Journal of Network and Innovative Computing
ISSN 2160-2174 Volume 1 (2013) pp. 119 -125

Improvement to the K-Means Algorithm Through a Heuristics Based on a Bee Honeycomb Structure

Joaquín Pérez[1], Adriana Mexicano[1] Rodolfo Pazos[2], René Santaolaya[1], Miguel Hidalgo[1], Alejandra Moreno[1] and Nelva Almanza[1]

[1] Centro Nacional de Investigación y Desarrollo Tecnológico,
Cuernavaca, México,
jpo_cenidet@yahoo.com.mx

[2] Instituto Tecnológico de Cd. Madero,
Tampico, México,
r_pazos_r@yahoo.com.mx

***Abstract*: The object clustering problem, according to their similarity measures, can be formulated as a combinatorial optimization problem. The K-Means algorithm has been widely used for solving such problem; however, its computational cost is very high. In this work a new heuristics is proposed for reducing the computational complexity in the classification step of the algorithm, based on a honeycomb structure, which the algorithm builds when clusters are visualized in a two-dimensional space. In particular it has been observed that an object can only change membership to neighboring clusters. The heuristics consists of performing distance calculations only with respect to centroids of neighboring clusters, which reduces the number of calculations. For assessing the performance of this heuristics, a set of experiments was carried out that involved 2,500, 10,000 and 40,000 objects uniformly distributed in a two-dimensional space, as well as real-world instances of 3,100 and 245,057 objects with 2 and 3 dimensions. The results were encouraging, since the calculation time was reduced as much as 65% on average, with respect to the standard K-Means algorithm for the synthetic instances, and up to 62% on average for the real-world instances, while the quality was reduced on average by 0.05% and 2.5%, respectively.**

***Keywords*:** Combinatorial Optimization, K-Means, Biologically Inspired Computing**.**

I. Introduction

Clustering is a process of assigning a set of objects into subsets, called clusters, such that objects in the same cluster are similar and objects in different clusters are quite distinct [1]. Clustering has been applied to many fields, and according to [2], the main five major groups are: a) biology and zoology; b) medicine and psychiatry; c) sociology, criminology, anthropology, and archeology; d) geology, geography and remote sensing; and e) information retrieval, pattern recognition, market research, and economics.

An important category of clustering algorithms are center-based algorithms, which use a center to represent a cluster and are very efficient for clustering high-dimensional databases. Among the center-based clustering algorithms are those described in [3], [4] and [5]; however, one of the most popular and simple clustering algorithms is K-Means [6], also known as Lloyd's algorithm [7]. A detailed description of the K-Means algorithm can be found in [6].

The main steps of the standard K-Means are the following:

- Initialization. Consists of defining the objects to be partitioned, the number of clusters, and a centroid for each cluster. With regard to the definition of the initial centroids, the random method is the most widely used.
- Classification. For each object, its distance to the centroids is calculated, the nearest centroid is determined and the object is assigned to the cluster related to this centroid.
- Centroid calculation. The centroid is recalculated for each cluster based on the current partition.
- Stopping criteria. Several convergence conditions have been used, such as: stopping when reaching a given number of iterations, when there is no change of constituent objects for each cluster, or when the difference of the centroids at any two consecutive iterations is smaller than a given threshold. If the convergence condition is not satisfied, then steps 2, 3, and 4 are repeated.

An aspect that affects the computational cost of K-Means is the number of iterations that the algorithm needs to carry out, since, for each iteration, it calculates the distance of each object to the clusters' centroids. In this work, we propose a new heuristics, called *Honeycomb Classification* (HC), to reduce the number of calculations in the classification step of K-Means.

This paper is organized as follows: Section 2 presents the

related work. Section 3 describes the Honeycomb heuristics proposed to improve the classification step of K-Means. Section 4 presents the computational experiments. Finally, Section 5 presents conclusions and directions for future work.

II. Related work

Several improvements have been proposed to enhance the standard K-Means algorithm in its main steps. Some works have focused on finding the best value for the initial number of clusters *k* and the best way of choosing the initial centroids as described in [8], [9], [10], [11], [12], [13], [14], [15], [16] and [17]. Other research works have focused on defining the best stopping criterion in order to avoid excessive iterations considering that K-Means converges at a local minimum [18], amongst others [19], [20]. Thus, in the following paragraphs, we focus on related works that propose improvements to minimize the number of calculations in the classification step of K-Means.

In [21] an improvement was proposed for the Filtering Algorithm (FA), a variant of the K-Means algorithm [22]. FA considers that objects are stored in a kd-tree, i.e., a binary tree that divides the objects into cubes using perpendicular hyperplanes. Each node in the tree is associated with a set of data points called a cell. At each iteration, FA determines the nearest centroids of every cell by calculating all object-to-centroid distances, and verifies whether each member of the centroid set should be pruned for each internal node. The improvement consists of identifying the centroids that, between the current and the previous iteration, were displaced. Results show that the improvement reduces the execution time by up to 33.6% in comparison to the original FA algorithm.

In [23] the heuristics proposed compresses and removes objects that are close to the centroid. An object is considered close to the centroid if the distance to its nearest centroid is smaller than the average distance of all the objects in the same cluster to their centroid. The heuristics is applied repeatedly until 80% of the objects are removed. Results show that this heuristics reduces execution time by up to 79% especially for high dimensional data sets; but unfortunately it reduces the cluster quality by 14.08% in comparison to the standard K-Means algorithm.

The improvement proposed by [24] consists of calculating and storing the shortest distance between each object and its nearest centroid at each iteration. For each object, the previous distance to the current one is compared. If the previous distance is less than or equal to the current one, the object remains in the cluster and is discarded for subsequent calculations; otherwise, it is necessary to determine the distance between the object and all cluster centroids as well as to identify the new nearest cluster. Results show that this improvement reduces the execution time without significantly decreasing the cluster quality.

Pérez et al. [25] proposed Early Classification, a heuristics for improving the classification step of the K-Means algorithm. The proposed heuristics identifies and excludes from future calculations objects that have low chance of changing cluster membership. The low chance of cluster-membership change is defined considering if the difference of the distances between an object and its two nearest clusters is greater than the sum of the two largest displacements of the centroids in the current iteration. Results show that for real instances, this approach decreases by 50.16% the execution time, but the quality is decremented by 6.94%.

Chukwudi et al. [26] developed an algorithm based on the principal component analysis. The algorithm helps to determine when all the data points in a cluster are stable in order to avoid calculation of the distances between the data points and the $k-1$ centroids. The improvement reaches a decrease in time execution of up to 14.17% with respect to the standard K-Means.

Hitendra et al. [27] proposed in 2013 a prototype-based method. The method consists of partitioning the data into small clusters, which are of varying sizes. Each cluster is represented by a prototype. Prototypes are partitioned using a version of K-Means that avoids empty clusters. After data partition, a correcting method is applied in order to avoid deviations between K-Means and the proposed approach. Results show that the prototype-based method allows reducing execution time by up to 74.9% without loss of quality.

Vijay et al. [28] proposed a modified version of K-Means that defines a threshold distance for each centroid. The aim of their work is comparing the distance between each object and the cluster's centroid with such threshold in order to minimize the calculation effort. The threshold distance is obtained from the distance calculation between each cluster centroids. If the distance from object *i* to centroid *j* is less than the threshold distance of cluster *j*, then object *i* moves to it. In the best case, for a single object, there is only one distance calculation. Unfortunately, there are not as many experiments as expected, but one that barely shows the difference in the number of iterations.

In the following section a heuristics for the K-Means algorithm is explained, which is inspired by a bee honeycomb and aims at improving the efficiency of K-Means.

III. The Honeycomb Classification Heuristics (HC)

The development of the HC heuristics arose from a detailed visual analysis of the grouping generated after each iteration by the K-Means algorithm. The visual analysis was conducted using synthetic instances with 2,500, 10,000, and 40,000 two-dimensional objects distributed uniformly. To perform the analysis, a graphical visualization tool was developed which indicates (by a point) the position of each object in a two-dimensional plane, each cluster centroid is represented by a square at the center, and each generated cluster is distinguished from another because the objects in each cluster have the same color, which differs from the colors of adjacent clusters.

After carrying out several executions with the K-Means algorithm using synthetic instances, in general, we noted that the cluster's geometry looks like the cells in a honeycomb (see Fig. 1.a). Likewise, Fig. 1.b shows as an example, a section of the cluster formation using an instance with 40,000 objects, that is obtained with the graphical visualization tool and it is possible to observe the similarity with a honeycomb structure.

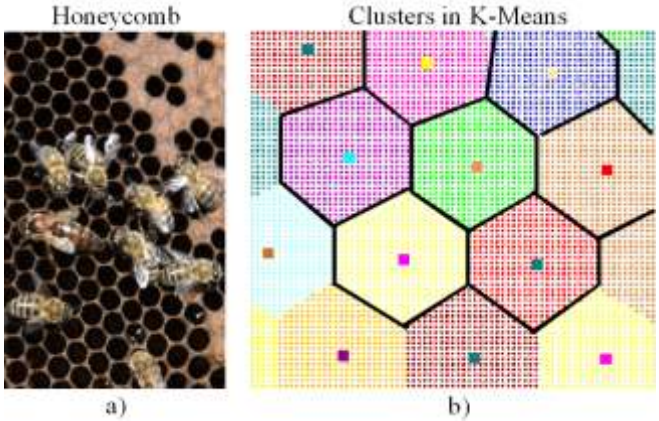

Figure 1. K-Means clusters look like a bee honeycomb on uniformly distributed objects.

Inspired by the visual analysis and the honeycomb structure, four characteristics in the behavior of the K-Means algorithm, that can help improve the efficiency of the algorithm, were identified:

1. In general, for large instances when the algorithm stops, the clustering has a geometry similar to a bee honeycomb, with hexagonal cells (see Fig. 1.b).
2. In the first iteration clusters with irregular polygon shapes are formed, generally with 5 or 6 sides on average. However, there were some clusters with 7, 8 and 9 sides; few clusters had 7 and 8 sides and others very rare had 9. As the number of iterations increases, the shape tends to be a hexagon.
3. The change in cluster membership of an object between the previous and the current iteration is towards one of its adjacent clusters. For a 40,000 object instance, Fig. 2 shows that at iteration 1 (Fig. 2.a) all the objects (dots) in the irregular marked polygon were assigned to cluster C_1. In the second iteration (Fig. 2.b), we can observe that the size and shape of cluster C_1 was modified; additionally, some objects assigned to cluster C_1 at iteration 1, were assigned to clusters C_2 and C_3 in the second iteration; and some objects assigned to cluster C_5 at iteration 1, were assigned to cluster C_1 in the second iteration.
4. The highest number of changes in centroid values occurs in the first two iterations, as shown in Figs. 2.a and 2.b. Also, it is observed how from iteration 1 to 2, the shapes of the polygons change dramatically; however, after the third iteration, the shapes tend to be hexagonal polygons, i.e., the final shape when the algorithm converges (see Fig. 1.b).

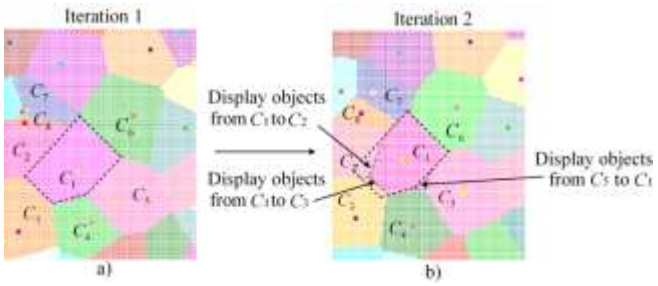

Figure 2. General behavior in object cluster change.

After analyzing these four characteristics, the extracted knowledge was used in order to generate the HC heuristics, whose objective is to reduce the number of calculations in the classification step of the standard K-Means algorithm.

For better understanding the HC heuristics, first some terms are introduced and then the proposed heuristics is presented.

A. *Calculations in the classification step of K-Means*

Let $I = \{i_1, ..., i_n\}$ be a set of n d-dimensional objects to be partitioned, $C = \{C_1, ..., C_k\}$ be the set of partitions of I into k sets ($2 < k < n$). For each iteration of the classification step, the standard K-Means algorithm calculates $\|i_p - \mu_l\|^2$, for $p = 1, ..., n$ and $l = 1, ..., k$; where μ_l is the centroid of the objects in C_l.

This step constitutes the most important computational cost of the algorithm, in terms of the number of calculations.

B. *Cluster Walls*

The cluster walls allow visualizing the polygons formed by the clusters in the final iteration of the K-Means algorithm. For each cluster, its wall is informally defined by the most distant objects from the cluster centroid (see Fig. 3).

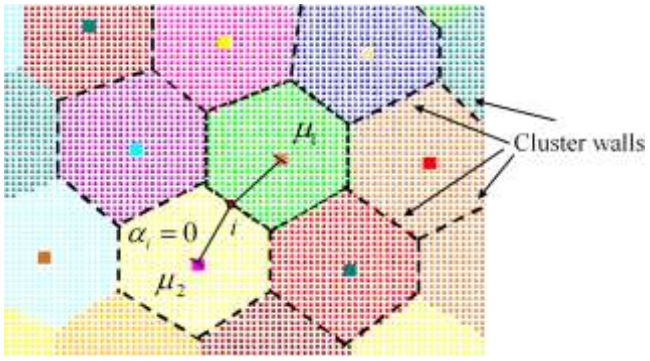

Figure 3. A honeycomb formed by cluster walls.

The objects in the *cluster walls* can be identified as follows: given an object i and its two nearest centroids μ_1 and μ_2, the object belongs to the cluster walls if it is equidistant to μ_1 and μ_2.

The equidistance α_i of an object i to the two nearest centroids μ_1 and μ_2 is defined as: $\alpha_i = \text{abs}(\|i-\mu_1\|^2 - \|i-\mu_2\|^2)$. The lower bound of α_i is 0, and the upper bound is $\|\mu_1-\mu_2\|^2$. The lower bound indicates that object i is located at an equidistant position to the centroids μ_1 and μ_2, whereas the upper bound indicates that object i is located at the same position of centroid μ_1. Each cell generated by the K-Means execution can be delimited by a set of cluster walls that in turn allows defining the number of *neighboring clusters of each object*. Similar concepts of equidistance are found in voronoi diagrams [29].

C. *Neighboring clusters*

The neighboring clusters of an object i are: (a) the object's cluster and (b) the clusters that are adjacent to the cluster to which the object belongs, and they correspond to the clusters which object i has the highest probability of cluster-membership change.

The HC heuristics was generalized so that the number of adjacent clusters be 7. Then, object i has to calculate the distance to the centroids of the adjacent clusters, $\{C_1, ..., C_7\}$ plus the distance to the centroid of its own cluster $\{C_i\}$, giving a total of 8 distance calculations to the centroids.

Hence, the set of neighboring clusters of an object i is defined as $B_i = \{C_i \cup \{C_1, ..., C_7\}\}$, at iteration 2, where C_i is the cluster to which object i belonged at iteration 2 and $\{C_1, ..., C_7\}$ are the adjacent clusters centroids to cluster C_i at iteration 2. Fig. 4 shows that the neighboring clusters of object i is the set $\{C_i, C_1, ..., C_7\}$ whose centroids are $\{\mu_i, \mu_1, ..., \mu_7\}$ respectively (see Fig. 5).

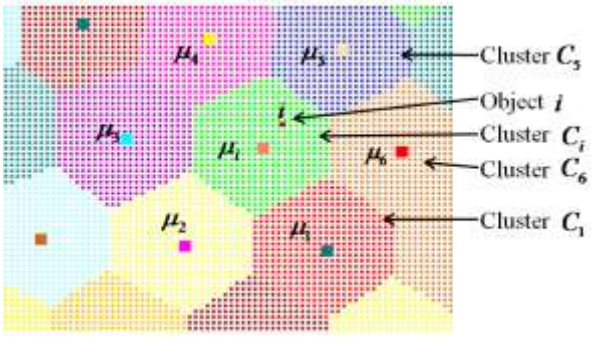

Figure 4. Generalized number of neighbors for the HC heuristics.

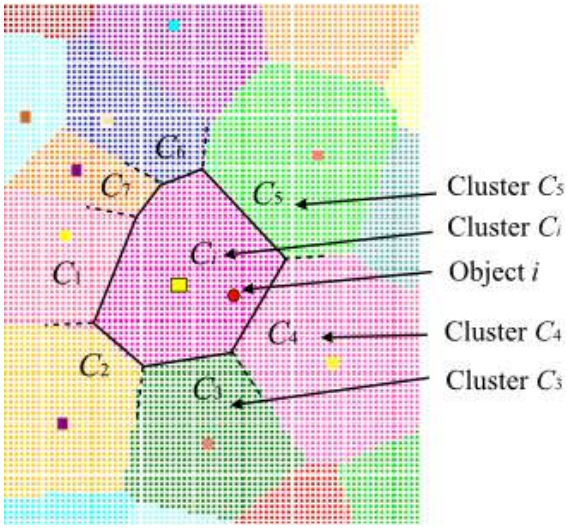

Figure 5. Distance calculations from object i to the neighboring centroids.

D. The HC heuristics

The HC heuristics is simple: it consists of identifying the neighboring *clusters* of an object in the third iteration in order to only calculate the distances from the object to its neighboring clusters centroids. This allows to exclude those calculations, in the classification step, for the non neighboring clusters centroids of the objects in subsequent iterations.

The HC heuristics included in the classification step of the K-Means is described below:

Algorithm 1 HC Heuristics (A section of the classification step of the algorithm)

```
for all iterations
if (iteration <= 3) then
    for each object i calculate the Euclidean distance from it
     to each of the clusters centroids in μ
    if (iteration = 3) then
      for each object i, store its 8 neighboring clusters
       centroids
    end if
else
    for each object i calculate the Euclidean distance from it
     to each one of the 8 neighboring cluster centroids
end if
assign the object to the centroid whose distance to the object
is the smallest
```

IV. Experimental results

This section presents the results from a set of experiments conducted to assess the improvement of HC with respect to the standard K-Means. The standard K-Means and HC were implemented in C language.

The experiments were performed on a computer with the following configuration: Intel(core) i5, 2.50 GHz processor, 6 GB of RAM and Ubuntu 12.04 operating system. The experiments were performed on three different kinds of synthetic instances with 2,500, 10,000 and 40,000 objects uniformly distributed in a two-dimensional space. In addition, two real-world instances were used called D31 [30] and skin segmentation [31], with 3,100 objects (2 dimensions) and 245,057 objects (3 dimensions), respectively.

Tables 1 and 2 show the comparative results of the standard K-Means and the HC heuristics in 7 and 9 columns respectively. Regarding Table 1, the first column indicates the number of objects for each instance. The second and third columns refer to the execution time that shows the average results for 30 runs for HC and K-Means expressed in milliseconds for each instance; the fourth and fifth columns indicate the average squared error for both HC and K-Means for 30 runs of each instance. The following column shows the average percentage of time reduction obtained with the HC algorithm for 30 runs of each instance. The last column indicates the average error difference obtained with the HC algorithm for 30 runs of each instance. Finally, the last row shows the overall averages for time reduction and error difference.

With regard to Table 2, the columns correspond to those of Table I, except for the first column which indicates the name of the dataset and the second column that indicates the number of clusters. Similarly, the last row shows the overall averages for time reduction and error difference.

At each run the initial centroids were randomly generated for each instance, and they were input to the K-Means standard and to the improved algorithm (HC). Each run considered 100 clusters and, in the case of the improved algorithm, the HC heuristics was applied from the third iteration and subsequent ones on every run, and the number of neighboring clusters was set to 8. Besides 100 clusters, the D31 instance was also tested with 31 clusters and skin segmentation was tested with 31, 200 and 400 clusters.

To compare the quality of the results, the sum of the squared errors was taken as reference in the final iteration of the algorithm.

According to the results of Table 1, notice that the average quality difference obtained with the HC heuristics does not increase more than 1%; i.e., in the case of the synthetic instance with 2,500 objects, the error difference is 0.01%. In contrast, the average time execution was reduced by 65%, i.e.,

Number of objects	Execution time		Squared error		% decreased time	% error difference
	HC	K-Means	HC	K-Means		
2,500	77.98	166.25	4,844.32	4,843.67	53.09	0.01
10,000	543.67	1,757.52	38,356.44	38,346.36	69.07	0.03
40,000	4,346.07	16,910.25	305,517.81	305,172.69	74.30	0.11

Table 1. Average results of 30 runs for each of the synthetic instances.

Dataset	Number of clusters	Number of objects	Execution time		Squared error		% decreased time	% error difference
			HC	K-Means	HC	K-Means		
D31	31	3,100	42.95	70.11	3,731.03	3,730.43	38.74	0.01
	100	3,100	95.46	201.17	2,102.57	2,102.26	52.55	0.01
Skin Segmentation	31	245,057	6,163.86	11,988.69	4,099,748.00	4,015,861.25	48.59	2.08
	100	245,057	17,478.13	80,590.11	2,405,388.04	2,322,294.84	78.31	3.57
	200	245,057	37,417.42	162,541.83	1,791,915.75	1,705,012.75	76.98	5.09
	400	245,057	79,402.46	402,213.31	1,408,255.25	1,350,002.38	80.26	4.31

Table 2. Average results of 30 runs for each of the real-world instances.

for the synthetic instance with 40,000 objects, there is a 74.30% reduction in execution time.

In the results of Table 2, the average quality difference obtained with the HC heuristics is 2.5%; i.e., in the case of the D31 instance with 3,100 objects, the error difference is 0.01%. In contrast, the average time execution was reduced by 62%; for example, for the skin segmentation instance with 245,057 objects and 400 clusters, there is an 80.26% reduction in execution time; therefore, the HC heuristics performs better as the number of objects increases as well as the number of clusters.

V. Conclusions and Future Work

This work shows that it is possible to reduce the execution time of the K-Means algorithm by using biologically inspired heuristics. Through a detailed visual analysis of the clusters generated by the K-Means algorithm and the displacement of the centroids throughout the iterations, it was observed that objects only change membership from a cluster to a contiguous one. It was also observed that the number of neighboring clusters to any given cluster was at most eight, at the first iteration. This knowledge was used in the classification step, so the distance of an object is calculated only to its eight adjacent clusters, and excluding the calculation for the rest of the centroids.

For assessing the performance of the proposed heuristics, a set of experiments were carried out on an instance that involved 2,500, 10,000 and 40,000 objects with two attributes (dimensions) and uniformly distributed objects. Also, two real-world instances were used with 3,100 and 245,057 objects. Based on the experimental results of Tables 1 and 2, it can be observed that, as the number of objects and the number of clusters increase, the results of the HC heuristics are better.

The experimental results were encouraging, since when comparing the standard K-Means algorithm and HC, the results showed that HC attained an average reduction of 65% of processing time with respect to the standard K-Means, while obtaining an average quality reduction of 0.05% for the synthetic instance, as shown in Table 1. In the case of the real-world instances, the average reduction time was 62% with respect to K-Means, with just an average quality reduction of 2.5%, as it can be seen in Table 2. It was observed that, for instances with the largest number of objects, the HC heuristics performs better in time reduction and, similarly, as the number of cluster increases, the results are better too.

In addition, the proposed heuristics can be used in combination with other improvement techniques for speeding up the execution of the K-Means algorithm, which would make possible to integrate HC with other heuristics for further speeding up the algorithm.

Finally, as future work, more experimentation will be conducted for testing the HC heuristics on real-world instances with more than two dimensions and, a comparison of the HC heuristics with other improved K-Means clustering algorithms will be made.

Acknowledgment

We would like to thank Nelva Nely Almanza Ortega and Emanuel Sotelo González (students of the MSc program at the Centro Nacional de Investigación y Desarrollo Tecnológico, CENIDET), Heladio Cisneros Reyes (student at the Universidad de la Sierra Juárez) and Juan Manuel Ortiz Alemán (student at the Instituto Tecnológico de Zacatepec) for their assistance in the algorithm coding and experimentation.

References

[1] A. Jain, M. Murty, and P. Flynn, "Data clustering: a review", *ACM Computing Surveys*, 31(3), pp. 264–323, 1999.

[2] J. Scoltock, "A survey of the literature of cluster analysis*", The Computer Journal*, 25(1), pp. 130–134, 1982.

[3] B. Zhang, "Comparison of the performance of center-based clustering algorithms," Advances in Knowledge Discovery and Data Mining, pp. 569–569, 2003.

[4] M. Teboulle, "A unified continuous optimization framework for center based clustering methods", *The Journal of Machine Learning Research*, 8, pp. 65–102, 2007.

[5] A. Chaturvedi, P. Green, and J. Caroll, "K-modes clustering", *Journal of Classification*, 18(1), pp. 35–55, 2001.

[6] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations". In *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability*, pp. 281–296, 1967.

[7] S. P. Lloyd, "Least Squares Quantization in PCM", *IEEE Trans. Information Theory*, 28(1) pp. 129–137, 1982.

[8] M. E. Agha and W. M. Ashour, "Efficient and Fast Initialization Algorithm for K-means Clustering", *International Journal of Intelligent Systems and Applications*, 1(1), pp. 21–31, 2012.

[9] A. H. Ahmed and W. Ashour, "An Initialization Method for the K-means Algorithm, using RNN and Coupling Degree", *International Journal of Computer Applications*, 25(1), pp. 1–6, 2011.

[10] M. B. Al-Daoud, "A New Algorithm for Cluster Initialization". In *Proceedings of the Second World Enformatika Conference*, pp. 74–76, 2005.

[11] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for K-means clustering", *Pattern Recognition Letters*, 25(11), pp. 1293–1302, 2004.

[12] M. F. Eltibi and W. M. Ashour, "Initializing K-Means Clustering Algorithm using Statistical Information", *International Journal of Computer Applications*, 29(7), pp. 51–55, 2011.

[13] S. J. Redmond and C. Heneghan, "A method for initializing the Kmeans clustering algorithm using kd-trees", *Pattern Recognition Letters*, 28(8), pp. 965–973, 2007.

[14] C. S. Li, "Cluster Center Initialization Method for K-means Algorithm Over Data Sets with Two Clusters". In *Proceedings of the International Conference on Advances in Engineering*, pp. 324–328, 2011.

[15] X. Zhanguo, C. Shiyu, and Z. Wentao, "An Improved Semi-supervised Clustering algorithm based on Initial Center Points", *Journal of Convergence Information Technology*, 7(5), pp. 317–324, 2012.

[16] R. Salman, V. Kecman, Q. Li, R. Strack, and E. Test, "Two-Stage Clustering with k-Means Algorithm", in *Recent Trends in Wireless and Mobile Networks*, A. Özcan, J. Zizka, and D. Nagamalai (eds.), Springer, 2011.

[17] X. Wang, J. Wang, "Research and Improvement on K-Means Clustering Algorithm", In *Proceedings of the 2012 2nd International Conference on Computer and Information Application* (ICCIA), pp. 1138—1141, 2012.

[18] J. Pérez, R. Pazos, L. Cruz, G. Reyes, R. Basave, and H. Fraire, "Improving the Efficiency and Efficacy of the K-means Clustering Algorithm Through a New Convergence Condition", in *Computational Science and Its Applications*, O. Gervasi and M. L. Gavrilova (eds.), LNCS, Germany, 2007.

[19] D. Rebollo, M. Solé, J. Nin, J. Forné, "A modification of the k-means method for quasi-unsupervised learning", *Knowledge-Based Systems*, 37, pp. 176--185, 2013.

[20] N. Kaur, J. Kaur, N. Kaur, "Efficient k-means clustering algorithm using ranking method in data mining", *International Journal of Advanced Research in Computer Engineering & Technology*, 1(3), pp. 85--91, 2012.

[21] J. Z. C. Lai and Y. Liaw, "Improvement of the k-means clustering filtering algorithm", *Pattern Recognition*, 41(12), pp. 3677–3681, 2008.

[22] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, pp. 881–892, 2002.

[23] C. Tsai, C. Yang, and M. Chiang, "A Time Efficient Pattern Reduction Algorithm for k-means Based Clustering". In *Conference on Systems, Man and Cybernetics*, pp. 504–509, 2007.

[24] A. M. Fahim, A. M. Salem, F. A. Torkey, and M. A. Ramadan, "An efficient enhanced k-means clustering algorithm", *J Zhejiang Univ SCIENCE A*, 7(10), pp. 1626–1633, 2006.

[25] J. Pérez, C.E. Pires, L Balby, A. Mexicano, M. Hidalgo, "Early Classification: A New Heuristic to Improve the Classification Step of K-Means", Simpósio Brasileiro de Bancos de Dados, pp. 185–192, Brasil, 2012.

[26] V. Chukwudi, E. Femi, J. Oyelade, S. Doumbia, "Reducing the Time Requirement of k-Means Algorithm", *PLOS ONE*, 7(12), pp. 1—10, 2012.

[27] T. Hitendra, P. Viswanath, B. Eswara, "A hybrid approach to speed-up the k-means clustering method", *Int. J. Mach. Learn. & Cyber*, 4, pp. 107–117, 2013.

[28] Vijay Singh R., Bhatia M.P.S., "Data Clustering with Modified K-Means Algorithm". In *Proceedings of the International Conference on Recent Trends in Information Technology*, pp.717-721, 2011.

[29] M. de Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer, 2008.

[30] Clustering datasets. School of Computing, Univ. of Eastern Finland. Available at http://cs.joensuu.fi/sipu/datasets/. (2012).

[31] C. Merz, P. Murphy, and D. Aha. UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California. Available at http://www.ics.uci.edu/ mlearn/MLRepository.html. (2012).

Author Biographies

Joaquín Pérez Ortega was born in León, Guanajuato in 1958. He received his PhD from the Tecnológico de Monterrey (ITESM) in 1999. He is professor at the Centro Nacional de Investigación y Desarrollo Tecnológico. He is a National Researcher of Mexico (SNI), Senior Member of IEEE, and author of more than 100 publications on databases and metaheuristic optimization.

Adriana Mexicano Santoyo was born in 1981. She received her PhD from CENIDET in 2012. Currently she is professor at the Centro Nacional de Investigación y Desarrollo Tecnológico. Her research interests include algorithm analysis, optimization, and knowledge discovery.

Rodolfo Pazos received his Ph.D. degree in computer science from the University of California at Los Angeles (USA) in 1983. He is professor at the Instituto Tecnológico de Cd. Madero. His scientific interests include natural language processing and distributed database design.

René Santaolaya was born in Cuernavaca, Morelos in 1953. He received a PhD in computer science from Instituto Politécnico Nacional (IPN) in 2003. He is professor at the Centro Nacional de Investigación y Desarrollo Tecnológico. He is a Senior Member of IEEE. His scientific interests include software process models, software architectures, software reuse, service oriented architectures and web services.

Miguel Hidalgo Reyes was born in 1977. He obtained a B.Sc. and a M.Sc. from Instituto Tecnológico de Orizaba (ITO) in 2000 and 2003, respectively. Currently, he is a PhD student at the Centro Nacional de Investigación y Desarrollo Tecnológico. His research interests include data mining, ubiquitous computing and data analysis and preparation.

Nelva Almanza was born in 1984. She received her B.Sc. from the Instituto Tecnológico de Celaya in 2011. Currently she is an MSc student at the Centro Nacional de Investigación y Desarrollo Tecnológico. Her research interests include software engineering, algorithm analysis, and data mining.

Alejandra Moreno was born in 1987. She received her B.Sc. from the Instituto Tecnológico de Tapachula in 2010. Currently she is an MSc student at the Centro Nacional de Investigación y Desarrollo Tecnológico. Her research interests include software engineering, algorithm analysis, and data mining.