Dealing with Repeated Objects in SNNagg

João Galvão, Maribel Yasmina Santos, João Moura Pires, and Carlos Costa

***Abstract*—Due to the constant technological advances and massive use of electronic devices, the amount of data generated has increased at a very high rate, leading to the urgent need to process larger amounts of data in less time. In order to be able to handle these large amounts of data, several techniques and algorithms have been developed in the area of knowledge discovery in databases, which process consists of several stages, including data mining that analyze vast amounts of data, identifying patterns, models or trends. Among the several data mining techniques, this work is focused in clustering spatial data with a density-based approach that uses the Shared Nearest Neighbor algorithm (SNN). SNN has shown several advantages when analyzing this type of data, identifying clusters of different sizes, shapes, and densities, and also dealing with noise. This paper presents and evaluates a new extension of SNN that is able to deal with repeated objects, creating aggregates that reduce the processing time required to cluster a given dataset, as repeated objects are excluded from the most time demanding step, which is associated with the identification of the *k*-nearest neighbors of a point. The proposed approach, SNNagg, was evaluated and the obtained results show that the processing time is reduced without compromising the quality of the obtained clusters.**

***Index Terms*—Spatial Data, Spatio-Temporal Data, Clustering, Density-based Clustering, SNN.**

I. Introduction

In 1996, [1] claimed that “There is an urgent need for a new generation of computational theories and tools to assist humans in extracting useful information from the rapidly growing volume of digital data”. Several years have passed and several computational theories and tools emerged to improve the capability to handle large amounts of data. However, due to the ever-increasing volume of data that organizations are able to collect, there is still the need for new algorithms with increased performance.

Clustering is an unsupervised learning method that constitutes a cornerstone of an intelligent data analysis process [2], being capable of grouping a set of objects in classes of similar objects [3]. From the several clustering approaches, the density-based one showed to be especially appropriated for the analysis of spatial data [4]–[7]. In particular, the Shared Nearest Neighbor (SNN) algorithm, which presents as main advantages the capability of identifying clusters of different sizes, shapes, and densities, as well as being able to deal with noise [8].

Although several improvements have been made to the SNN performance [5], [6], which is constrained by the identification of the *k*-nearest neighbors of an object, new developments are still needed as the volume of data that needs to be processed continues to grow.

This paper presents a new extension of the SNN algorithm, which is able to exclude repeated objects from the clustering process, thus reducing the number of points in the most demanding step, being able to add those points to the obtained clusters, without compromising the quality of the results. The proposed approach is named SNNagg (SNN with aggregates) due to the fact that a given point may represent an aggregate of points in the clustering process, assuming that huge datasets may include a relevant number of repeated points.

The obtained results show that the time needed to process the dataset decreases, also presenting results that do not compromise the analytical task, as the identified clusters are able to properly represent the expected reality, here evaluated using synthetic datasets where the results are previously known, and real datasets where the results are evaluated by previously defined quality measures.

The outline of this paper is as follows. Section III briefly describes clustering approaches and emphasizes the advantages of a density-based approach in the analysis of spatial data, as well as presents in more detail the SNN algorithm. Section II introduces the related work about SNN and its several extensions or variants. Section IV describes the new approach proposed in this paper, the SNNagg (SNN with aggregates), while Section V discloses the quality of the obtained clusters and evaluates the processing time and the impact of the number of repeated objects in the clustering results. Section VI summarizes the presented work and reveals directions for future work.

II. Clustering and The SNN Algorithm

This section provides an overview on clustering approaches and describes the SNN algorithm, the basis for the work here presented.

A. Clustering Approaches

Clustering is the process of grouping large datasets where objects in the same group should be as similar as possible and different to objects in other groups. It is known as unsupervised learning as no a priori information about the data is required [3]. Clusters emerge naturally from the data under analysis using some distance function used to measure the similarity among objects. This technique is classified in

Manuscript received December 24, 2015. This work has been supported by FCT, *Fundação para a Ciência e Tecnologia*, within the Project Scope UID/CEC/00319/2013.

João Galvão is with the ALGORITMI Research Centre, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal

Maribel Yasmina Santos is with ALGORITMI Research Centre, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal (corresponding author, phone: +351-253-510308; fax: +351-253-510300; e-mail: maribel@dsi.uminho.pt).

João Moura Pires is with the NOVA-LINCS Lab, New University of Lisbon, Quinta da Torre P-2829-516, Lisboa, Portugal (jmp@fct.unl.pt).

Carlos Costa is with the ALGORITMI Research Centre, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal.

four main categories [3], [9]: Partition Clustering, Hierarchical Clustering, Density-based Clustering and Grid-based Clustering.

In the first category (Partition), the dataset is decomposed into a set of clusters, being the number of clusters defined by a *k* number, which is a parameter given by the user. Each cluster must contain at least one object and each object belongs to exactly one or none group. Most partitioning methods are distance-based and can be divided into two groups: Centroid-based or Representative Object-based techniques. The first one defines the centroid of a cluster as the mean value of the points within the cluster, being *k-means* one of the most well-known centroid-based algorithms [10]. The second technique derives from the first one. The way the cluster is characterized is defined from a measure (for example, average distance) between a point and the point that defines the cluster. Some of the algorithms used in this technique are *k-Medoids*, PAM (Partitioning Around Medoids), CLARA (Clustering Large Applications) and CLARANS (Clustering Large Applications based on Randomized Search) [3].

The hierarchical clustering category groups the data objects into hierarchies or “trees” of clusters, following two different approaches: divisive algorithms or agglomerative algorithms. In divisive, the algorithm starts by considering that all the objects are in one group and, after that, it starts to successively divide this group into two or more groups if necessary. This iterative process stops when the maximum number of clusters is reached or the adopted metric indicates that the obtained set of clusters is the best possible solution. The second strategy is the opposite of the first one. It starts by considering that each object is a group and then successively integrates clusters to form new clusters. The most cited algorithm that uses divisive techniques is CURE (Clustering Using REpresentatives) while some algorithms that use agglomerative techniques are BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), Chameleon and ROCK (RObust Clustering using linKs) [9].

Unlike partitioning and hierarchical methods, density based algorithms identify clusters independently of their shape. Typically, they classify dense regions as clusters and classify as noise regions with low density of objects. Some density-based algorithms are SNN, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify the Clustering Structure) and DENCLUE (DENsity-based CLUstEring) [3].

The last category is the Grid-based clustering in which the space is divided into a finite number of cells creating a grid structure. After that, all the operations for clustering are performed in each cell. Some of the algorithms used in this category are STING (Statistical Information Grid), WaveCluster and CLIQUE (CLustering In QUEst) [9].

After reviewing the several categories of clustering and considering that this work aims to analyze spatial data, density-based approaches were selected as these algorithms can handle noise, outliers and can create clusters of different sizes and shapes. Moreover, other advantages have been pointed, including [9], [11]–[14] that previous knowledge of the data set is not required (there is no need for the number of clusters as an input parameter) and that it is possible the identification of an arbitrary number of clusters with different densities to better fit the data under analysis.

Density-based algorithms usually require a set of input parameters like the radius of the neighborhood or the number of neighbors, which can be used to control the type of expected result, from less clusters with more points, to more clusters with less but more similar points [4], [15].

B. Shared Nearest Neighbor (SNN)

The SNN algorithm is a density-based clustering algorithm proposed by [8]. It has the capability of identifying clusters of different shapes, sizes and densities, as well as the ability to deal with noise, which makes it particularly suited for the analysis of spatial data. The algorithm computes a list of the *k*-nearest neighbors for each point using a distance function, usually the Euclidean or the geographical distance. The SNN is based on the notion of similarity and defines this similarity between points by calculating the number of nearest neighbors that two points share. The density of a point is the number of neighbor points within a given radius. Points with high density are classified as core points and points with low density will become noise points [9]. This similarity definition between points allows the algorithm to deal with datasets of variable density, being able to identify clusters with those different densities [8].

This algorithm needs three input parameters: *k*, *Eps* and *MinPts*. *K* is the number of neighbors, *Eps* defines the threshold density and *MinPts* is the minimum density that a point has to have to be considered a core point [8].

The most important input parameter is *k* (neighborhood list size) because it strongly influences the granularity of the clusters. If *k* is too small, even a uniform cluster will be split into several clusters and because of that, the algorithm will have a tendency to find many small, but tight, clusters. On the contrary, if *k* is too high, the algorithm will find only a few large, well separated clusters [8].

The main steps of the SNN algorithm can be briefly summarized as [8]:

1. Compute the similarity matrix. This is a similarity graph in which objects are represented as nodes and whose edges include a weight that define the similarity between objects;
2. Sparse the similarity matrix by keeping only the *k* most similar neighbors of a point. Keep only the *k* strongest links of the previous similarity graph;
3. Construct the shared nearest neighbor graph from the sparse similarity matrix. In this step we can apply the similarity threshold and find the connected components to obtain the clusters [16];
4. Find the SNN density of each point. Using the user-defined parameter *Eps*, find the number of points that have a SNN similarity equal or greater than *Eps* to each point. This is the SNN density of the point;
5. Find the core points. Using other user-defined parameter *MinPts*, find the core points, meaning all points that have a SNN density equal or greater than *MinPts*;
6. Form clusters from the core points. If two core points are within the radius of *Eps* of each other, they are placed in the same cluster;

7. Discard all noise points. All non-core points that are not within a radius of *Eps* of a core point are discarded;
8. Assign all the other points to clusters: non-noise and non-core points are assigned to the nearest core point.

Considering these 8 steps and the need of 3 input parameters (*k*, *Eps*, and *MinPts*), the SNN algorithm is now represented as in Fig. 1, where *k* is used to set the number of nearest neighbors that need to be identified for each point, *Eps* for filtering points attending to the calculated densities and *MinPts* to identify the core points used in the clusters construction. The representation will be later used to identify the steps added by the SNNagg approach.

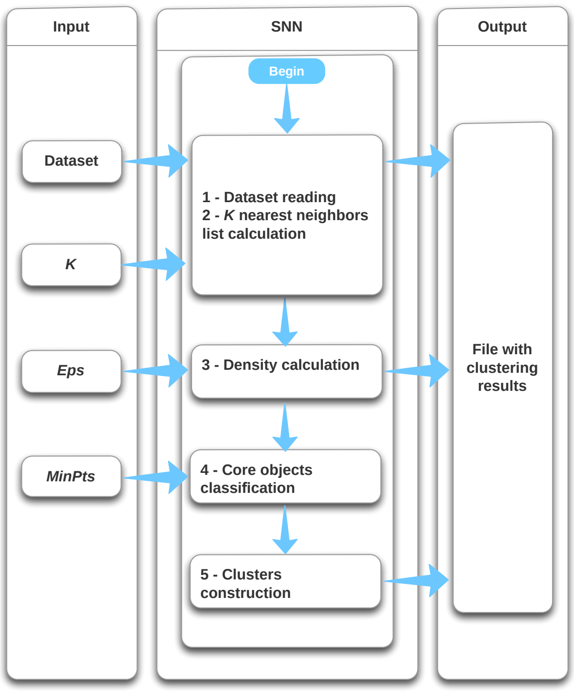

Input
SNN
Output
Begin
Dataset
K
Eps
MinPts
1 - Dataset reading
2 - K nearest neighbors list calculation
3 - Density calculation
4 - Core objects classification
5 - Clusters construction
File with clustering results

Fig. 1. SNN main steps

For measuring the distance between points, a distance function is needed. In the context of this work, given two objects or data points, p_1 $(<x_1, y_1>)$ and p_2 $(<x_2, y_2>)$, the distance between them is measured using Equation 1, which considers the Euclidean distance between the points.

$$Ds(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (1)$$

As will be seen in the following section, this distance function can integrate more dimensions of analysis, as the time dimension or semantic attributes, which enrich the analyses that can be done over a dataset.

III. Related Work

First introduced by [16] and then extended by [8], the SNN algorithm has been used due to its capability of identifying clusters with convex and non-convex shapes, having different sizes and densities, as well as its ability to deal with noise.

Several works were undertaken in order to improve SNN performance or to understand its behavior in what concerns the influence of its input parameters. As already mentioned, SNN has three input parameters, which values strongly influence the results that can be obtained.

To understand the behavior of the input parameters and their influence on the results, [17] identified an heuristic that, in an automatic way, finds a set of adequate input parameters. In this work, a strong correlation between *k* and *MinPts* was found and, also, it is mentioned that *Eps* is a less sensitive parameter, due to the wide range of values it can adopt for a pair of *k* and *MinPts* values. Although being *Eps* less sensitive, the authors show that it is possible to obtain better results for specific *Eps* values [18]. These values rely on the average number of arcs per node in the SNN graph, a value that depends of *k*. In terms of input parameters, the authors propose that *k* must be between 0,7 and 1% of *n*, being *n* the number of objects in the dataset. *MinPts* must have a value between 92% and 94% of *k* and *Eps* must be 18,5% of *MinPts*. The obtained values can be used as initial values for starting a clustering process with SNN. For a more detailed analysis of the influence of the input parameters in the clustering results, please see [17-18].

For improving the SNN performance, [6] implemented two versions of SNN that make use of metric data structures to improve the search in the *k*-nearest neighbors list. These two implementations benefit from different metric data structures: the *kd-tree*, which works on primary memory, and the *df-tree*, which works on secondary memory. Although the results from applying the *df-tree* were not impressive, the performance results for the primary memory implementation showed an effective improvement of the SNN performance. Also, the work of [5] presented an extension of the SNN, the Fast-SNN (F-SNN) approach, which divides the space into a matrix that optimizes the search for the *k*-nearest neighbors of a point, as the authors identified an heuristics that, depending on the used distance function, is able to limit the search space to the cells that can have possible neighbors. This approach emerged from the identification of the most inefficient step of the algorithm, which is the calculation of neighbors' list with a complexity of $O(n^2)$, due to the need of calculation of the similarity matrix between all points [19]. The obtained results showed an impressive decrease in the needed processing time.

Other extensions to the SNN include the work of [20], where improvements are made making available an incremental clustering approach that does not require the processing of all algorithm's steps every time new objects need to be added to the previously identified clusters. The SNN^{++}, an incremental version of SNN, maintains most of the SNN steps, with the advantage that new objects are included in previous existing clusters without the need to recalculate the nearest neighbor list, and consequently, to redo all the clustering process. Afterwards, this incremental version was extended to automatically adapt the input parameters following the [17] heuristics and also to be able to consider several dimensions in the distance function [21], like the spatial, temporal and one or more semantic dimensions, with a dynamic incremental version of SNN for clustering spatio-temporal data [7]. Considering the distance function expressed by Equation 1, this function is an instance of a more generic distance function defined by [21] and used by [7], in which more than 4 dimensions of analysis can be simultaneously considered. In this approach, 4D$^+$SNN, and considering two objects, p_1 ($< x_1, y_1, t_1, a_1 >$) and p_2 ($< x_2, y_2, t_2, a_2 >$), the distance between them is measured using Equation 2. Typically, *x* and *y* are the spatial coordinates, *t* the timestamp and *a* an additional attribute.

$$4D(p_1, p_2) = w_s * \frac{Ds(x_1,x_2,y_1,y_2)}{MaxS} + w_t * \frac{Dt(t_1,t_2)}{MaxT} + w_a * \frac{Da(a_1,a_2)}{MaxA} \qquad (2)$$

In the 4D function, the user can use any distance function (*Ds*, *Dt* and *Da*) to calculate the differences (respectively, spatial, temporal and semantic attribute) between the points. w_s, w_t and w_a are used to assign a weight to each one of these components (spatial, temporal and semantic dimensions). To guarantee that these weights are an effective way for controlling the dimension's relative importance in the distance function, it is necessary that the range of values for all distance components are on the same order of magnitude. One way to achieve that is by normalizing the computed values *Ds*, *Dt* and *Da*, in such a way that *Ds/MaxS*, *Dt/MaxT*, *Da/MaxA* become values of the same order of magnitude when the algorithm is calculating the *k*-neighbours lists. Without this normalization process, the integrated distance function may become strongly dependent of a single distance component if their values are excessively high, causing the other distance components to become irrelevant. In [21], a method for extracting *MaxS*, *MaxT* and *MaxA* from the dataset is proposed. Using this approach, the user can control the pretended results attending to the analytical context.

Besides all the improvements already made around SNN, there is still the need to optimize the analysis of vast amounts of data, as the size of the datasets continue to growth. In that sense, other works focus their attention in the aggregation of data using the hubness concept, as some points, in high density datasets, appear as neighbors in the neighbors' list more often than other points in the dataset [22-23]. This concept of hubness points recall for the need of reducing the number of points to cluster without compromising the quality of the clustering results. This was done by [23] with a two-steps methodology for clustering at different granularity levels, with a multi-granular hierarchical model where the datasets are generalized to obtain less detailed representations. With this approach, the clustering process can be first applied to a less detailed dataset and then be extended, to the objects that were filtered in the first step, without losing precision.

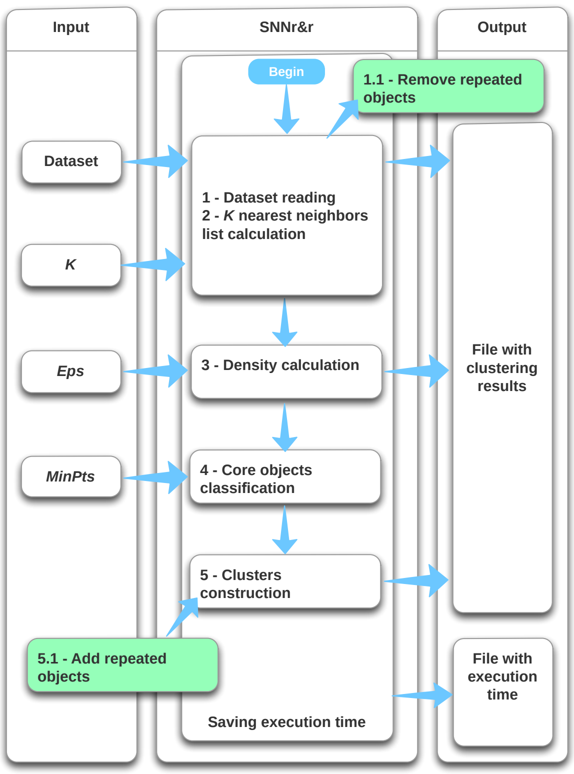
Following the principles of hubness and a clustering process that can filter out data points without compromising the clustering results, this paper presents an approach that removes repeated objects from the dataset to cluster, being able to add them latter to the obtained clusters without compromising the clustering results. This approach is presented in the next section.

IV. SNNAGG (SNN WITH AGGREGATES)

As the volume of data increases, also increases the similarity between the objects of those datasets. The main objective of this work is to improve the SNN performance on large datasets taking into consideration two premises: (1) the execution time needs to be reduced; (2) the number of objects in the dataset is also reduced introducing the notion of repeated objects. As the number of different objects to cluster decreases, also decreases the time needed to process those objects. In this work, the input dataset is analyzed in order to identify those repeated objects, to exclude them from the clusters' construction step, being those points later added to the identified clusters without compromising the clustering results.

As mentioned, when the size of a dataset grows, it is expected that the number of equal or similar objects increases. In this work, we are interested in repeated objects, as for similar objects an analogous approach can be followed as long as a similarity function is defined.

Removing repeated objects from the dataset would optimize the search for the *k*-nearest neighbors of an object, as fewer objects need to be tested, improving the overall SNN performance. We started our developments by removing repeated objects in the beginning of the clustering process, adding them at the end to the obtained clusters. The proposed approach was named SNNr&r (SNN Remove and Replace) and is shown in Fig. 2. Two steps were added to the scenario previously presented in Fig. 1, namely steps 1.1 (for removing repeated objects) and 5.1 (for adding the repeated points to the identified clusters). Moreover, and besides the file with the clustering results, another file is created to store the processing time, is order to verify the performance of this approach.


Input
SNNr&r
Output
Begin
1.1 - Remove repeated objects
Dataset
K
Eps
MinPts
1 - Dataset reading
2 - K nearest neighbors list calculation
3 - Density calculation
4 - Core objects classification
5 - Clusters construction
5.1 - Add repeated objects
Saving execution time
File with clustering results
File with execution time
Fig. 2. Main steps for the SNNr&r approach

The reading of a dataset (Fig. 2 – step 1) is a sequential process in which a new attribute is added to each object to store the information about the number of repeated objects it has. By default, this new attribute (NumberOfObjects) has the value of 0, being incremented by 1 each time a repeated object is found. All objects with a NumberOfObjects greater than 0 are called representative objects, as they will represent all the repeated ones. As we can see in Fig. 3, and taking into consideration a spatial dataset where five repetitions were introduced in a very specific area, the NumberOfObjects presents a summary of the dataset, giving a clear overview of the number of repeated objects that will be removed in step 1.1.

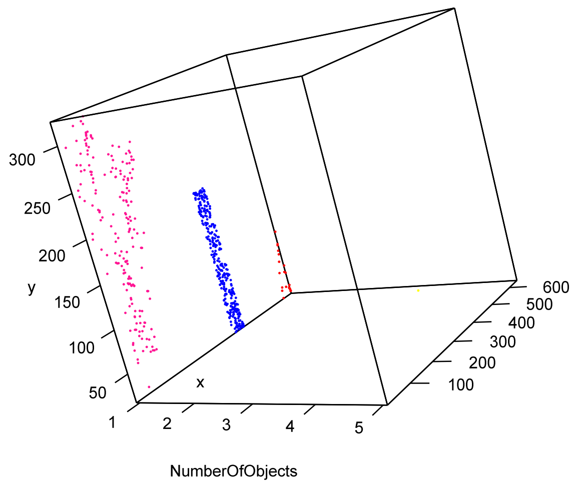


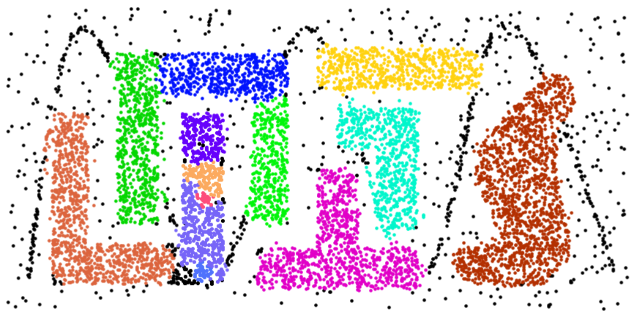
Fig. 3. Repeated objects spatial representation

After the identification and removal of all repeated objects, the clustering process proceeds as usually with a reduced dataset. The clusters are identified and for each representative object, all the repeated objects are assigned to the cluster where the representative is located (Fig. 2 – step 5.1).

The preliminary results obtained with this approach were very promising, showing in all tests a reduction of the processing time, being more efficient as more repeated objects are present in the analyzed dataset. However, when analyzing the quality of the results in terms of the clusters constitution, several differences were identified with regard to the Original SNN algorithm, mainly when the density of the datasets is supposed to affect the clustering results in the formation or merging of different clusters. Taking into consideration the sample dataset shown in Fig. 3, we can see that running the Original SNN the obtained result is different from the SNNr&r. While the first identified different clusters in the area where repeated objects were introduced (Fig. 4 a)), it also identified a new cluster were no changes were made to the original distribution of the points. In terms of the SNNr&r, it was unable to detect a denserest area, as the density of the points was not considered in the clustering process (Fig. 4 b)). In all the presented clustering results, points identified as noise are plotted in black.

Although the results do not need to be exactly the same, as the user may accept to have different results, as long as they are considered good enough for the analytical task in hands, this approach can be improved if the number of repeated objects has impact in the clustering process, as the nearest neighbors' lists are severely affected by the repeated objects. Although this and other tested datasets will be introduced in more detail in section V, it is worth mentioning that this dataset integrates in its original version 8000 points and that, in this case, 2872 repeated objects were randomly introduced in the area identified in Fig. 3.

In terms of the input parameters for each run of the SNN, either for the Original one or the SNNagg, it is important to mention that all of them were calculated considering the heuristics identified in the work of [17]. Although afterwards a specific section is dedicated to the comparison of the different approaches, for analyzing the results in terms of the expected clusters, each approach uses its specific *n*, the number of objects that are processed in the clusters construction step that, in the case of the SNNagg, excludes the number of repeated objects. Only when performance is in examination, both approaches are compared using the same input parameters.


a) Original SNN (*k=92, MinPts=88, Eps=17*)

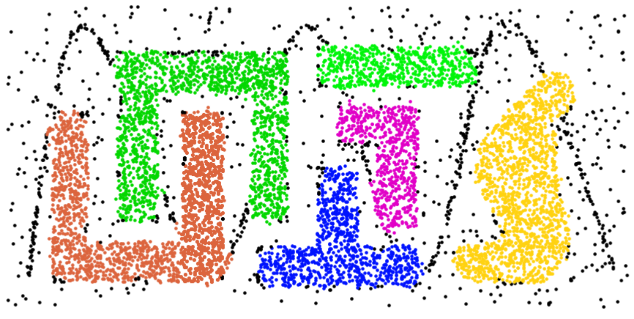

b) SNNr&r (*k=68, MinPts=64, Eps=12*)

Fig. 4. Spatial representation of the clustering results

To overcome the mentioned drawback, the SNNr&r approach advanced to the SNN_{agg} (SNN with Aggregates), in which a new task is taking care of incrementing the density of the representative points, considering this way more dense regions of objects and how this needs to affect the process of clusters construction (Fig. 5 - step 3.1). For representative objects, their density value corresponds to the number of repeated objects they represent. The main steps of the SNNagg approach can now be summarized as:

1) Read objects from the dataset; remove all the repeated ones and increment the representative objects;
2) Identify the nearest neighbors' lists without considering repeated objects. Each object in this list has the information of the number of repeated objects it represents;
3) Calculate the density of the points. It starts by being the same as the original SNN, being later updated for representative points. An increment of 1 is added for each repeated object;
4) Classify core objects attending to the density threshold;

5) Identify the clusters only considering non-repeated objects and the representative objects. Add to the obtained clusters the previously removed objects, considering the respective representative objects.

With the SNNagg, the clustering results, although not being the same as the original SNN, are the expected ones if we consider that, without the addition of the repeated points, 6 clusters are expected, one for each geometric figure. After the addition of the repeated points, and as this was randomly made in a specific area, it is expected that this area gives origin to a new cluster, which is properly identified by SNNagg as can be seen in Fig. 6, and was not identified by the Original SNN, as shown in (Fig. 4 a)).

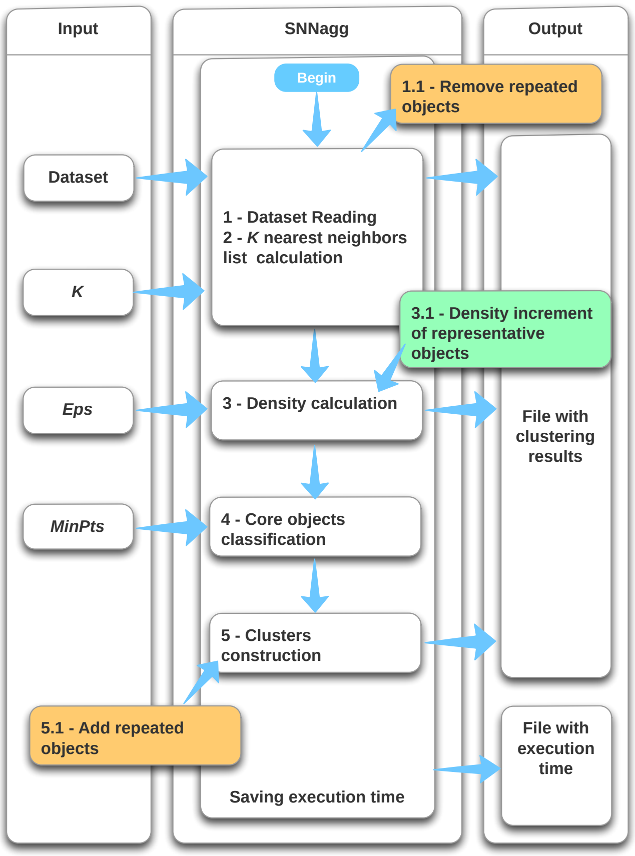

Input
SNNagg
Output
Begin
1.1 - Remove repeated objects
Dataset
1 - Dataset Reading
2 - K nearest neighbors list calculation
K
3.1 - Density increment of representative objects
Eps
3 - Density calculation
File with clustering results
MinPts
4 - Core objects classification
5 - Clusters construction
5.1 - Add repeated objects
File with execution time
Saving execution time

Fig. 5. Main steps for approach SNNagg

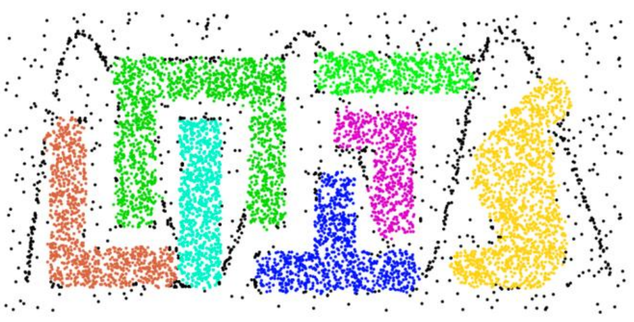


Fig. 6. Spatial representation of the clustering results with SNNagg (*k=68, MinPts=64, Eps=12*)

To evaluate in more detail the results that can be obtained by the SNNagg approach, next section presents the proposed quality indicators as well as the obtained values.

V. Evaluation and Results

As it was possible to see at the end of the previous section, the visual analysis of the results can provide useful hints about the quality of the clusters a specific approach is able to provide. However, this is a subjective evaluation. In order to make available objective measures about the quality of the results, the measures proposed in [24], and used in [25] are adopted and updated, including: (1) the *Intercluster* metric, (2) the *Intracluster* metric, and (3) the *ModelQuality* metric.

Next sections present how each one of these metrics are calculated as well as the datasets used to test the quality of the results. This section ends with the presentation and discussion of the obtained results, and the trend verified in terms of the number of repeated objects the approach is able to consider.

A. Model Quality

The *Intracluster* metric evaluates the similarity inside each cluster and its value is obtained by calculating the average distance of the objects to the cluster's average point. This means that for each cluster, the number of points of that cluster divides the sum of the distances to the average point inside the cluster. This process is repeated for each cluster, adding all values and dividing them by the number of clusters, allowing the calculation of the *Intracluster* of a model. In equation (3), t is the number of clusters, l is the number of objects inside cluster i, o stands for an object in cluster i and m is de average point of cluster i. The distance function (*Fdist*) must be the same as the used in the clustering process (in the case of this work, it is the Euclidean distance). The average point of each cluster is calculated as the arithmetic mean of all objects inside the cluster.

$$Intracluster = \frac{\sum_{i=1}^{t} \frac{\sum_{j=1}^{l} Fdist(o_j, m_i)}{l}}{t} \tag{3}$$

The *Intercluster* metric, shown in equation 4, evaluates the similarity between pairs of clusters and is calculated by the sum of the similarity between all objects from pairs of clusters (Fig. 7), values that then divided by the number of pairs of objects. This process is repeated for all pairs of clusters, being the sum of all these values divided by the number of clusters of a model. In equation 4, t represents the number of clusters, l_i is the number of objects inside clusters i, while l_j is the number of objects inside cluster j. *Fdist* stands for the distance function and, again, must be the same as the used in the clustering process. The objects o_y and o_z represent the objects inside clusters y and z, respectively.

$$Intercluster = \frac{\sum_{i=1}^{t-1} \sum_{j=i+1}^{t} \frac{\sum_{y=1}^{l_i} \sum_{z=1}^{l_j} Fdist(o_y, o_z)}{l_i * l_j}}{t} \tag{4}$$

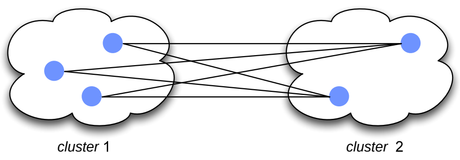


Fig. 7. Pairs of distances used for the calculation of the *Intercluster* metric

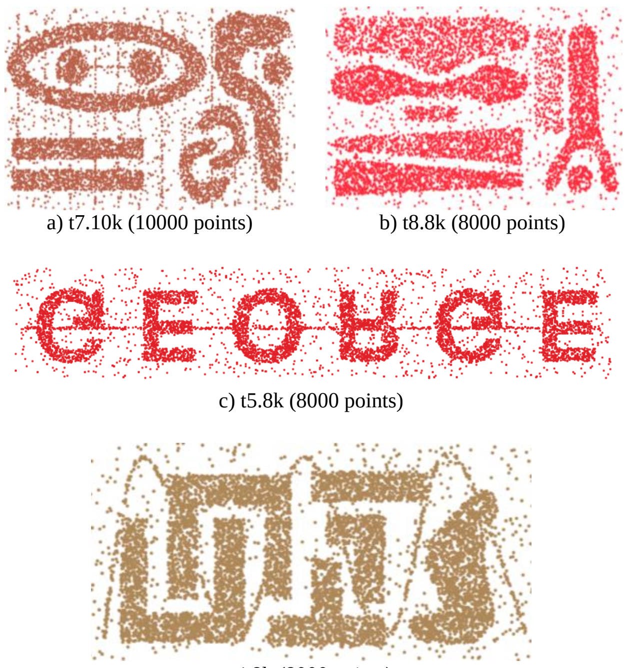
The *ModelQuality* metric is the absolute difference between the *Intracluster* and *Intercluster* values, as shown in equation 5. This metric gives an objective value of the quality of a clustering result, although in analytical contexts the best model may depend on the users' needs and may vary from user to user.

$$ModelQuality = |Intracluster - Intercluster| \quad (5)$$

The analysis of the *Intracluster* and *Intercluster* measures shows that the best clustering results usually emerge when there is a balance between these two metrics. A high *Intercluster* value means that clusters have a low similarity between pairs, while a low *Intracluster* value means that similarity is bigger inside each cluster. As the number of clusters increases, the *Intracluster* similarity decreases, as objects inside those clusters are more similar to each other, while the *Intercluster* similarity increases as more clusters allow more distinction between clusters. Although it cannot be defined as a rule, the best clustering results seem to emerge for lower *ModelQuality* values.

B. *Artificial Datasets*

In this paper, and for the evaluation of the proposed approach, several artificial datasets were used. The option for artificial datasets is justified by the need of knowing, beforehand, the expected results. The selected ones are the four artificial Chameleon datasets, presented in Fig. 8, integrating from 8000 to 10000 distinct objects. As these datasets do not include, in their original versions, any repeated objects, new datasets were created randomly introducing repeated objects.


a) t7.10k (10000 points)

b) t8.8k (8000 points)

c) t5.8k (8000 points)

t4.8k (8000 points)

Fig. 8. Spatial representation of datasets Chameleon

In a first stage, and to test how points are randomly generated and what the distribution of the repeated objects is, three versions of the t4.8k dataset were created adding 2000 repeated objects. As can be seen in Table 1, for the three tests, the distributions are equivalent so different random processes would probably lead to the same clustering results.

Having checked this, and for all the four artificial datasets, new datasets with repeated objects were created. In particular: i) for the t4.8k, three new versions, with 25%, 50% and 100% of repeated objects, now named as t4.8k+25%ro, t4.8k+50%ro and t4.8k+100%ro, respectively; ii) for the t5.8k, t7.10k and t8.8k, with 100% of repeated objects, now named as t5.8k+100%ro, t7.10k+100%ro, and t8.8k+100%ro, respectively. The results obtained with the clustering of these datasets are presented in the following subsection.

Table 1. Number of repeated objects by test

Type	Test 1	Test 2	Test 3
1 Object	6239	6234	6232
2 Repeated objects	1541	1553	1548
3 Repeated objects	201	193	209
4 Repeated objects	19	19	10
5 Repeated objects	0	1	1

C. *Obtained Results*

To evaluate the obtained results, the processing time of the Original SNN and the SNNagg approaches are compared, as well as the number of expected clusters. For comparing processing times, both algorithms use the same input parameters, as the *k* value highly influences their performance. When analyzing the expected clusters, the input parameters of the SNNagg approach must consider a lower *k*, as fewer objects are present in the processed dataset when looking for the *k*-nearest neighbors and, as consequence, the *k* value must be adapted to this scenario.

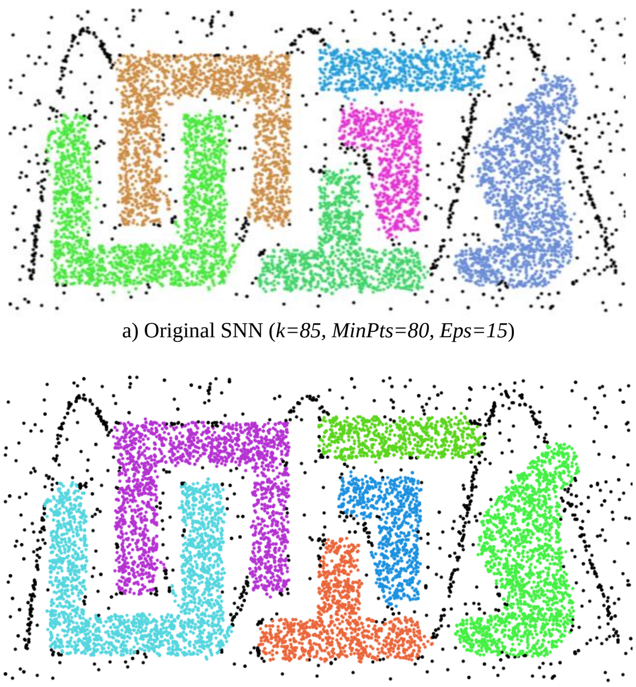
Starting by the t4.8k+25%ro dataset, Table 2 shows in its first line that the number of obtained clusters (NC) is the same, but that the *ModelQuality* (MQ) metric is lower in the SNNagg approach, meaning that probably the points are more adjusted in the obtained clusters. However, SNNagg needs more processing time (Time), as new steps were added to the algorithm. When the number of repeated objects starts to growth, the processing time of SNNagg starts to be lower than the Original SNN, decreasing substantially. For comparing the obtained clusters and their quality metrics, Table 3 presents the obtained results taking into consideration the number of objects effectively processed in the clusters construction step. Taking as an example the t4.8k+25%ro dataset, with the adjusted parameters, SNNagg continues to identity the 6 expected clusters, with a lower value for the *ModelQuality* (Fig. 9).

Table 2. Original SNN vs. SNNagg with the same input parameters

		Original SNN			SNNagg		
		Time	MQ	NC	Time	MQ	NC
t4	25%	18,45	539,880	6	18,87	468,323	6
	50%	38,05	668,966	8	28,27	448,940	6
	100%	99,21	501,179	6	59,52	513,284	6
t5	100%	90,87	946,665	7	55,05	756,090	6
t7	100%	167,31	1144,048	10	115,38	570,960	5
t8	100%	97,72	1197,971	9	57,46	588,144	6

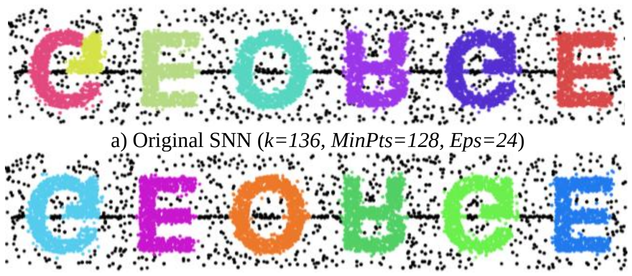
Table 3. SNNagg results with *k* calculated without repeated objects

		SNNagg		
		Time	MQ	NC
t4	25%	12,42	411,214	6
	50%	13,17	517,704	6
	100%	13,55	527,534	6
t5	100%	12,33	717,441	6
t7	100%	23,35	1217,795	10
t8	100%	13,55	1093,475	8


a) Original SNN (*k=85, MinPts=80, Eps=15*)

b) SNNagg (*k=68, MinPts=64, Eps=12*)

Fig. 9. Spatial representation of the clustering results for t4.8k+25%


a) Original SNN (*k=136, MinPts=128, Eps=24*)

b) SNNagg (*k=68, MinPts=64, Eps=12*)

Fig. 10. Spatial representation of the clustering results for t5.8k+100%

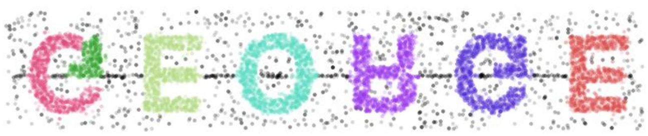
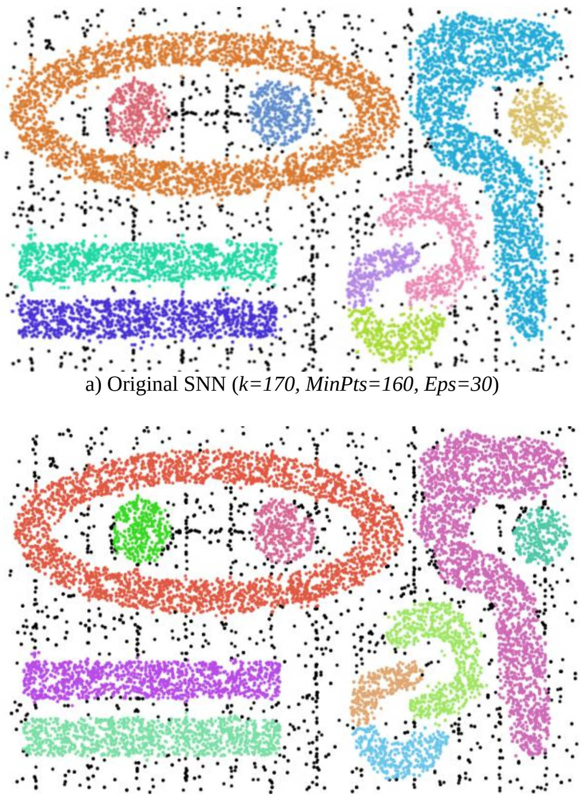


Fig. 11. Spatial representation of the clustering results for t5.8k+100%, with 90% of transparency, Original SNN (*k=136, MinPts=128, Eps =24*)


a) Original SNN (*k=170, MinPts=160, Eps=30*)

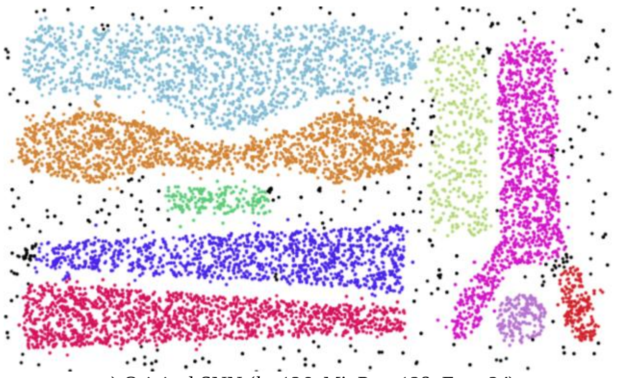
b) SNNagg (*k=85, MinPts=80, Eps=15*)

Fig. 12. Spatial representation of the clustering results for t7.10k+100%

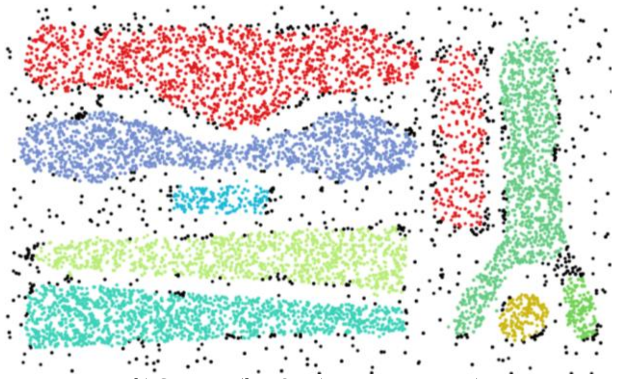
Looking now to the t5.8k+100%ro dataset, the Original SNN identified one more cluster, as previously seen in Table 2 and also depicted in Fig. 10. Visually, we are expecting the same 6 clusters unless the density of the points justifies the creation of another cluster. Although the two approaches may present different results, a visual analysis of the density was done using a new representation of these results with 90% of color transparency (Fig. 11). By the analysis of the figure it does not seem to exist a relevant change in the density of the two clusters, mainly in the transition area, but a detailed analysis of the density of the points would be needed before taking any conclusions.

For the dataset t7.10k+100%ro, the time results present in Table 2 are very relevant with about 1/3 of time saving. The quality comparison is made between the Original SNN approach in Table 2 and SNNagg in Table 3. Both approaches produced the same number of clusters, with SNNagg presenting a slightly higher measure for the *ModelQuality*. Although this difference, both results in terms of visual analysis are equivalent, as Fig. 12 shows.

For the t8.8k+100%ro dataset, and looking to Table 2, the gain in terms of processing time is about 40% using the SNNagg approach. While the Original SNN identifies 9 clusters (Fig. 13 a)), SNNagg identifies 8 using the input parameters adapted to the number of different objects (Table 3). In both cases, the algorithms failed to correctly identify one of the geometric figures (the inverted Y). Besides this, the SNNagg joins two different clusters, plotted in red in Fig. 13 b).



a) Original SNN (*k=136, MinPts=128, Eps=24*)



b) SNNagg (*k=68, MinPts=64, Eps=12*)

Fig. 13. Spatial representation of the clustering results for t8.8k+100%

To understand the joining of these two clusters by the SNNagg approach, a representation of the points with 90% of transparency was done (Fig. 14), allowing the visual verification of a high number of repeated objects in the boundaries of the two merged clusters, being this probably the reason for the merging of the clusters. In the opposite way, the inverted Y cluster, a variation on the density of the repeated objects may explain the identification of two different clusters.

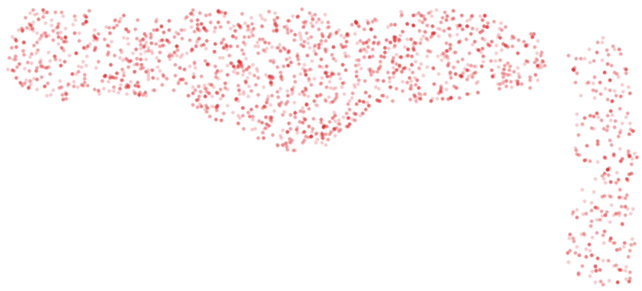
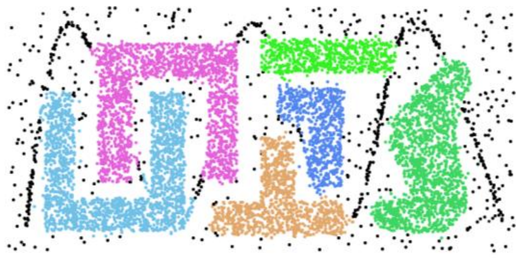


Fig. 14. T8.8k+100% spatial representation of clustering process with 90% of transparency, SNNagg (*k=136, MinPts=128, Eps =24*)

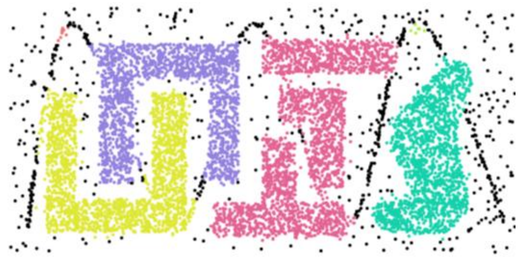
To conclude this subsection, it is important to mention that SNNagg can reduce the processing time by the elimination of the repeated objects during the clustering process, being able to add them to the identified clusters without compromising the clustering results. This can be achieved considering the repeated objects in step 3 (Density calculation). However, this reduction in terms of processing time is only verified when the number of repeated objects starts to grow. To evaluate when the clustering results, in terms of quality, start to be affected by the increasing number of repeated objects, next section tries to answer the question: how many repeated objects the SNNagg approach is able to support without compromising the results?

D. Impact of the number of repeated objects

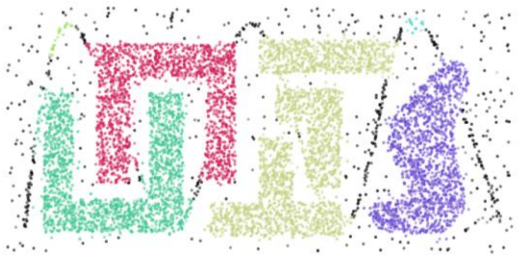
To evaluate the impact of the number of repeated objects in the quality of the clustering results, an increasing number of repeated objects were added to the t4.8k and to the t7.10k datasets. Starting by the t4.8k dataset, four new datasets were created with more 500%, 750%, 1000% and 2000% of repeated objects. As all the datasets were created with random distributions, it is expected the same number of clusters, in this case 6. As can be seen in Fig. 15 a), for 500% of repeated objects, SNNagg still identifies the 6 expected clusters. For 750% (Fig. 15 b)), the approach starts to aggregate clusters, downgrading the quality of results. This can also be seen for 1000% (Fig. 15 c)), where previously noise points start to be integrated in existing clusters or give origin to new, very small, clusters. When we reach the 2000% dataset (Fig. 15 d)), the tendency of aggregation of clusters is intensified, as the number of repetitions is so high that the algorithm is unable to detect transitions.



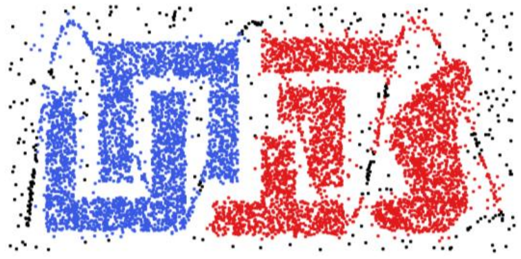
a) More 500% of repeated objects



b) More 750% of repeated objects



c) More1000% of repeated objects



d) More 2000% of repeated objects

Fig. 15. Evolution of clustering results for t4.8k with different amounts of repeated objects, SNNagg (*k*=68, *MinPts* =64, *Eps*=12)

To have a clearer overview of the number of repeated objects, Fig. 16 presents a histogram with the distribution of the number of repeated objects considering the several created datasets. As can be seen in this figure, as we increase the number of repeated objects, this number starts to be superior to *Eps*, the input parameter used to separate zones of high and low density of points.

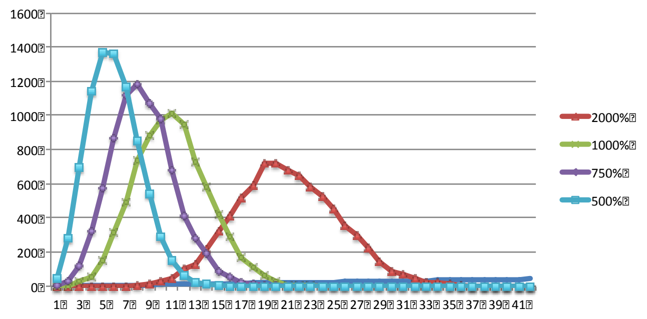


Fig. 16. Distribution of the number of repeated objects for t4.8k

For *Eps*, and considering the extensive work done in [18] for understanding the SNN input parameters, the authors show that although *Eps* is a less sensitive input parameter, different values can provide different valid results, with several degrees of quality. In the case of the t4.8k dataset, and due to the increasing number of repeated objects, *Eps* can be increased in order to allow the algorithm the separation of the zones of high and low densities. Looking to the histogram in Fig. 16 and having into consideration the normal distribution of the 2000% dataset, and *Eps* of 32 was tested providing the results shown in Fig. 17. Though the results are not excellent, the algorithm is able to separate again some of the clusters, although some probable noise points are not detected as such.

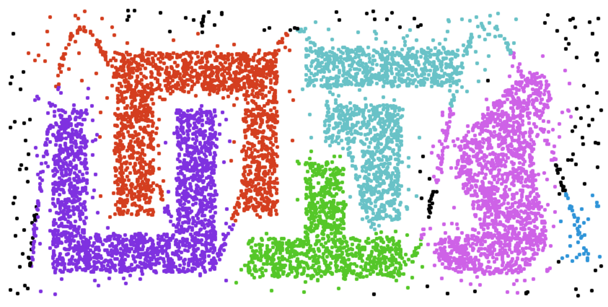
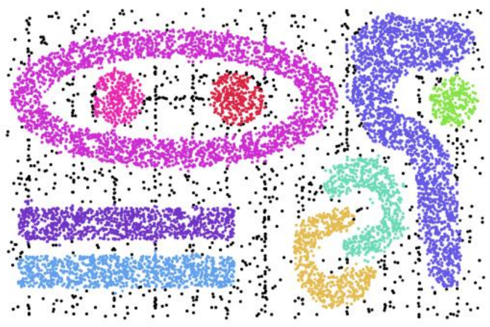
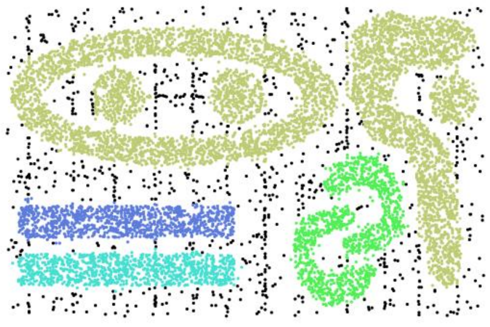


Fig. 17. Clustering results for t4.8k for 2000% of repeated objects, SNNagg (*k*=68, *MinPts* =64, *Eps*=32)

The other dataset tested was the t7.10k, mainly because of the difference in the number of objects (10000 original points). Once again we have created four new datasets with more 250%, 500% and 2000% of repeated objects. In this case different repetitions were made as the density of the repeated points started to affect sooner the clustering results. As can be seen in Fig. 18 a), 250% of repeated objects still allow the identification of the 9 expected clusters, well formed with high separation from noise. As the number of repeated objects starts to increase, the clusters start to be merged, finishing in Fig. 18 c) with only 3 clusters and with an increasing number of previously noise points as non-noise points.


a) More 250% of repeated objects


b) More 500% of repeated objects

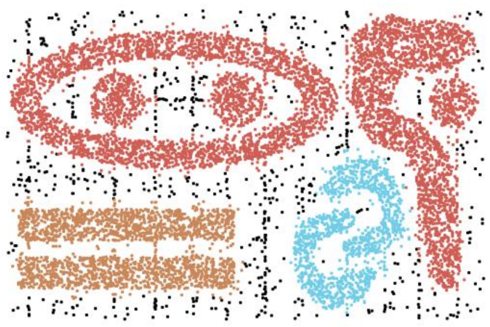

c) More 2000% of repeated objects

Fig. 18. Evolution of clustering results for t7.10k with different amounts of repeated objects, SNNagg (*k*=85, *MinPts* =80, *Eps*=15)

Once again (Fig. 19), it is possible to see the curves with the normal distributions of the number of repeated objects for the several considered datasets.

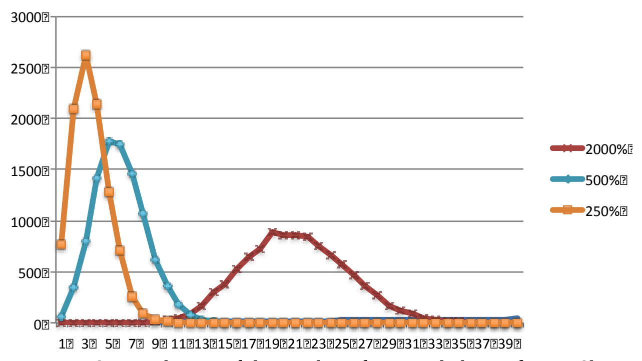


Fig. 19. Distribution of the number of repeated objects for t7.10k

In this case it is also necessary to change the *Eps* parameter, considering the number of repeated objects that is verified. Once again, looking at the distribution to see that a value higher than 32 is necessary, the one that provided better results was 40, as can be seen in Fig. 20, with the 9 expected clusters.

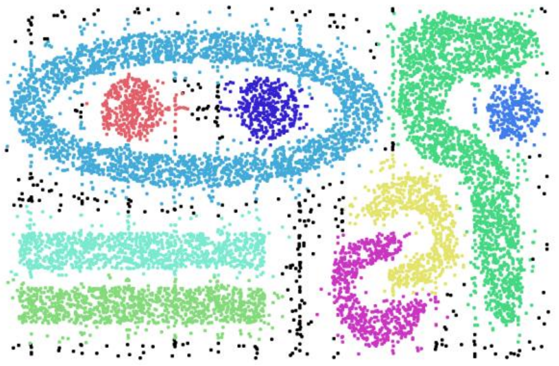


Fig. 20. Clustering results for t7.10k for 2000% of repeated objects, SNNagg (*k*=85, *MinPts* =80, *Eps*=40)

The presented results show that SNNagg is able to remove repeated objects, adding them latter to the obtained clusters. However when the number of repeated objects starts to increase, the *Eps* input parameter also needs to be increased. As in any clustering process, the tuning of the input parameters may help to improve the clustering results.

VI. Conclusion

Clustering with SNN is a very demanding task in terms of processing time, mainly when searching for the *k*-nearest neighbors of the objects. With the increasing size of the datasets, the time needed to obtain the results is even more critical.

This paper presented an approach for dealing with repeated objects, SNNagg, which is able to remove the repeated objects in the beginning of the clustering process, being able to add those repeated objects to the identified clusters. SNNagg showed important gains in the reduction of the processing time, mainly with increasing numbers of repeated objects. In terms of quality of the results, and for the used artificial datasets used, depending on the number of repeated objects present in the dataset, the results can range from the expected ones to approximate results that can be acceptable if the user is able to have those approximations as long as they are obtained in a short period of time.

As future work, the notion of repeated object can be extended to the notion of similar object, where a function to measure the similarity between objects is needed. Also important is to verify the impact of the repeated objects in the input parameters, mainly in the *Eps* value.

References

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI Mag.*, vol. 17, no. 3, p. 37, 1996.

[2] S. Kotsiantis and P. Pintelas, “Recent advances in clustering: A brief survey,” *WSEAS Trans. Inf. Sci. Appl.*, vol. 1, no. 1, pp. 73–81, 2004.

[3] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[4] M. Y. Santos, J. P. Silva, J. Moura-Pires, and M. Wachowicz, “Automated traffic route identification through the shared nearest neighbour algorithm,” in *Bridging the Geographic Information Sciences*, Springer, 2012, pp. 231–248.

[5] A. Antunes, M. Y. Santos, and A. Moreira, “Fast SNN-Based Clustering Approach for Large Geospatial Data Sets,” in *Connecting a Digital Europe Through Location and Place*, Springer, 2014, pp. 179–195.

[6] B. F. Faustino, J. Moura-Pires, M. Y. Santos, and G. Moreira, “kd-SNN: A Metric Data Structure Seconding the Clustering of Spatial Data,” in *Computational Science and Its Applications–ICCSA 2014*, Springer, 2014, pp. 312–327.

[7] M. Y. Santos, J. M. Pires, G. Moreira, R. Oliveira, F. Mendes, and C. Costa, “Geo-spatial analytics using the dynamic ST-SNN Approach,” in *Proceeding of the World Congress on Enginnering 2015, WCE 2015*, London, 2015, vol. I, pp. 285–290.

[8] L. Ertoz, M. Steinbach, and V. Kumar, “Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data,” in *Proc. Third SIAM Intl Conf. Data Min*, 2003, pp. 285–290.

[9] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, “On Clustering Validation Techniques,” *J. Intell. Inf. Syst.*, vol. 17, no. 2–3, pp. 107–145, Dec. 2001.

[10] J. A. Hartigan and M. A. Wong, “Algorithm AS 136: A K-Means Clustering Algorithm,” *J. R. Stat. Soc. Ser. C Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.

[11] D. Birant and A. Kut, “ST-DBSCAN: An algorithm for clustering spatial–temporal data,” *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, Jan. 2007.

[12] J. A. M. R. Rocha, G. Oliveira, L. O. Alvares, V. Bogorny, and V. C. Times, “DB-SMoT: A direction-based spatio-temporal clustering method,” in *Intelligent Systems (IS), 2010 5th IEEE International Conference*, 2010, pp. 114–119.

[13] M. Nanni and D. Pedreschi, “Time-focused clustering of trajectories of moving objects,” *J. Intell. Inf. Syst.*, vol. 27, no. 3, pp. 267–289, 2006.

[14] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko, “Visually Driven Analysis of Movement Data by Progressive Clustering,” *Inf. Vis.*, vol. 7, no. 3–4, pp. 225–239, Sep. 2008.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.,” in *Kdd*, 1996, vol. 96, pp. 226–231.

[16] R. A. Jarvis and E. A. Patrick, “Clustering using a similarity measure based on shared near neighbors,” *Comput. IEEE Trans. On*, vol. 100, no. 11, pp. 1025–1034, 1973.

[17] G. Moreira, M. Y. Santos, and J. Moura-Pires, “SNN Input Parameters: how are they related?,” in *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, 2013, pp. 492–497.

[18] G. Moreira, M. Y. Santos, J. M. Pires, and J. Galvão, “Understanding the SNN Input Parameters and How They Affect the Clustering Results,” *Int. J. Data Warehous. Min. IJDWM*, vol. 11, no. 3, pp. 26–48, 2015.

[19] H. B. Bhavsar and A. G. Jivani, “The Shared Nearest Neighbor Algorithm with Enclosures (SNNAE),” in *2009 WRI World Congress on Computer Science and Information Engineering*, 2009, vol. 4, pp. 436–442.

[20] F. Mendes, M. Y. Santos, and J. Moura-Pires, “Dynamic Analytics for Spatial Data with an Incremental Clustering Approach,” in *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, 2013, pp. 552–559.

[21] R. Oliveira, M. Y. Santos, and J. M. Pires, “4D+ SNN: A Spatio-Temporal Density-Based Clustering Approach with 4D Similarity,” in *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, 2013, pp. 1045–1052.

[22] N. Tomasev, M. Radovanovic, D. Mladenic, and M. Ivanovic, “The Role of Hubness in Clustering High-Dimensional Data,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 739–751, Mar. 2014.

[23] E. Camossi, M. Bertolotto, and T. Kechadi, “Mining spatio-temporal data at different levels of detail,” in *The European Information Society*, Springer, 2008, pp. 225–240.

[24] L. Rokach and O. Maimon, “Clustering Methods,” in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer US, 2005, pp. 321–352.

[25] A. Moreira, M. Y. Santos, and S. Carneiro, “Density-based clustering algorithms–DBSCAN and SNN, 2005. Technical Report, available at: *http://ubicomp.algoritmi.uminho.pt/local/download/SNN&DBSCAN.pdf*, accessed in 2015.