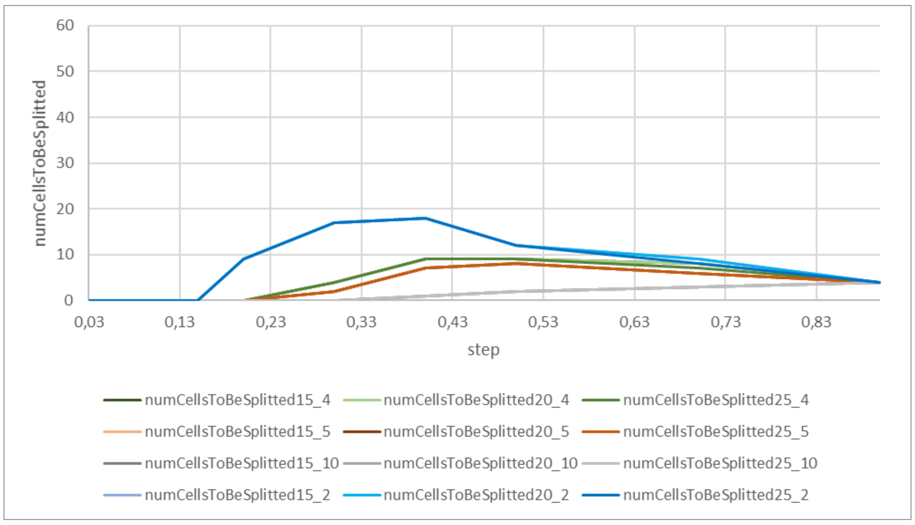

Figure 4.22: Numbers of cells to be splitted for cluster 1 using *thPercStddev* = {15, 20, 25} and *thPercRepresentedPoints* = {2, 4, 5, 10}. *NumCellsToBeSplittedx_y* indicates the number obtained using *thPercStddev* = x and *thPercRepresentedPoints* = y

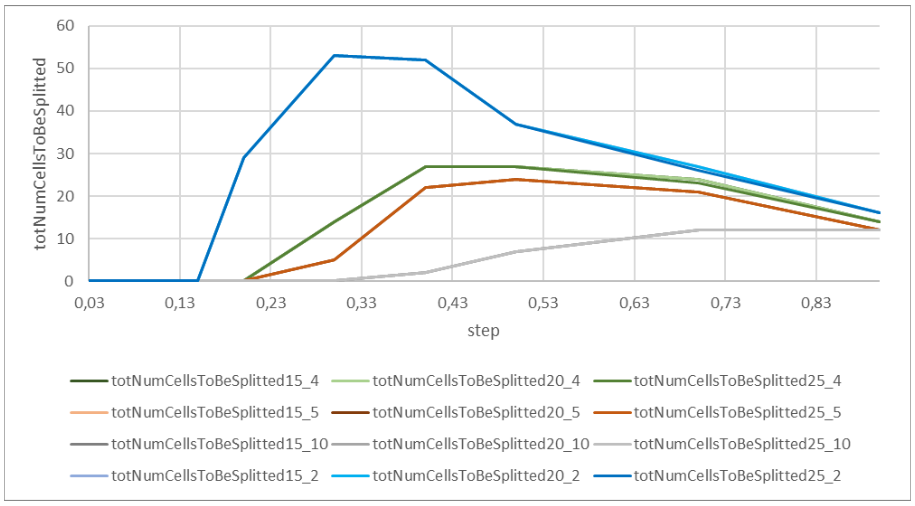

Figure 4.23: Numbers of cells to be splitted for the entire data set using *thPercStddev* = {15, 20, 25} and *thPercRepresentedPoints* = {2, 4, 5, 10}. *TotNumCellsToBeSplittedx_y* indicates the number obtained using *thPercStddev* = x and *thPercRepresentedPoints* = y

Since the total number of cells is greater for lower *steps* and the number of cells selected to be splitted is affected by this, the percentage of cells to be splitted has been considered.

Percentages of cells to be splitted for cluster 1, obtained as $numCellsToBeSplitted_{cluster1} \cdot 100 \ / \ numCells_{cluster1}$ are shown in Figure 4.29. Percentages for the entire clustering $totNumCellsToBeSplitted \cdot 100 \ / \ totNumCells$ are displayed in Figure 4.30. Finally, in Figure 4.31 the average of the percentages of all clusters is represented.

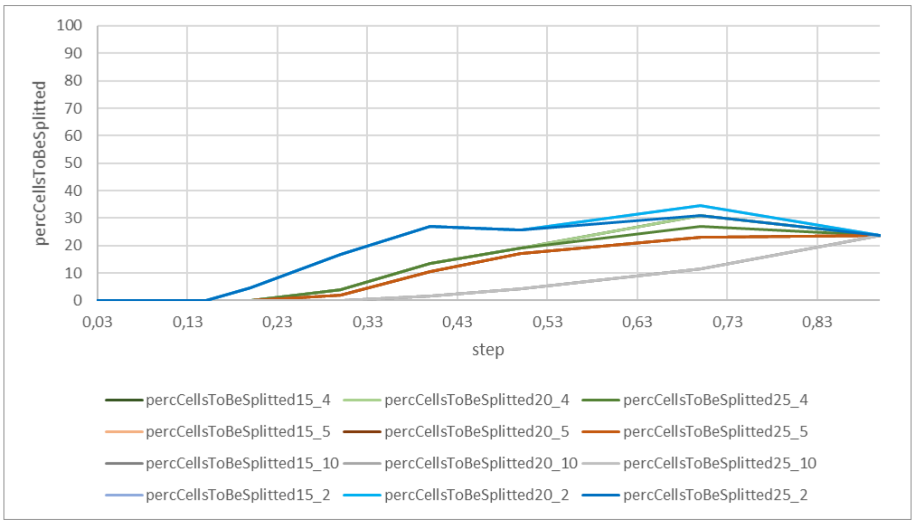

Figure 4.24: Percentage of cells to be splitted for cluster 1: $percCellsToBeSplitted_{cluster1} = numCellsToBeSplitted_{cluster1} \cdot 100 \ / \ \mathrm{numCells}_{\mathrm{cluster1}}$. $PercCellsToBeSplittedx\_y$ indicates the percentage obtained using $thPercStddev = x$ and $thPercRepresentedPoints = y$

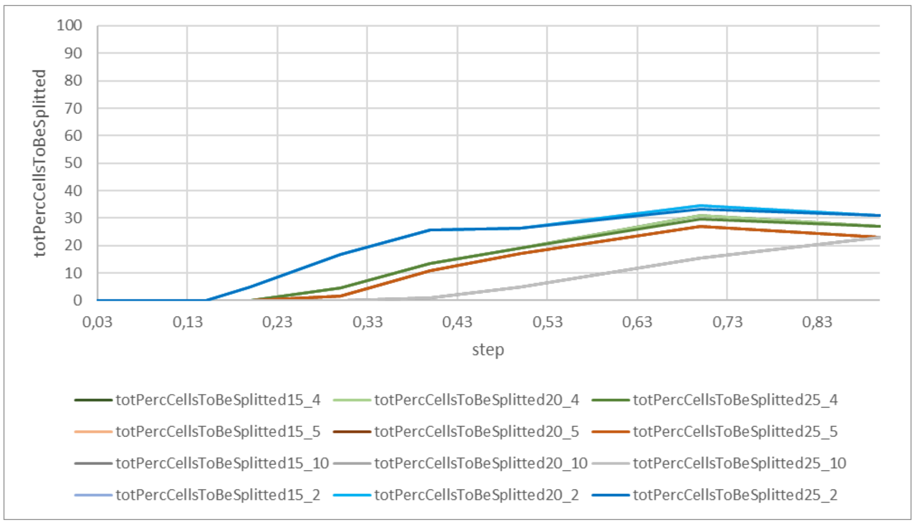

Figure 4.25: Percentage of cells to be splitted for the entire clustering: *totPercCellsToBeSplitted* = *totNumCellsToBeSplitted* · 100 / totNumCells. *TotPercCellsToBeSplitted$x$_y* indicates the percentage obtained using *thPercStddev* = x and *thPercRepresentedPoints* = y

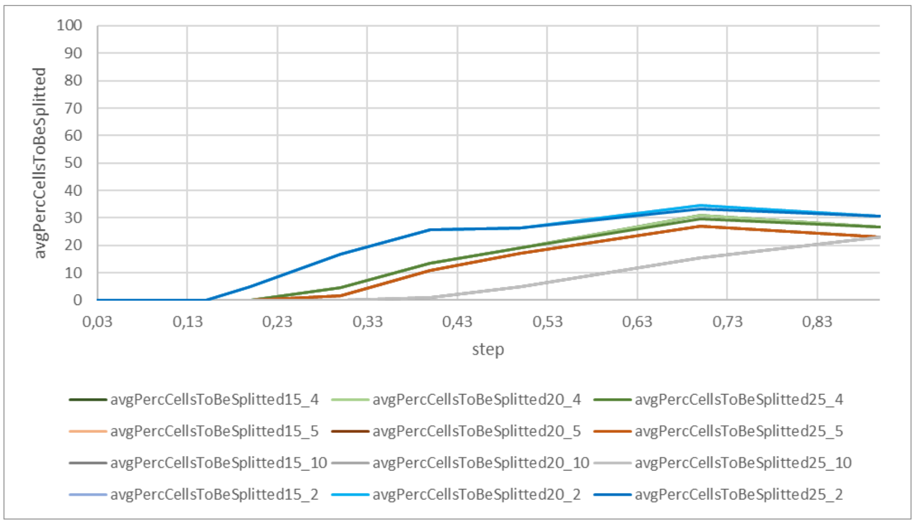

Figure 4.26: Average of *percCellsToBeSplitted* among all clusters: *avgPercCellsToBeSplitted*. *AvgPercCellsToBeSplitted$x$_y* indicates the average obtained using *thPercStddev* = x and *thPercRepresentedPoints* = y

It can be noticed that for all thresholds values the percentage of cells to be splitted grows up for higher *steps*. This is good because the worsening of the results happens for high values of *step*.

The use of both *thPercStddev* and *thPercRepresentedPoints* allows the selection of cells with an high number of points and an high standard deviation, i.e. the quantity of points in the selected cell is large and these points are scattered. This is the situation that leads to the increase of weighted silhouette values, since all these scattered points are represented in one single macropoint with an high weight. The obtained clusters of macropoints result to have a cohesion factor much higher than the original one, leading to bad results of silhouette. This condition becomes worse as the *step* assumes greater values, because the resultant cells are larger. The split of an higher number of cells for higher values of *step* can solve this problem.

For the choice of thresholds values, the silhouettes graphs need to be considered together with the *percCellsToBeSplitted* graphs. From Figure 4.12 that displays values of silhouette, weighted silhouette and average of the random silhouettes, it is clear that the worsening of the weighted silhouette starts from *step* values between 0.2 and 0.3. Value of thresholds such as *thPercStddev* = 15 and *thPercRepresented-Points* = 4 can be used, since they cause a percentage of cells to be splitted greater than 0 starting from *step* values around 0.3. Performing the same analysis on other data sets, same values of thresholds results to be a good choice.

Looking at the previous figures, it can be easily noticed that, considering both the thresholds, the trend is only slightly influenced by the *thPercStddev* value. Since for the same value of *thPercRepresentedPoints* we have almost the same trend for any values of *thPercStddev*, the *thPercRepresentedPoints* seems to have a way more impact than the other threshold. It can be useful to show the charts about numbers and percentages of cells to be splitted considering only the *thPercRepresentedPoints* threshold (i.e. setting the other threshold to 0). These charts are reported in the following figures.

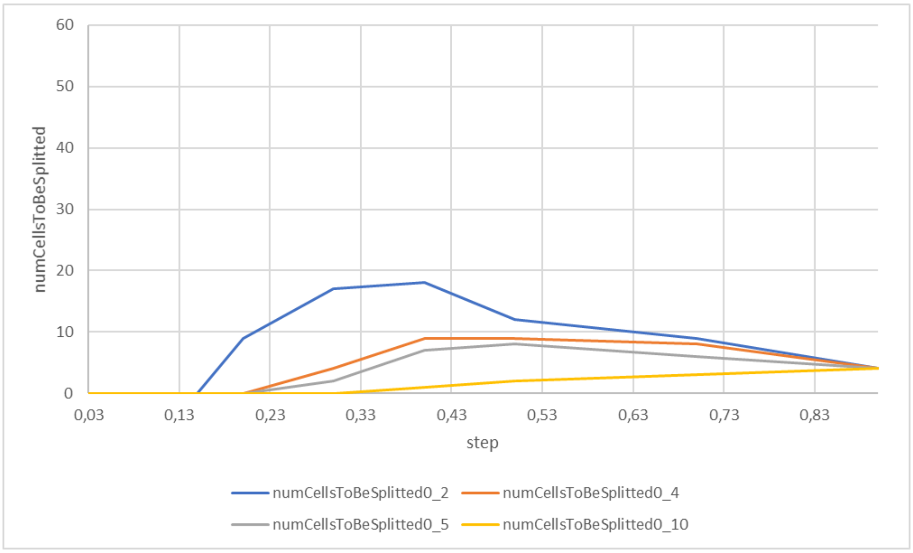

Figure 4.27: Numbers of cells to be splitted for cluster 1 using *thPercStddev* = 0 and *thPercRepresentedPoints* = {2, 4, 5, 10}. *NumCellsToBeSplitted0_y* indicates the number obtained using *thPercStddev* = 0 and *thPercRepresentedPoints* = *y*

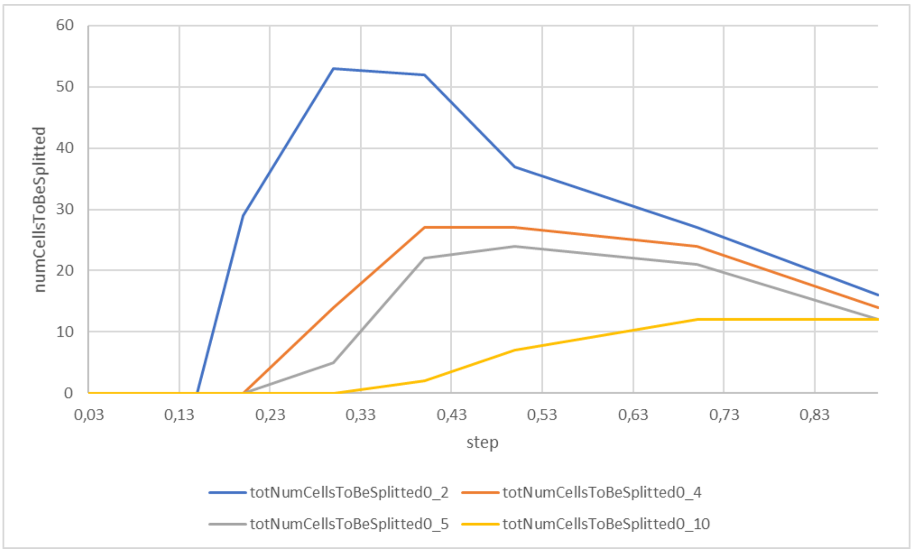

Figure 4.28: Numbers of cells to be splitted for the entire data set using *thPercStddev* = 0 and *thPercRepresentedPoints* = {2, 4, 5, 10}. *TotNumCellsToBeSplitted0_y* indicates the number obtained using *thPercStddev* = 0 and *thPercRepresentedPoints* = *y*

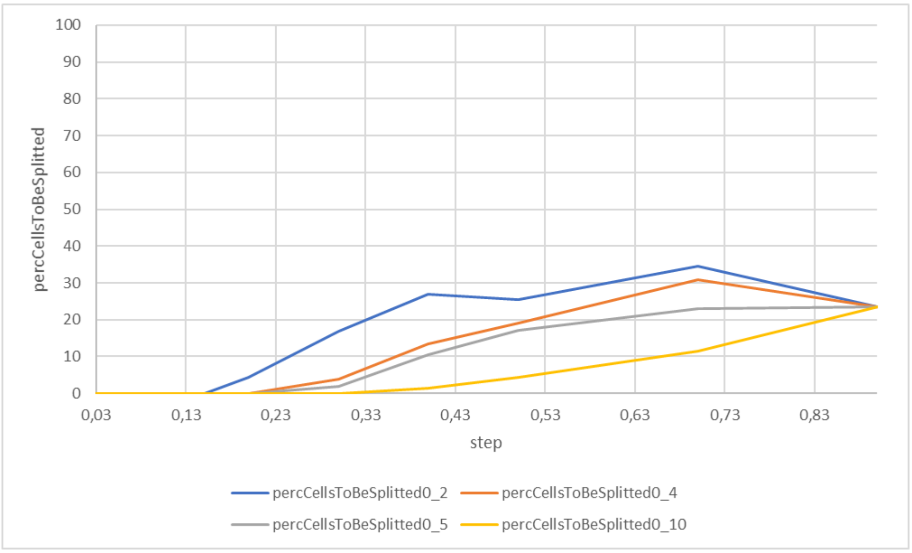

Figure 4.29: Percentage of cells to be splitted for cluster 1: $percCellsToBeSplitted_{cluster1} = numCellsToBeSplitted_{cluster1} \cdot 100 \ / \ \mathrm{numCells}_{\mathrm{cluster1}}$. $PercCellsToBeSplitted0\_y$ indicates the percentage obtained using $thPercStddev = 0$ and $thPercRepresentedPoints = y$

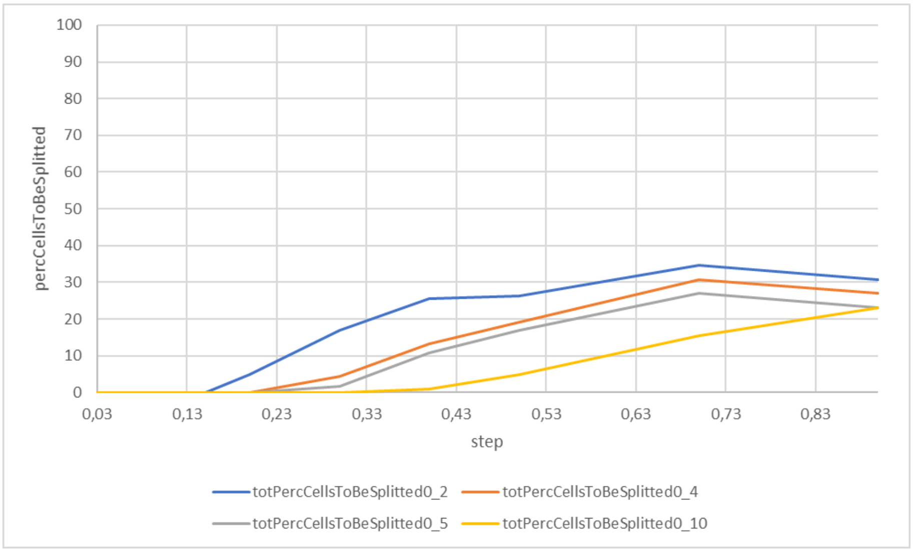

Figure 4.30: Percentage of cells to be splitted for the entire clustering: *totPercCellsToBeSplitted* = *totNumCellsToBeSplitted* · 100 / totNumCells. *TotPercCellsToBeSplitted0_y* indicates the percentage obtained using *thPercStddev* = 0 and *thPercRepresentedPoints* = *y*

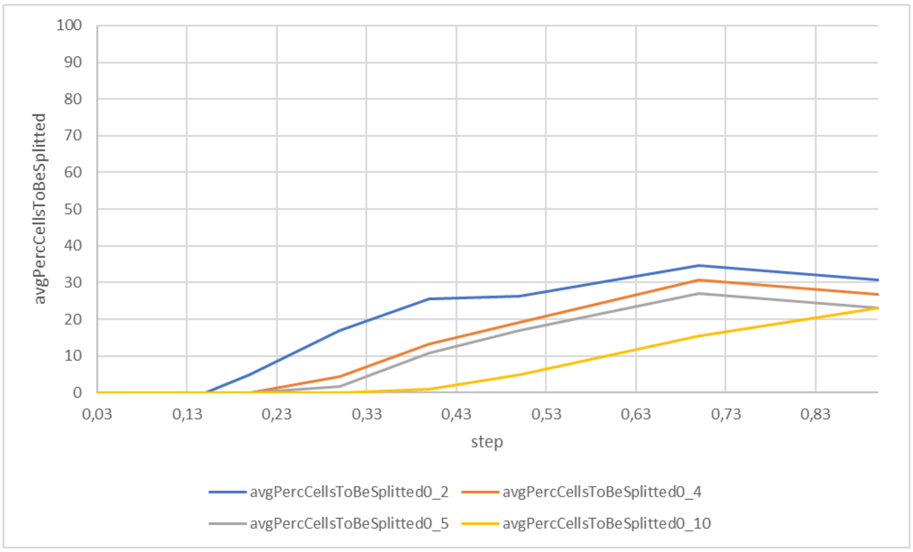

Figure 4.31: Average of *percCellsToBeSplitted* among all clusters: *avgPercCellsToBeSplitted*. *AvgPercCellsToBeSplitted0_y* indicates the average obtained using *thPercStddev* = 0 and *thPercRepresentedPoints* = *y*

Let us consider *thPercStddev* = 4. Some further charts are reported in the following, in order to highlight the changes in number of macropoints and in the value of the weighted silhouette using the modified weighted clustering with this threshold. In Figure 4.32 the number of macropoints obtained with the normal weighted clustering and the one obtained with the modification are compared. In Figure 4.33 the value of the weighted silhouette computed on the normal weighted clustering result and the value computed on the modified weighted clustering result are compared. In the modification only the threshold *thPercRepresentedPoints* = 4 is considered.

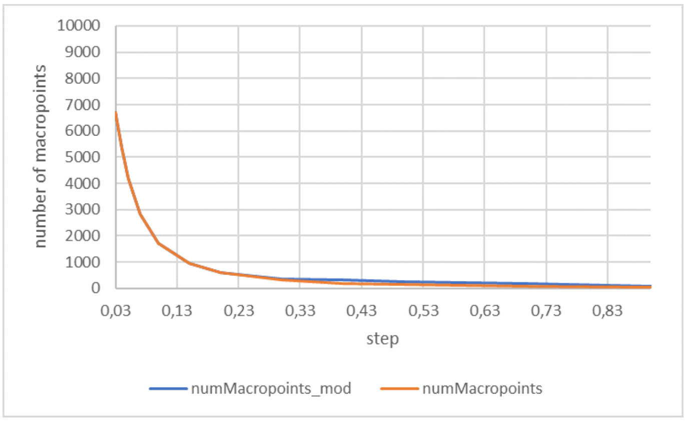

Figure 4.32: Number of macropoints obtained with the normal weighted clustering (*numMacropoints*) and with the modified weighted clustering (*numMacropoints_mod*). In the modification only the threshold *thPercRepresentedPoints* = 4 is considered.

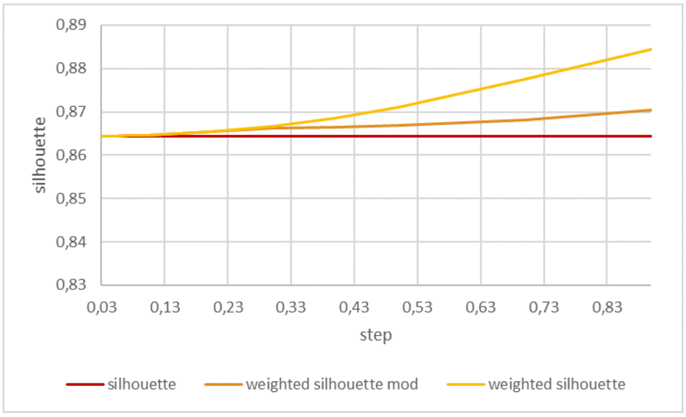

Figure 4.33: Weighted silhouette computed on the normal weighted clustering result (*weighted silhouette*) and on the modified weighted clustering result(*weighted silhouette mod*). In the modification only the threshold $thPercRepresentedPoints = 4$ is considered.

As expected, the trends are totally similar to the previous ones, so the *thPercStddev* threshold seems to be useless. It is necessary to say that the apparent uselessness of the threshold has been noticed only for this particular data set and for these values of *step*. For time reasons no more studies on this topic have been done. It has to be noticed that the aim of the *thPercStddev* in the weighted clustering algorithm is to detect the sparsity of points in cells, while the *thPercRepresentedPoints* is used to detect the cells with an high number of points. Using only the second threshold would mean miss the check on the sparsity. Moreover, the standard deviation shold be computed anyway because it is used in the calculation of the *newStep*, so the execution time of the algorithm would be only slightly reduced by the elimination of the *thPercStddev*. In future studies, if this threshold would be discovered to be not necessary, a different way to compute the *newStep* could be defined in order to totally avoid the computation of the standard deviation. In this way, the execution time of the weighted clustering algorithm would be reduced by a significant factor.

4.4.3 Silhouettes results on weighted clustering modification

As in Section 4.4.1, the *Blobs* data set clustered using K-means algorithm is considered (reported in Figure 4.1 of Section 4.1).

The weighted silhouette computed on the modification of the weighted clustering algorithm result is used to evaluate the quality of this algorithm. The chosen *step* values are the same used in Section 4.4.1, but using the modified weighted clustering the number of obtained macropoints is greater because some cells are splitted. Remember that the cells selected using the thresholds are then splitted using a new value of *step*: the *newStep*. This new value is computed for each cell C as:

$$newStep_C = step - \frac{\text{step} \cdot maxPercStddev_C}{100} \tag{4.5}$$

where *step* is the original value received as input parameter, $maxPercStddev_C$ is the maximum $percStddev_j$ among all dimensions j for cell C.

As $newStep < step$, the number of macropoints obtained with the modified weighted clustering will be higher than the one of the original weighted clustering. In Table 4.2 and in Figure 4.34, correspondences *step* - obtained number of macropoints with original weighted clustering - obtained number of macropoints with modified weighted clustering are displayed. The number of macropoints obtained with the modified version of the weighted clustering process is computed as the sum of the number of old macropoints (i.e. macropoints associated to cells not selected as to be splitted) and the number of new macropoints (i.e. macropoints computed for the new cells obtained with the *newStep*).

step	*numMacropoints*	$numMacropoints_{modified}$
0.03	6694	6694
0.04	5329	5329
0.05	4211	4211
0.07	2822	2822
0.1	1730	1730
0.15	953	953
0.2	607	607
0.3	315	361
0.4	203	322
0.5	141	260
0.7	78	175
0.9	52	96

Table 4.2: Number of macropoints obtained for *Blobs* data set with the weighted clustering (*numMacropoints*) and with the modified weighted clustering ($numMacropoints_{modified}$), depending on the value of *step*

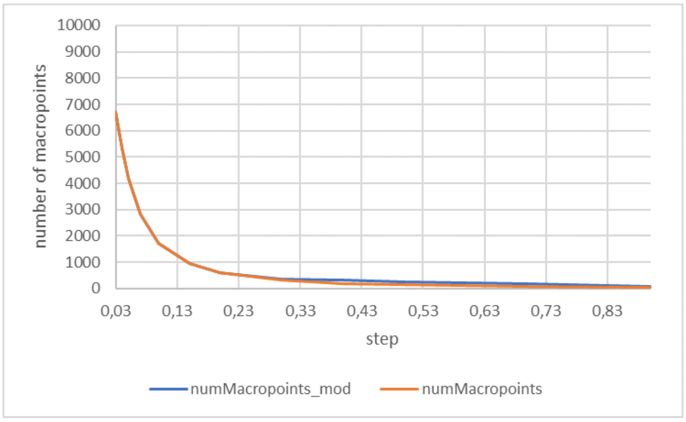

Figure 4.34: Comparison between number of macropoints obtained for *Blobs* data set with the weighted clustering (*numMacropoints*) and with the modified weighted clustering (*numMacropoints_mod*) depending on the *step* for *Blobs* data set

The number of macropoints obtained with the modification is the same as before for low *steps* and it grows up of a little factor for greater *steps*. The increasing is clearly minimal but the improvement is remarkable as shown in Figure 4.35, where the desired silhouette, the weighted silhouette on weighted clustering result and the weighted silhouette on modification of weighted clustering result are compared for each *step*.

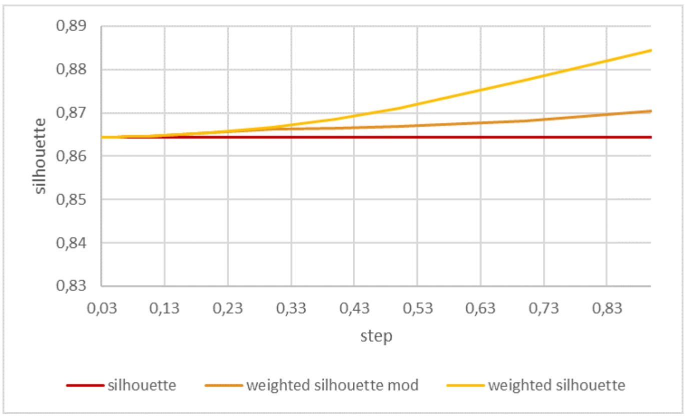

Figure 4.35: Comparison between desired silhouette (*silhouette*), weighted silhouette on weighted clustering result (*weighted silhouette*) and weighted silhouette on modified weighted clustering result (*weighted silhouette mod*)

The silhouette on the original data, the weighted silhouette on the result of the weighted clustering modification, the average of silhouettes on the sets of random clusters obtained with strategy 0, and the average of silhouettes on the sets of random clusters obtained with strategy 1 are computed for each *step* and are compared in the following figure. The number of points randomly selected in the two random strategies has been set to the number of macropoints obtained using the modified weighted clustering.

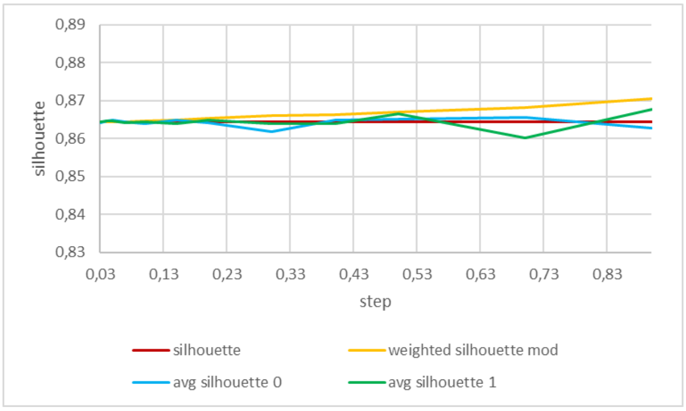

Figure 4.36: Silhouette on the original data (*silhouette*), weighted silhouette on weighted clustering modification (*weighted silhouette*), average silhouette on random clusters strategy 0 (*avg silhouette 0*), and average silhouette on random clusters strategy 1 (*avg silhouette 1*)

Even if for *Blobs* data set the two random strategies are the same, silhouettes on random clusters strategy 0 and on random clusters strategy 1 are slightly different among them and compared to previous results, because the selection of points is random.

The weighted silhouette calculated on modified weighted clustering result is much better than without the modification, but still worse than the average of random strategies. It has to be said that this happens only with this particular data set, probably because *Blobs* shapes are regular and compact so the choice of random points that are a good representation of the clustering is easy. Moreover, the average silhouette on random clusters is retrieved as the arithmetic mean of 10 silhouettes computed on 10 different results of random strategies. The resultant average of these silhouettes does not evaluate the quality of a clustering, so even if it is closer to the desired silhouette this does not mean that there is a set of random clusters that represents the original clustering better than the weighted clustering.

If only one set of random clusters is considered, the silhouette on it will probably be worse than our weighted silhouette. This is clear in the next figures, where the worst cases (minimum and maximum) of silhouettes on random clusters are shown specifically for strategies 0 and 1.

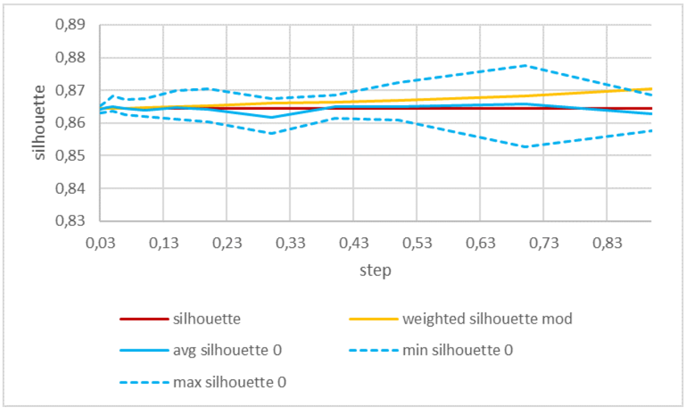

Figure 4.37: Details of the worst cases (*min silhouette 0* and *max silhouette 0*) of silhouettes on random clusters strategy 0

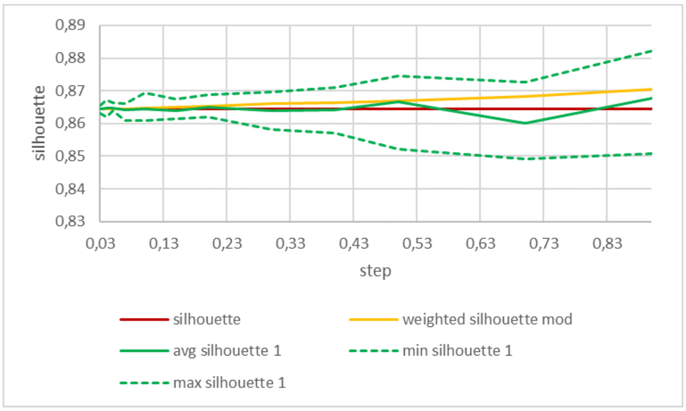

Figure 4.38: Details of the worst cases (*min silhouette 1* and *max silhouette 1*) of silhouettes on random clusters strategy 1

Comparing the weighted silhouette on modified weighted clustering with only one silhouette computed on only one set of random clusters, the modified weighted clustering results to be a better representation of the original clustering.

4.4.4 Time analysis

Times information have been collected during the execution of the algorithms for the reduction of cardinality (weighted clustering and random strategies) and also for the silhouette and weighted silhouette computation. In the following charts all times are reported in nanoseconds. Times collected during the executions, with the different values of *step*, of all these algorithms are displayed in Figure 4.39.

Silhouette time indicates the time to compute the silhouette index on the original clustering containing all the data. *Weighted clustering time* is the time to perform the weighted clustering algorithm: split the clusters using the *step* and calculate the macropoints and their weights. The time to compute the weighted silhouette on the result of the weighted clustering (without modification) is the *Weighted silhouette time*. *Random clusterings 0 time* refers to the time to generate the 10 sets of random clusters applying strategy 0 for 10 times. *Silhouettes 0 time* is the time to compute the silhouettes indices on the 10 results of random strategy 0 and calculating their average. *Random clusterings 1 time* refers to the time to generate the 10 sets of random clusters using strategy 1. *Silhouettes 1 time* indicates the time to compute the silhouettes index on the results of random strategy 1 and computing their average.

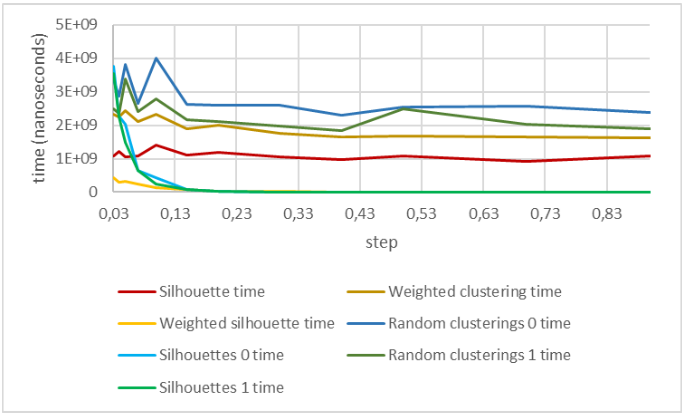

Figure 4.39: Comparision between times of separated processes

The *Silhouette time* should be a constant since the silhouette computation on the original clustering is always the same. Little variations in this value are probably caused by variations of computer performance during the execution. It can be noticed that the *weighted clustering time* is the lowest among the clustering times (weighted clustering and random strategies) for all values of *step*. Also the *Weighted silhouette time* is lower than all other silhouettes computation times.

In the following figure, times of entire processes have been considered: starting from the original clustering until the silhouette computed on the representation of the original clustering with a reduced cardinality.

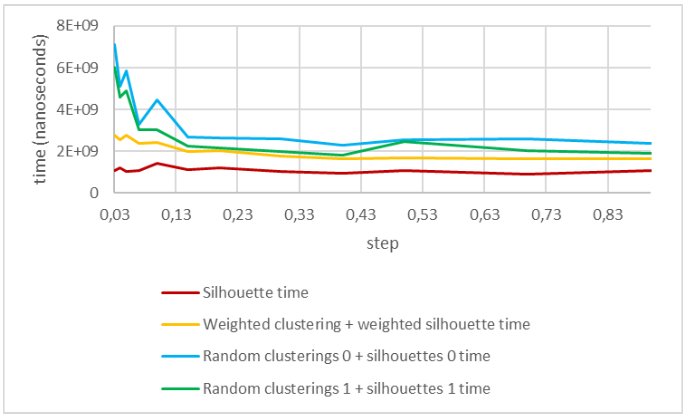

Figure 4.40: Comparison between times of entire processes

The *Silhouette time* is the lowest since it does not have the clustering part before the computation of the silhouette. Moreover, this value is low because the Blobs data set contains only 10000 points. Remember that the time complexity for the silhouette index is quadratic in the number of data. For higher cardinalities the computation of the silhouette on the whole original clustering is not even feasible. In these situations the use of algorithms like the weighted clustering is necessary.

Among the three processes that creates and evaluate the representation of the original clustering, the weighted clustering together with the weighted silhouette computation is clearly the fastest.

Considering the modification of weighted clustering algorithm, *Weighted clustering modification time* indicates the time necessary to apply the modified weighted clustering on the original clustering: primary split into cells, computation of standard deviation and other parameters, filter and selection of cells to be splitted, secondary split, calculation of the macropoints. *Weighted silhouette modification time* refers to the time for the computation of the weighted silhouette on the result of modified weighted clustering.

In figures 4.41 and 4.43, the same times considered before are shown for the modification of the weighted clustering.

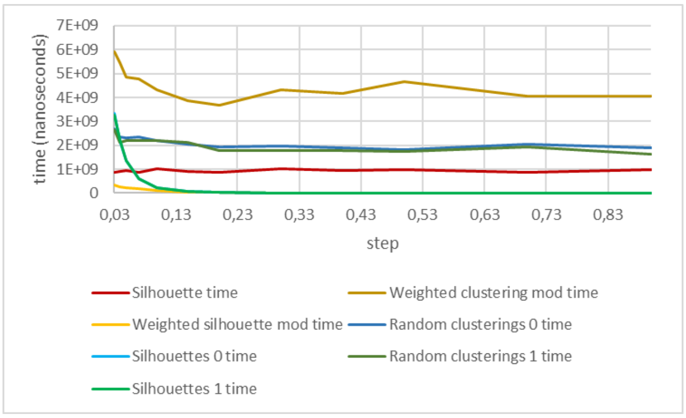

Figure 4.41: Comparison between times of separated processes

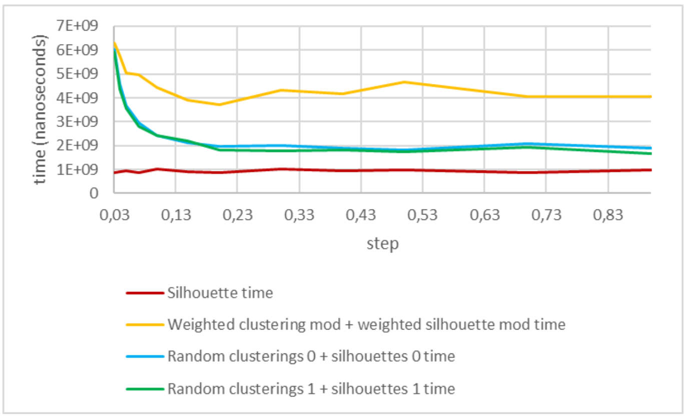

Figure 4.42: Comparison between times of entire processes

It is clear that the modification of weighted clustering causes a terrific increase in the required time for obtaining the weighted clusters. In the modification, even when no cells are splitted, a lot of time is used for the computation of all standard deviations and percentages in order to decide if a cell is critical or not. In the following figure times for weighted clustering with and without modification are compared.

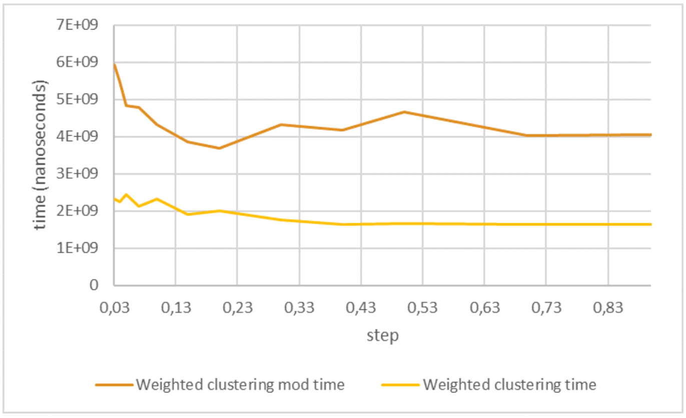

Figure 4.43: Times comparison between weighted clustering and modified weighted clustering

The rise of the computation time introduced by the modification is clearly caused by the computation of parameters used to decide if a cell has to be splitted or not. If the increase in the necessary time were caused by the split, the time would have been lower for low *steps*, were no cells are splitted. Since the time is high even for low *steps*, the cause is the computation of standard deviations, range of variations and other parameters for all the cells in order to select the ones to be splitted.

No longer research has been performed about this issue, but in the future further studies need to be done about how to lower computation time for the modified weighted clustering.

4.4.5 Clustering evaluation on randomly clustered data sets

In order to better evaluate the weighted and the random representations, another experiment has been performed. Starting from the *Blobs* data set, a new assignment of points to clusters has been performed. Given the number of clusters to form (three for *Blobs*), each point is randomly assigned to one of the clusters. The resultant clustering will have a very low quality since the position of points has not been considered during the partition in clusters. An example of random assignment of *Blobs* data to clusters can be found in Figure 4.7 of Section 4.1.

This randomly obtained clusters can be evaluated using the silhouette index. The index will be near -1 as the quality of the clustering is low. Anyway, applying the weighted clustering and the two random strategies, the weighted silhouette and the silhouettes on random results should be as close as possible to the desired silhouette.

The more they are close to the desired value, the more their reduced cardinality representation is a good representation of the original randomly obtained clusters. The experiment is the following.

- The random assignment of points to clusters is performed a number of times (four in this experiment), obtaining a number (four) of sets of random clustered data
- For each set, the silhouette index is computed, obtaining four values of desired silhouettes
- These values are sorted in descending order, obtaining a ranking of the four partitions in clusters based on their quality
- Weighted clustering, random strategy 0 and random strategy 1 are applied on each one of the four sets of random clustered points, using different values of *step*
- The weighted silhouette is calculated for all the obtained weighted clusterings and the silhouettes are computed on all the random results
- For each strategy and evaluation, the silhouettes retrieved for the four partitions in clusters are sorted
- The rankings of the four sets based on these silhouettes, should theoretically be the same as the ranking obtained using the silhouette index on all original data

The silhouettes obtained are shown in the following four figures, divided by clustering strategy. The desired silhouettes in the desired order are in the first figure, where the silhouette indexes computed on the four original sets of randomly clustered points are displayed. The quality ranking is indicated by the relative position of the silhouette lines.

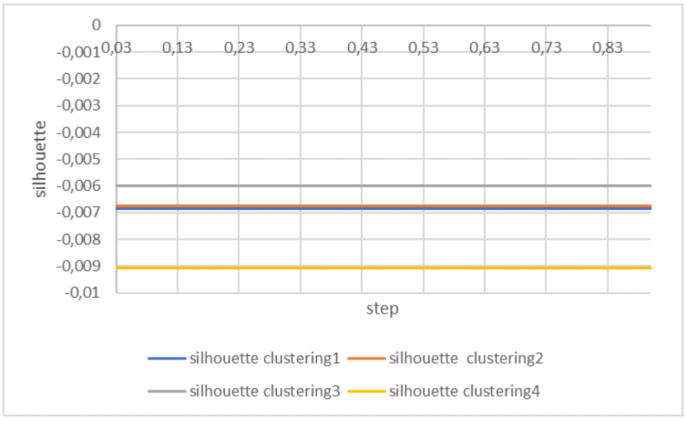

Figure 4.44: Silhouettes on the four original sets of randomly clustered points

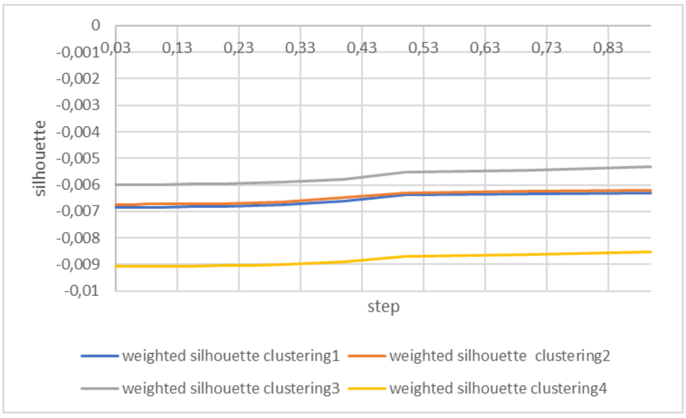

Figure 4.45: Weighted silhouettes on the weighted clusterings obtained from the four sets of randomly clustered points

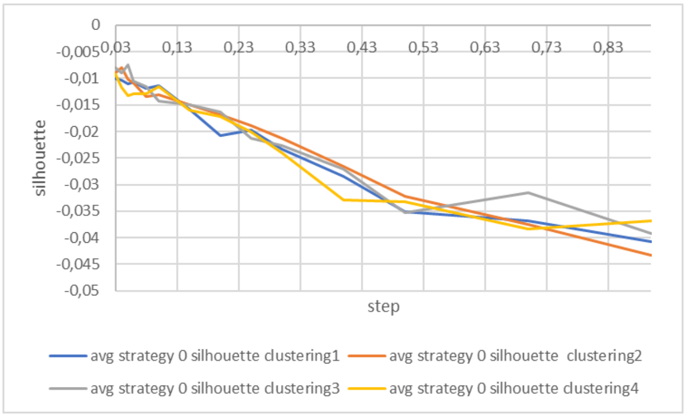

Figure 4.46: Averages of 10 silhouettes computed on 10 sets of clusters of random points, obtained applying strategy 0 on each of the the four sets of randomly clustered points

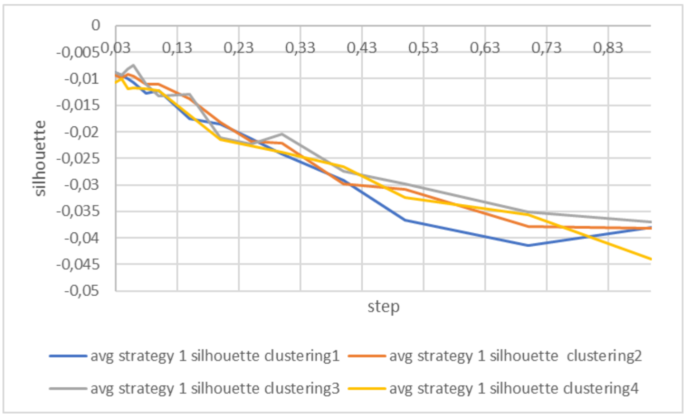

Figure 4.47: Averages of 10 silhouettes computed on 10 sets of clusters of random points, obtained applying strategy 1 on each of the the four sets of randomly clustered points

The figures show that only the weighted silhouettes, that represent the qualities of weighted clusterings, maintain the relative positions of the desired silhouettes, that represent the qualities of original random clusters. Moreover, it is evident that in these cases the weighted silhouette works way better than the random strategies. A comparison between the weighted silhouette on weighted clustering and the silhouettes on random results for strategies 0 and 1 is reported in Figure 4.48, where only one of the sets of randomly clustered points is considered.

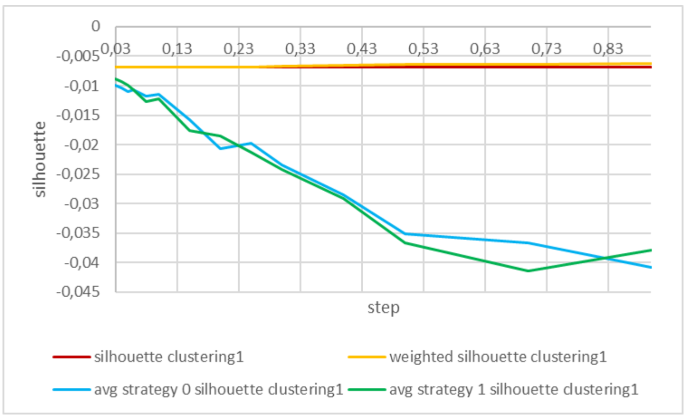

Figure 4.48: Silhouettes comparison for one of the sets of randomly clustered points

Clearly the weighted silhouette gives very good results in this case. This means that the quality of the weighted clustering is similar to the quality of the original clustering, and so for our criterion the weighted clustering is a good representation of the original. An explanation of why in this case the weighted clustering works much better than the random strategies is that the shapes obtained with the random assignment to clusters are really complex and so the random choice of some points that represents these clusters does not lead to good results (while with the simple circular shapes of the *Blobs* data set the random selection of points easily gave a good representation of the original clusters).

4.5 Other data sets results

4.5.1 Complex 8

The *Complex 8* data set (Figure 4.3) is way more complex than *Blobs*. It contains eight clusters with complex shapes that differ widely in shape, cardinality and density of points. For this reason, the strategy 1 for the random selection of points, that selects the same quantity of points from each cluster, works really bad. This proofs that if the distribution of data into clusters is not known, techniques of random selection of points can not be used.

Figure 4.49 shows the comparison for different *step* values between silhouette result on the original clustering, weighted silhouette on the weighted clustering and averages of silhouettes on sets of random clusters obtained with the two random strategies. The *step* values are always chosen in a way to guarantee numbers of macropoints that cover the range between $N_D/1.5$ (cardinality reduced by a $1/1.5$ factor) and $N_D/150$ (cardinality reduced by a $1/150$ factor), where N_D is the cardinality of the original data set (2551 for *Complex 8*).

In Figure 4.50 the same comparison is done with the weighted silhouette on the modified weighted clustering result. The thresholds values used in the modification of weighted clustering are the ones described in Section 4.4.2: *thPercStddev* = 15 and *thPercRepresentedPoints* = 4. The weighted silhouette on the modified weighted clustering is clearly closer to the desired value, compared with the previous result (i.e. the one without the modification). For high values of *step* the modified weighted clustering is the best representation of the original clustering. About random strategies, while the strategy 1 keeps giving bad results, the distance between the average silhouette on strategy 0 clustering and the desired silhouette is lower. This is caused by the modification of the weighted clustering that leads to a greater number of macropoints (remember that the number of points to be randomly selected in random strategies is set to the number of macropoints obtained with the weighted clustering). Since the number of randomly selected points in random strategies is equal to the number of macropoints, in Figure 4.50 also the average random silhouette on strategy 0 results knows an improvement.

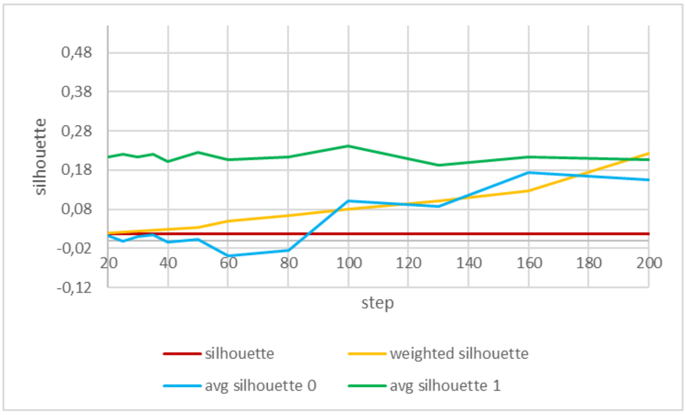

Figure 4.49: *Complex 8* data set: silhouette on original data (*silhouette*), weighted silhouette on weighted clustering (*weighted silhouette*), average silhouette on results of strategy 0 (*avg silhouette 0*), and average silhouette on results of strategy 1 (*avg silhouette 1*)

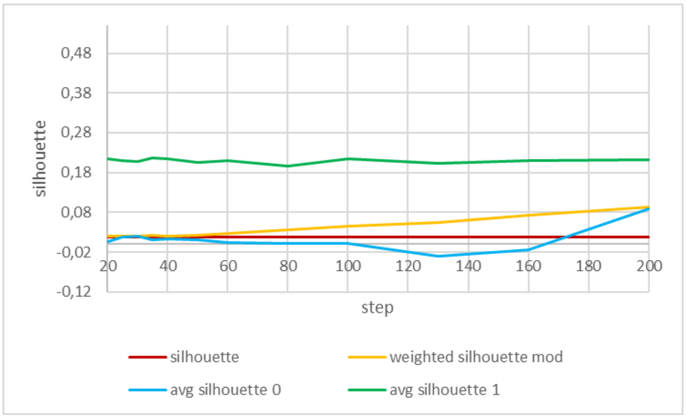

Figure 4.50: *Complex 8* data set: silhouette on original data (*silhouette*), weighted silhouette on weighted clustering modification (*weighted silhouette mod*), average silhouette on results of strategy 0 (*avg silhouette 0*), and average silhouette on results of strategy 1 (*avg silhouette 1*)

4.5.2 Complex 9

The *Complex 9* data set (Figure 4.4) contains nine clusters different in shape, size and density as in *Complex 8* but with more complex shapes. As in *Complex 8*, random strategy 1 gives terrible results while strategy 0 knows a little improvement in the second figure, where the modified weighted clustering is considered and so the number of considered points is higher. Remember that the values reported in the graphs are obtained as an average of 10 silhouettes computed on 10 different set of random clusters so these silhouettes do not indicate the qualities of real representations.

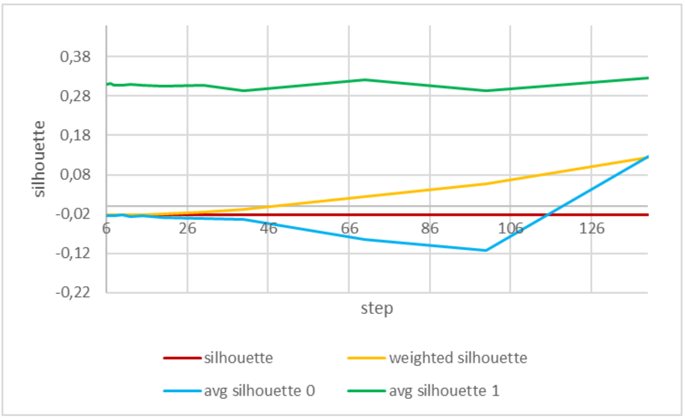

Figure 4.51: Silhouettes comparison for *Complex 9* data set

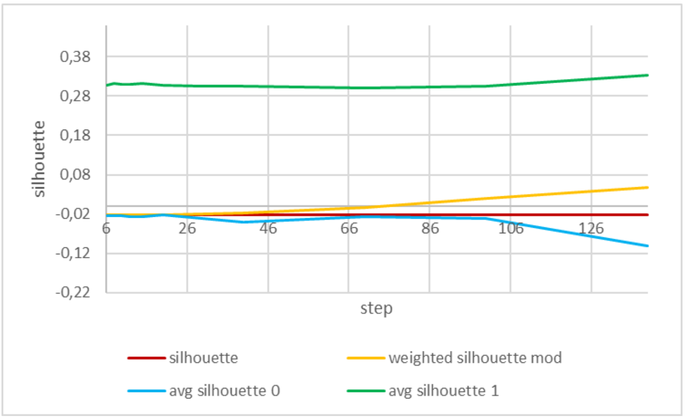

Figure 4.52: Silhouettes comparison for *Complex 9* data set considering the modified weighted clustering

4.5.3 Compound

The *Compound* data set (Figure 4.5) contains six clusters that again differ a lot in shape, size and density of points. In particular it contains some clusters with really scattered points. Once again, since the number of points in each cluster is not the same, strategy 1 gives terrible results. In Figure 4.53, where the standard weighted clustering is considered, the average of silhouettes obtained using strategy 0 starts to fall for *step* values around 5, and falls to 0 for *steps* greater than 8. This happens because the *Compound* data set contains only 399 points distributed among six clusters. When a low number of points is selected for the random strategies (i.e. when the used *step* leads to a low number of macropoints) it is possible that for some clusters just one point is randomly selected. By definition, if a cluster contains only one point, the silhouette of the point is 0. In this work, the silhouette index is set to 0 also for empty clusters. This leads to a decrease in the silhouette value of the whole data set. In case of strategy 0, the number of points is distributed among the clusters proportionally to their points, so even when it is low there are clusters with more than one points. These points have silhouette greater than zero and so the resultant silhouette index, being the average of all silhouettes, is greater than zero. In strategy 1, being the number of selected points equal for all the clusters, for high *step* values all clusters contain zero or one point and so the resultant silhouette is 0. In Figure 4.54 the modified weighted clustering is considered. Being the number of macropoints higher due to the split of some cells, also the number of points to be randomly selected is higher and so the average silhouette on the result of random strategy does not fall to 0. In both figures the distance between the weighted silhouette on weighted clustering and the desired silhouette has a dramatic increase for high *step*. This is probably caused by the spread of the points in some clusters. Looking at Figure 4.5 it is evident that points in some clusters are really scattered. This is partially corrected by the modification of the weighted clustering with the split of cells with scattered points. Figure 4.55 shows the average among all the clusters percentages of splitted cells using the thresholds $thPercStddev = 15$ and $thPercRepresentedPoints = 4$. This percentage is obtained for each cluster as:

$$percCellsToBeSplitted_k = \frac{cellsToBeSplitted_k \cdot 100}{cells_k} \tag{4.6}$$

where C_k is the considered cluster, $cellsToBeSplitted_k$ is the number of cells in the cluster selected to be splitted and $cells_k$ is the total number of original cells i the cluster.

It is clear that for high *steps* almost all cells are splitted, so the thresholds values are not the problem. The reason why the split of the cells is not enough (i.e. random strategy 0 gives better results) can be the value of *newStep* which is not enough low. Remember that the *newStep* is obtained for each cell as:

$$newStep_C = step - \frac{\text{step} \cdot maxPercStddev_C}{100} \tag{4.7}$$

where C is the considered cell, *step* is the original value received as command line argument and $maxPercStddev_C$ is the maximum percentage of standard deviation computed for cell C among all the dimensions.

Even if the $maxPercStddev_C$ is high, it is not enough to guarantee an adequate value of *newStep*. It has to be said that a too low value of *newStep*, since almost all cells are splitted, would lead to a weighted clustering very similar to the one obtained with a *step* near to the hypothetical low value of *newStep*. The number of macropoints retrieved with a low *newStep* would be high (i.e. it wold be almost the same as the number of macropoints obtained with a $step = newStep$) and so the aim of the reduction of cardinality as the *step* grows wold not be accomplished.

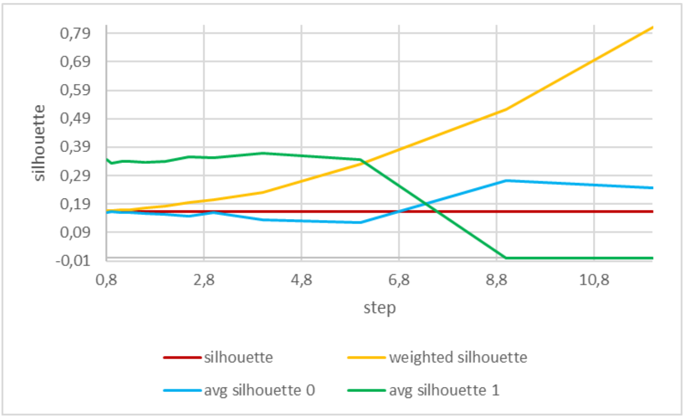

Figure 4.53: Silhouettes comparison for *Compound* data set

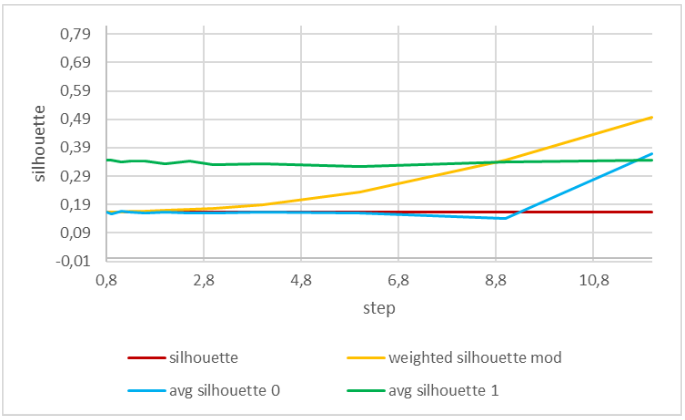

Figure 4.54: Silhouettes comparison for *Compound* data set considering the modified weighted clustering

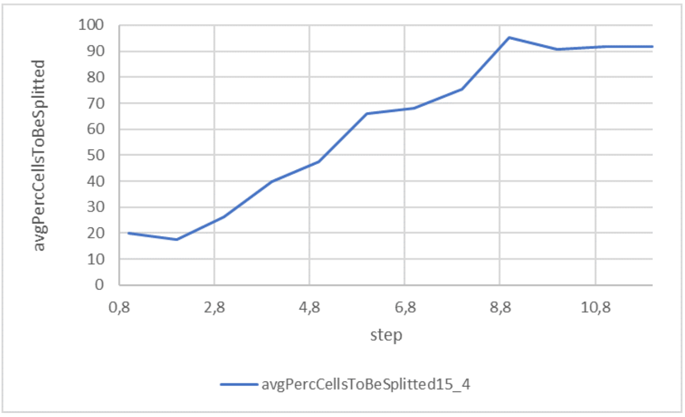

Figure 4.55: Average percentage of cells selected to be splitted for data set *Compound*, obtained as the mean of percentages of all clusters

4.5.4 Cure-t1-2000n-2d

The *Cure-t1-2000n-2d* data set (Figure 4.6) contains six clusters that differ a lot in shape and size. Differently from *Compound*, clusters of *Cure-t1-2000n-2d* data set are dense.

Strategy 1 gives bad results because the number of points in each cluster is not the same. Strategy 0 works well, probably due to the simple round shapes. Always remember that the average silhouette represented for random strategies is the mean of 10 different silhouettes. The modification of the weighted clustering works well in this case. For high *steps* the weighted silhouette is the one that gives the best results (the closest to the desired value).

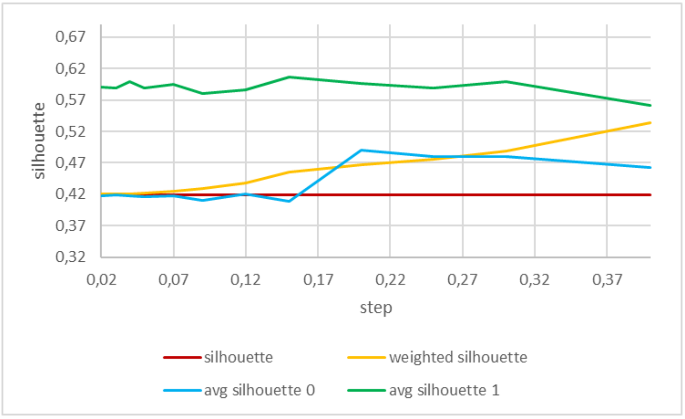

Figure 4.56: Silhouettes comparison for *Cure-t1-2000n-2d* data set

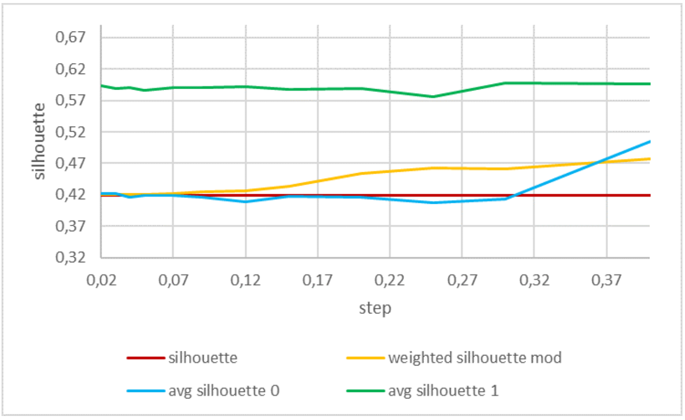

Figure 4.57: Silhouettes comparison for *Cure-t1-2000n-2d* data set considering the modified weighted clustering

4.5.5 Moons

The *Moons* data set (Figure 4.2) is really simple. It contains only two clusters with the same shape and size. Strategy 1 is the same as strategy 0 since the distribution of points in clusters is regular. The division in clusters does not follow the two moons that human eyes immediately notice because the considered clustering is not the optimal one but the one obtained applying the K-means algorithm.

Given the simplicity of this data set all strategies give pretty good results. The weighted clustering is better than the random strategies since it is predictable and gives almost perfect results. The quality of random strategies results is partly fortuitous, it is obtained as an average of 10 trials and, as can be seen in next figures, it is always worse than the quality of weighted clustering. The modified weighted clustering gives results even better than the weighted clustering, splitting some cells only for high values of *step* were the weighted silhouette is slightly worse. The split happens only for high *steps* where the weighted silhouette starts to be greater than the desired silhouette. Since the distance between the weighted silhouette and the desired value is always minimal, it is desirable for the number of macropoints obtained with the modified weighted clustering to remain low. As shown in Figure 4.60 this is satisfied, so not useless increase of cardinality is introduced with the modification of the weighted clustering.

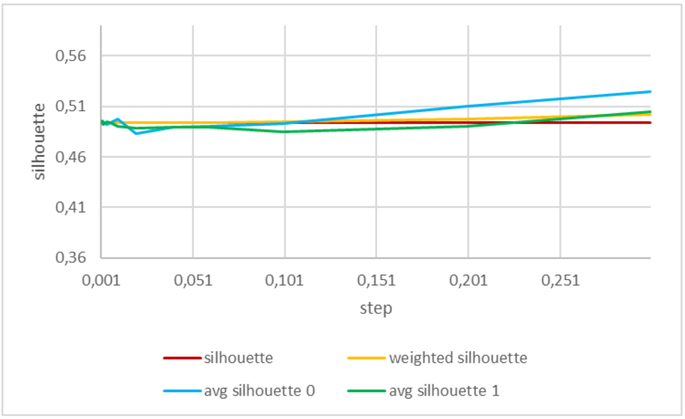

Figure 4.58: Silhouettes comparison for *Moons* data set

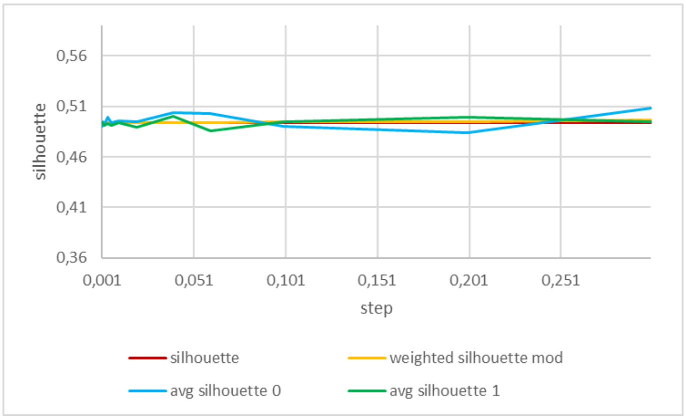

Figure 4.59: Silhouettes comparison for *Moons* data set considering the modified weighted clustering

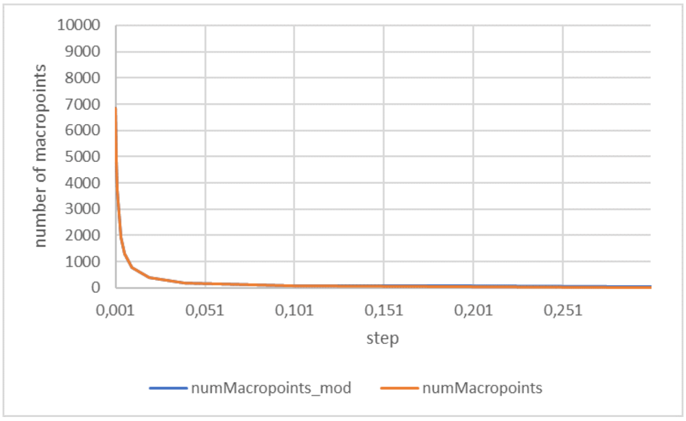

Figure 4.60: Comparison between number of macropoints obtained with the weighted clustering and with the modified weighted clustering for the *Moons* data set

Chapter 5

Conclusions and future works

The goal of the representation of a number of clustered data using a lower number of records has been achieved. Using the weighted clustering algorithm it is possible to represent an original set of clustered data with a new set of clustered weighted data. The number of weighted data is lower than the original, so operations on the obtained weighted clusters will be faster and easier. The reduction factor can be set as needed by setting the value of the *step* parameter before applying the weighted clustering algorithm.

If the aim is the reduction of the cardinality, the algorithm presented in this work is a valid solution among the few ones available at the moment. As showed in this work, the random strategy that does not consider the distribution of objects into clusters (i.e. random strategy 1) makes no sense for all data sets containing no evenly distributed data. The random strategy that considers the data distribution (i.e. random strategy 0) gives pretty good results of silhouette index only when performed a number of times, computing the silhouette index on a number of clusters of randomly selected points, and finally considering an average of these silhouettes. These silhouette results are not referred to any existing single representation of the original data, so if a single representation is needed random strategies are not valid solutions.

The goal can also be the computation of the silhouette index for huge data sets, when the index can not be applied since its time complexity is quadratic in the number of considered data. Even if in some cases random strategy 0 seems to give results with the same quality in less time, it has to be considered that this happens only with simple shapes. In real world clusters shapes are more complex, and the records are probably more than two-dimensional. In these situations, a randomly selection of some points that successfully represent original points is clearly not usual. For this reason the weighted clustering used together with the weighted silhouette seems to be a better solution to obtain an approximation of the silhouette index.

With the introduction of the modification in the weighted clustering algorithm, a little increase in the number of macropoints leads to a huge improvement in the quality of the weighted clustering, measured by the weighted silhouette. The modification consists in the computation of standard deviation and cardinality of each cell, the checks on them, the possible split of some cells and the calculation of new macropoints for the new cells. It causes a crucial rise in the weighted clustering computation time. Even if no cells or a low number of cells are critical (i.e. have high standard deviations and contain high number of points), the requested time is long because of the computation of all the parameters used to decide if a cell is critical.

This part can be optimized, for example avoiding the calculation of all parameters when the cell is clearly not critical. A simple improvement that can be tried is to check the number of points represented by the macropoint of the cell before the computation of all other parameters: if *percRepresentedPoints* $<$ *thPercRepresentedPoints* the cell will not be splitted so the computation of range of variation and standard deviation is not useful. Other experiments can be performed to find if the *thPercStddev* is really needed to select the cells to be splitted. If it is not, another way to compute the *newStep* value can be find and the computation of the standard deviation can be avoided. In this thesis, for time reasons, no more researches and trials have been performed.

About the weighted silhouette, the computation time is low and the results are the desired ones. As shown in the weighted silhouette example (Section 3.2.2) the weighted silhouette of a weighted clustering containing macropoints is exactly equal to the silhouette computed on a clustering containing standard points obtained as follows: given a macropoint with coordinates $(m_x,\ m_y)$ and weight w, a number w of standard points are considered, all with coordinates $(m_x,\ m_y)$. Concerning the weighted silhouette no further work is necessary. It can be used as it is to evaluate the results obtained with the future developments of weighed clustering algorithm. Given any process that produces clusters of weighted objects that are an approximation of some original clusters, the weighted silhouette index presented in this thesis is a solution of great value to retrieve an approximate value of the silhouette index.

Bibliography

[1] A. Starczewski and A. Krzyfffdfffdak, *A Modification of the Silhouette Index for the Improvement of Cluster Validity Assessment.* LNCS, 2016.

[2] A. K. Jain, *Data clustering: 50 years beyond K-means.* Elsevier, 2010.

[3] R. Xu and D. Wunsch, *Survey of Clustering Algorithms.* IEEE, 2005.

[4] A. Jain, M. Murty, and P. Flynn, *Data Clustering: A Review.* ACM Computing Surveys, 1999.

[5] T. R. dos Santos and L. E. Zfffdfffdrate, *Categorical data clustering: What similarity measure to recommend?* Elsevier, 2014.

[6] X. Gao and M. Yang, *Understanding and Enhancement of Internal Clustering Validation Indexes for Categorical Data.* MDPI, 2018.

[7] G. Menardi, *Density-based Silhouette diagnostics for clustering methods.* Springer Science+Business Media, 2010.

[8] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, *On Clustering Validation Techniques.* Journal of Intelligent Information Systems, 2001.

[9] *Spark 2.4.0 documentation - clustering*, https://spark.apache.org/docs/latest/-ml-clustering.html.

[10] *Euclidean and euclidean squared*, http://www.improvedoutcomes.com/docs/-WebSiteDocs/Clustering/Clustering_Parameters/Euclidean_and_Euclidean-_Squared_Distance_Metrics.htm.

[11] J. Yu, M.-S. Yang, and E. S. Lee, *Sample-weighted clustering methods.* Elsevier, 2011.

[12] J. Li, X. Gao, and L. Jiao, *A novel typical-sample-weighted clustering algorithm for large data sets.* LNAI, 2005.

[13] C. Zhang, Q. Shi, and D. Xue, *Document clustering algorithm based on sample weighting*. Journal of The China Society For Scientific and Technical Information, 2008.

[14] M. Studer, *WeightedCluster Library Manual: A practical guide to creating typologies of trajectories in the social sciences with R*. LIVES Working Papers 24, 2013.

[15] S. Petrovic, *A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters*. Proceedings of the 11th Nordic Workshop of Secure IT Systems, 2006.

[16] *Scikit learn - api reference*, https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets.

[17] Q. Yuan, H. Shi, and X. Zhou, *An Optimized Initialization Center K-means Clustering Algorithm based on Density*. IEEE, 2015.