SNN Input Parameters: how are they related?

Guilherme Moreira, Maribel Yasmina Santos
Algoritmi Research Centre
University of Minho
Guimarães, Portugal
pg19786@alunos.uminho.pt, maribel@dsi.uminho.pt

João Moura-Pires
Faculty of Science and Technology
New University of Lisbon
Lisbon, Portugal
jmp@fct.unl.pt

***Abstract*—Nowadays, organizations are facing several challenges when they try to analyze generated 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 prior 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 Neighbor) algorithm and proposes specific guidelines for the identification of the appropriate input parameters that optimizes the processing time.**

Keywords-clustering; density-based clustering; shared nearest neighbor; input parameters tuning

I. INTRODUCTION

Current technological developments allow the collection of huge amounts of data that are usually stored and analyzed to support the decision-making process. 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 [1], in which clustering algorithms are common techniques used to analyze data, not requiring any prior knowledge about the data set [2]. Being unsupervised data mining techniques, clustering has the advantage of identifying clusters that emerge naturally from data. However, 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 [3]. 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 – k, *MinPts* and *Eps* – of the SNN (Shared Nearest Neighbor) algorithm and proposes specific guidelines to identify these parameters, which are used to calibrate the clustering's results, attending to the data available for analysis. k represents the size of nearest neighbors 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 [4].

The problem of input parameters identification was addressed in previous works [5-8]. Birant [8] suggested that k can be approximated using the natural logarithm of the number of objects to be clustered ($ln(n)$). Although providing a good approximation for small data sets, the value of k obtained for large data set is not appropriate, as this paper will show. Ertoz [4] indicates that knowing the value of k, the *MinPts* input parameter can be calculated as being a fraction of k. For *Eps*, some heuristics have been proposed mainly for the DBSCAN algorithm [7], but they cannot be applied to SNN as the *Eps* parameter in SNN is used with a different semantic.

This paper is organized as follows. Section II summarizes related work on density-based clustering approaches, and the SNN algorithm, and their attempt to identify the input parameters. Section III analyses synthetic data sets, establishing relationships between the input parameters. Section IV concludes with a summary of the main findings and proposals of future work.

II. RELATED WORK

Clustering is the task of identifying sets of segments or 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 [9].

Density-based clustering approaches were developed based on the notion of density [9]. 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 [10]. DBSCAN [7] and OPTICS [11] are major representatives of this class of clustering algorithms, being DBSCAN the most representative density-based clustering algorithm. Many of the available density-based algorithms were derived from DBSCAN, which was introduced by Ester [7] and was specially designed to treat spatial databases.

DBSCAN needs two input parameters: *Eps* and *MinPts*. *Eps* is the radius distance of a point, in other words the neighborhood of a point. *MinPts* is the minimum number of points that the neighborhood 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 do not belong to any cluster are considered noise points [7].

The SNN algorithm is an extension of the Jarvis-Patrick [12] and the DBSCAN [7] algorithms. The main difference and contribution from [4] is the similarity measure implemented looking at the number of nearest neighbors that two points share.

In this research, we chose to use the SNN algorithm, because some studies have proven that SNN performs better since it can detect clusters with different densities while DBSCAN cannot [13]. While similar to DBSCAN, the SNN approach is slightly different. The number of neighbors that two points share defines the similarity of points. As mentioned, the user must specify three input parameters. The algorithm needs a distance function to define the list of *k*-nearest neighbors. Generally the Euclidean distance function is used when points are inside a two dimensional space.

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. One example, in which authors make several tests until they achieve meaningful cluster results, in found in [14].

One of the possible approaches, to solve the need of several input parameters, regardless of the algorithm, is to minimize the number of input parameters by finding relations between parameters and identifying a function that give us one parameter given other input parameter. Ertoz [4] give some insights about this parameterization in the SNN algorithm, saying that *MinPts* should be a fraction of the neighborhood list size *k,* which determines the granularity of the clusters. If *k* is too small, even a uniform cluster will be broken up into pieces due to local variations in the similarity, and the algorithm will tend to find many small, but tight, clusters. If *k* 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. Once *k* is fixed, the nature of the clusters is also fixed [4].

As we don't know beforehand the appropriate input parameters, Ester [7] proposed a heuristic to determine the parameters *Eps* and *MinPts* of the thinnest cluster in the database. The first step of the heuristic is to determine the distances to the *k*-nearest neighbors for each object, where *k* is equal to *MinPts*. The density distribution of the data set when plotted using the *k*-distance values sorted in a descending order, gives some insights. The first "valley" in the density distribution highlights a threshold. All points with higher *k-dist* value, placed on the left of the threshold are considered noise points and all other points, on the right of the threshold, are assigned to some cluster. This heuristic is also followed by Birant [8] and Silva et al. [15].

Silva et al. [15] modified Ester's heuristic, changing the way the first "valley" is identified and giving the possibility of choosing other breaks (valleys) in the graph. This means that the clustering process can produced more or less clusters depending on the break position.

In [8], it is also concluded that should be $MinPts \approx ln(n)$, where *n* is the size of the database, and *Eps* less than the distance defined by the first valley. In [13], the values used for *Eps* and *MinPts* were dependent of the *k* value. The authors suggested, using the results of clustering small data sets between 300 and 523 points, that $Eps=0.3*k$ and $MinPts=0.7*k$. In another study by Liu [16], STSNN (Spatial Temporal Shared Nearest Neighbor), a similar approach was taken. The purpose of their work was to find clusters in spatio-temporal data sets. The algorithm needs three input parameters, k, k_t (for time) and *MinPts*, but the authors verified that both k_T and *MinPts* are highly correlated to *k* and so can be defined as a percentage of *k*. After several tests, the authors suggested that both k_T and *MinPts* can be set to a value around $0.5*k$.

After the analysis of several works, 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.

For a short demonstration of the influence of the input parameters in the clustering results, a small sample data set with 322 objects is used. With a valid combination of the input parameters, k=7, Eps=2, $MinPts$=5, the results shown in Figure 1 are obtained. The clustering algorithm was able to identify 7 clusters of variable density and shape, also identifying noise points. Each cluster is plotted with a different color, while noise points are presented in black.

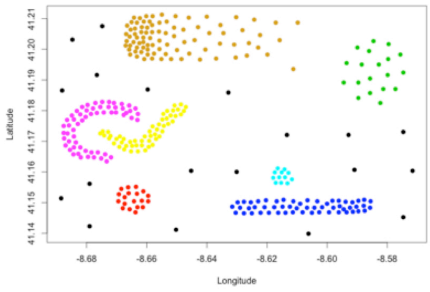

Figure 1. Sample-322 expected clusters

An example of an inappropriate clustering result due to the inadequacy of the chosen input parameters is shown in Figure 2. 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 (13 clusters). This 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.

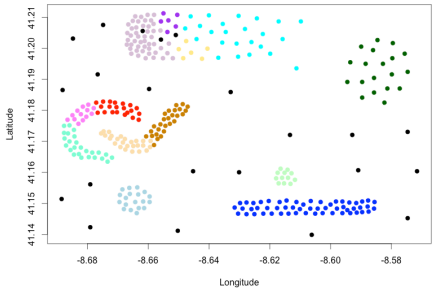

Figure 2. Inappropriate clustering result for the sample-322 data set

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

III. Analysis of the SNN Input Parameters

In this paper, the behavior of the input parameters is tested using four artificial Chameleon data sets [5]. Figure 3 presents the t5.8k and Figure 4 shows the t4.8k, both with 8.000 points. It were also used the data sets t8.8k with 8.000 points and t7.10k with 10.000 points. For the analysis of the input parameters of the t5.8k and t4.8k data sets, several initial experiments using heuristics proposed by other authors were tested without any satisfactory results.

To understand the behavior of the SNN input parameters, the used strategy 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.

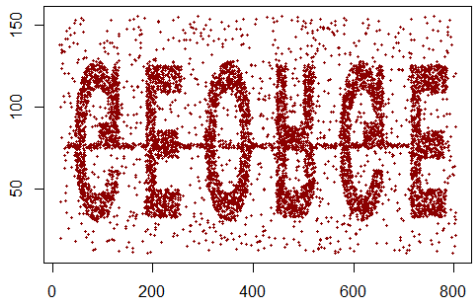

Figure 3. Spatial distribution of t5.8k

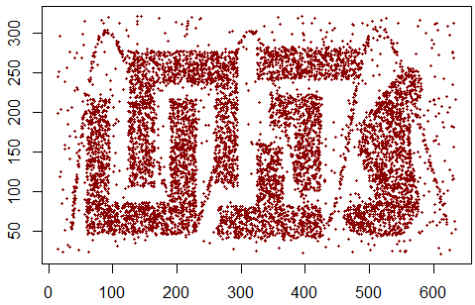

Figure 4. Spatial distribution of t4.8k

A. Methodology

The implementation used to perform clustering with the SNN algorithm was the F-SNN [17]. 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. The methodology used contains the following steps: i) set the maximum number for k (*k-max*); ii) run the clustering algorithm using all possible combinations of *MinPts* and *Eps* with a k between 2 and *k-max*; iii) identify the combinations that produced the number of expected clusters; iv) rerun the clustering algorithm, for the combinations identified in step iii), counting the number of points per cluster; and, v) assign a quality label to the identified clusters.

Initially, and following the SNN steps, the nearest neighbors' 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)*. The neighbors' list for a k equal to 80 (*k-max*) was created and used in all the executions.

Using the t5.8k and t4.8k data sets with 6 clusters each, allowed the identification of the input parameters that produce the expected results. 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. Note that for a range of k between 2 and 80, 170.640 executions of the SNN were performed. All the combinations that produced the 6 expected clusters were identified and the number of points that each cluster has is recorded, as this value allows an evaluation of the quality of the obtained results.

The approach used to measure the quality of the clusters was to analyze 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.

Next subsections describe the obtained results for the two data sets, while the last subsection summarizes the main findings of these analyses.

B. Results for the t5.8k data set

After processing all possible combinations for the values of k, *Eps* and *MinPts*, 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 analyzed. For this data set, a large number of valid combinations were obtained (1.527 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. This assignment of a quality label allowed the analysis of the several combinations among input parameters and the clusters' quality. The pattern that emerged clearly reveals the influence of the input parameters on the clustering results quality. Figure 5 only consider those combinations that generate results classified as Excellent. As shown in Figure 5, all the Excellent results were obtained with values of k and *MinPts* that are strongly related (R^2=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*.

On Figure 6 is shown the correlation between k and *MinPts*. These are the results for the 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* behavior keeps as previous observation.

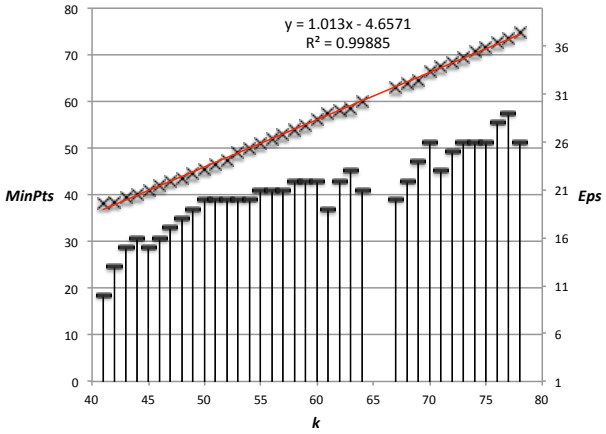

Figure 5. Input parameter combinations that produce Excellent results (t5.8k data set)

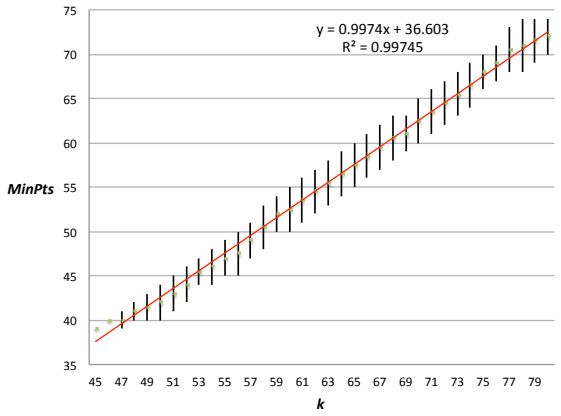

Figure 6. Correlation between *k* and *MinPts* of Very Good clustering results (t5.8k data set)

C. *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. However, a high correlation between *k* and *MinPts* is still verified and, also, the high variability of *Eps*.

For the two data sets, Table 1 presents a summary with the intervals that produced the six clusters.

Table 1. Range of input parameters that produce 6 clusters

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 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.

The best clustering results are achieved for t4.8k with a combination of parameters *k-Eps-MinPts* of 67-22-64 (Figure 7) and of 46-15-43 for the t5.8k data set (Figure 8). 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 7. Clustering result for the t4.8k data set

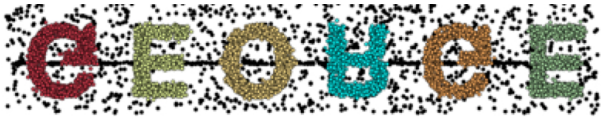

Figure 8. Clustering result for the t5.8k data set

D. *Main findings*

Figure 9 shows a summary of the results obtained in the test of several data sets, t4.8k, t5.8k, t7.10k and t8.8k, where a coherence in terms of the values *minK*, *maxK*, *avgK*, *bestK* and the value of *n* (size of the data set) can be observed.

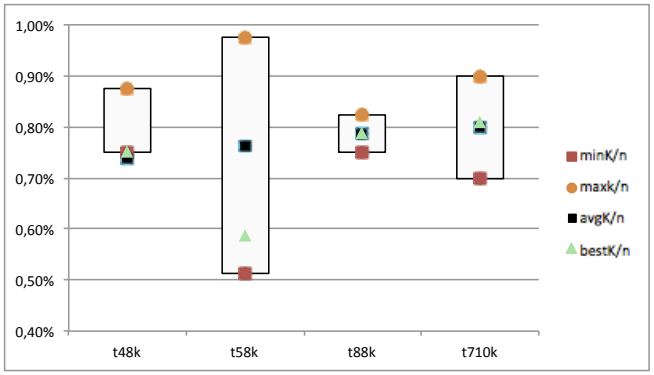

Figure 9. Results for t4.8k, t5.8k, t7.10k and t8.8k data sets

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*. 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.

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, the four data sets available from the Chameleon algorithm were tested. Table 2 shows the estimated range for *k* using the proposed heuristics, as well as the value of *k*, classified as excellent, obtained after processing and analyzing all the results.

Table 2. 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

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.

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 [4], mainly looking for the *k* nearest neighbors of a point, this means that as *n* increases, the estimation of *k* will increase too, making the calculation of the list of nearest neighbors a more difficult task.

The t5.8k data set was replicated, as shown in Figure 10. 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 11). Moreover, independently of the undertaken replication, with for example 32.000 points, the new estimated values for *k* identifies the expected clusters as well as the previous combinations. This means that a pattern exist in the possible input parameters for a data set. This is very important as we can avoid large values for *k*, which severely penalize the time needed to compute the clusters.

Figure 10. Duplication of the t5.8k data set

Figure 11. Clustering result for the duplicated t5.8k data set

To test this behavior of the input parameters, two other data sets were used (http://cs.joensuu.fi/sipu/datasets/). Birch1 and Birch2 have 100.000 objects each one (Figure 12 and Figure 13). In both data sets, the clustering process must identify the 100 well-defined clusters.

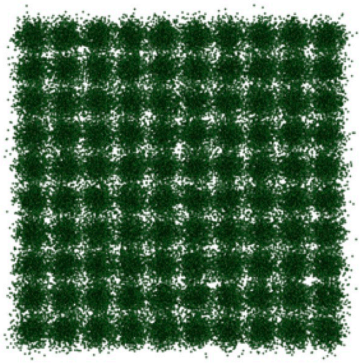

Figure 12. Spatial distribution of Birch1 data set

Figure 13. Spatial distribution of Birch2 data set

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

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. Considering a *k* value of 175, a *MinPts* value of 162 (since the heuristic rule choose a value between 92% and 96% of *k* value) and an *Eps* value of 30 (since the heuristic rule choose a value around 18.5% of *MinPts* value), the 100 clusters were perfectly identified in both data sets (Figure 14 and Figure 15), providing excellent results.

Figure 14. Clustering results for the Birch1 data set

Figure 15. Clustering results for the Birch2 data set

IV. Conclusions and Future Work

This paper analyzed the three input parameters of the SNN algorithm in order to find guidelines that can be used in an analytical process to present the most appropriate results to the user. This research 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.

A significant number of SNN executions were needed. From a total number of 1.253.210 executions, 102.034 were needed to count the number of points in each cluster, as this was the measure used to evaluate the quality of the clustering results.

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 paper. This tool will be public available and will allow users to explore their data sets obtaining results that try to optimize 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.

Acknowledgment

This work was partly funded by FEDER funds through the Operational Competitiveness Program (COMPETE), by FCT with the project: FCOMP-01-0124-FEDER-022674 and by *Novabase Business Solutions* with a co-funded QREN project (24822).

References

[1] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, "Advances in knowledge discovery and data mining," 1996.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR),* vol. 31, no. 3, pp. 264-323, 1999.

[3] M. Bouguessa, "A practical approach for clustering transaction data," *Machine Learning and Data Mining in Pattern Recognition*, pp. 265--279: Springer, 2011.

[4] L. Ertoz, M. Steinbach, and V. Kumar, *Finding Clusters of Different Size, Shapes and Densities in Noisy, High-Dimensional Data, Army High Performance Center*, technical report, April, 2003.

[5] G. Karypis, E. H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer,* vol. 32, no. 8, pp. 68-75, 1999.

[6] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," in ACM SIGMOD Record, 1998, pp. 73--84.

[7] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "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, 1996, pp. 226--231.

[8] D. Birant, and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial--temporal data," *Data \& Knowledge Engineering,* vol. 60, no. 1, pp. 208--221, 2007.

[9] J. Han, and M. Kamber, "Data mining: concepts and techniques," *San Francisco, CA, itd: Morgan Kaufmann,* vol. 5, 2001.

[10] N. Ye, and others, *The handbook of data mining*: Lawrence Erlbaum Associates, Publishers, 2003.

[11] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," *ACM SIGMOD Record,* vol. 28, no. 2, pp. 49-60, 1999.

[12] R. A. Jarvis, and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *Computers, IEEE Transactions on,* vol. 100, no. 11, pp. 1025-1034, 1973.

[13] A. Moreira, M. Y. Santos, and S. Carneiro, "Density-based clustering algorithms--DBSCAN and SNN," 2005.

[14] G. Schoier, and G. Borruso, "Spatial Data Mining for Highlighting Hotspots in Personal Navigation Routes," *International Journal of Data Warehousing and Mining (IJDWM),* vol. 8, no. 3, pp. 45-61, 2012.

[15] R. Silva, J. Moura-Pires, and M. Y. Santos, "Spatial Clustering in SOLAP Systems to Enhance Map Visualization," *International Journal of Data Warehousing and Mining (IJDWM),* vol. 8, no. 2, pp. 23-43, 2012.

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

[17] A. A. F. Antunes, "F-SNN. A Fast SNN-based clustering approach for large geospatial data sets.," 2013-08-13, http://ubicomp.algoritmi.uminho.pt/projects/f-snn/