Universidade do Minho
Escola de Engenharia

José Guilherme da Cruz Moreira

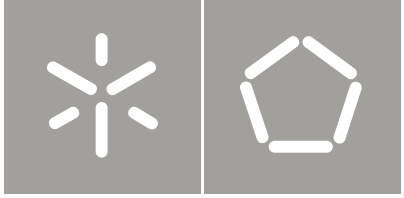
Input Parameters Self-tuning on the SNN Algorithm (Shared Nearest Neighbour)

outubro de 2013

Input Parameters Self-tuning on the
SNN Algorithm (Shared Nearest Neighbour)

José Guilherme da Cruz Moreira

UMinho | 2013



Universidade do Minho
Escola de Engenharia

José Guilherme da Cruz Moreira

Input Parameters Self-tuning on the SNN Algorithm (Shared Nearest Neighbour)

Tese de Mestrado
Mestrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação da
Professora Doutora Maribel Yasmina Santos

outubro de 2013

Anexo 3

DECLARAÇÃO

Nome: José Guilherme da Cruz Moreira

Endereço electrónico: jgcmoreira@gmail.com

Telefone: 964965783

Número do Bilhete de Identidade: 7665033

Título dissertação □/tese □ Input Parameters Self-tuning on the SNN Algorithm (Shared Nearest Neighbour)

Orientador(es): Professora Doutora Maribel Yasmina Santos

Ano de conclusão: 2013

Designação do Mestrado ou do Ramo de Conhecimento do Doutoramento:

Mestrado em Engenharia e Gestão de Sistemas de Informação

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

Universidade do Minho, 31/12/2013

Assinatura: _____

To my parents, José and Olinda and to my brother Mário

Acknowledgments

First, I would like to express my deep gratitude to my advisor, Professora Maribel Yasmina Santos, for her supervision, great engagement, guidance and motivation. I also would like to thank Professor João Moura-Pires for sharing his opinions and insights. A general appreciation goes also to all my teachers and masters that have accompanied me in my academic and personal growth so far. I need to include here also the institutions that provide the needed learning environment, including of course, the University of Minho and the Information Systems Department.

This acknowledgement wouldn't be completed if I didn't include all my colleagues, friends and family that were with me all these years sharing good and less good moments and unforgettable experiences. A special thank you to Ricardo, *companion de route* during the graduation, and to Fernando and Ricardo, my fellow colleagues, during the master period.

This thesis was done with the support of the research project "GIAP - GeoInsight Analytics Platform (LISBOA-01-0202-FEDER-024822)", funded by Comissão de Coordenação e Desenvolvimento Regional de Lisboa e Vale do Tejo (PORLisboa), included in Sistema de Incentivos à Investigação e Desenvolvimento Tecnológico (SI I&DT), through the research fellowship UMINHO/BI/305/2012.

Abstract

Recent technological developments have lead to a ever increasing rate in data collection. Organisations are facing several challenges when they try to analyse this vast amount of data with the aim of extracting useful information. This analytical capacity needs to be enhanced with tools capable of dealing with big data sets without making the analytical process a difficult task. Clustering is usually used, as this technique does not require any *a priori* knowledge about the data. However, clustering algorithms usually require one or more input parameters that influence the clustering process and the results that can be obtained.

This work analyses the relation between the three input parameters of the SNN (Shared Nearest Neighbour) algorithm through extensive brute-force executions and finds some strong relations between them. These findings help to propose an heuristic suitable for the identification and suggestion of the SNN input parameters. The proposed heuristic is validated using different data sets that the ones used for the heuristic development.

The solution is very useful because it allows the user to avoid a considerable time spent on trial and error executions. It suggests the user an initial quality clustering result, that while not definitive, it is a good starting point for the clustering analysis.

Keywords: density-based clustering, SNN, shared nearest neighbour, input parameters tuning

Resumo

Os recentes avanços tecnológicos têm levado a um ritmo cada vez maior na recolha de dados. As organizações enfrentam diversos desafios quando tentam analisar essa imensa quantidade de dados, com o objetivo de extrair informação útil. Esta capacidade analítica precisa de ser melhorada com ferramentas capazes de lidar com grandes conjuntos de dados, sem que isto transforme o processo de análise, numa tarefa difícil. O agrupamento (*clustering*), é normalmente utilizado, tratando-se de uma técnica que não requer conhecimento, *a priori*, sobre os dados. No entanto, os algoritmos de agrupamento, normalmente requerem um ou mais parâmetros de entrada que influenciam o processo de agrupamento e os resultados que podem ser obtidos.

Este trabalho, analisa a relação entre os três parâmetros de entrada do algoritmo SNN (*Shared Nearest Neighbour*) através de execuções de força-bruta e encontra algumas relações fortes entre eles. Estes resultados ajudam a propor uma heurística adequada para a identificação e sugestão dos parâmetros de entrada do algoritmo SNN. A heurística proposta é então validada utilizado conjuntos de dados diferentes daqueles que foram utilizados para o desenvolvimento da heurística.

A solução encontrada é de grande utilidade, pois permite ao utilizador evitar consumir uma quantidade considerável de tempo em execuções recorrendo à tentativa e erro. A heurística sugere ao utilizador um resultado de agrupamento inicial com qualidade, que embora não definitivo, é um bom ponto de partida para a análise do agrupamento.

Palavras-chave: agrupamento baseado em densidade, SNN, *shared nearest neighbour*, ajuste dos parâmetros de entrada.

Contents

List of abbreviations and acronyms

BIRCH - Balanced Iterative Reducing and Clustering using Hierarchies

CURE - Clustering Using REpresentatives

DBSCAN - Density-Based Spatial Clustering of Applications with Noise

OPTICS - Ordering Points To Identify the Clustering Structure

SNN - Shared Nearest Neighbour

SSDBSCAN - Semi-Supervised Density Based Clustering

STSNN - Spatial Temporal Shared Nearest Neighbour

VDBSCAN - Varied Density Based Spatial Clustering Applications with Noise

List of Figures

List of Tables

1 Introduction

In this introductory chapter it's described the context and motivation, the problem that this dissertation will try to solve and the document structure.

1.1 Context and Motivation

"The purpose of computing is insight, not numbers" [Hamming, 1987]

Current technological development allows the collection of huge amounts of data that are usually stored and analysed to support the decision-making process. In every step of our life, data are collected: when we ride our car; we buy our groceries; we watch a sports event; etc. But storing these data isn't enough. We need tools to get information and knowledge out of these stored data.

Analytical tools, like data mining algorithms, support the analysis of such vast amount of data. Data mining is one of the steps of the knowledge discovery process [Fayyad et al., 1996], in which clustering algorithms are common techniques used to analyse data, not requiring any prior knowledge about the data set [Jain et al., 1999]. Being unsupervised data mining techniques, clustering has the advantage of identifying clusters that emerge naturally from data.

Increasing use of spatial databases, which have localisation information of the data, allows to cluster behaviours of subjects, animals, etc. There are several spatial algorithms to fulfil this job. Among them the most well known are DBSCAN [Ester et al., 1996], OPTICS [Ankerst et al., 1999], SNN [Ertoz et al., 2002, Ertoz et al., 2003]. SNN (Shared Nearest Neighbour) has proved to be one of the appropriate spatial clustering algorithms, because it can perform well with various clusters shapes [Ertoz et al., 2003] and with variable densities [Moreira et al., 2005]. For the purpose of this dissertation it's going to be used the SNN algorithm. First because it's one of the appropriate solutions, as stated before, and secondly because there are freely available two recent software artefacts, that implement the SNN algorithm: F-SNN [Antunes, 2013] and Kd-SNN [Faustino, 2012].

1.2 Aim and objectives

Most of the clustering algorithms require input parameters that influence the results that can be obtained. The process of tuning the input parameters is usually a trial and error process in which the user changes the input parameters until the results satisfy the analysis requirements [Bouguessa, 2011]. This process can be difficult and time consuming as no strict rules exist about the definition of these parameters. Moreover, any new trial requires a new run of the algorithm so more processing

time is needed. To overcome this trial and error process, this work analyses in detail the three input parameters of the SNN (Shared Nearest Neighbour) algorithm and proposes specific guidelines to identify these parameters attending to the data available for analysis. SNN has three input parameters, *k*, *MinPts* and *Eps*, which are used to calibrate the clustering's results. Being a density-based algorithm, *k* represents the size of nearest neighbours list; *Eps* is the density threshold; and, *MinPts* is the minimum density that a point needs to satisfy to become a core point of a cluster [Ertoz et al., 2003].

The aim of this dissertation is try to answer the following research question: ***"How to establish mechanisms to self-tune the SNN (Shared Nearest Neighbour) algorithm input parameters?"***.

In order to answer that question, there are some objectives we need to reach, that will be important to the elaborate the solution. The objectives are:

- Literature review:
  - Do a complete study on clustering and density-based clustering. It's the general domain of the problem. Before go to the specifics of the SNN algorithm, it is necessary to understand the general concepts of the area;
  - Completely understand the SNN algorithm. It is necessary to understand the most of the algorithm, hence is the technique that will be used and improved by this dissertation. Analise problems that other researchers are facing to solve similar problems;
- Analyse the SNN input parameters and identify possible relations between them:
  - This is the specific object of the proposed work. To find mechanisms, inferences or heuristics, that help to solve the problem of the research question;
- Propose an heuristic that helps to minimise the work needed to find the proper SNN input parameters and save the user of the long time of trial and error process, that is the current practice.
- Validate the proposed solution working with different data sets than the ones used to develop it.

1.3 Document structure

This document is organised in five chapters. The first one, an introductory chapter, contains the context and motivation, the general problem, the research question and the organisation of this

document. On the second chapter is going to be presented the literature review, explaining the concepts and techniques that are the state-of-the-art of clustering, spatial clustering, types of clustering algorithms, focusing on density-based clustering algorithms. The chapter finishes with the research of what other authors are doing about the input parameters tuning. The third chapter is dedicated to the objectives of the work showing the approach, tests results and the proposed heuristics. On the fourth chapter is made a validation of the proposed heuristics. The fifth chapter contains the conclusions, including the main achievements of this thesis and the proposals of future work.

2 Clustering Algorithms and Input Parameters Tuning

To understand the principles of clustering as a data mining technique, we need to go for some ideas and concepts beforehand. Three fundamental definitions are now given, due to their relevance to this work.

- Data - We live in the information era. Data are collected in every step of our lives. When we buy our groceries, when we take a picture, when we use our car in a motorway, when we make a phone call or when we update our status on Facebook. The collection of data increases everyday and produces huge datasets. The Cambridge online dictionary defines data as *"information, especially facts or numbers, collected to be examined and considered and used to help decision-making, or information in an electronic form that can be stored and processed by a computer"*. [Cios et al., 2000] highlight the fact that we live in a information-rich but at the same time in a knowledge-poor environment. They think that there is a wide gap between data generation and data comprehension. Thats when comes the need of processes of knowledge discovery and data mining;

- Data Mining - *"is the analysis of (often large) observational data sets to find unsuspected relationships and to summarise the data in novel ways that are both understandable and useful to the data owner"* [Hand et al., 2001]. [Cios et al., 2000] remembers the statement of [Hamming, 1987] that said: *"the purpose of computing is insight, not numbers"* to remind that knowledge discovery and data mining is fundamental to make sense of data. Knowledge discovery was first used by [Piateski and Frawley, 1991] when they defined it as *"Knowledge discovery in databases is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data"*.

- Spatial Data - To understand spatial data mining we need to understand what type of data is stored in a spatial database. A spatial database stores large amount of space-related data, such as maps, medical imaging data, etc. Spatial databases differs from relational databases because they carry topological and/or distance information, usually organised by sophisticated, multidimensional spatial indexing structures that are accessed by spatial data access methods and often require spatial reasoning, geometric computation, and spatial knowledge representation techniques [Han and Kamber, 2001].

Next sections presents the understanding of clustering as a data mining technique, the different types of clustering algorithms, the SNN algorithm, and what researchers are doing to minimise the hassle of input parameters tuning.

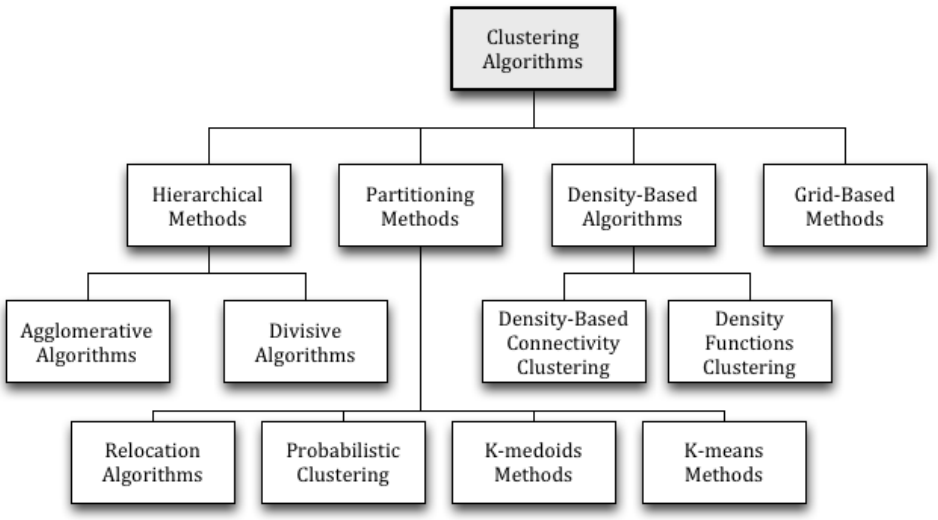


Figure 2.1: Clustering algorithms. Adapted from [Berkhin, 2004]

2.1 Clustering as a Data Mining Technique

Clustering is a popular data-mining technique due to its ability to extract synthetic information from complex and multi-dimensional data sets without a prior classification. Clustering is the task of identifying sets of clusters that group similar objects. A cluster is a collection of data objects that have more similarities between them and are dissimilar to objects that belong to other clusters [Han and Kamber, 2001]. Dissimilarities are often assessed by distance measures. In this data mining process, clustering is an unsupervised learning technique, because it can treat large amount of raw data, without being previously classified. On the spatial clustering field, the processes are very similar adding also the position that the objects occupy in the space. So, in this case the distance measures are even more important because if we have similar objects but very distant from each other they will be in different clusters. The importance of spatial clustering is very high in several areas of our life, like in medicine, territorial administration, telecommunications, biology and other nature sciences, etc.

2.2 Approaches and Types of Clustering Algorithms

[Berkhin, 2004] classifies clustering algorithms in four main classes: hierarchical, partitioning, density-based, grid-based, and some other evolutions that combine one or more of the previous techniques. The different types of clustering algorithms can be organised like it is shown on fig. 2.1.

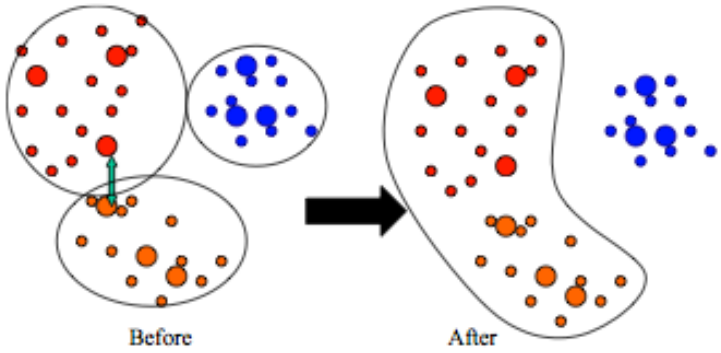


Figure 2.2: Agglomeration in CURE. Adapted from [Berkhin, 2004]

Next sections presents the different types of clustering algorithms: hierarchical clustering, partition clustering , grid-based clustering, and density-based clustering algorithms.

2.2.1 Hierarchical clustering algorithms

Also known as connectivity based clustering, hierarchical clustering algorithms aim to establish a hierarchical relation between clusters.

The graphic representation of these types of clusters is made by a dendrogram. The dendrogram is a representation tree-like that shows the hierarchy of the clusters. There are two types of hierarchical clustering, agglomerative and divisive:

- agglomerative algorithms have a bottom-up approach. The observation starts on a single cluster and as clusters are being merged they move up on the hierarchy until there is only one big cluster. Examples of this kind of technique are the Chameleon [Karypis et al., 1999] and CURE [Guha et al., 1998] algorithms, the last one showed on fig. 2.2.

- divisive algorithm (figure 2.3) use a top-down approach and start with one cluster and as the clusters are being divided they go down on the hierarchy until they reach the intended number of clusters.

2.2.2 Partition clustering algorithms

The most used partitioning methods are the k-means and the k-medoids and their own variations. These methods work well for finding spherical shaped clusters [Han and Kamber, 2001], but don't perform well with variable sized clusters or non-spherical shaped clusters [Ertoz et al., 2003].

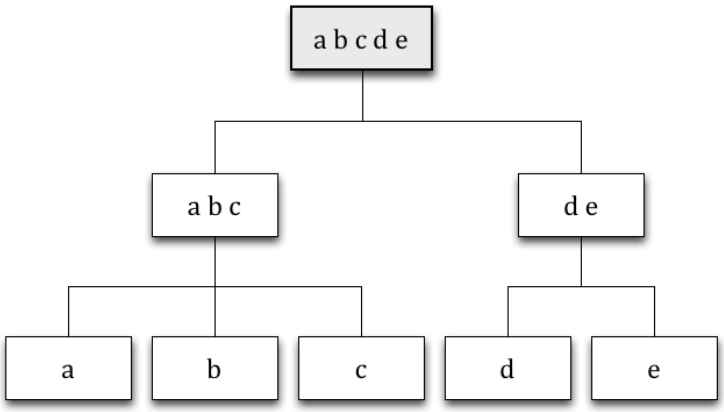


Figure 2.3: Divisive clustering

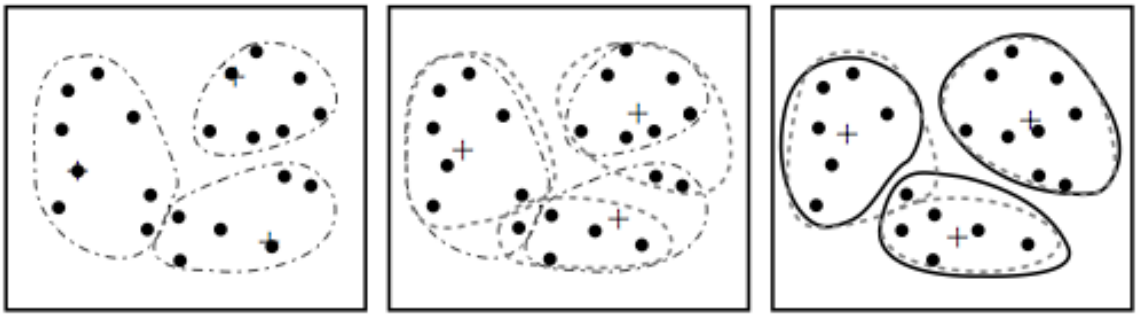


Figure 2.4: *k*-means algorithm. Adapted from [Han and Kamber, 2001]

- on the *k*-means algorithm, each cluster is represented by the mean value of the objects included in the cluster [Han and Kamber, 2001]. The process is shown on fig. 2.4. First the algorithm choose *k* arbitrary objects as the initial clusters centers, then iteratively it reassigns each object to the cluster that have the mean value more similar to the object value, updates the new means of each cluster and repeats until there is no more possible changes. This algorithm is not outlier-friendly, because an object with a large value easily distorts the distribution of data [Han and Kamber, 2001].

- the *k*-medoids algorithm, represents each cluster by one object located near the center of the cluster [Han and Kamber, 2001]. This algorithm tries to solve the outlier problem that the *k*-means suffers. Instead of using the mean value as a reference point, the algorithm performs a technique to choose a representative object for each cluster. This way it is less sensitive to noise and outliers.

One of the limitations of this kind of algorithms is that they use k as an input parameter, so the user indicates the total number of clusters. They don't "naturally" result from the data.

2.2.3 Grid-based clustering

Grid-based methods divide the space occupied by the objects of the dataset in a finite number of cells that form the grid structure. Then the clustering operations are made inside the grid allowing a fast processing time. This way we limit the search combinations inside a segment (cell) [Han and Kamber, 2001]. This method transforms the data partitioning to space partitioning. Cells that contain more than a pre-determined number of objects are treated as dense and then clusters are formed connecting dense cells [Kotsiantis and Pintelas, 2004].. Instead of numerical types, this method works best with attributes of different types [Berkhin, 2004].

2.2.4 Density-based clustering algorithms

These methods were developed based on the notion of density [Han and Kamber, 2001]. These algorithms perceive clusters as dense regions of objects in a space separated by regions of relatively low density. This kind of algorithms is useful to filter out noise and for discovering clusters of arbitrary shapes [Ye et al., 2003]. DBSCAN [Ester et al., 1996] and OPTICS [Ankerst et al., 1999] are major representatives of this class of clustering algorithms.

Once DBSCAN is the most representative density-based clustering algorithm and many of the available algorithms in this area were derived or are evolution versions of it, it's important to analyse DBSCAN with more detail.

DBSCAN, which stands for Density Based Spatial Clustering of Applications with Noise, introduced by [Ester et al., 1996] was specially designed to treat spatial databases. The authors stated three primary requirements for clustering algorithms: 1- Minimal requirements of the domain knowledge to determine the input parameters, as these algorithms are non-supervised and they don't require knowledge of the dataset *a priori*; 2- The algorithm must be able to discover clusters of arbitrary shape, because shapes of clusters in spatial databases can be spherical, linear, elongated etc; 3- Clustering algorithms must have good efficiency, because spatial databases can have hundreds of thousand objects easily.

DBSCAN needs two input parameters: *Eps* and *MinPts*. *Eps* is the radius distance of a point, in other words the neighbourhood of a point. The *MinPts* parameter is the minimum number of points that the neighbourhood must have to be considered a cluster. Points within the radius are considered core points and points on the border of the cluster are border points. Points that don't belong to any cluster are noise points [Ester et al., 1996]. Examples of DBSCAN types of points are shown on figure 2.5.

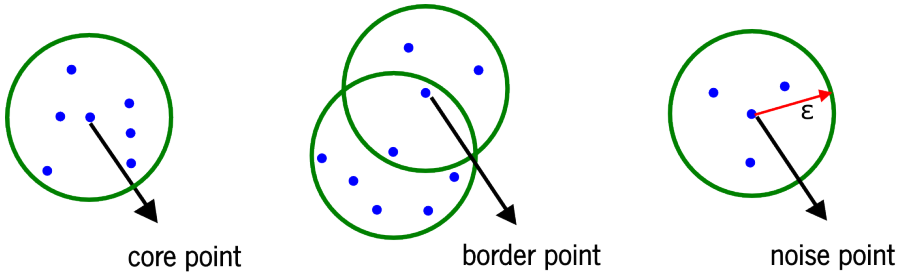


Figure 2.5: Types of points on DBSCAN using *Eps* = ε and *MinPts* = 6

To a better understanding of the DBSCAN algorithm we go through the formal explanation that includes six definitions.

Let D be a database of points of some euclidean 2D space:

Definition 1 (*Eps* - neighbourhood of a point) - The *EPS*-neighbourhood of a point *p*, denoted $N_{Eps}(p)$, is defined by:

- $N_{Eps}(p) = \{ q \in D \mid dist(p,q) \leq Eps\}$

Definition 2 (directly density-reachable) - A point *p* is directly density-reachable from a point *q* if:

- $p \in N_{Eps}(q)$, and
- $\mid N_{Eps}(q) \mid \geq MinPts$, *q* is a core point

Meaning that for each point *p* in a cluster C , there is a point *q* in C so that *p* is inside of the *Eps*-neighbourhood of *q* and that $N_{Eps}(p)$ at least contains *MinPts* points. This relation is not necessarily symmetric. Figure 2.6 shows definition 2 in practice in an asymmetric case.

Definition 3 (density-reachable)

- A point *p* is density-reachable from a point *q* if there is a chain of points $p_1, ..., p_n$, p_1=*q*, p_n= *p* such that p_{i+1} is directly density-reachable from p_i.

Density-reachability is a canonical extension of direct density-reachability. This relation is transitive but not symmetric. Figure 2.7 in (a) shows an example of density-reachability.

Definition 4 (density-connected)

- A point *p* is density connected to a point *q* if there is a point o such that both, *p* and *q* are density-reachable from *o*.

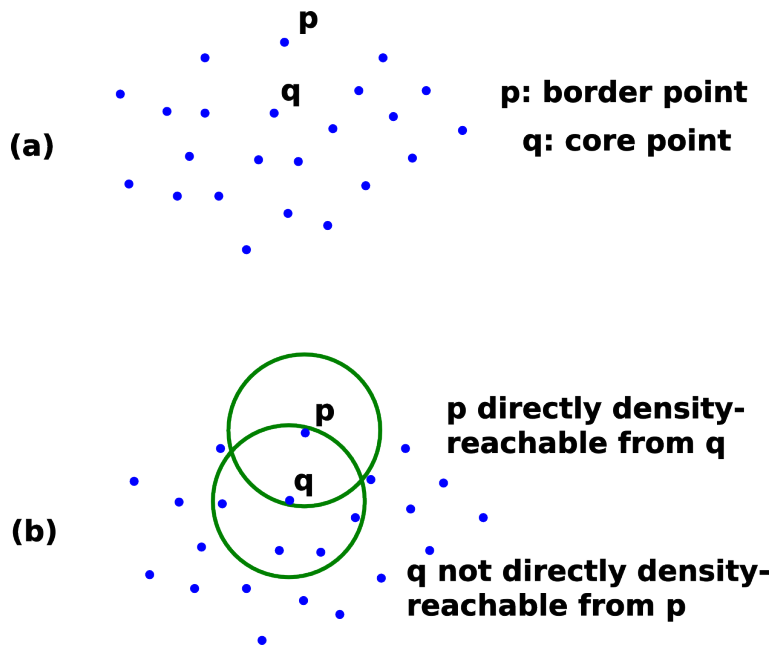


Figure 2.6: DBSCAN core points and border points. Adapted from [Ester et al., 1996]

Density-connectivity is a symmetric relation and for density-reachable points is also reflexive. Density-connectivity is shown in figure 2.7 (b).

Definition 5 (cluster) - A cluster C is a non-empty subset of D if:

- $\forall$ *p*, *q*: if $p \in$ C and *q* is density-reachable from *p* then $q \in$ C (maximality), and
- $\forall$ *p*, $q \in$ C: *p* is density-connected to *q* (Connectivity)

So, the cluster is defined to be a set of density-connected points which is maximal density-reachability.

Definition 6 (noise) - Let C_1, ..., C_k be the clusters of a database D and parameters Eps_i and $MinPts_i$, $i = 1, ..., k$. Noise is the set of points in the database D that don't belong to any cluster C_i, so:

- noise = $\{p \in D \mid \forall i: p \notin C_i\}$

Noise is a set of points in a database D that don't belong to any of its clusters.

DBSCAN performs well with clusters of different shapes, but according to [Moreira et al., 2005], is not efficient with clusters of variable densities.

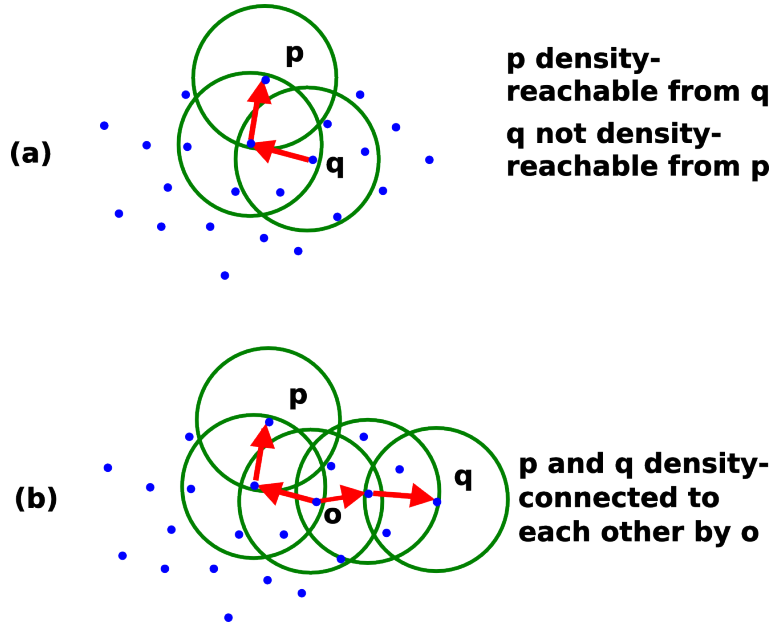


Figure 2.7: DBSCAN density-reachability and density-connectivity. Adapted from [Ester et al., 1996]

2.3 Shared Nearest Neighbour

Like DBSCAN, SNN is also a density-based clustering algorithm. [Jarvis and Patrick, 1973] suggested for the first time the idea of using as a similarity measure the number of closer neighbours that two points share. [Ertoz et al., 2002] and [Ertoz et al., 2003] present and improve the Shared Nearest Neighbour - SNN, that allows to find clusters in varying shapes, sizes and densities, even in the presence of noise and outliers. In this new evolution, the algorithm can manipulate large datasets with variable densities and automatically determines the total number of clusters. While similar to DBSCAN, the SNN approach it slightly different. The number of neighbours that two points share defines the similarity of points. The algorithm requires three input parameters:

- *k* - the neighbourhood list size;
- *Eps* - the density threshold;
- *MinPts* - the threshold to consider that a point is a core point

As the similarity in this kind of algorithms is set by the distance between points, the algorithm also needs a distance function to define the list of *k*-nearest neighbours. Generally the euclidean distance function is used when points are inside a two dimensions plan. But sometimes depending of the domain of the application, it may be needed another more specific distance function, for instance, if we need a more precision distance on a spheroid, like planet earth, Vicenty's formulae [Vincenty, 1975] must be used.

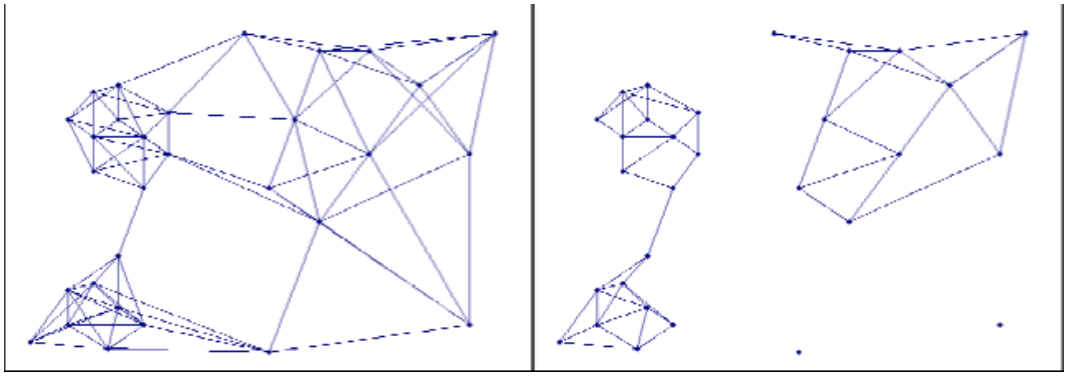


Figure 2.8: Nearest Neighbour Graphs. [Ertoz et al., 2003]

SNN algorithm address the problem where DBSCAN fails, i.e., to identify clusters with different densities. Figure 2.8 helps to visualise two properties of the shared nearest neighbour graph. Left graph shows link of every point to its five most similar neighbours. Right graph shows unweighted shared nearest neighbour graph. On this graph there is a link between two points p_1 and p_2 only if p_1 and p_2 have each other on their nearest neighbourhood lists [Ertoz et al., 2003]. When previous condition is not verified and point loose their links they become noise points. Also observable is the maintenance of the links in regions of uniform density and the break of links in transition regions, solving the question of the difficulty of identify clusters in varying densities.

[Ertoz et al., 2003] explain the eight steps that the algorithm take for the clustering process:

1. Compute the similarity matrix.

   This is a similarity graph with points for nodes and borders, whose weights define the similarity between points;

2. Sparsify the similarity matrix by keeping only the the k most similar neighbours.

   Keep only the k strongest links of the previous similarity graph;

3. Construct the shared nearest neighbour graph from the sparsified similarity matrix.

   In this step we can apply the similarity threshold and find the connected components to obtain the clusters [Jarvis and Patrick, 1973];

4. Find the SNN density of each point.

   Using the user-defined parameter *Eps*, find the number of points that have an 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 an 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 on 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 non-noise, non-core points to clusters.

   This is done by assigning those points to the nearest core point.

This algorithm is an extension of Jarvis-Patrick [Jarvis and Patrick, 1973](step 3) and DBSCAN [Ester et al., 1996](steps 4-8). The great difference and contribution from [Ertoz et al., 2003] is the similarity measure implemented in steps 1 and 2, looking at the number of nearest neighbours that two point share, as said before.

[Ertoz et al., 2003] explains that input parameter *k* is very important to the final cluster definition, because it determines the granularity of the clusters. If *k* value is too small, even an uniform cluster can be broken into pieces due to the local variations in the similarity will tend to find many small, but tight, clusters. On the other hand if *k* value is too large, then the algorithm will tend to find only a few large well-separated clusters, and small local variations in similarity will not have an impact on the clustering process. Once the neighbourhood size is fixed, the nature of the clusters that will be produced is also fixed [Ertoz et al., 2003].

For a short demonstration of the SNN clustering algorithm, and the influence of the input parameters in the clustering results, a small sample data set with 322 objects is used. The spatial distribution is shown in figure 2.9

Performing the SNN algorithm with a valid combination of the input parameters, *k*=7, *Eps*=2, *MinPts*=5, the results shown in figure 2.10 are obtained. The clustering algorithm was able to identify clusters of variable density and shape, also identifying noise points. Each cluster is plotted with a different colour, while noise points are presented in black.

An example of an inappropriate clustering result due to the inadequacy of the chosen input parameters is shown in figure 2.11. In this case, the input parameters were *k*=7, *Eps*=3, and *MinPts*=5. A small change in the parameters can produce a significant difference in the output results. This

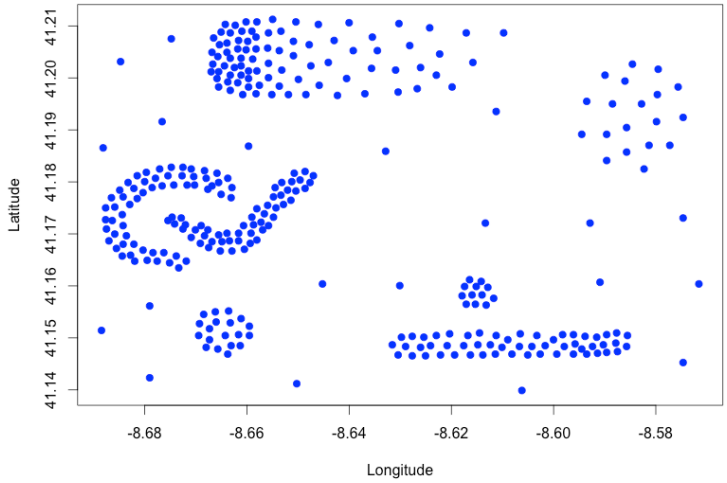


Figure 2.9: Spatial distribution of sample-322

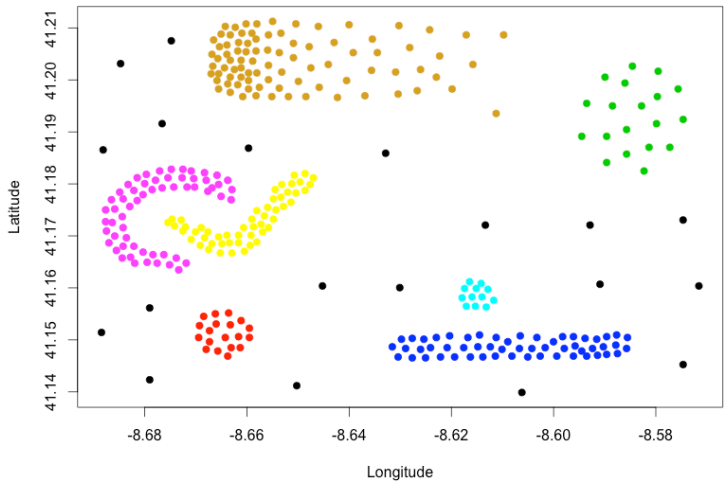


Figure 2.10: Sample-322 expected clusters

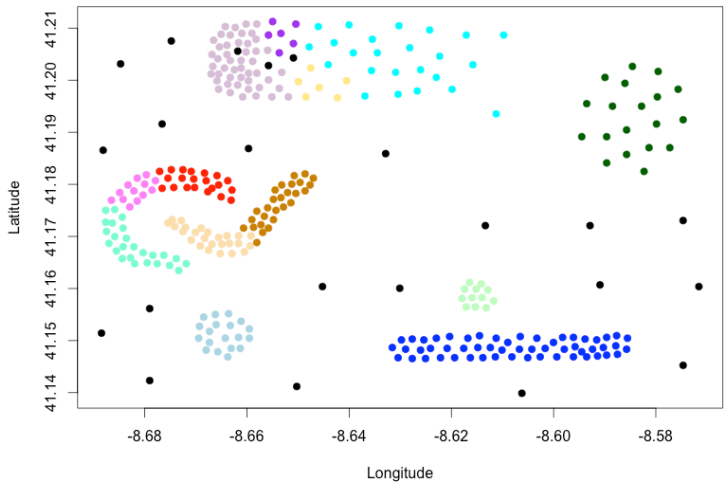


Figure 2.11: Inappropriate clustering result for the sample-322 data set

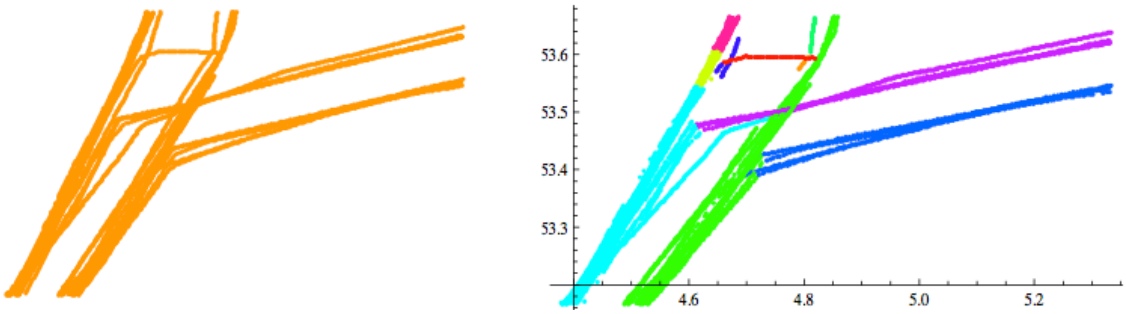


Figure 2.12: Clusters identifying routes of ships. [Santos et al., 2012]

example shows the relevance of setting the algorithm input parameters and the difficulty of this task as no objective guidelines are available to help the analyst.

Another example of a real life application is shown in figure 2.12. It shows the utilisation of the SNN algorithm to find the traffic routes of some kind of ships [Santos et al., 2012]. Left graph shows all points of the trajectories, while right graph shows clustered traffic routes.

2.4 Input Parameters Tuning

One of the biggest issues in clustering algorithms is the need for user input parameters, which usually is a trial and error process in which the data set and the algorithm are adapted to each other. In most cases it is necessary that the user have domain knowledge to determine the input parameters. It is also important to know the behaviour of the algorithm and the impact that parameters

change may take on the clustering result. One of the main challenges that researchers are trying to improve in spatial clustering algorithms, is to find workarounds to minimise the need for user input parameters. In this chapter, several proposed approaches are analysed, mainly associated with the DBSCAN algorithm.

2.4.1 Followed approaches

One of the possible approaches, to solve the need of several input parameters, regardless of what algorithm, is to minimise the number of input parameters by finding relations between parameters and identifying a function that gives us one parameter given other input parameter.

[Ertoz et al., 2003] give some insights for solutions of this kind, saying on the parameterisation section of the SNN, that the MinPts parameter should be a fraction of the neighbourhood list size k:

$$MinPts = \frac{k}{x}$$

[Ester et al., 1996] dedicate a section of the DBSCAN paper explaining a heuristic they developed to determine the parameters *Eps* and *MinPts* of the thinnest cluster in the database. This heuristic is also followed by [Birant and Kut, 2007]. First step of the heuristic is to determine the distances to the *k*-nearest neighbours for each object. Then these *k*-distance values need to be sorted in descending order. According to [Ester et al., 1996], the graph of this function gives some hints about density distribution on the database. The threshold is the first "*valley*". All points with higher *k*-dist value, placed left of the threshold are considered noise points and all other points, on right of the threshold, are assigned to some cluster. This value should be used as the *Eps* parameter.

This solution is very simple working with a graphical representation, because user can identify the "*valley*" on the graph but is very difficult to determine automatically. An example of a sorted *k*-dist graph is shown on figure 2.13.

Silva's work [Silva et al., 2012] also followed Ester's heuristic of the "*first valley*", but modified it not only to choose the first "*valley*" but also to give the user the possibility of choosing other breaks(valleys) in the graph. This means that the clustering process can be influenced by producing more clusters or less clusters depending if the break is more to the left or to the right respectively.

In Birant's work [Birant and Kut, 2007] it is also concluded that $MinPts \approx \ln(n)$, where n is the size of the database. *Eps* should be selected to less than the distance defined by the first valley.

[Liu et al., 2007] introduced VDBSCAN (Varied Density Based Spatial Clustering Applications with Noise), improving DBSCAN difficulty of working with different densities. The solution they found

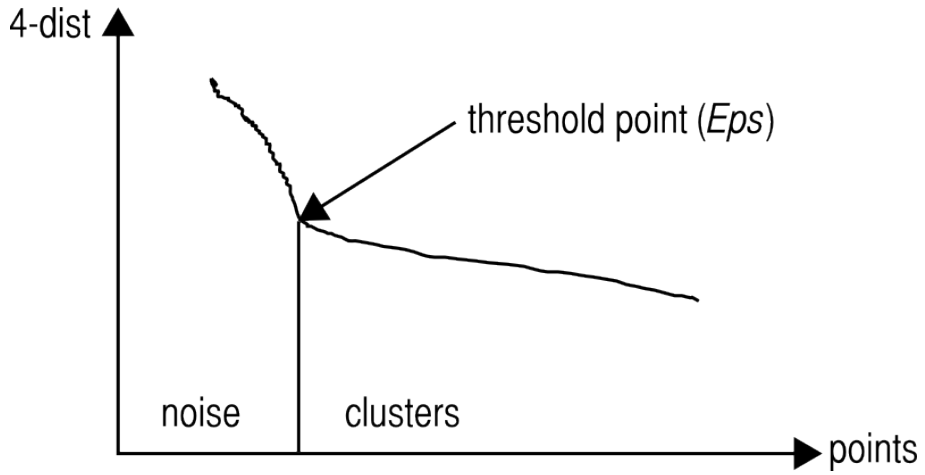


Figure 2.13: Sorted 4-dist graph for sample database. Adapted from [Ester et al., 1996]

was based on previous solution based on the *k*-dist plot of figure 2.13. They don't define just one threshold value. They plot the graph that usually has several sharp changes, each one corresponding to a variation of the density. Then each variation corresponds to a different value of the input parameter *Eps*. Even so, the algorithm still needs the parameter *k* to be chosen by the user subjectively.

LD-BSCA (A Local-DensityBase Spatial Clustering Algorithm) [Wei and Liu, 2009], is an algorithm also based on the DBSCAN algorithm. The authors modified the original algorithm creating a notion of local MinPts automatically generated. Their solution still requires that the user inputs just one parameter *Eps*(ε). However user must have enough knowledge of the domain, because this parameter is crucial for the clustering result.

Like many other authors, [Tripathy et al., 2011] try to solve the limitations of DBSCAN. In their proposal FDCA (A Fast Density Based Clustering Algorithm for Spatial Database System), the authors create the concept ***cluster constant*** and use a new measure for calculating the density of a point. The parameter *Eps* is calculated by the algorithm while the parameter *MinPts* is user-dependent.

Another tentative to reduce user intervention in density based clustering algorithms was taken by [Lelis and Sander, 2009] with SSDBSCAN (Semi-Supervised Density Based Clustering). This algorithm unlike other unsupervised algorithms, uses labeled objects to help determine the clusters. These labeled objects are used to find the density parameters in the dataset. It reduces the user input to the parameter *MinPts*. Needless to refer that this process of using labeled objects, can bias the results.

In a comparison between DBSCAN and SNN [Moreira et al., 2005], used values for *Eps* and *MinPts* were dependent of the *k* value. The authors suggested that, using the results of clustering small

data sets between 300 and 523 points with SNN, *Eps*=0.3**k* and *MinPts*=0.7**k*.

In another study by [Liu et al., 2012], STSNN (Spatial Temporal Shared Nearest Neighbour) a similar approach as the previous one, was taken. The purpose of their work was to find clusters in spatio-temporal datasets. Their algorithm, at first needed three input parameters, k, k_t and *MinPts*, but they verified that both k_t and *MinPts* are highly correlated to k. They realised that those parameters can be defined as a percentage of k. After their tests and analysis, they suggest that both k_t and *MinPts* can be set to a value around 0.5k.

In this research, we chose to use the SNN algorithm, because although the DBSCAN is really more effective in discovering clusters of arbitrary shapes than previous density-based algorithms, some studies have proven that SNN performs better since it can detect clusters with different densities while DBSCAN cannot [Moreira et al., 2005].

2.4.2 Existing Limitations and Research Opportunities

After the analysis of several works available in the literature, it was possible to verify that no conclusive guidelines for the input parameters tuning on spatial clustering algorithms in general and SNN in particular are available. There is here a research opportunity to propose and develop a solution that can solve this need and help users to minimise their intervention in the clustering process. [Biçici and Yuret, 2007] states, referring to DBSCAN algorithm, that ***"finding the correct parameters for standard density based clustering, is more of an art than a science"***.

As related before there is probably two paths to be followed. One is to develop heuristics based on the experimental tests with several different combinations of input parameters, and compare the clustering results to see if it's possible to find a correlation between parameters regarding the behaviour of the algorithm. Another possible path is to find or create a density measure for the database, and make the input parameters depend on that density. The problem, not being so new, is recent, and there has been several studies that try to achieve a solution, but always is compromised by some additional parameter, user intervention or knowledge about the problem domain *a priori*. There is here, definitely, a research opportunity of a actual and challenging problem that need a solution.

Next chapter describes a detailed analysis of the three input parameters and also the main findings that emerged from this analysis.

3 SNN Input Parameters Self-Tuning

This chapter presents the preliminary experiments made in a exploratory way to find if suggestions from other authors could lead to a solution, after is reported the proposed approach, the preliminary results and finally the adopted heuristics.

3.1 Preliminary experiments and analysis

The starting point on this research was to test and develop Ester's heuristic [Ester et al., 1996], that was reviewed in the related work. Recalling their findings, the first step of their heuristic is to determine the distances to the *k*-nearest neighbours for each object. Then these *k*-distance values need to be sorted in descending order. According to Ester, the graph of this function, shown in figure 3.1, gives some hints about the density distribution on the database. The threshold is the first "*valley*". All points with higher *k*-dist value, placed left of the threshold are considered noise points and all other points, on right of the threshold, are assigned to some cluster. This value should be used as the *Eps* parameter.

Although this solution is very simple working with a graphical representation, because user can visually identify the *"valley"* on the graph, it is very difficult to determine automatically. To determine this value using the computer it is necessary to develop an algorithm that logically infers the "*valley*" analysing the sorted values and determining where the break is most significantly.

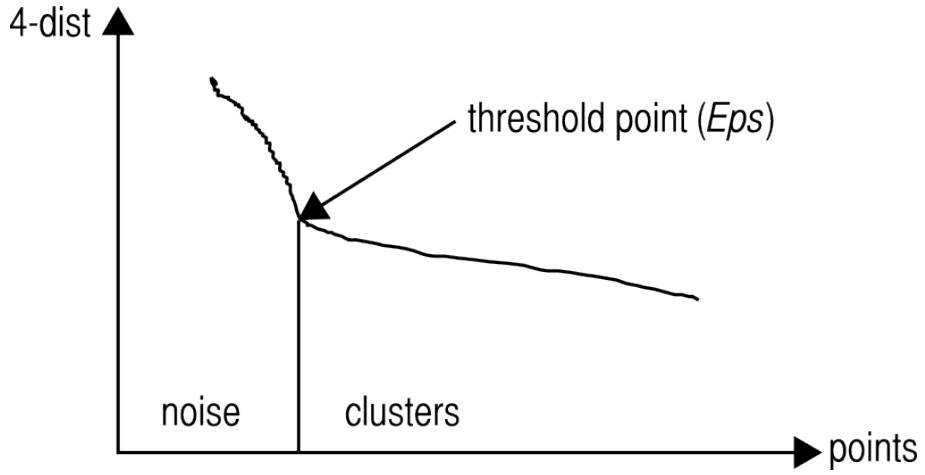


Figure 3.1: Sorted 4-dist graph for sample database. Adapted from [Ester et al., 1996]

Ester et al, also stated that, in their experiments *k*-dist graphs for *k*>4 do not significantly differ from *k*=4, and also at the time, in 1996, computation power requested for these calculations was an

issue to consider. So, they proposed the 4-dist graph, and the 4-dist value threshold point is used as the *Eps* value for DBSCAN [Ester et al., 1996].

In this work, and in order to validate the heuristic proposed by Ester, several programs were developed in R, using R-Studio.

R (http://www.r-project.org/) is a language and environment for statistical computing and graphics. RStudio (http://www.rstudio.com/) is a powerful development environment and user interface for R language. Both are available for free.

Also used for visualisation purposes was used QGIS, previously known as "Quantum GIS"), which is a cross-platform free and open source desktop geographic information systems (GIS). It can be found on (http://www.qgis.org/)

One of used data sets is the MARIN_AIS_Dataset_2a-LPG, a real data set that contains 4168 points with ships locations is show in figure3.2, seen in the working environment of QGIS.

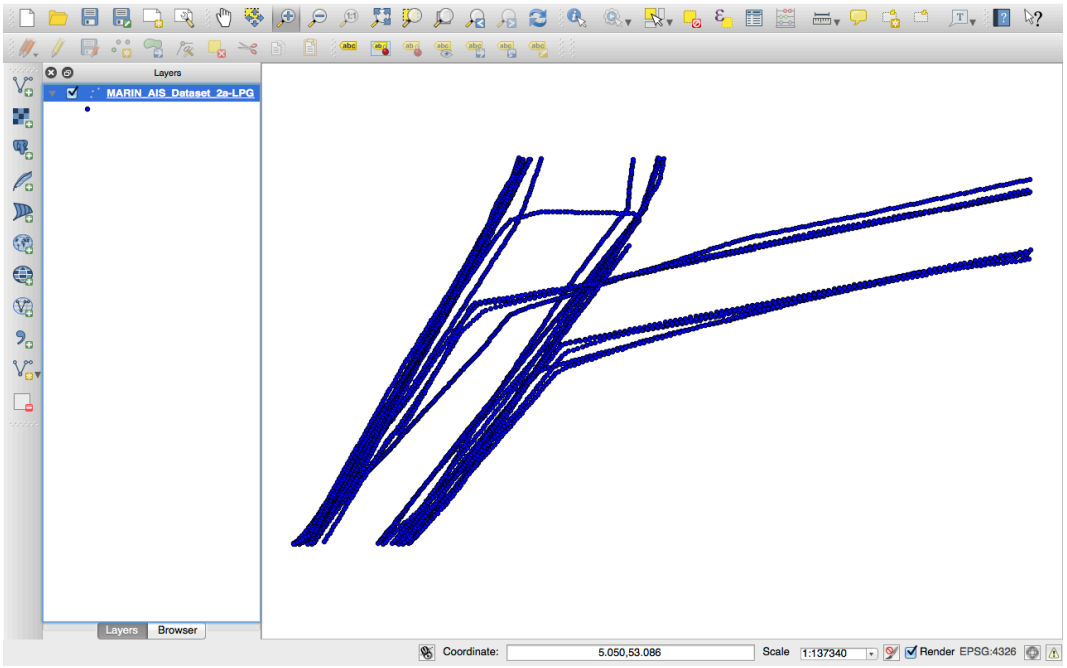


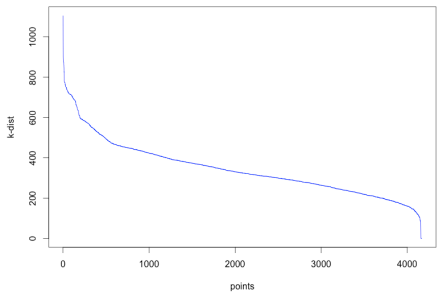
Figure 3.2: Marin_LPG data set spatial distribution

An example of the code developed in R to create the *k*-dist graph is shown in program listing 1.

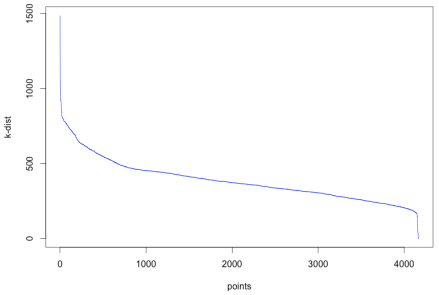
Program Listing 1 Sample code in R to plot *k*-dist graph

```
1  #sorted k-dist graph
2
3  #Load database
4  database <- read.csv2("/Users/gm/Dropbox/MEGSI/Dissertacao/R/Marin-LPG/MARIN_AIS_Dataset_2a-LPG.txt")
5
6  #input value for k to design k-dist
7  k <- 4
8
9  #input number of points to analise
10 p <- 4168
11
12 #definition of dataset
13 dataset <- database[1:p,]
14 coord <- dataset[,c(6,5)]
15
16 #create variable to keep k-dist of every point
17 k_dist <- array(1:nrow(dataset))
18
19 #create variable to keep distances to each point
20 dadosDist <- array(1:nrow(dataset))
21
22 library(geosphere)
23
24 #calculate distances between all points
25 for (i in 1:nrow(dataset)) {
26         actualPoint <- dataset[i,c(6,5)]
27         for (j in 1:nrow(dataset)) {
28                 dadosDist[j] <- distVincentySphere(actualPoint,coord[j,])
29         }
30         t <- sort(dadosDist) #sort distance array
31         k_dist[i] <- t[k+1] #save distance of k(th) point to array
32 }
33
34 #sort descending order
35 sorted_k_dist <- rev(sort(k_dist))
36 plot(sorted_k_dist, col="blue", type = "l", xlab = "points", ylab = "k-dist")
```

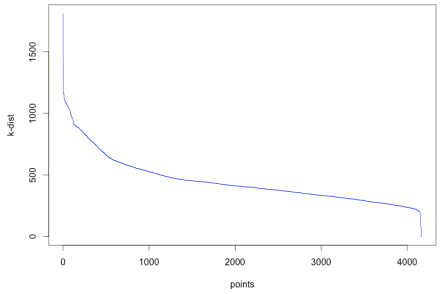
The *k*-dist script was run with *k* values between 3 and 10. Outputs of sorted those *k*-dist graphs are shown in figure 3.3. Ester, that was also followed by [Birant and Kut, 2007], suggested that starting from *k*=4, the graph doesn't change significantly. However, the outputs are not so similar as Ester said.



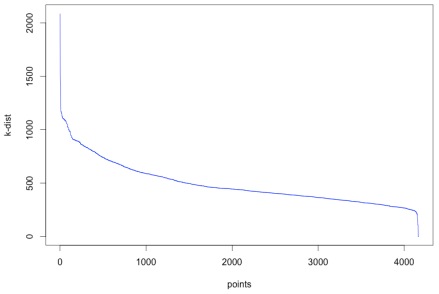
(a) k=3



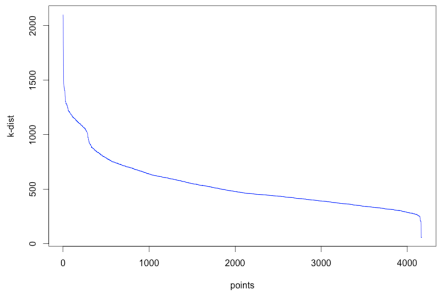
(b) k=4



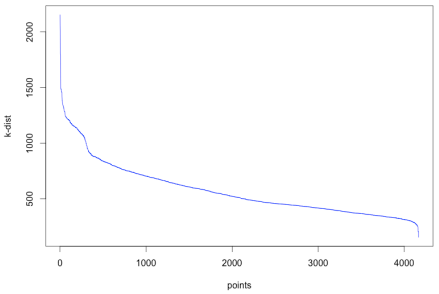
(c) k=5



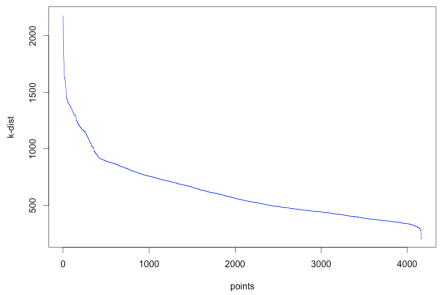
(d) k=6



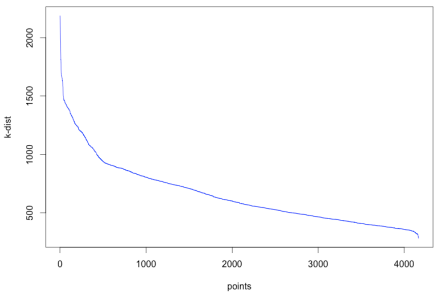
(e) k=7



(f) k=8



(g) k=9



(h) k=10

Figure 3.3: Sorted k-dist graphs with k between 3 and 10

As the looks of the outputs were not conclusive, the R application was modified to infer the break value ("*first valley*") in each *k-dist* graph.

As said before, to determine these values it was necessary to develop an algorithm that logically infers the break points. The strategy used, was adapted from the one used by [Silva et al., 2012].

Previous script developed in R to plot the sorted *k-dist* graph was modified to include the inference of the *"valley"* or break point.

As we have already an array with the sorted *k-dist* values, the idea is to read the values in that sequence, and for each point *k-dist*(*x*) it is calculated the difference between previous point, *k-dist*(*x*) - *k-dist*(*x*-1) and keep and update the *average* of these differences. We also define a *factor* that we think it is big enough to consider a break, started with value = 3. So, when we reach a point where (*k-dist*(*n*) - *k-dist*(*n*-1)) > (*factor* * *average*), we consider to have found the break point.

After implemented these changes in the R application and executed those k-dist plots, the results were very different, not proving Ester's theory. Another approach was needed to continue the research, explained in the next section.

3.2 Proposed Approach

This section presents the proposed approach to find a solution, the test data sets used, and finally the methodology followed.

3.2.1 Used data sets

In this document, the behaviour of the input parameters is tested using four artificial Chameleon data sets [Karypis et al., 1999]: t4.8k (figure 3.4), t5.8k (figure 3.5) and t8.8k (figure 3.7), all three with 8.000 points and the data set t7.10k (figure 3.6) with 10.000 points.

As the analysis made in initial experiments, testing heuristics proposed by other authors did not produce satisfactory results, an exhaustive processing of two data sets was carried out using data sets t4.8k and t5.8k. To understand the behaviour of the SNN input parameters, the strategy used was to perform brute force tests. Executions were made covering all possible combinations of the input parameters within a reasonable range. Before presenting the results, next subsection describes the adopted methodology.

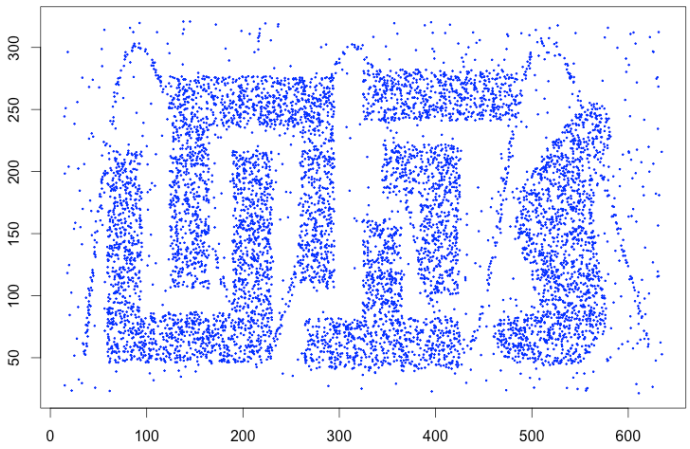


Figure 3.4: Spatial distribution of t4.8k data set

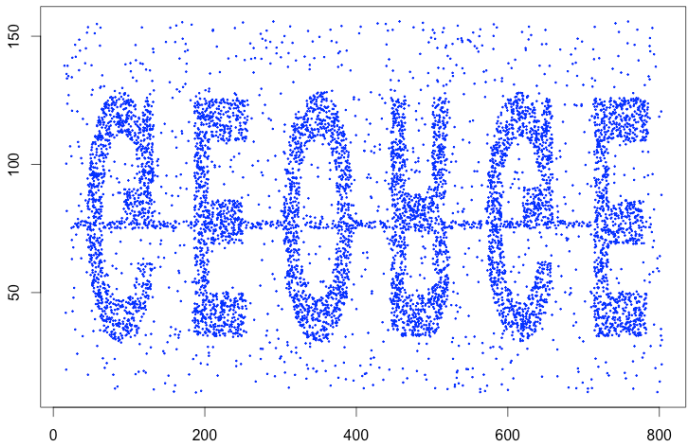


Figure 3.5: Spatial distribution of t5.8k data set

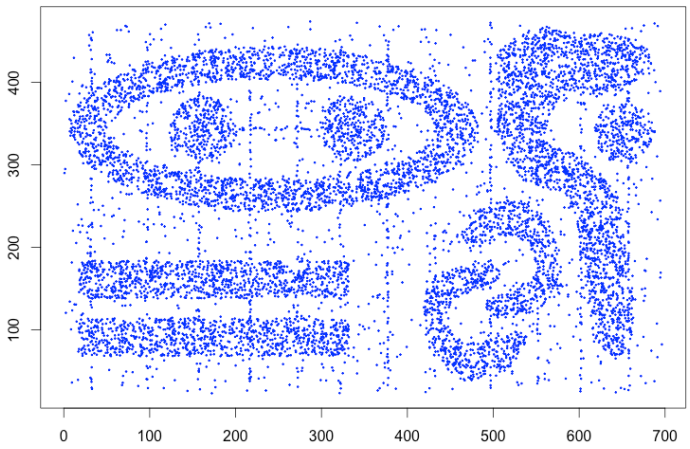


Figure 3.6: Spatial distribution of t7.10k data set

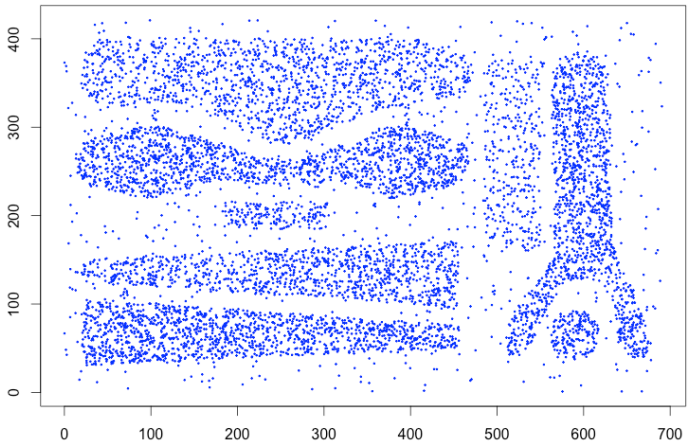


Figure 3.7: Spatial distribution of t8.8k data set

3.2.2 Methodology

The implementation used to perform clustering with the SNN algorithm was the F-SNN [Antunes, 2013]. To speed up the experiments, several scripts were created to run batch SNN executions

with cycles that test all possible combinations within a pre-determined range. An overview of the methodology is shown in 3.8. The steps used in the methodology are:

i) set the maximum number for *k* (*k*-max);

ii) run the F-SNN to create the neighbour's list;

iii) run the clustering algorithm using all possible combinations of *MinPts* and *Eps* with a *k* between 2 and *k*-max;

iv) identify the combinations that produced the number of expected clusters;

v) rerun the clustering algorithm, for the combinations identified in step *iv*), counting the number of points per cluster; and,

vi) assign a quality label to the identified clusters.

Initially, and following the SNN steps, the nearest neighbours' list was created. This is the most time consuming task in a SNN run. Considering the size of the testing data sets, the first batch execution needs to cover a *k* value between 2 and 80 (*k*-max), trying to establish a relationship between *k* and the size of data sets (*n*). The neighbours' list for a *k* equal to 80 (*k*-max) was created and used in all the executions. The limit of 80 for k-max was decided due to the time that the application takes to make an execution with such value. After some executions, that limit was defined, so we could have results available to have this research ready on time.

Using synthetic data sets with 6 clusters each (t4.8k and t5.8k), will allow the identification of the input parameters that produce the expected 6 clusters result. The script was prepared to log the combinations of *k*, *Eps* and *MinPts* that produce 6 clusters, independently of the constitution of each one of these clusters. For covering all possible combinations, batches of nested cycles like the ones shown in program listing 2, were used to run the SNN. This way, it was possible to cover the pre-determined range of *k*'s, and for each *k* it will combine all *Eps* between 1 and *k*-1 (by algorithm definition) and for each pair *(k* and *Eps)* will run SNN combined with *MinPts* between 1 and *k*.

Program Listing 2 Brute force nested cycles

```
for (int i = inf_k; i <= sup_k; i++) {
        for (int j = inf_eps ; j < i; j++) {
                for (int k = inf_minPts; k <= i; k++) {
                        //operations;
                }
        }
}
```

Note that for a range of *k* between 2 and 80, 170.640 executions of the SNN were performed. Due to the time that this process take, these executions were subdivided in small k ranges and were executed simultaneously in different machines. In some occasions 6 different machines were running batches of SNN executions. This allowed to obtain results in a timely manner.

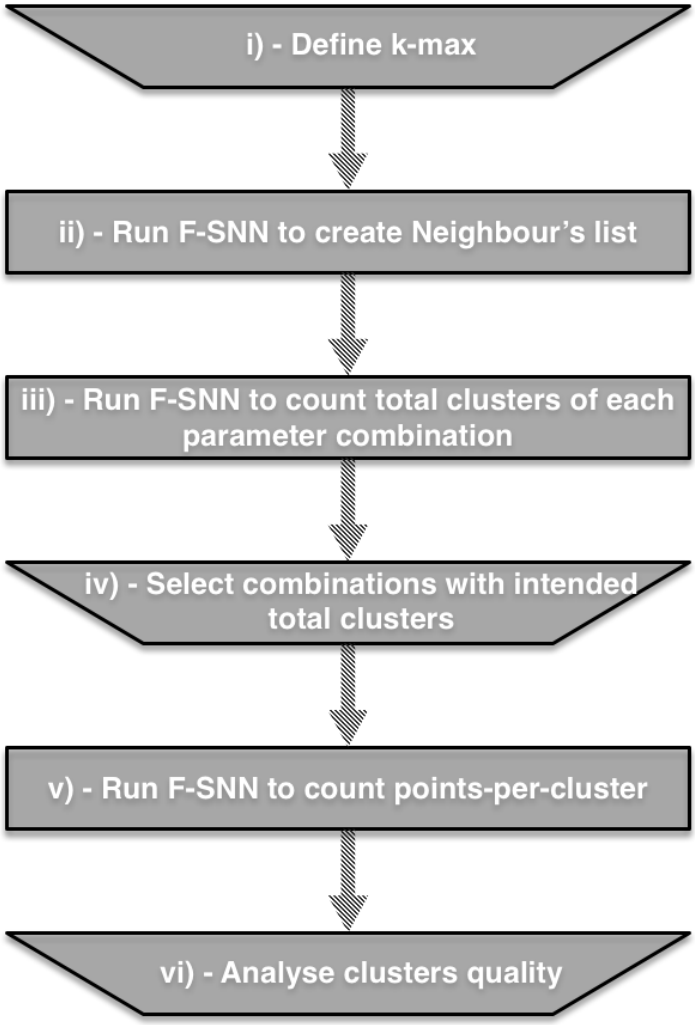


Figure 3.8: Methodology overview

After previous step, all the combinations that produced the 6 expected clusters were identified for further processing. A sample of the output log is shown in program listing 3. Then all combinations that produce a different number of the intended total clusters are deleted. At this point, another script was used to rerun the algorithm for the identified combinations (sample in program listing 4). The aim is to record the number of points that each cluster has, as this value allows an evaluation of the quality of the obtained results. A crop of the output produced in this phase is shown in 5. This

task was not performed earlier due to the time that is needed to execute all possible combinations and to also record the number of points for all of them.

The approach used to measure the quality of the clusters was to analyse the total number of points of each cluster. As it will be explained in the following subsection, the clusters that presented better results were classified with an ordinal scale ranging from excellent to sufficient.

Program Listing 3 Sample of output produced by counting clusters for each parameter combination

```
k eps minPts clusters
_________________
41 16 08   1
41 16 09   1
41 16 10   2
41 16 11   5
41 16 12   5
41 16 13   6
41 16 14   6
41 16 15   6
41 16 16   6
41 16 17   6
41 16 18   7
41 16 19   5
41 16 20   5
41 16 21   4
41 16 22   6
41 16 23   8
41 16 24   9
41 16 25   10
41 16 26   11
```

Program Listing 4 Sample of combinations to count points-per-cluster

```
k eps minPts
____________
41 16 13
41 16 14
41 16 15
41 16 16
41 16 17
41 16 22
```

Program Listing 5 Partial log produced by counting of the points-per-cluster.

```
k eps minPts noise cl1 cl2 cl3 cl4 cl5 cl6
_____________________________________
41 10 38  1085 1190 1214 1175 1063 1060 1213
42 01 38  1039 1202 1224 1186 1063 1067 1219
42 01 39  1080 1194 1216 1176 1061 1059 1214
42 02 38  1039 1202 1224 1186 1063 1067 1219
42 02 39  1080 1194 1216 1176 1061 1059 1214
42 03 38  1039 1202 1224 1186 1063 1067 1219
42 03 39  1080 1194 1216 1176 1061 1059 1214
42 04 38  1039 1202 1224 1186 1063 1067 1219
42 04 39  1080 1194 1216 1176 1061 1059 1214
42 05 38  1039 1202 1224 1186 1063 1067 1219
42 05 39  1080 1194 1216 1176 1061 1059 1214
42 06 38  1039 1202 1224 1186 1063 1067 1219
42 06 39  1080 1194 1216 1176 1061 1059 1214
42 07 38  1039 1202 1224 1186 1063 1067 1219
42 07 39  1080 1194 1216 1176 1061 1059 1214
42 08 38  1039 1202 1224 1186 1063 1067 1219
42 08 39  1080 1194 1216 1176 1061 1059 1214
42 09 38  1039 1202 1224 1186 1063 1067 1219
42 09 39  1080 1194 1216 1176 1061 1059 1214
42 10 38  1039 1202 1224 1186 1063 1067 1219
42 10 39  1080 1194 1216 1176 1061 1059 1214
42 11 38  1040 1202 1224 1186 1063 1066 1219
42 11 39  1081 1194 1215 1176 1061 1059 1214
42 12 38  1040 1202 1224 1186 1063 1066 1219
42 12 39  1082 1193 1215 1176 1061 1059 1214
42 13 38  1042 1202 1224 1186 1063 1066 1217
42 13 39  1083 1193 1215 1176 1060 1059 1214
43 01 39  1034 1205 1223 1187 1064 1066 1221
43 01 40  1074 1197 1214 1178 1059 1063 1215
43 02 39  1034 1205 1223 1187 1064 1066 1221
```

Next subsection describes the obtained results for the two data sets, while the last subsection summarises the main findings of these analyses.

3.3 Preliminary results

This section presents the results of the tests performed with the test data sets t5.8k and t4.8k.

3.3.1 Results for the t5.8k data set

After processing all possible combinations for the values of *k*, *Eps* and *MinPts*, with *k* between 2 and 80, the obtained clusters were compared with the solution that presented better results. This means that the number of noise points as well as the number of points in each cluster was analysed. For this data set, a large number of valid combinations were obtained (23.686 of 170.640 tested combinations) . Due to this large number, the obtained combinations were classified as Excellent, Very Good, Good and Sufficient depending on the number of points clustered correctly. The thresholds used were a correspondence in term of points of 90%, 80%, 70%, 60% or higher for the clusters classified as Excellent, Very Good, Good and Sufficient, respectively. The summary of the classification is shown in figure 3.10

This assignment of a quality label allowed the analysis of the several combinations among input parameters and the clusters' quality. figure 3.9 shows the relation between the different valid combinations of the input parameters and the quality of the results.

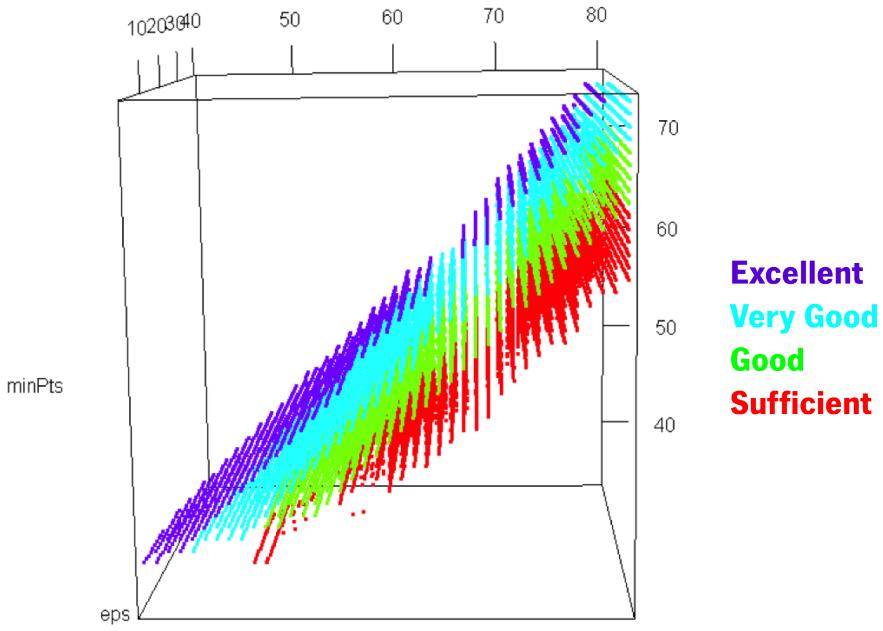


Figure 3.9: t5.8k results of combinations vs. quality

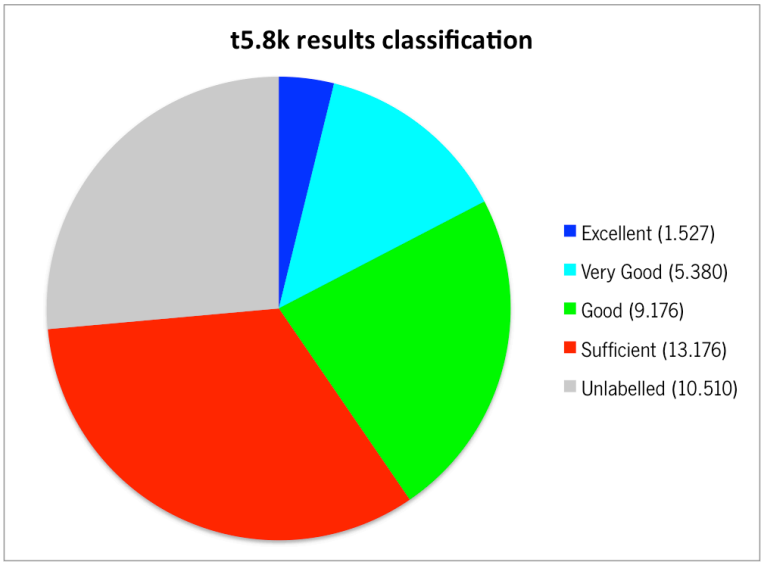


Figure 3.10: t5.8k correct results and quality classification

The relations between input parameters *k* and *Eps* produce de results shown in figure 3.11. Similar analysis can be made regarding relations between *k* and *MinPts* on figure 3.12.

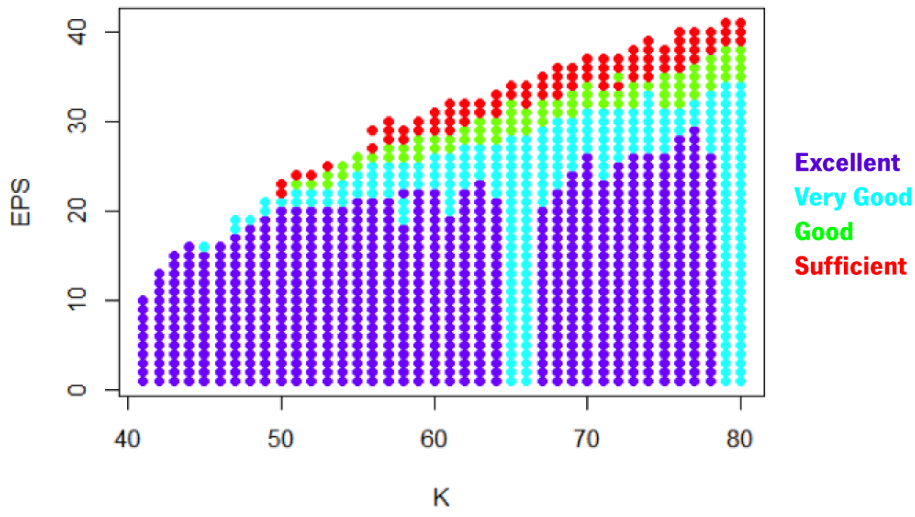


Figure 3.11: t5.8k results regarding *k* and *Eps* relation.

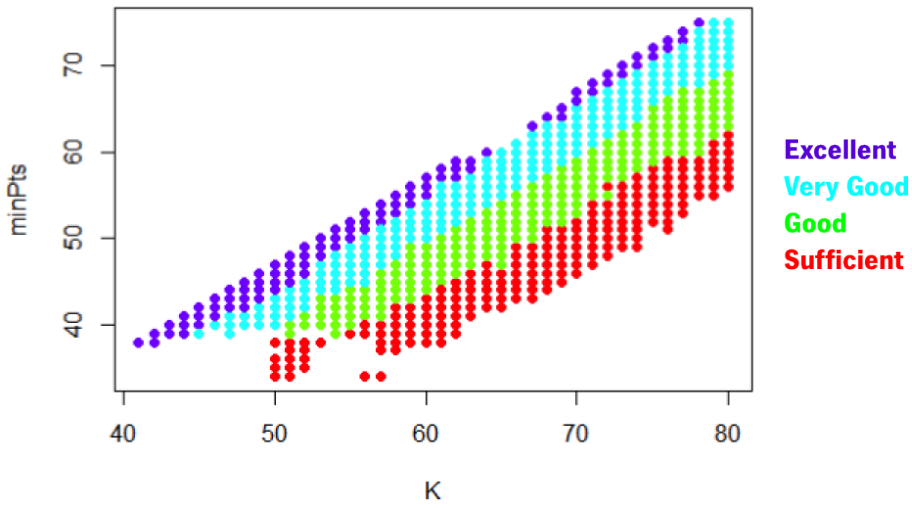


Figure 3.12: t5.8k results considering *k* and *MinPts* relation.

A pattern emerged among the input parameters that clearly influence the clustering results. figure 3.13 and figure 3.14 only consider those combinations that generate results classified as Excellent.

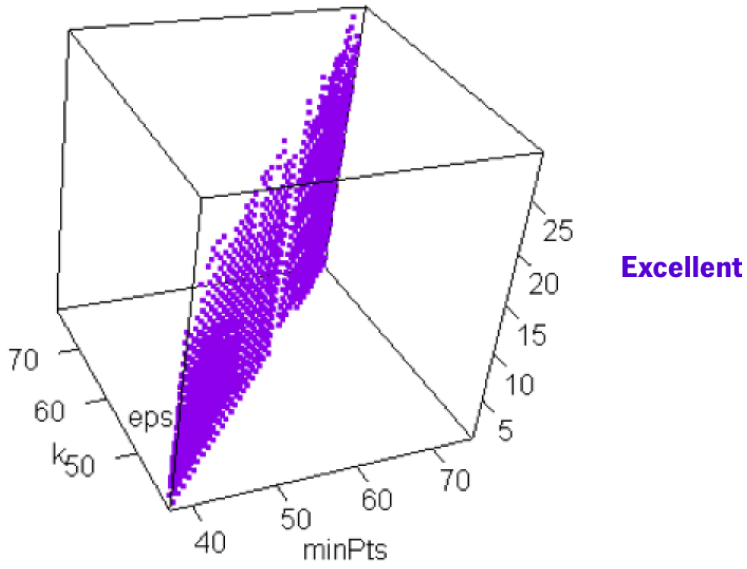


Figure 3.13: t5.8k results classified as Excellent.

As shown in figure 3.14, all the Excellent results were obtained with values of *k* and *MinPts* that are strongly related (R2=0.99885). For each pair of (*k*, *MinPts*), a large range of values for *Eps* can be used without affecting the result quality. In fact, almost any value of *Eps* is acceptable if it is slightly lower that *MinPts*. This analysis is very clear when still in the Excellent results we analyse the both

relations between *k* and *MinPts* and *k* and *Eps*, shown in figure 3.15 and figure 3.16 respectively. We can conclude that there is a strong relation between *k* and *MinPts* while a large quantity of *Eps* values can be used without compromising quality results.

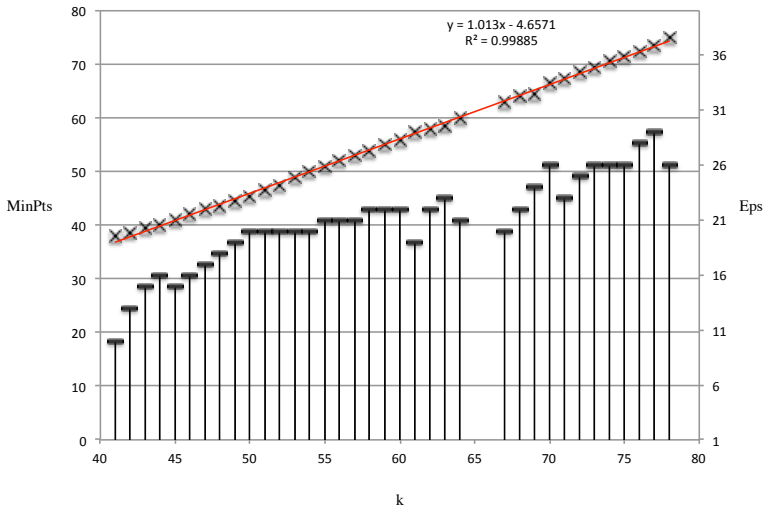


Figure 3.14: Input parameter combinations that produce Excellent results (t5.8k data set)

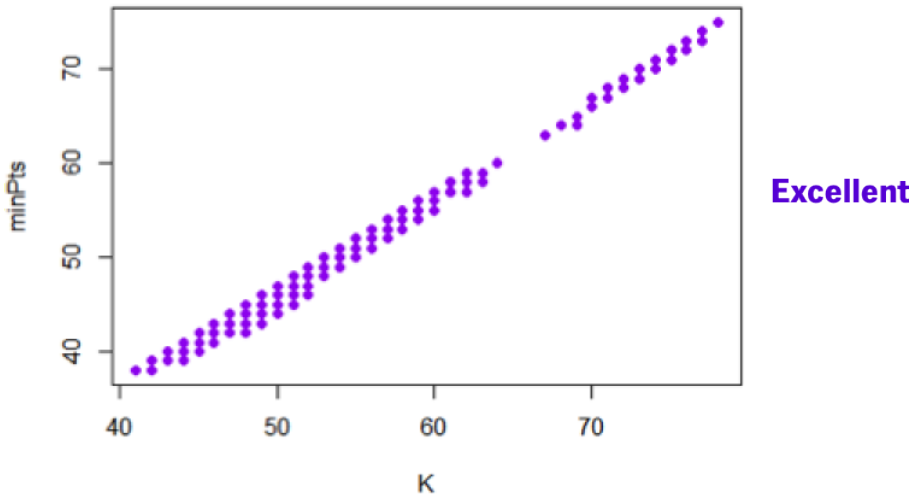


Figure 3.15: t5.8k Relation between *k* and *MinPts* for Excellent results.

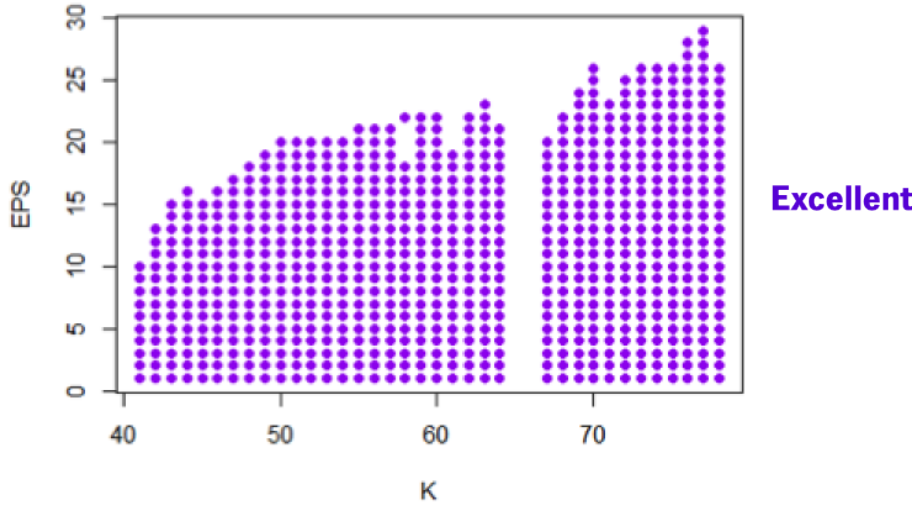


Figure 3.16: t5.8k Relations between k and Eps for Excellent results.

On figure 3.17 is shown the correlation between *k* and *MinPts* on the results for clusters classified as Very Good. The correlation maintains although less stronger as we can notice that for each *k* value there are now some acceptable *MinPts* values. *Eps* behaviour keeps as previous observation.

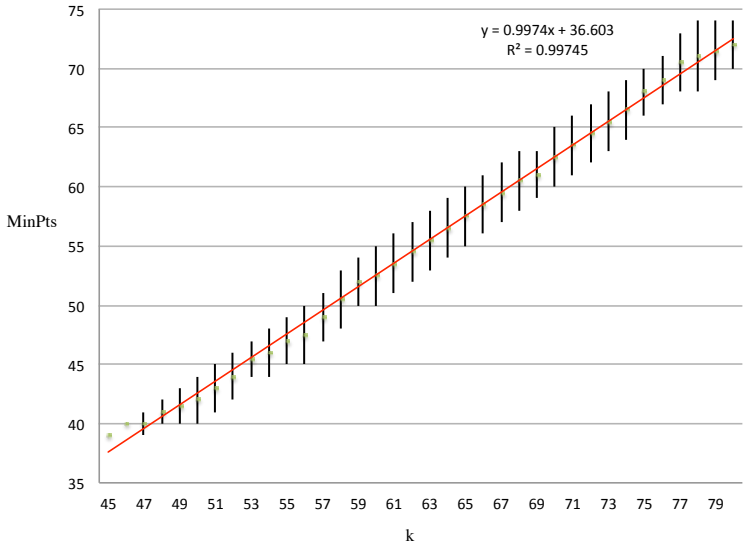


Figure 3.17: Correlation between *k* and *MinPts* of Very Good clustering results (t5.8k data set)

It is worth mentioning that a strong correlation between *k* and *MinPts* maintains despite the decreasing of the quality of the clusters. We can observe the correlation between the parameters in 3.18. The decreasing of the correlation is also explainable due to the fact that the quantity of clusters classified as correct is higher as the quality of the clusters is getting lower. From the total of 23.686

combinations of input parameters that produced correct clustering results for the data set t5.8k, only 1.527 were classified as Excellent unlike Sufficient labelled clusters results that were 13.176 as seen before.

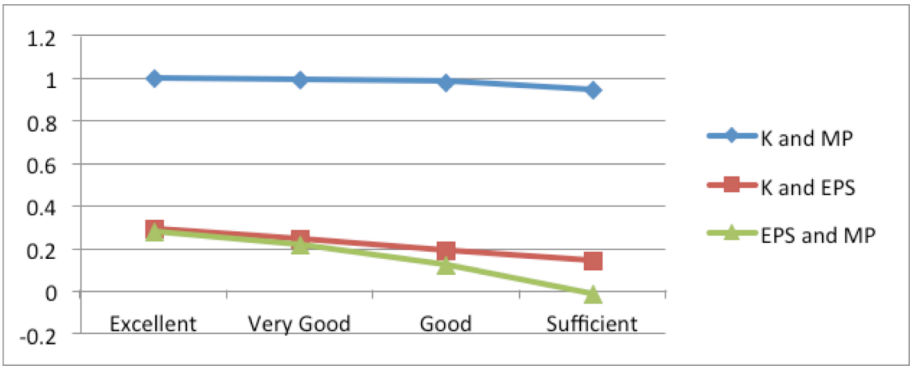


Figure 3.18: Correlation between input parameters in t5.8k data set

3.3.2 Results for the t4.8k data set

Repeating the same process to the t4.8k data set, the assignment of a quality label to the obtained clusters allowed the identification of a similar pattern found in the previous data set.

For the t4.8k data set, fewer possible combinations of the input parameters are available, mostly for the excellent combinations, only 230, as we can observe from table 1 and figure 3.19.

Table 1: t4.8k data set clustering results and classification

Quality	**Results**
Excellent	230
Very Good	1.253
Good	2.758
Sufficient	3.761
Total	4.763

Like it already happened with data set t5.8k, a high correlation between *k* and *MinPts* is still verified (figure 3.20) and, also maintains the high variability of Eps shown in figure 3.21.

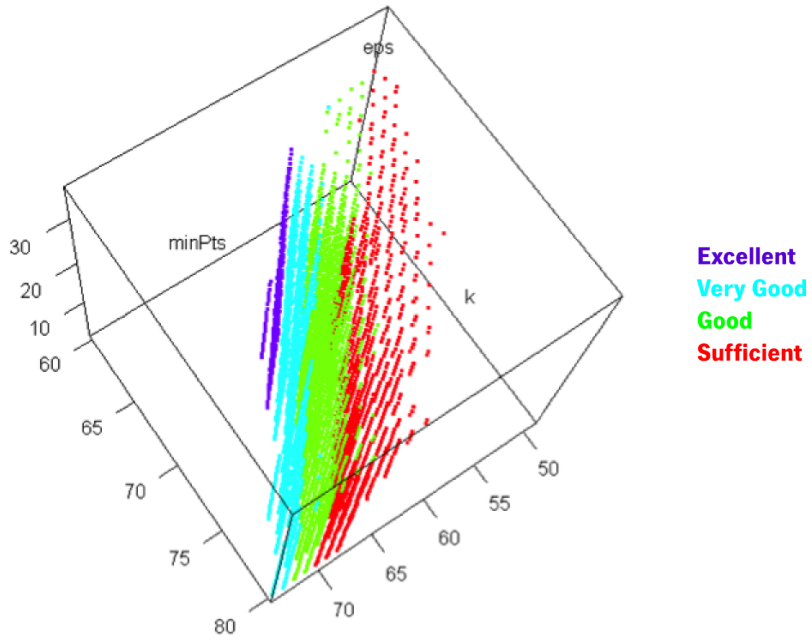


Figure 3.19: t4.8k results of combinations vs. quality

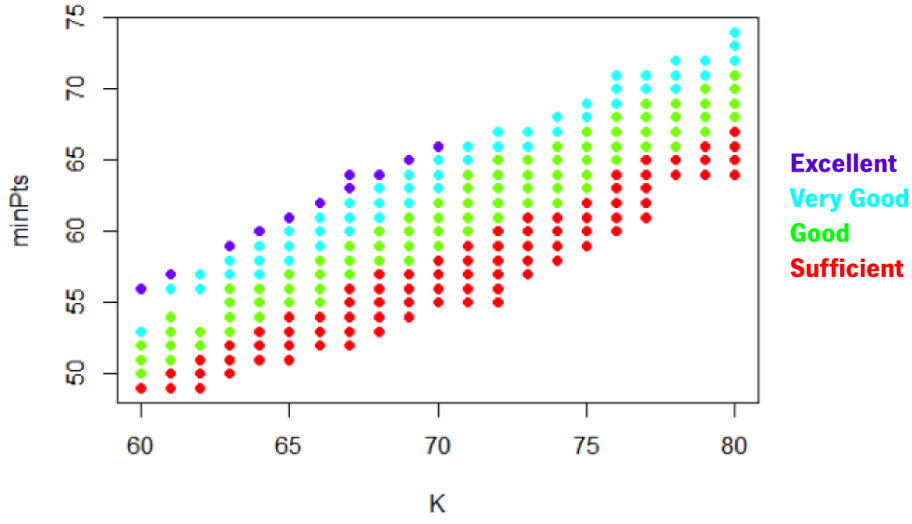


Figure 3.20: t4.8k results concerning *k* and *MinPts* relation

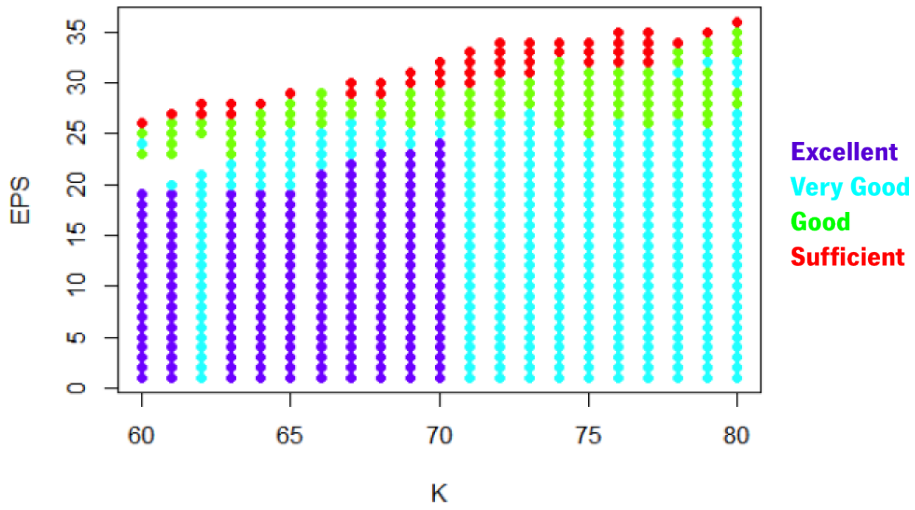


Figure 3.21: t4.8k results regarding *k* and *Eps* relation

For the two data sets, table 2 presents a summary with the intervals that produced the six clusters. The range of valid results for *k* shows that, for both data sets, the expected results started to emerge with a *k* around 40. For *Eps*, the wide range of possible values is verified. This table also shows the possible values for the Excellent clusters, with a visible reduction in the amplitude of the possible values for each parameter.

Table 2: Range of possible input parameters

Data set	***k***	**Avg *k***	***Eps***	**Avg *Eps***	***MinPts***	**Avg *MinPts***
t4.8k (all)	[38,80]	72	[1,50]	19	[24,74]	61
t5.8k (all)	[41,80]	69	[1,54]	19	[17,75]	51
t4.8k (excellent)	[60,70]	66	[1,24]	11	[56,66]	62
t5.8k (excellent)	[41,78]	59	[1,29]	11	[38,75]	55

The best clustering results are achieved for t4.8k with a combination of parameters *k-Eps-MinPts* of 67-22-64 (figure 3.22) and of 46-15-43 for the t5.8k data set (figure 3.23). The semantic of best result can depend of the analytic context in which the data analysis task is being undertaken. In this work, and as already mentioned, it is associated to the ability to identify the higher number of noise points as possible without compromising the constitution of each one of the clusters that must be identified.

Figure 3.22: Clustering result for the t4.8k data set

Figure 3.23: Clustering result for the t5.8k data set

3.4 Adopted Heuristics

This section presents the main findings of the research as well as the adopted heuristic to find the appropriate input parameters.

3.4.1 Main findings

After processing so many combinations and the results provided by each one, it was possible to identify a strong correlation between *k* and *MinPts* and to verify that *Eps* can be inferred knowing *MinPts*. A summary of the results obtained in the tests with the data sets t4.8k, t5.8k, t7.10k and t8.8k is shown in figure 3.24, where we can observe a coherence in terms of the values of *minK*, *maxK*, *avgK*, *bestK* and the value of *n* (the size of the data set).

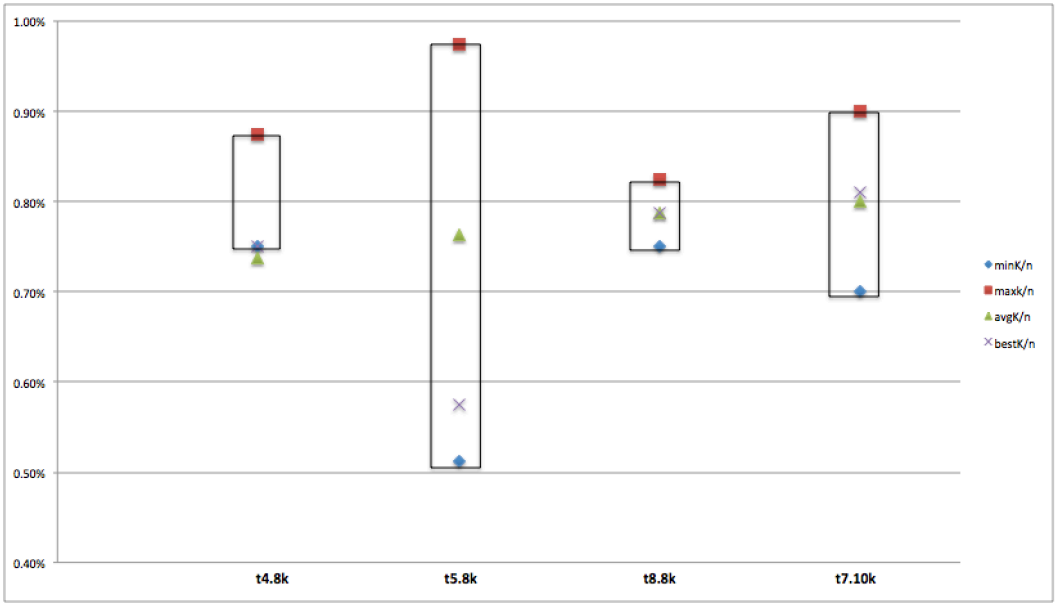


Figure 3.24: Results for t4.8k, t5.8k, t7.10k and t8.8k data sets.

With the available results it is possible to suggest that: *MinPts* should be a value ranging from 92% to 96% of *k* value; *Eps*, a more flexible parameter, should be around 18,5% of *MinPts*. In both cases, and as already mentioned, several valid combinations exist. For the three input parameters, *k* is the most difficult one to estimate. After testing also several random data sets extracted from t4.8k and t5.8k with different number of points, 4.000, 5.000, 6.000 and 7.000, it was possible to verify that *k* is contained in an interval that ranges from 0,70% and 1% of the size of the data set.

3.4.2 Adopted heuristic

Summarising the findings, it is possible to propose the following heuristic to find the starting parameters to run the SNN algorithm:

n = size of the database;
k = *n* * 0,7%;
MinPts = *k* * 94%
Eps = *MinPts* * 18,5%

4 Evaluation of the proposed heuristics

4.1 Chameleon data sets

In order to validate these findings, and also to check if they are independent of the number of points and the number of clusters present in the data set, two other data sets available from the Chameleon algorithm were tested: the t8.8k that integrates 8.000 points and 8 clusters, and the t7.10k that integrates 10.000 points and 9 clusters. For both data sets, table 3 shows the estimated range for *k* using the proposed heuristics, as well as the combinations, classified as excellent, obtained after processing and analysing all the results.

Table 3: Range of estimated *k* and valid input parameter *k*

			Estimated *k*			
Data set	***n***	**Expected clusters**	**0.70%**	**1%**	**Valid *k***	
t4.8k	8.000	6	56	80	60	70
t5.8k	8.000	6	56	80	41	78
t8.8k	8.000	8	56	80	60	66
t7.10k	10.000	9	70	100	70	90

Continuing the validation process, it was necessary to test how sensitive the clustering process is to the number of points in the data set. As the SNN algorithm has a time complexity evaluated in $O(n^2)$ in the worst case [Ertoz et al., 2003], mainly looking for the *k* nearest neighbours of a point, this means that as *n* increases, the estimation of *k* will increase too, making the calculation of the list of nearest neighbours a more difficult task.

Several mutations of the Chameleon data sets were created in order to validate the proposed heuristic. The t5.8k data set was replicated, as shown in figure 4.1. With 16.000 points, the proposed heuristic for k estimates that the range of possible values is between 112 and 160. Processing these values allowed the identification of the expected clusters. However, the initial estimated values, for 8.000 points, also identified the same result (figure 4.2).

Figure 4.1: Duplication of the t5.8k data set

Figure 4.2: Clustering result for the duplicated t5.8k data set

Moreover, independently of the undertaken replication, with for example 32.000 points, shown in figure 4.3, the new estimated values for *k* identifies the expected clusters as well as the previous correct combinations for the 8.000 points data set, as we can observe in without the noise points for better viewing. This means that a pattern exist in the possible input parameters for a data set. If we keep doubling the data set size and the original correct combination of input parameters keeps valid, this means that the reverse path can also be taken. If we have really huge data sets, maybe we can make samples of those data sets continuously dividing them by two, until we can have workable parameters. This is very important as we can avoid large values for *k*, which severely penalise the time needed to compute the clusters.

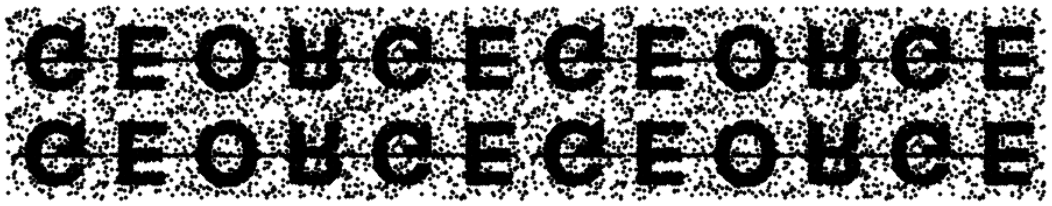


Figure 4.3: Density distribution of the 4 times replicated t5.8k data set

Figure 4.4: Clustering result for t5 dataset with 32.000 points without the noise points

A similar process was used to replicate two times and three times the t4.8k dataset, transforming it in 16.000 (figure 4.5) and 24.000 (figure 4.6) points respectively. The same way as already happened with t5.8k data sets variations, the original correct combinations for t4.8k dataset produced correct clustering results as we can observe in figure 4.7 and in figure 4.6, this one without the noise points plotted.

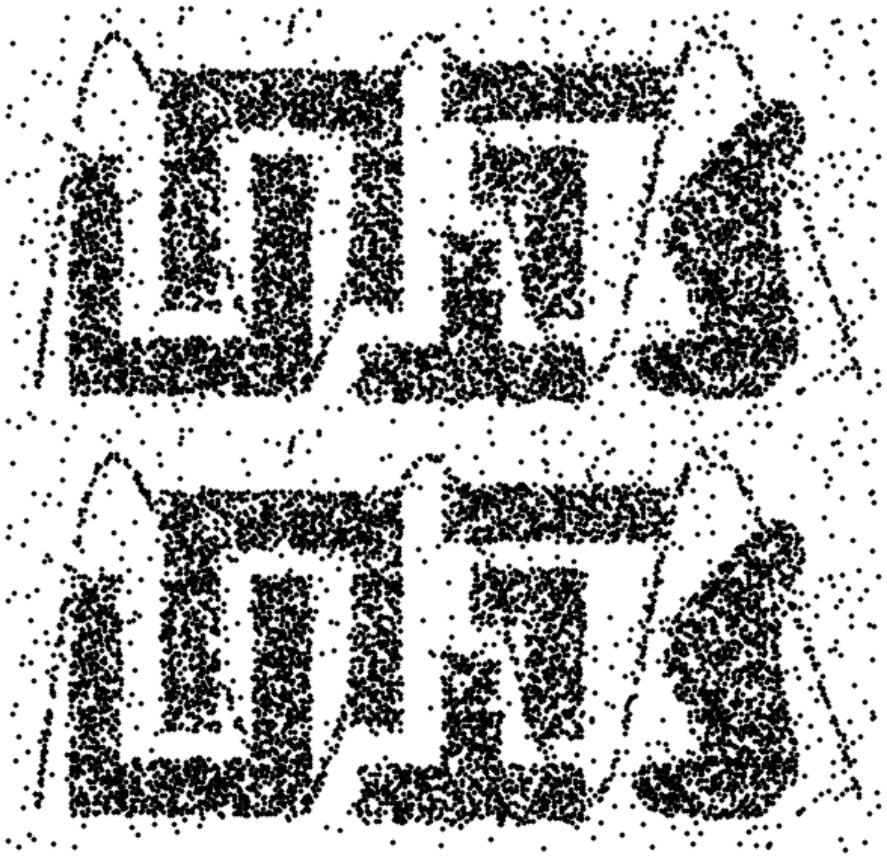


Figure 4.5: Plot of unclustered 2x t4.8k data set

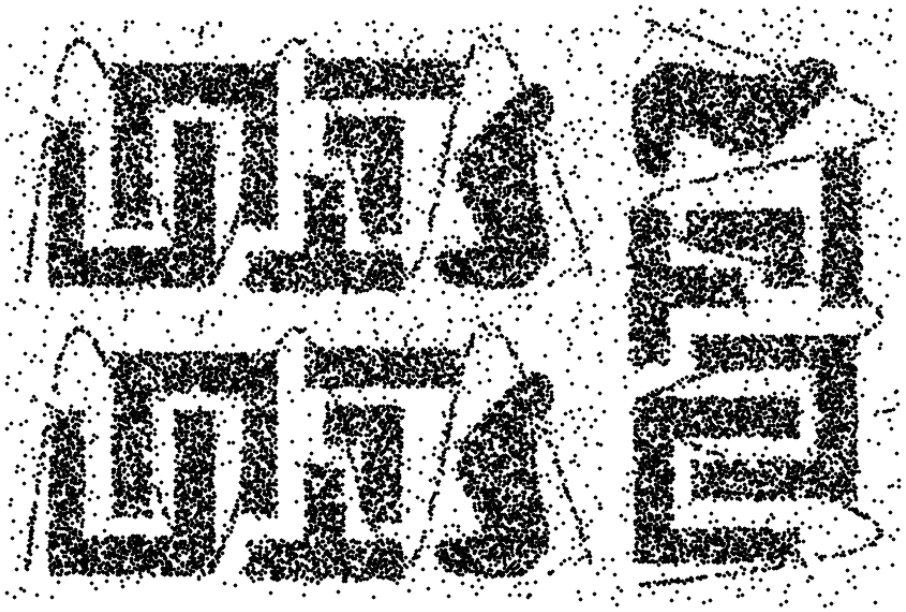


Figure 4.6: Plot of unclustered 3x t4.8k data set

Figure 4.7: Clustering results for the 2x t4.8k data set

Figure 4.8: Clustering results for the 3x t4.8k data set

Other data sets were created resizing the t4.8k Chameleon data set. In order to test if the resizing of a data set, could imply a change in the results, two versions of the data set t4.8k were created in the WSG84 (a standard for use in cartography of the World Geodetic System). Original Chameleon data sets were created in a bi-dimensional (x,y) space. A smaller and a larger data sets in terms of occupied area were created. Applying the proposed heuristic, both datasets produced correct results, the same way as the original did and with the same combination of input parameters *k-Eps-MinPts* of 67-22-64. An image with both overlaid plots in their relative positions is shown in 4.9. The clustering results can be seen in 4.10. This test help us to conclude that the density of the terrain area does not have influence in the clustering process.

Figure 4.9: Plots of two versions of t4.8k data set with different terrain areas occupied

Figure 4.10: Clustering results for both versions of t4.8k data set

4.2 Birch data sets

Another set of test data sets were used to validate the proposed heuristic. It is important to validate the findings in different data sets than the ones that were used to develop the heuristic. There are not many available test data sets suitable to test the heuristic. Birch synthetic 2-d data sets were used (http://www.uef.fi/fi/sipu/datasets). Birch1 is a square shaped data set, which spatial distribution is shown in figure 4.11. Birch2 with a sine curve shape, is shown in figure 4.12. Both data sets have 100.000 objects. In both data sets, the clustering process must identify 100 well-defined clusters. Note that both data sets have an amount of a priori undetermined noise points. Both data sets were generated and used by [Zhang et al., 1997] on the research of the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm.

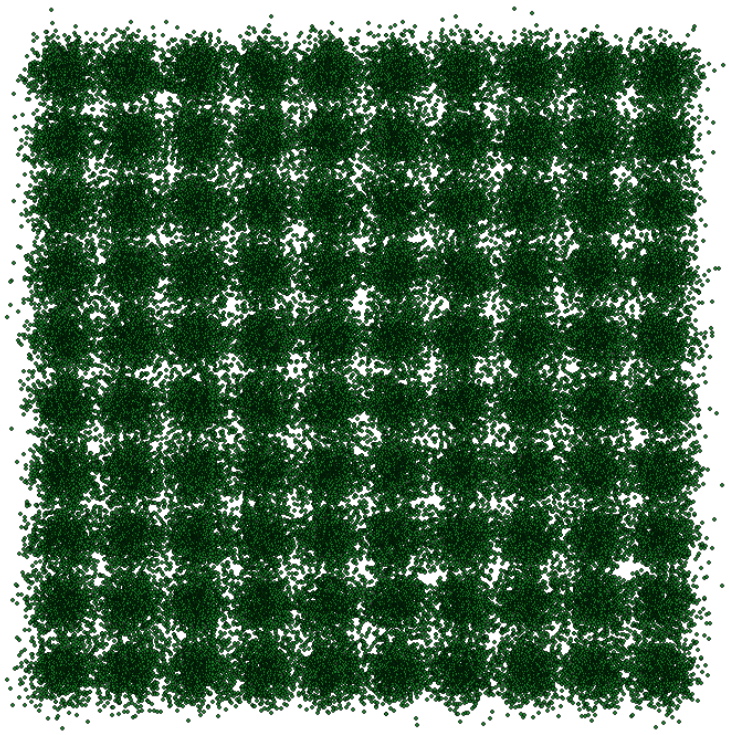


Figure 4.11: Spatial distribution of Birch1 data set

Figure 4.12: Spatial distribution of Birch2 data set

As these data sets have 100.000 objects each one, the heuristic says that a *k* range between 700 and 1.000 is estimated. Finding the *k* nearest neighbours list for this magnitude of values is not possible in a reasonable amount of time.

Taking into consideration the pattern in the input parameters verified for the t5.8k data set, the initial estimated ranges were successively divided by 2 until a *k* lower than 200 is found. This corresponds to a situation where a sample of the initial data set is used to estimate the parameters but all the data set is used in the clustering process. A *k* range between 175 and 250 is estimated. So, taking 700 and dividing it by two, two times, we get a *k* value of 175. MinPts value should be 162, since the heuristic rule choose a value between 92% and 96% of *k* value. The *Eps* value would be 30, since the heuristic rule choose a value around 18.5% of *MinPts* value.

The result of this validation was extraordinary, since the 100 clusters were perfectly identified in

both data sets (figure 4.13 and figure 4.14), providing excellent results. For better reading, both images are shown without noise points.

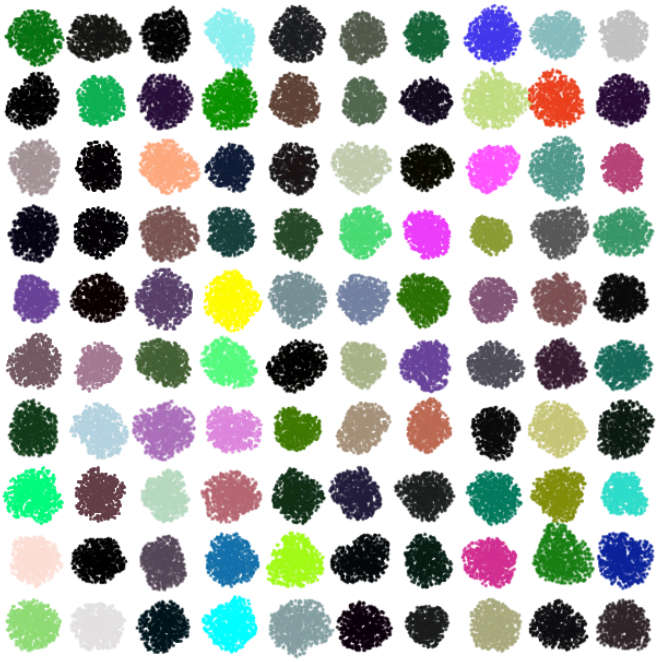


Figure 4.13: Clustering results for the Birch1 data set

Figure 4.14: Clustering results for the Birch2 data set

Normally, what is expected from the heuristic, is the SNN input parameters suggestion for the first execution. Then the user would adjust those values for better adequate the results to the domain of the problem that is being analysed. However in this situation, and for both data sets, the first solution happened to be the perfect one, giving the user what he was expecting to find.

5 Conclusions and Future Work

This section presents the main achievements of this thesis as well as the future work suggestions.

5.1 Main achievements

As a conclusion of this work, it is possible to say that the objectives proposed initially were completed. It was made a literature review to understand clustering and density clustering principles in general and the SNN clustering algorithm in particular. It was tried to follow some other researchers suggestions, but inconclusive. Then, the research option was to perform brute-force test executions, in an exhaustive way.

Using the results of these performed tests it was possible to analyse them visually, analytically and statistically and this allowed the identification of a strong correlation between *k* and *MinPts* and also the verification that *Eps* is the less sensitive input parameter, as it presents a wide range of possible values for each *MinPts* value. It was also possible to identify a pattern for the appropriate *k* value, or range of values, which depends on the size of the data set.

Knowing the number of points, *k*, *MinPts* and *Eps* can be suitable defined. It was also possible to show that, in order to reduce the needed processing time, a sample of the data set, in terms of the number of points, can be used to estimate the input parameters, as the obtained values will work well with the entire data set.

Recalling the main finding of this thesis, the proposed heuristic, that allows to find the combination of input parameters to an execution of the SNN algorithm is the following:

n = size of the database;
k = *n* * 0,7%;
MinPts = *k* * 94%
Eps = *MinPts* * 18,5%

The proposed heuristic was validated successfully as intended initially.

This contribution is not definitive, as the heuristic needs to be tested in other data sets to fully verify it's utility. Nonetheless is an import contribute so the user can have a starting point of analysis and don't loose the historic trial and error time that was needed so far.

A significant number of SNN executions were needed. A total number of 1.253.210 executions were done to count the total number of clusters per combination of input parameters. 102.034 executions were needed to count the number of points in each cluster of selected combinations parameters,

as this was the measure used to evaluate the quality of the clustering results. To be possible to complete this work on time, six different computers, PC's and Mac's were used to process batches of SNN executions. The total run-time of these executions was 138.253 minutes what makes merely 96 days of combined runtime.

5.2 Future Work

As future work, and as the size of the data sets is continuously growing, random samples of the original data sets will be used not only to estimate the algorithm input parameters, but also in the clustering process, through a methodology that would infer the clustering of the entire data set based on the result of a sample. Also, and as part of ongoing work, an analytical tool is under development integrating the findings presented in this thesis. This tool will be public available and will allow users to explore their data sets obtaining results that try to optimise the insight on data. In this tool, the input parameters are auto-tuned attending to the user's guidelines to see more detailed or more aggregated clusters. Other paths should be taken to find alternative causes for the discovery of the appropriate input parameters, other than the size of the database.

References

[Akasapu et al., 2011] Akasapu, A., Rao, P., Sharma, L., and Satpathy, S. (2011). Density based k-nearest neighbors clustering algorithm for trajectory data. Int. J. on Advanced Science and Technology, 31:47–57.

[Andrienko et al., 2011] Andrienko, G., Andrienko, N., Bak, P., Keim, D., Kisilevich, S., and Wrobel, S. (2011). A conceptual framework and taxonomy of techniques for analyzing movement. Journal of Visual Languages & Computing, 22(3):213–232.

[Ankerst et al., 1999] Ankerst, M., Breunig, M., Kriegel, H., and Sander, J. (1999). Optics: ordering points to identify the clustering structure. ACM SIGMOD Record, 28(2):49–60.

[Antunes, 2012] Antunes, A. A. F. (2012). Análise espacial de grandes quantidades de dados de movimento usando técnicas de clustering baseadas em densidade. Master's thesis, Universidade do Minho.

[Antunes, 2013] Antunes, A. A. F. (2013). F-snn. a fast snn-based clustering approach for large geo-spatial data sets, retrieved july 15, 2013, from http://ubicomp.algoritmi.uminho.pt/projects/f-snn/.

[Berkhin, 2004] Berkhin, P. (2004). Survey of clustering data mining techniques. Accrue Software: San Jose, CA.

[Bhavsar and Jivani, 2009] Bhavsar, H. and Jivani, A. (2009). The shared nearest neighbor algorithm with enclosures (snnae). In Computer Science and Information Engineering, 2009 WRI World Congress on, volume 4, pages 436–442. IEEE.

[Biçici and Yuret, 2007] Biçici, E. and Yuret, D. (2007). Locally scaled density based clustering. Adaptive and Natural Computing Algorithms, pages 739–748.

[Birant and Kut, 2007] Birant, D. and Kut, A. (2007). St-dbscan: An algorithm for clustering spatial–temporal data. Data & Knowledge Engineering, 60(1):208–221.

[Bouguessa, 2011] Bouguessa, M. (2011). A practical approach for clustering transaction data. In Machine Learning and Data Mining in Pattern Recognition, pages 265–279. Springer.

[Cambridge, 2011] Cambridge, U. (2011). Cambridge online dictionary.

[Cios et al., 2000] Cios, K., Pedrycz, W., and Swiniarski, R. (2000). Data mining methods for Knowledge discovery. Kluwer Academic Publishers.

[Ertoz et al., 2002] Ertoz, L., Steinbach, M., and Kumar, V. (2002). A new shared nearest neighbor clustering algorithm and its applications. In Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining, pages 105–115.

[Ertoz et al., 2003] Ertoz, L., Steinbach, M., and Kumar, V. (2003). Finding clusters of different size, shapes and densities in noisy, high-dimensional data, army high performance center. Technical report, technical report, April.

[Ester et al., 1996] Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining, volume 1996, pages 226–231. AAAI Press.

[Faustino, 2012] Faustino, B. F. (2012). Implementation for spatial data of the shared nearest neighbour with metric data structures. Master's thesis, Universidade Nova de Lisboa.

[Fayyad et al., 1996] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1996). Advances in knowledge discovery and data mining.

[Foss and Zaïane, 2002] Foss, A. and Zaïane, O. (2002). A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, pages 179–186. IEEE.

[Guha et al., 1998] Guha, S., Rastogi, R., and Shim, K. (1998). Cure: an efficient clustering algorithm for large databases. In ACM SIGMOD Record, volume 27, pages 73–84. ACM.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. ACM SIGKDD Explorations Newsletter, 11(1):10–18.

[Hamming, 1987] Hamming, R. (1987). Numerical methods for scientists and engineers. Dover Publications.

[Han and Kamber, 2001] Han, J. and Kamber, M. (2001). Data mining: concepts and techniques. San Francisco, CA, itd: Morgan Kaufmann, 5.

[Hand et al., 2001] Hand, D., Mannila, H., and Smyth, P. (2001). Principles of data mining. MIT press.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. ACM computing surveys (CSUR), 31(3):264–323.

[Jarvis and Patrick, 1973] Jarvis, R. and Patrick, E. (1973). Clustering using a similarity measure based on shared near neighbors. Computers, IEEE Transactions on, 100(11):1025–1034.

[Karypis et al., 1999] Karypis, G., Han, E., and Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. Computer, 32(8):68–75.

[Kisilevich et al.,] Kisilevich, S., Mansmann, F., Nanni, M., and Rinzivillo, S. Spatio-temporal clustering: a survey. Technical report, Technical Report, ISTI-CNR, Italy, Submitted to Data Minig and Knowledge Discovery Handbook.

[Kotsiantis and Pintelas, 2004] Kotsiantis, S. and Pintelas, P. (2004). Recent advances in clustering: A brief survey. WSEAS Transactions on Information Science and Applications, 1(1):73–81.

[Lelis and Sander, 2009] Lelis, L. and Sander, J. (2009). Semi-supervised density-based clustering. In Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, pages 842–847. IEEE.

[Liu et al., 2007] Liu, P., Zhou, D., and Wu, N. (2007). Vdbscan: varied density based spatial clustering of applications with noise. In Service Systems and Service Management, 2007 International Conference on, pages 1–4. IEEE.

[Liu et al., 2012] Liu, Q., Deng, M., Bi, J., and Yang, W. (2012). A novel method for discovering spatio-temporal clusters of different sizes, shapes, and densities in the presence of noise.

[Manso et al., 2010] Manso, J., Times, V., Oliveira, G., Alvares, L., and Bogorny, V. (2010). Db-smot: A direction-based spatio-temporal clustering method. In IEEE International Conference on Intelligent Systems, London, England.

[Moreira et al., 2005] Moreira, A., Santos, M., and Carneiro, S. (2005). Density-based clustering algorithms–dbscan and snn.

[Peffers et al., 2007] Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of Management Information Systems, 24(3):45–77.

[Piateski and Frawley, 1991] Piateski, G. and Frawley, W. (1991). Knowledge discovery in databases. MIT press.

[Rinzivillo et al., 2008] Rinzivillo, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., and Andrienko, G. (2008). Visually driven analysis of movement data by progressive clustering. Information Visualization, 7(3-4):225–239.

[Santos et al., 2012] Santos, M., Silva, J., Moura-Pires, J., and Wachowicz, M. (2012). Automated traffic route identification through the shared nearest neighbour algorithm. Bridging the Geographic Information Sciences, pages 231–248.

[Schoier and Borruso, 2012] Schoier, G. and Borruso, G. (2012). Spatial data mining for highlighting hotspots in personal navigation routes. International Journal of Data Warehousing and Mining (IJDWM), 8(3):45–61.

[Silva et al., 2012] Silva, R., Moura-Pires, J., and Santos, M. Y. (2012). Spatial clustering in solap systems to enhance map visualization. International Journal of Data Warehousing and Mining (IJDWM), 8(2):23–43.

[Tripathy et al., 2011] Tripathy, A., Maji, S., and Patra, P. (2011). Fdca: A fast density based clustering algorithm for spatial database system. In Computer and Communication Technology (ICCCT), 2011 2nd International Conference on, pages 21–26. IEEE.

[Vincenty, 1975] Vincenty, T. (1975). Geodetic inverse solution between antipodal points.

[Wei and Liu, 2009] Wei, G. and Liu, H. (2009). Ld-bsca: A local-density based spatial clustering algorithm. In Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on, pages 291–298. IEEE.

[Witten and Frank, 2005] Witten, I. and Frank, E. (2005). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

[Yang et al., 2011] Yang, Y., Gong, Z., et al. (2011). Identifying points of interest by self-tuning clustering. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pages 883–892. ACM.

[Ye et al., 2003] Ye, N. et al. (2003). The handbook of data mining. Lawrence Erlbaum Associates, Publishers.

[Zhang et al., 1997] Zhang, T., Ramakrishnan, R., and Livny, M. (1997). Birch: A new data clustering algorithm and its applications. Data Mining and Knowledge Discovery, 1(2):141–182.

[Zhou et al., 2004] Zhou, D., Cheng, Z., Wang, C., Zhou, H., Wang, W., and Shi, B. (2004). Sudephic: Self-tuning density-based partitioning and hierarchical clustering. In Database Systems for Advanced Applications, pages 69–108. Springer.