ATLANTIS PRESS

Advances in Computer Science Research (ACSR), volume 85

Third Workshop on Computer Modelling in Decision Making (CMDM 2018)

The Clusterization of Geo-Tagged Data for Finding City Sights with Use of a Modification of k-MXT Algorithm

Anastasia Stepanova*, Sergei V. Mironov†[0000−0003−3699−5006],
Eugene Korobov, Sergei Sidorov[0000−0003−4047−8239]
Saratov State University, Saratov, Russia
*ste.nasty@gmail.com, †mironovsv@info.sgu.ru

***Abstract*—The paper considers the task of detection of the most attractive for tourists city sights using the database with geo-tagged data of photographs.We form a graph on the basis of the geo-tagged spot coordinates and rewrite the problem as the graph clusterization task. In our work we use two clustering algorithms, DBSCAN and k-MXT. Moreover, we develop a modification of the k-MXT algorithm called the k-MXT-Gauss algorithm, where the calculation of the weights of the graph edges is transformed using the Gaussian distribution density. We compare the performance of k-MXT-Gauss algorithm with the performance of k-MXT and DBSCAN algorithms both on simulated data and real data.**

***Index Terms*—graph clusterization, weighted graph, geo-tagged data**

I. Introduction

Using ideas of the work [1] we perform the task of graph clusterization for a graph constructed with use of geo-tagged data. First, we employ such clustering algorithms as k-MXT algorithm [1] and the DBSCAN algorithm [2] for the task of clustering geo-tagged data using both on simulated data and real data (one of Russian cities as an example). The graph will be constructed on the basis of a dataset of geo-tagged data of photographs taken from Flickr[1]. In the graph a vertex (photograph) connects to other vertices (photographs) if the geo-distance between them is less than a fixed threshold. We examine the constructed in such way graph to explore its structural properties. It turns out that the subgraphs at some location are very dense but the links between such dense subgraphs is quite sparse. Empirical results presented in this paper are close to the experimental results obtained in the works [1] and [3] in which it was shown that the k-MXT algorithm is applicable to the solution of the clusterization problem and produces clusters which are comparable to the clusters obtained by DBSCAN algorithm. An extended version of DBSCAN algorithm was examined in [4] to detect stops in individual trajectories using GPS data. The paper [5] proposes a novel niche genetic algorithm (NGA) with density and noise for K-means clustering and apply the algorithm on taxi GPS data sets.

The paper [6] presents and examines a family of clustering algorithms (including density-based and partition-based algorithms). One of analyzed algorithms is K-means, which is partition-based and is well-known and employed in different applications. The algorithm is simple and effective tool with respect to other methods of clusterization (such as density-based or model-based). K-means algorithm has the number of clusters K as an input [7], [8].

In this paper we present a modification of the k-MXT algorithm, where the calculation of the weights of the graph edges is transformed using the Gaussian distribution density (the k-MXT-Gauss algorithm). We compare the performance of k-MXT-Gauss algorithm with the performance of k-MXT [1] and DBSCAN [2] algorithms on simulated data obtained using the Python machine-learning library scikit-learn. This paper also shows the performance of algorithms for the city of St. Petersburg based on data taken from the Flickr site.

The algorithms were implemented in Python using the Jupyter environment.

II. Clustering Algorithms

A. DBSCAN Algorithm

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed in the paper [2]. It is one of the most well-known clustering algorithms and most popular in research literature. DBSCAN algorithm is based on the following idea. For a given set of points in some space, it groups together the points that have many nearby neighbors. The points whose nearest neighbors are too far away, are marked as outliers points by the DBSCAN algorithm.

Let $G = (V, E)$ be a graph, where V is the set of vertices of the graph and E is the set of edges of the graph. The vertices of graph G are the points on the map where the photographs were taken. Initially, the graph G does not contain any edges. We need some definitions and notations. Let

- $\rho(p, q)$ be the distance between the vertices p and q;
- $E(p)$ be ε-neighborhood of the vertex (object) p, defined by the formula $E(p) = \{q \mid \rho(p, q) \leq \varepsilon; p, q \in V\}$, where ε is a certain constant;
- a kernel or root object of degree $minPts$ is an object, whose ε-neighborhood contains no less than $minPts$

[1] "Flickr." [Online]. Available: https://www.flickr.com/

elements, i. e. $|E(p)| \geq minPts$, where $minPts$ is a constant. Objects that are not root are noise;
- if $q \in E(p)$ and p is a root object, then the object q is directly densely attainable from the object p;
- if there exist $p_1, p_2, \ldots p_n$, such that $p_1 = p$, $p_n = q$ and p_{i+1} is directly densely attainable from p_i for all $i \in 1 \ldots n-1$, then the object q is densely attainable from the object p.

If root vertex p of the graph G is not assigned to any cluster, then all vertices directly densely attainable from p to be added to the traversal list. For each root node q in the traversal list, we find all directly densely attainable vertices and add them to the same traversal list. The vertex p and all the vertices from the traversal list that do not belong to any cluster are assigned to a new cluster.

B. k-MXT Algorithm

k-MXT algorithm was proposed in [1]. The algorithm considers a graph fragmentation process in the following way: each vertex v selects the k adjacent vertices which have the largest number of common neighbours. For each selected neighbour u, we retain the edge (v, u) to form subgraph S of the input graph. The object of interest in [1] are the components of S, the k-Max-Triangle-Neighbour (k-MXT) subgraph, and the vertex clusters they produce in the original graph.

The vertices of graph G are the points on the map in which the pictures were taken. Initially, the graph G does not contain edges.

Each vertex of a graph is connected to all vertices located at a distance less than ε. No two vertices are connected more than once.

For the k-MXT algorithm, we need to construct a graph G' based on the graph G. Initially, the graph G' contains all vertices of the graph G, but does not contain any edges.

Let the set E contain all edges emanating from the vertex p. We add k edges from the set E to the graph G' according to the following rule: for each vertex p of the graph G, the weight of all edges originating from this vertex is calculated. The weight of the edge (q, r) is equal to the number of common neighbors of vertex q and vertex r. Let the set E contain all the edges emanating from the vertex p. Add k edges to the graph G' from the set E by the following rule.

- If the set E has less than k edges or exactly k edges, then all edges from the set E are added to the graph.
- If the set E has more than k edges, then k edges with the maximum weight are added to the graph. If it is needed to select from several edges with the same weight, then the edges are selected randomly.

Clusters will be the connected components of the resulting graph G'. In this paper, the connectivity components are distinguished by traversing the graph in depth-first.

C. k-MXT-Gauss Algorithm

As in the k-MXT algorithm, the vertices of the graph G are points on the plane. Initially, the graph G contains no edges edges.

Each vertex of a graph is connected to all vertices located at a distance less than ε. No two vertices are connected more than once.

For the k-MXT-Gauss algorithm, we need to construct a graph G' based on the graph G. Initially, the graph G' contains all vertices of the graph G, but does not contain any edges.

For each vertex p of the graph G, the weight of all edges incident to this vertex is calculated. The weight of the edge (q, r) is equal to the number of common neighbors of the vertex q and the vertex r multiplied by the value of the Gaussian distribution density at the point x equal to the distance between the vertices of q and r; $\sigma = \frac{1}{3}\varepsilon$; $\mu = 0$.

$$N(x, \sigma, \mu) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

Let the set E contain all edges emanating from the vertex p. Add to the graph $G^{'}$ k edges from the set E according to the following rule:

- If the set E has less than k edges or exactly k edges, then all edges from the set E are added to the graph.
- If the set E has more than k edges, then k edges with the maximum weight are added to the graph. If it is needed to select from several edges with the same weight, then the edges are selected randomly.

The clusters will be the strongly connected components of the resulting graph $G^{'}$.

When we use the gaussian distribution density function to modify arc weights, we choose the standard deviation of the distribution using three sigma rule: to fully cover all points located at a distance of ε from the given vertex, it suffices to let $\sigma = \varepsilon/3$.

III. Comparison of algorithms on simulated data

A. Clustering Validation Metrics

We use the ARI (Adjusted Rand Index) metric to evaluate the correct operation of the k-MXT, k-MXT-Gauss, DBSCAN clustering algorithms.

Let Y be the proper partitioning of the vertices into clusters, X be the result of clustering.

Define the following variables:

- a is the number of pairs of elements that belong to the same clusters in partitions X and Y
- b is the number of pairs of elements that belong to different clusters in partitions X and Y

Then the Rand Index (RI) is defined as follows:

$$RI = \frac{a+b}{\binom{n}{2}}, \tag{2}$$

where n is the number of vertices, $\binom{n}{2}$ is the total number of all possible pairs of vertices.

Figure 1. The graphs of the k-MXT and k-MXT-Gauss algorithms for the ARI metric values of ε and illustrations of the best ARI metric of both algorithms for a given k are presented. Data type is blobs

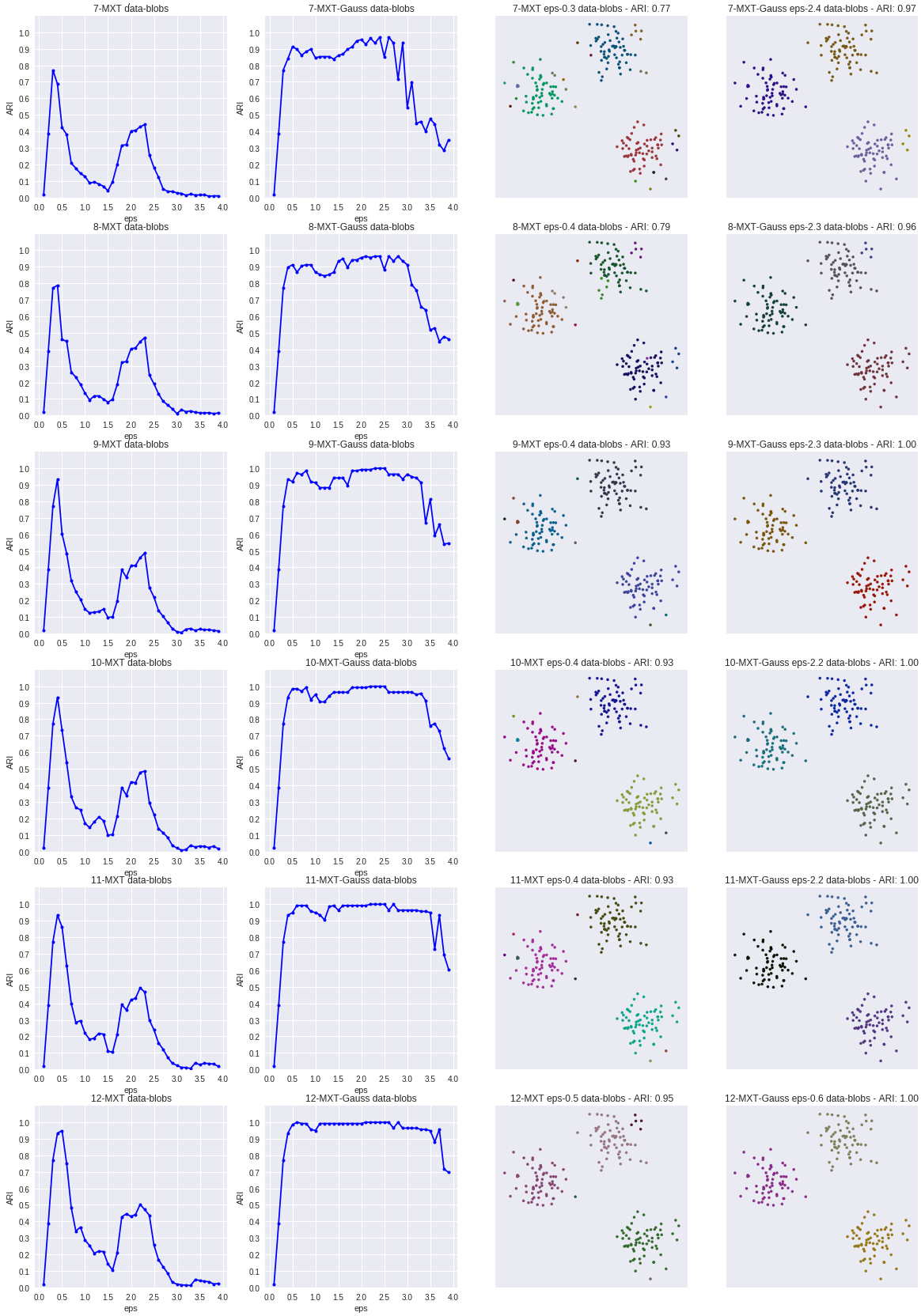

Figure 2. The graphs of the k-MXT and k-MXT-Gauss algorithms for the ARI metric values of ε and illustrations of the best ARI metric of both algorithms for a given k are presented. Data type is blobs

However, the RI metric does not guarantee that its value is close to zero when we analyze a random assignment of cluster labels. In order to avoid this, the ARI metric is introduced, which is calculated by the formula:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]} \quad (3)$$

ARI is a measure of the distance between different partitions. The ARI value varies between -1 and 1. Negative values correspond to partitions into clusters that are far from being similar, values close to zero correspond to random partitions, positive values show that the two partitions X, Y are similar ($ARI = 1$ if the partitions X, Y coincide completely)

B. Simulated data

We model the data using the Python scikit-learn library.

We obtain the blobs data using the $sklearn.datasets.make\_blobs$ function with the following parameters:

- total number of generated points is 200 ($n\_samples = 200$).
- cluster standard deviation is 0.5 ($cluster\_std = 0.5$).

In the Fig. 1, 2 with $k = 3$, the k-MXT-Gauss algorithm has a larger ARI metric value ($ARI = 0.93$), the k-MXT algorithm has the value of the metric $ARI = 0.39$. For $k \geq 9$ the k-MXT-Gauss algorithm has the value of the metric $ARI = 1.00$. At the same time, k-MXT for the considered values of k does not reach the value of $ARI = 1.00$.

When considering the graphs in the Fig. 1, 2, it can be seen that the k-MXT algorithm has no more than one point where $ARI = 1.00$ is reached, the DBSCAN algorithm (Fig. 3) has a small range of values ε, for which $ARI = 1.00$, and the k-MXT-Gauss algorithm on most graphs has a large range of ε, for which $ARI = 1.00$.

Good performance of k-MXT-Gauss on smaller values k allows us to get results on a large data in a shorter program runtime. The presence of a large range of ε values resulting in $ARI = 1.00$ simplifies the process of selecting the ε parameter.

IV. Application and comparison of the results of the work of algorithms for clustering photos in St. Petersburg

A. Data description

The data with coordinate labels we took from the site Flickr. In total for St. Petersburg, we received about 200 thousand photographs with geo-data and usernames.

The data is filtered in the following way: if there are multiple photos from the same user having the same geodata, we consider only one of those photos (about 2000 photos).

B. Graph Data Representations

For DBSCAN, k-MXT, k-MXT-Gauss algorithms, the vertices correspond to the photos with geotags. An edge between two vertices is added if the distance between two vertices does not exceed ε. The distance between two points was considered

Figure 3. The graphs of the DBSCAN algorithms ARI metric values from $minPts$ are presented. Data type is blobs

as the shortest distance between two points on the surface of the sphere calculated with the following formula:

$$\Delta\sigma = 2R \times$$
$$\times \arcsin\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \cos\phi_2 + \sin^2\left(\frac{\Delta\lambda}{2}\right)},$$

where R is the Earth radius, ϕ_1, ϕ_2 are the latitudes of the first and second points, λ_1, λ_2 – the longitudes of the first and second points accordingly,

$$\Delta\phi = |\phi_1 - \phi_2|, +$$
$$\Delta\lambda = |\lambda_1 - \lambda_2|.$$

C. Results of applying the algorithms to geodata for St. Petersburg

The clusters allocated by the k-MXT-Gauss algorithm with $\varepsilon = 50$ (Fig. 4) correspond to the actual separation of objects into clusters. The algorithm allocates clusters of both large sizes and small sizes.

DBSCAN algorithm with $\varepsilon = 50$, $minPts = 2$ also performs cluster allocation well (Fig. 5), but unlike the k-MXT-Gauss algorithm, DBSCAN does not select clusters of

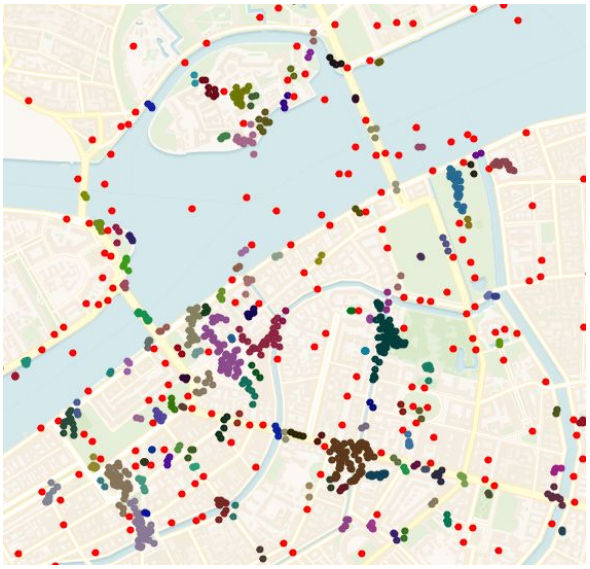

Figure 4. 7-MXT-Gauss with $\varepsilon = 50$

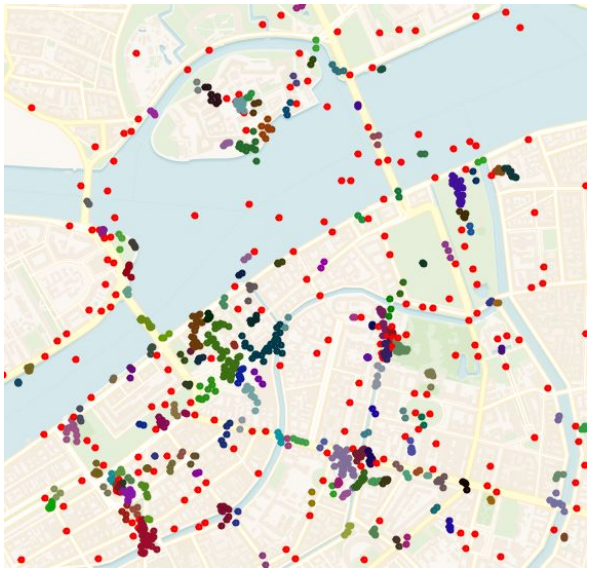

Figure 6. 7-MXT with $\varepsilon = 50$

small sizes. Instead, the vertices that would belong to small clusters are considered to be noise.

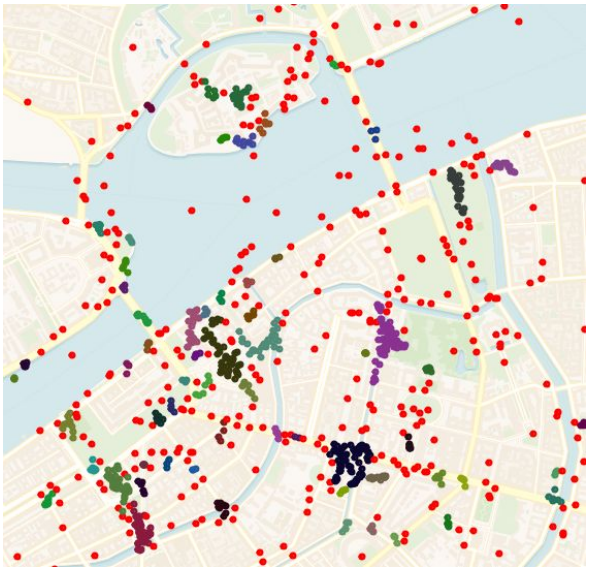

Figure 5. DBSCAN with $\varepsilon = 50$, $minPts = 2$

The algorithm 7-MXT $\varepsilon = 50$ (Fig. 6) performs cluster allocation worse if the data contains a point having a large number of vertices near it (clusters are mixed, some vertices become noise).

V. Conclusions

The comparison of the results of the k-MXT-Gauss algorithm and the results of the k-MXT, DBSCAN algorithms on the simulated data shows that the k-MXT-Gauss algorithm, starting from a certain value of k, has a large range of $epsilon$, for which $ARI = 1.00$. In contrast to the k-MXT algorithm, the k-MXT-Gauss algorithm reaches $ARI = 1.00$ with a smaller value of k.

The application of the k-MXT-Gauss algorithm to the data obtained from the Flickr website for St. Petersburg shows that the k-MXT-Gauss algorithm works well not only on the simulated data. Thus, the k-MXT-Gauss algorithm can be employed to solving various applied problems without the need for careful selection of parameters for cluster allocation.

Acknowledgments

This work was supported by the Russian Fund for Basic Research, project 18-37-00060.

References

[1] C. Cooper and N. Vu, "An experimental study of the k-MXT algorithm with applications to clustering geo-tagged data," *LCNS*, vol. 10836, pp. 145–169, 2018, 15th Workshop on Algorithms and Models for the Web Graph, WAW 2018 - Moscow, Russian Federation.

[2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining KDD'96*. AAAI Press, 1996, pp. 226–231.

[3] A. Stepanova, S. Mironov, and E. Korobov, "The clusterization of geo-tagged data for finding tourists attractions," *Mathematical modeling, computer and field experiment in natural sciences*, no. 3, p. 1971, 2018.

[4] T. Luo, X. Zheng, G. Xu, K. Fu, and W. Ren, "An improved DBSCAN algorithm to detect stops in individual trajectories," *ISPRS International Journal of Geo-Information*, vol. 6, no. 3, p. 63, 2017.

[5] X. Zhou, J. Gu, S. Shen, H. Ma, F. Miao, H. Zhang, and H. Gong, "An automatic k-means clustering algorithm of GPS data combining a novel niche genetic algorithm with noise and density," *ISPRS International Journal of Geo-Information*, vol. 6, no. 12, p. 392, 2017.

[6] J. Han, M. Kamber, and J. Pei, *Data mining concepts and techniques, third edition*. Waltham, Mass.: Morgan Kaufmann Publishers, 2012.

[7] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010, award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[8] M. A. Rahman and M. Z. Islam, "A hybrid clustering technique combining a novel genetic algorithm with k-means," *Knowledge-Based Systems*, vol. 71, pp. 345–365, 2014.