AGILE: Augmented GrId based cLustEring

Amin Javidan
Department of Computer Science & IT
Shiraz University
Shiraz, Iran
a.javidan@cse.shirazu.ac.ir

Reza Boostani
Department of Computer Science & IT
Shiraz University
Shiraz, Iran
boostani@shirazu.ac.ir

ABSTRACT
Clustering of large databases is still a challenge, especially with high dimensional inputs. Conventional clustering methods suffer from high sensitivity to noisy samples and bad initial conditions. To overcome the mentioned drawbacks, a new kernel-based method, termed as Augmented GrId based cLustEring (AGILE) is proposed. AGILE is designed to cluster large databases, regarding the number of attributes. It is an in-memory clustering algorithm which starts to partition the input space into small grids and tries to form arbitrary shaped clusters with eliminating outlier samples. Experimental results demonstrate the higher performance of AGILE in terms of computational complexity and accuracy compared to the other methods.

CCS Concepts
• **Information systems → Clustering**

Keywords
high dimensional clustering; data analyzing; density based clustering; grid-based clustering;

1. INTRODUCTION
Clustering is a data mining technique which tries to distinguish input vectors into different groups (clusters) based on their similarity. The computational complexity of conventional clustering methods is still a big challenge for online grouping, since current databases contain a huge number of samples, sometimes with a high number of attributes. A wide variety of clustering algorithms is available for data analyzing which broadly divides into three categories: a. Partition based b. Hierarchical c. Density-based methods.

Partition based methods try to determine *k* clusters by partitioning search space into *k* parts, based on optimizing an objective function iteratively [7, 12]. *k* is the input parameter which should be determined by the expert. Usually, the exact value of *k* is unknown, especially in large databases. Most of the partitional clustering methods are unable to find arbitrary shaped clusters since they result in convex shapes [4].

Hierarchical methods use top-down or bottom-up approaches for retrieving clusters in a nested mode [6, 11, 17], the challenge in these methods is finding a proper termination criterion. Hierarchical methods can retrieve arbitrary shaped clusters; consequently, they act more accurate than the partitional approaches. Although, they fail with elongated clusters.

ICICM'17, August 28-30, 2017, Moscow, Russian Federation.

ACM ISBN 978-1-4503-5279-6/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3134383.3134412

Density-based methods retrieve clusters by using the distribution of data, so the nature of clusters will be preserved [4, 9]. This paper proposes a new method called Augmented GrId based cLusetEring (AGILE), which uses kernel density estimation (KDE) [3] with an adaptive parameter for efficient clustering. The AGILE run time and accuracy are considerable because of information compression in each cell and applying a novel method to distinguish valid clusters.

The rest of this paper is structured as follows. In section 2 related works are examined. Section 3 states a method for determining the appropriate length of grid cells. Section 4 proposes the dynamic structure of grid cells. The main approach will be discussed in section 5 and section 6 provides the result of the AGILE method. Finally, the paper is concluded in section 7.

2. RELATED WORKS
In this section, state of the art clustering algorithms are discussed. These methods are capable of efficient clustering by reducing time complexity or enhancing the final results.

Clustering Large Applications based on RANdomized Search (CLARANS) [12] is a partition based clustering approach which is based on randomized sampling. CLARANS finds clusters by creating a graph which each node of it has a feasible solution of the clustering problem. It is a faster version of *k*-means but still cannot find arbitrary shaped clusters.

One of the most powerful density based approaches is DBSCAN [4]. This is a fast and noise robust method which requires two parameters: a. MinPts (minimum points) b. Eps (epsilon) for clustering in noisy databases by defining reachability and connectivity concepts. The main flaw of DBSCAN is that performance of this algorithm is significantly decreased in high dimensional space by expensive computation of nearest neighbors.

OPTICS introduces a cluster analysis method which generalizes DBSCAN approach [1]. It is capable of retrieving density varying clusters by assuming a wide range of parameter settings. This assumption makes OPTICS an offline cluster analysis approach.

The BIRCH algorithm is a hierarchical method which uses CF tree for bottom-up clustering [17]. This algorithm clusters high dimensional data dynamically by considering memory limits. BIRCH performs well on large databases by just one data scanning, but it cannot retrieve non-spherical clusters.

CHAMELEON is another strong clustering algorithm [11], it is an agglomerative hierarchical approach that retrieves clusters by dynamic modeling. The result of CHAMELEON is significantly accurate, although it is not efficient in processing time.

Another efficient clustering method that uses density function is DENsity based CLUstEring (DENCLUE) [9]. This method speeds up the clustering process by applying grid based partitioning

along with defining a kernel-based method for detecting clusters. In spite of the significant improvement in DENLCUE compared to DBSCAN in terms of computational complexity, DENCLUE applies hill climbing to inputs for detecting arbitrary shaped clusters. Due to its nonconvex solution space, this approach does not have an exact solution and therefore it should be solved in an iterative manner, which is a time-consuming process.

In addition to the mentioned problems these approaches mostly fail in handling large datasets. There are such alternative density based methods which improve the accuracy of DENCLUE by replacing hill climbing with simulated annealing or genetic algorithm [10] or accelerate its procedure by eliminating the hill climbing phase [13].

This paper improves the performance of clustering in noisy large databases in terms of accuracy and computational complexity.

3. DETERMINING GRID LENGTH

One of the fundamental approaches for detecting sparsity of clusters is *k* nearest neighbors (*k*NN) algorithm [4]. Sparsity calculation in noisy databases should be coupled with noise detection, for avoiding invalid calculations.

Definition 1. Assume *D* is a distribution in *n*-dimensional space, for each instance of *D*, *k*NN creates a S_k which is a k layered hyper-spherical structure.

$$\text{Applying } k\text{NN} \Rightarrow \forall\, x \in D: \quad \exists\ S_k \quad D, S_k \in \mathbb{R}^n$$

$$S_k^i : i = 1,2,\dots,N \quad |D| = N$$

Definition 2. Assume X^i is a *k*-sized group of nearest neighbors of point *i*. The center of the group is point *i* itself which is created because of *k*NN approach.

$$V_{X^i} = \max_j d_i^j$$

d_i^j : Difference between *j*th and (*j-1*)th layers in S_k^i

V_{X^i} : Maximum radius of surrounded vacancy around point *i*

Fig. 1 depicts the S_k structure of point p. It is the result of applying 3NN, which creates a 3 layered circle. The notations are defined by definition 2.

Fig. 2 illustrates the expected value of each vacancy surrounding its corresponding point. The curve is sketched by applying 1NN to the limit number of samples down from a uniform distribution which sorted increasingly by $E[V_{X^i}]$. Thus each $E[V_{X^i}]$ illustrates the expected radius of space surrounding just one sample *i*. The sharp slope (Knee phenomenon) at the end of the curve illustrates points which surrounded by very big vacancies.

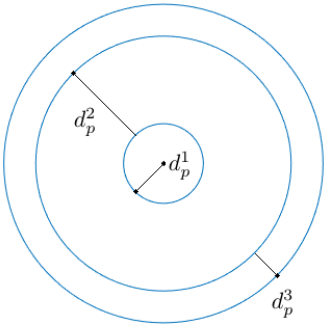

Figure 1. Illustration of the S_k^p structure.

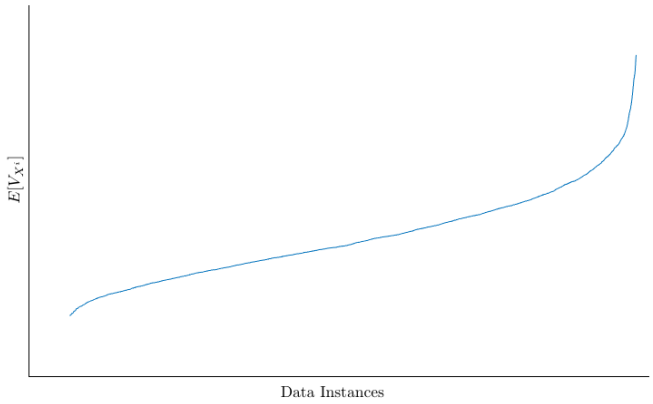

Figure 2. Expected values of vacancy on any uniform distribution.

Definition 3. Noisy points have more surrounded vacancies than the desired ones.

$$V_{C'} \gg V_C \quad |C| = |C'| = k+1 \quad \alpha \le k < \beta$$

C : Group with valid center $\quad C'$: Group with noisy center

β : Number of instances of the smallest desired cluster

α : Number of instances of the biggest bunch of noisy samples

The hyper-spherical structure of a noisy sample should contain null space between desired clusters and noisy ones to illustrate its real vacancy; thus, k should be equal or more than α. Also, preserving valid clusters needs that all neighbors be in the same cluster, hence k should be lower than β.

3.1 *k* Parameter of *k*NN

By increasing *k*, the chance of changing each V_{X^i} to a bigger value becomes more, hence an offset will be added to the *k*NN curve, but the original shape will remain. Eliminating uniform distribution assumption from Fig. 2 just causes the sharp slope of the *k*NN curve be sharper since noisy samples have larger vacancies.

Definition 3 states a condition for a valid value of *k*, but the point is that *k*NN is a spherical approach and may not follow the shape of clusters. Therefore, by increasing *k*, the probability of getting invalid results will be increased. By increasing k to ∞, (e.g. k = size of the instances in a dataset), the sharp slope of the *k*NN curve illustrates edge points in the database, these edge points can be either noise or marginal points of a cluster due to losing the locality. Thus *k*NN results are not reliable with big *k* values.

3.2 Obtaining Appropriate Length

The sharp slope of the *k*NN curve illustrates points which satisfy definition 3 conditions. Consequently, the appropriate length of cells is the starting point of the knee in the *k*NN curve.

$$\Lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix} \tag{1}$$

λ_i : Starting point of the knee in *i*th dimension

4. GRID CELL STRUCTURE

Capturing the nature of data takes $O(n^2)$ time by calculating the similarity matrix. One of the most fundamental ways for overcoming this problem is space partitioning which bounds behavior of data into cells. Partitioning space into grid cells cannot

map data distribution appropriately, because of its sensitivity to the grid length, the start and end points of cells.

Grid cells in AGILE have a dynamic structure for overcoming the mentioned problems. The dynamism of each cell gained by applying kernel to each point in the database and connectivity condition for neighbor grid cells which will be discussed in the next two subsections.

4.1 Kernel Density Estimation (KDE)

AGILE uses KDE for density estimation. Hence a kernel should be applied to each point of the distribution. Since Gaussian kernel considers the distance between each pair of points [9], this kernel is employed in the proposed method.

$$\widehat{K}_{Gauss}(x) = \sum_{x' \in N(x)} \exp(\frac{-d(x,x')^2}{2\sigma_x^2}) \quad (2)$$

$N(x)$: Nearest neighbors set of x

$d(x,x')$: Distance between point x and point x'

When using this kernel, farther x'has lower effect in shaping density. Hence in (2) for each point, just its neighbors are considered to be applied to the Gaussian kernel for reducing the running time. One way of determining nearest neighbors quickly is that we assume all points of the grid cell that contains the point x and its adjacent grid cells as nearest neighbors, e.g. bucketing [15].

$$N(x) = \{x' \mid x' \in A(G_\iota) \cup x' \in G_\iota\} \quad (3)$$

G_ι : ιth cell containing point x

$A(G_\iota)$: Adjacent grid cells of the G_ι

According to (3) $|A(G_\iota)|$ denotes the number of adjacent cells of the G_ι. For any non-edge cell G_ι, $|A(G_\iota)| = 3^n - 1$, which n is the dimension of the input space.

$$\sigma_x = \frac{\lambda^*}{Card(G_\iota)} \quad (4)$$

$$Card(G_\iota) : |\{x' \mid x' \in G_\iota\}| \quad \lambda^* = \min_i\{\lambda_i \mid \lambda_i \in \Lambda\}$$

σ_x of the Gaussian kernel in (2) is an instance dependent parameter which is set by (4). This parameter causes better density estimation since, for dense areas, it will be decreased to strengthen the effect of its neighbors.

The upper bound of the density estimation error in the proposed method is determined as follows:

$$\widehat{K}_{Gauss}(x) = \sum_{x' \in N(x)} \exp(\frac{-d(x,x')^2}{2\sigma_x^2})$$

$\widehat{K}_{Gauss}(x)$: Estimated density of point x

$$K_{Gauss}(x) = \sum_{all\ x'} \exp(\frac{-d(x,x')^2}{2\sigma_x^2})$$

$K_{Gauss}(x)$: True density of point x

$$\text{Error} = K_{Gauss}(x) - \widehat{K}_{Gauss}(x)$$

$$\text{Error} = \sum_{x' \notin N(x)} \exp(\frac{-d(x,x')^2}{2\sigma_x^2}) \leq \sum_{x' \notin N(x)} \exp(\frac{-\lambda^{*2}}{2\sigma_x^2})$$

$$\text{Error} \leq \sum_{x' \notin N(x)} \exp(\frac{-1}{2}\, Card(G_\iota)^2)$$

In spite of the accurate estimation in dense areas, the estimation error in sparse areas becomes high. Hence, by taking them into account, a considerable gap between sparse and dense areas is created.

4.2 Neighboring Connection Condition

Each grid cell G_ι has two attributes γ and p^* beside its data points which result in speeding up the AGILE run time.

$$\gamma = \max_{p \in G_\iota} K_{Gauss}(p) \quad (5)$$

$$p^* = \operatorname*{argmax}_{p \in G_\iota} K_{Gauss}(p) \quad (6)$$

Based on (5) and (6) γ is the maximum height of each cell and p^* is the point creating the corresponding γ. These parameters provide adequate information for their corresponding grid cell by encoding the affinity of data in each cell.

Theorem 1. Assume distribution of data and the kernel are both Gaussian, for The Euclidean distance we have:

$$\forall\, G_i \in \mathbb{R}^n : \max\{Dist(p_i^*, p_{i+1}^*)\} = \sqrt{\sum_{j=1}^{n} \lambda_j^2}$$

Proof. Applying Gaussian kernel to a Gaussian distribution results in creating a smooth Gaussian surface which is an approximation of the real model. In each region of the space, changes of this surface are monotonically either increasing or decreasing. Thus in any adjacent regions, the maximum difference between two consecutive optima is equal to the diagonal of the grid.

Theorem 1 is valid just for Gaussian distribution, while in real applications, distribution of data does not follow any well-known model. After applying Gaussian kernel, a bumpy surface should be created which violates the rule. We add an extra parameter τ to tune the upper bound of adjacency condition.

$$\max\{Dist(p_i^*, p_{i+1}^*)\} = \tau \sqrt{\sum_{j=1}^{n} \lambda_j^2} \quad \tau \in [1,2] \quad (7)$$

τ in (7) is a separating operator. By decreasing τ, the number of final clusters should increase because of the tight condition for merging cells and vice versa.

5. THE AGILE ALGORITHM

Assume k^* is the ideal choice of kNN parameter in a dataset. Employing k^*NN, results in almost eliminating all noisy clusters. In real applications, determination of k^* is difficult due to its heavy load in large datasets. As mentioned before, kNN is a spherical method, hence k^* cannot accurately be calculated. Here $k \ll k^*$is considered, regardless of achieving a good speed up rate, this causes preserving large bunches of noisy samples.

$$k \ll k^* \Rightarrow \exists\, C' :\ k < |C'| \leq k^*$$

C' : Noisy samples (noisy bunch)

The main reason of preserving noisy samples is violating the condition of definition 3. AGILE is capable of eliminating survived bunch of noisy samples. Hence there is no restriction on the value of the k parameter. After determination of Λ by kNN, the whole space of data distribution divides into Λ-sized hyper-rectangles and each point should be assigned to its cell. For each cell, (5) is applied for obtaining γ (the maximum height of density in each cell) separately. In addition to the γ, p^* (the pointer to the maximum height of density in each cell) should be stored.

A connected component labeling method should be applied to the partitioned database for retrieving clusters from cells. Then (8) will be applied to each retrieved cluster for calculating its Γ parameter.

$$\Gamma_C = \max_{p \in C} K_{Gauss}(p) = \max_{G_i \in C} \gamma_i \quad (8)$$

In fact, Γ_C is the maximum height of its corresponding retrieved cluster *C*.

Definition 4. Assume C' is a bunch of noisy samples and C is a valid cluster.

$$V_{C'} \gg V_C \Rightarrow S_{C'} \ll S_C \quad or \quad D_{C'} \ll D_C$$

S_c : Spread of cluster *C*

D_C : Density of cluster *C*

Employing *k*NN causes that a noisy cluster (a bunch of noisy samples) illustrates its big vacancy faster than a desired one. Therefore, it shows that the desired clusters are denser than the noisy ones.

This paper suggests a measurement for distinguishing noisy clusters from desired ones by the below equation.

$$P_C = \max_{p \in C} K_{Gauss}(p)\ n(C) = \Gamma\, n(C) \quad (9)$$

P_C : Power of cluster *C*

$n(C)$: Number of grid cells in cluster *C*

Noisy samples or noisy clusters can be artifacts of data measurement; thus, by definition, there should exist a considerable gap between a desired and a noisy one. By sorting P_C of each cluster and finding the first gap, AGILE eliminates noisy clusters.

5.1 Fusion Phase (optional)

The sparse clusters in a dataset would be broke down because of the tight condition of cell merging for avoiding coupling the valid clusters. For overcoming this problem, a fusion phase is applied by AGILE.

After finding the first gap, if still there exists a meaningful gap, the fusion phase can be applied. The low-rated clusters will be selected for the fusion procedure since they have the potential to be merged. Each candidate cluster $\bar{C}$ is encoded to just one point in the new space. This mapping transforms a cluster into the pointer (spatial position) to its corresponding Γ.

By applying grid cells with greater size than τ, connected points illustrate merged clusters. An approximation for the new grid length based on the number of candidate clusters and their cells is:

$$\tau' \approx \tau \frac{\sum_{\bar{C} \in X} n(\bar{C})}{|X|} \quad (10)$$

X : Set of candidate clusters for fusion

5.2 Complexity

The below pseudo code shows the main procedure of AGILE.

AGILE (points, Λ, τ)

1. cells ← CreatePartition (points, Λ)
2. optima ← AffinityCalculation (cells)
3. connected_cells, new_optima ← ConnectAdjacentCells (cells, optima, τ)
4. dense_clusters ← FindFirstGap (new_optima, connected_cells)
5. sparse_clusters ← FindSecondGap (new_optima, connected_cells)
6. merged_clusters ← Fusion (sparse_clusters, τ')
7. final_clusters ← dense_clusters + merged_clusters

The computational complexity of AGILE is $O(N)$ for loading dataset and assign each point to its corresponding cell when the size of the dataset is *N*. In step 2, the complexity of applying KDE is $O(MN)$, where *M* is the average number of proximity points of each instance ($M << N$). Connecting adjacent cells in Step 3 needs $O(2nK)$, where *K* is the number of non-empty cells and *n* is the data dimension. Step 4 and 5 have $O(K' \log K')$ because of the worst case of sorting, K' is the number of retrieved connected cells, $K' \ll K$. Step 6 time complexity is $O(2nK'')$, because of connecting adjacent cells (second time with looser grid length), which K'' is the number of sparse clusters ($K'' < K'$).

In conclusion, AGILE has 3 phases; *k*NN, KDE, and fusion. The *k*NN phase is offline, and it is just for calculating the Λ for partitioning input space. The main procedure of AGILE is KDE phase. The fusion phase is designed for tuning clustering in the datasets which have diverse clusters in term of sparsity. The worst order of applying *k*NN is $O(N^2/2 + KN^2)$, $O(N^2/2)$ part for creating adjacency matrix and (KN^2) part for finding k nearest neighbors of each data point. Due to the offline nature of this phase, the time complexity can be ignored [4, 9]. It can be seen that the computational complexity is decreased compared to the other density based methods.

6. RESULT AND DISCUSSION

In this part, the efficiency of the proposed AGILE algorithm along with *k*-means, DBSCAN, OPTICS, and DENCLUE was assessed on three different types of datasets, in terms of accuracy and computational complexity.

The main advantage of AGILE is its high execution speed (runtime) which makes it suitable for the real-world applications, compared to the state-of-art clustering algorithms.

6.1 Computational Complexity

In Fig. 3, to evaluate AGILE against the state-of-art methods, some uniform datasets are produced with a different number of samples. The selected number of points is the power of 2 including 4k, 8k, ..., 1024k.

The worst results belong to *k*-means which has a big difference to the others. Therefore, we limit the cluster size up to 256k to visually compare them. As we can see in Fig. 3, AGILE results in the lowest run time among them same as DENCLUE-IM. Also, the speed of AGILE is much higher (more than two times) than that of DNECLUE 2.0.

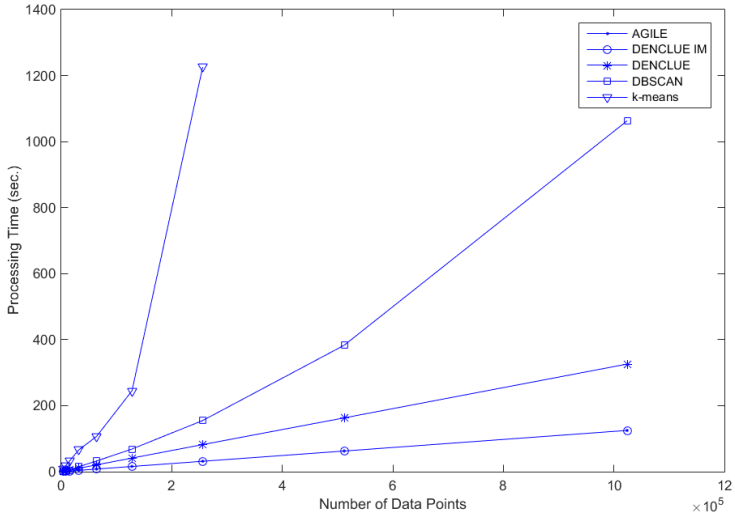

Figure 3. Comparison between clustering methods in term of running time.

Table 1. Comparison between clustering methods in term of accuracy (5 times execution).

	k-means	DBSCAN	OPTICS	DENCLUE 2.0	DENCLUE-IM	AGILE
Iris	76.26±16.6	67.33±0.00	75.54±0	48.00±0	47.33±0	**80.67±0**
Glass	43.00±1.76	45.79±0.00	44.39±0	42.06±0	40.65±0	**49.07±0**
BUPA	54.43±0.51	42.60±0.00	57.85±0	57.97±0	57.97±0	**58.55±0**
Vehicle	**36.97±0.79**	25.33±0.26	33.21±0	26.12±0	26.00±0	35.34±0

6.2 Accuracy

To evaluate the accuracy of AGILE, a few datasets from UCI repository database [14] is selected. The clustering accuracy [16] is considered as the evaluation metric. Approaches are well-tuned for the datasets, hence the result is near to the best case scenario.

As we see in Table 1, AGILE has a higher accuracy in comparison with the others. *k*-means performs well on various UCI datasets due to its iterative manner that can correct the clusters several times.

To graphically demonstrate the ability of AGILE, synthetic datasets containing complex and asymmetric shapes [11], is simulated here. AGILE and the other methods are applied to these synthetic datasets and the results depicted in Fig. 4 – Fig. 8. According to these Figures, AGILE outperforms the others except OPTICS.

The main flaw of AGILE is that it is not a self-tuning method. Using global parameters makes the clusters, similar in size, such that so closed valid clusters are merged, and in contrast, it breaks down the long or sparse desired clusters. As an example, two closed valid clusters are merged in Fig. 8(c) to avoid separating the other ones.

Although the fusion phase corrects the retrieved clusters, it should be considered that this phase just applied to the sparsest clusters which identified by the second gap. Thus it cannot merge lost parts of large and dense clusters.

7. CONCLUSION

This paper proposes a novel scheme, termed as AGILE, that benefits from both high speed of partition based and high accuracy of density-based clustering methods. The achieved results on different datasets imply its high efficiency in terms of runtime and accuracy. As a future work, the aim is to make the AGILE self-tuning such that it determines its parameters locally, in order to increase its flexibility and accuracy in retrieving asymmetric, long and sparse clusters.

8. REFERENCES

[1] Ankerst, M., Breunig, M. M., Kriegel, H., & Sander, J. (1999). Optics. *ACM SIGMOD Record*,*28*(2), 49-60. doi:10.1145/304181.304187

[2] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*,*18*(9), 509-517. doi:10.1145/361002.361007

[3] Bishop, C. (2006). *Pattern recognition and machine learning*. Springer Verlag.

[4] Ester M., Kriegel H. P., Sander J., Xu X. (1996). A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, pp. 226-231.

[5] Freidman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. on Mathematical Software*,*3*(3), 209-226. doi:10.1145/355744.355745

[6] Guha, S., Rastogi, R., & Shim, K. (1998). Cure. *ACM SIGMOD Record*,*27*(2), 73-84. doi:10.1145/276305.276312

[7] Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Applied Statistics*,*28*(1), 100. doi:10.2307/2346830

[8] Hinneburg, A., & Gabriel, H. (n.d.). DENCLUE 2.0: Fast Clustering Based on Kernel Density Estimation. *Lecture Notes in Computer Science Advances in Intelligent Data Analysis VII*,70-80. doi:10.1007/978-3-540-74825-0_7

[9] Hinneburg A., Keim D. (1998). An efficient approach to clustering in large multimedia databases with noise, in Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD98), pp.58-65.

[10] Idrissi, A., Rehioui, H., Laghrissi, A., & Retal, S. (2015). An improvement of DENCLUE algorithm for the data clustering. *2015 5th Int. Conf. (ICTA)*. doi:10.1109/icta.2015.7426936

[11] Karypis, G., Han, E., & Kumar, V. (1999). Chameleon: hierarchical clustering using dynamic modeling. *Computer,32*(8), 68-75. doi:10.1109/2.781637

[12] Ng R. T., Han J. (1994). Efficient and Effective Clustering Methods for Spatial Data Mining, Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, pp. 144-155.

[13] Rehioui, H., Idrissi, A., Abourezq, M., & Zegrari, F. (2016). DENCLUE-IM: A New Approach for Big Data Clustering. *Procedia Computer Science,83*, 560-567. doi:10.1016/j.procs.2016.04.265

[14] UCI ML Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2017.

[15] Welch, T. A. (1971). *Bounds on information retrieval efficiency in static file structures* (Unpublished master's thesis).

[16] Yan, D., Huang, L., & Jordan, M. I. (2009). Fast approximate spectral clustering. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 09*. doi:10.1145/1557019.1557118

[17] Zhang, T., Ramakrishnan, R., & Livny, M. (1996). Birch. *Proc. ACM SIGMOD Int. conf. on Management of data - SIGMOD 96*. doi:10.1145/233269.233324

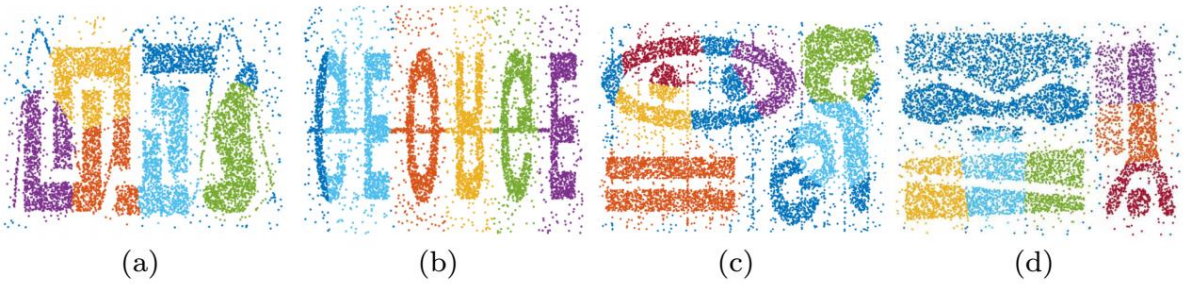

Figure 4. Illustration of the k-means performance.

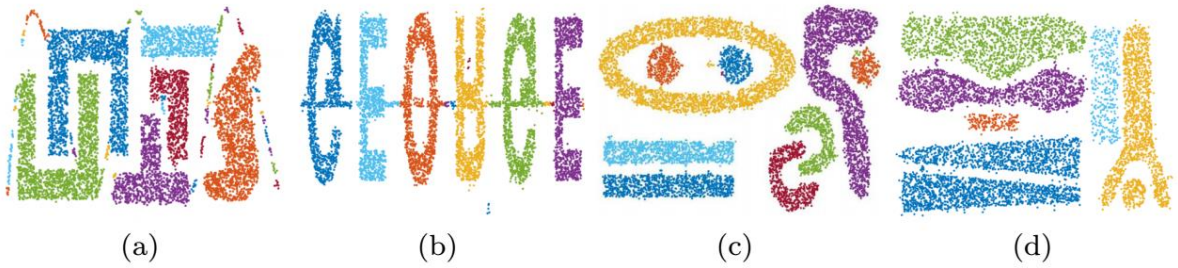

Figure 5. Illustration of the DBSCAN performance.

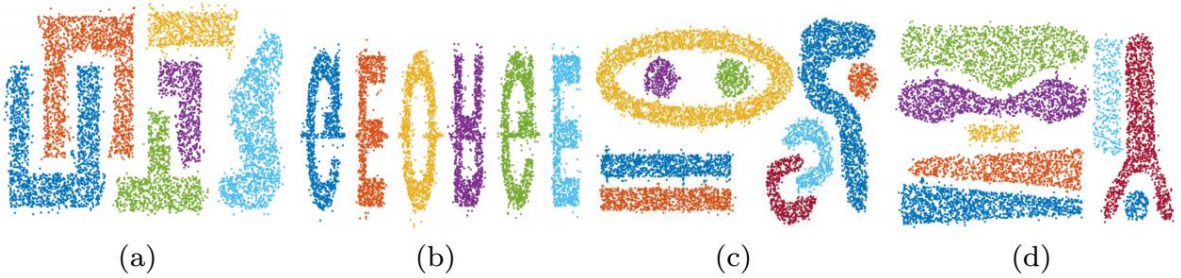

Figure 6. Illustration of the OPTICS performance.

Figure 7. Illustration of the DENCLUE performance.

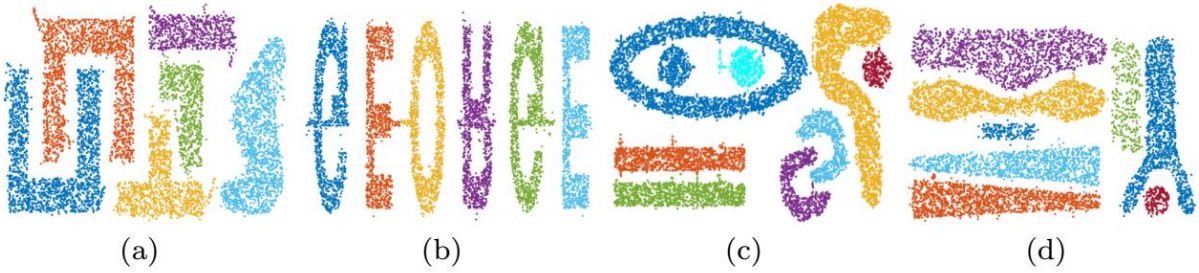

Figure 8. Illustration of the AGILE performance.