I.J. Modern Education and Computer Science, **2018, 7, 29-36**
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2018.07.03

An Efficient Clustering Algorithm for Spatial Datasets with Noise

Akash Nag
Department of Computer Science, M.U.C. Women's College, Burdwan, WB, India
Email: nag.akash.cs@gmail.com

Sunil Karforma
Department of Computer Science, The University of Burdwan, Burdwan, WB, India
Email: sunilkarforma@yahoo.com

Received: 16 June 2017; Accepted: 05 July 2017; Published: 08 July 2018

***Abstract*—**Clustering is the technique of finding useful patterns in a dataset by effectively grouping similar data items. It is an intense research area with many algorithms currently available, but practically most algorithms do not deal very efficiently with noise. Most real-world data are prone to containing noise due to many factors, and most algorithms, even those which claim to deal with noise, are able to detect only large deviations as noise. In this paper, we present a data-clustering method named SIDNAC, which can efficiently detect clusters of arbitrary shapes, and is almost immune to noise – a much desired feature in clustering applications. Another important feature of this algorithm is that it does not require apriori knowledge of the number of clusters – something which is seldom available.

Index Terms—Clustering, data mining, spatial datasets, noisy data.

I. Introduction

Clustering is a method of grouping objects based on the notion of similarity with respect to some given attribute. The entire population of objects is usually not homogenous and this lends itself to grouping together of objects into clusters, which are internally homogenous. This process of deriving clusters however is not always objective and two different clustering results may be viewed as being valid with respect to different measures of similarity. The most common method for measuring this similarity is some form of distance metric, computed using the attribute values of each object or data point. However, to obtain the best possible clustering, it is imperative to obtain the pairwise distance between every two such points, which is time-consuming, and often downright impractical when the dataset contains millions of points.

One of the most simple and widely used clustering algorithms is the K-Means [1] algorithm; however it suffers from the major drawback is that it requires the user to specify the number of clusters apriori – an information which is seldom available. A similar algorithm is the K-Medians [2] algorithm, and the K-Modes [3]. Another algorithm called CLARANS [4] is an improvement over the basic K-Modes algorithm. Some algorithms like GRID [5], BANG [6], CURE [7], and the Single-Link method [8] hierarchically decompose the dataset into clusters of higher-order, in turn decomposing them into smaller clusters. A popular approach is the density-based approach, wherein densities of clusters are measured, and spaces with similar densities are merged to form a single cluster. DBSCAN [9], DenClue [10], and CLIQUE [11] are some popular examples of density-based clustering. Some other popular clustering algorithms are BIRCH [12], FLAME [13], and OPTICS [14].

Complicating further the process of clustering is the presence of noise in the dataset that can adversely affect the clustering results. Most of the early clustering algorithms like K-Means are prone to this problem, while some of the modern algorithms like BIRCH can detect noise. Another problem in clustering is the density of the clusters. Many algorithms fail to obtain clusters with widely varying densities, while some like DBSCAN can handle this efficiently. Finally, not all algorithms are capable of detecting clusters of arbitrary shapes.

In this paper we propose a clustering algorithm called SIDNAC (Spatial Iterative Distance-based Noise-aware Agglomerative Clustering) that effectively addresses all of the aforesaid problems. In Section 2, we present the proposed method, and we discuss the results in Section 3.

II. The SIDNAC Algorithm

The SIDNAC algorithm proceeds in four phases which are shown in Fig. 1. The input to the algorithm is a set of points. SIDNAC is specially suited for clustering spatial datasets, but can be equally applied to any data as long as the number of dimensions is low. In this paper, we shall focus on 2-dimensional spatial data only, and hence each point will be thought of as a 2D point in the Cartesian coordinate system with *x* and *y* coordinates.

The algorithm also requires four user parameters, summarized in Table 1 that can be tweaked to obtain better results. The OVERLAP parameter informs SIDNAC whether to allow one cluster to be surrounded, but not actually overlapping, by another cluster. The other three parameters determine the number of clusters formed. It should be noted that none of these three parameters actually asks the user for the number of clusters – they only serve to tweak the results and it is up to the user to determine the desired number of clusters and hence the appropriate parameter values. We now discuss each of the four phases of the algorithm.

2.1 Initialization

Initialization is the pre-clustering step in SIDNAC, which forms initial clusters. Later steps attempt to make changes to this initial clustering to arrive at a better result. The central idea is to reduce the number of Independent Points. Independent Points are groups of points which cannot be separated and put into different clusters. In the initial dataset, all points are independent, and so, the Initialization phase attempts to reduce their number in order to bring down the total number of points to a reasonable amount.

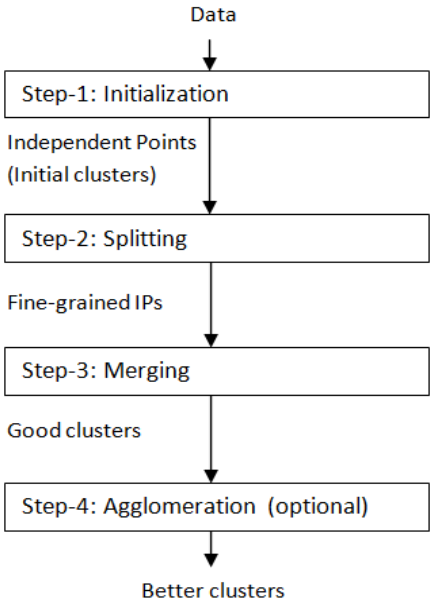

Fig.1. The phases of the SIDNAC algorithm

Table 1. User-parameters required by SIDNAC

Parameter	Default Value	Value Range	Description
CLUSTER_COUNT	1	>=1	Fine tunes the number of clusters formed. Larger the value, lesser is the no. of clusters
OVERLAP	False	True/False	Whether the dataset contains overlapping clusters or not
PROXIMITY_INDEX	2	>=1	Fine tunes the cluster size (in terms of area). Larger the value, lesser is the no. of clusters
MERGE_ORDER	1	>0	Fine tunes the merging process. Larger the value, lesser is the number of clusters formed

Each point is thought to have an Aura of Influence (AOI), initially zero. The Granularity (G) of the dataset is the distance between the closest 2 points in the data space, and is defined as follows, for all pairs of points *a* and *b*:

$$G = \min\left(\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}\right) \quad \forall\, a \neq b \quad (1)$$

Distance, here, refers to the Euclidean distance between 2 points. The AOI of each point is incremented in steps of G. After each such increment, if the sum of the AOIs of 2 points is found to be greater or equal to the distance between them, then the pair is merged to form a new cluster. Only those pairs are compared in which at least one of them has not been assigned to any cluster.

The Initialization phase proceeds as follows:

1. Determine the Granularity G of the dataset.
2. For each point A that has not been assigned to any cluster, repeat steps 3 and 4 until all points have been assigned.
3. Increment the AOI of A by G units.
4. For each point B in the dataset, such that A≠B, do the following:
    a. Compute the distance D between A and B.
    b. Compute the sum S of the AOI of A and AOI of B.
    c. If S exceeds D, then:
        i. If B has already been assigned to a cluster, add A to that cluster.
        ii. If B remains unassigned, create a new cluster C, and assign both A and B to this cluster.

The working of this phase can be understood from Fig. 2. At the end of the initialization phase, we are left with some initial clusters, which form the input to the next phase of the algorithm.

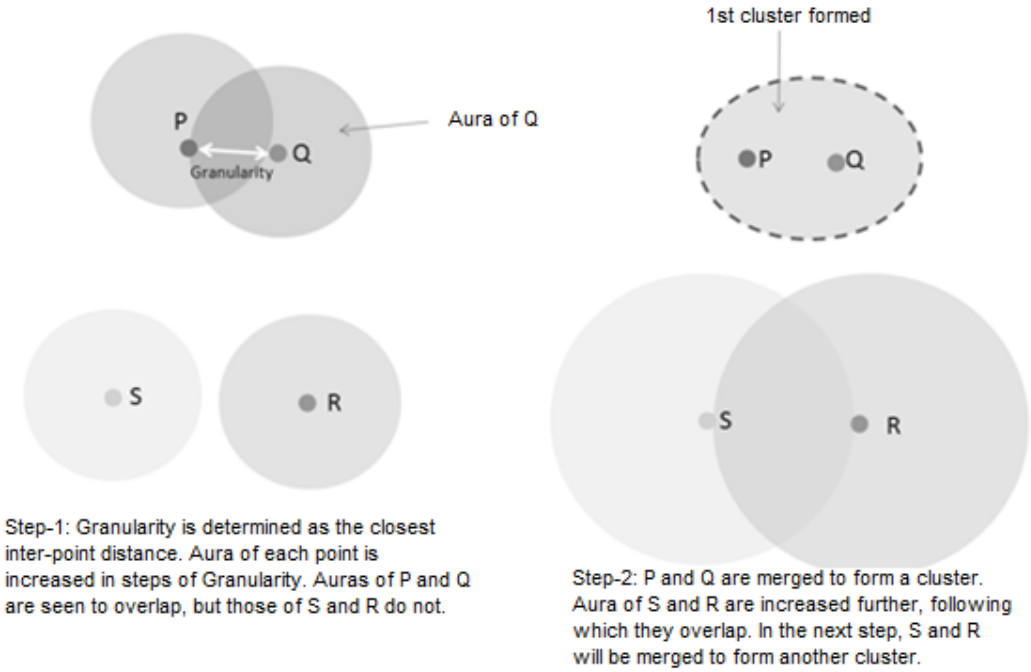

Fig.2. The Initialization phase of SIDNAC

2.2 Splitting

Splitting improves on the initial clustering, to provide a clean sand-bed for the subsequent steps, ensuring that no incorrect cluster assignments have been made in the initialization step, and breaking up the dataset into the finest grains possible, so that the clusters produced after this step would be a true representative sample of the original dataset, but with far fewer points than there was initially. This phase is also necessary to remove the data-order-sensitivity in the clusters that is inherent in the initialization phase due to the fact that the clustering proceeded in the same order in which the data was scanned.

The splitting phase is itself subdivided into 2 sub-steps:
1. Distance-based Splitting
2. ACS-based Splitting

The above steps are performed sequentially as shown in Fig. 3, with each step repeated for every cluster.

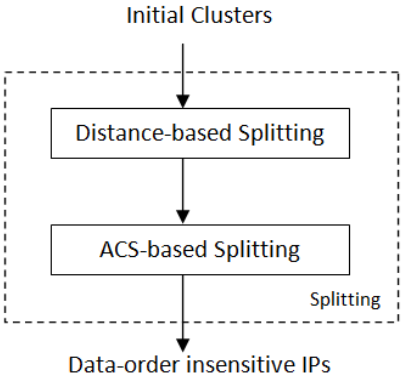

Fig.3. Splitting phase of SIDNAC

2.2.1 Distance-based Splitting

In this step, we split the initial clusters based on the fact that each member's closest neighbor should be within its own cluster. Supposing that there is a Cluster C with 2 members A and B, such that B is the closest point to A in that cluster. Now, if A is closer to a point P (belonging to a different cluster D) than to B, then A is split, i.e. taken out from Cluster C and assigned to a new cluster of its own, as shown in Fig. 4.

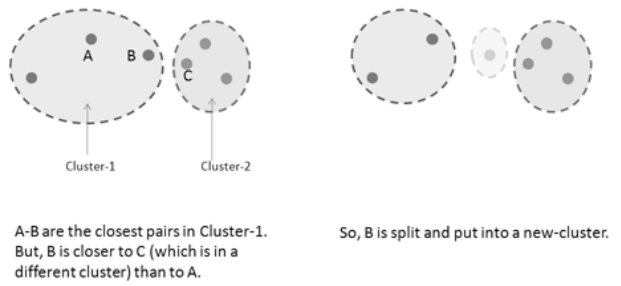

Fig.4. Distance-based Splitting

The algorithm works as follows:

1. For each cluster C do the following:
   a. For each point A in C, do the following:
i. Find a point B which is closest to A in C
ii. Find a point P which is closest to A in the entire dataset
iii. If B ≠ P, then create a new cluster D, and assign A to D

2.2.2 ACS-based Splitting

In this step, the Average Cluster Size (ACS) is determined. The ACS is the average of the cluster-sizes of all clusters; where the size of a cluster C is the maximum separation of any 2 members of C. The ACS is then determined using the following formula:

$$ACS = \frac{1}{N}\left[\sum_{i=1}^{N}\left\{\max_{j=1}^{n(C^i)}\left(\sqrt{\left(A_x^{i,j}-B_x^{i,j}\right)^2+\left(A_y^{i,j}-B_y^{i,j}\right)^2}\right)\right\}\right] \quad (2)$$

Where, N is the number of clusters obtained till now, $n(C^i)$ is the number of points belonging to the i-th cluster, and A and B are points belonging to the i-th cluster.

After ACS is determined, we iterate through each point A belonging to a cluster C, and determine the point B such that B is the closest point to A in cluster C. Let the distance between A and B be D. if D is more than ACS by a certain factor, then A is assigned to a new cluster. This factor is called the Proximity Index, and is a user-specified parameter to the algorithm. This process is repeated for every cluster.

The algorithm works as follows:

1. Determine ACS
2. For each cluster C, do the following:
   a. For each point A in C, do the following:
   i. Determine a point B which is closest to A in C
   ii. Calculate D to be the distance between A and B
   iii. If D > (ACS * PROXIMITY_INDEX) then create a new cluster and assign A to this new cluster

2.3 Merging

The merging process starts the clustering proper, by examining the inter-cluster distances. Two clusters are merged if their inter-cluster distance is less than the Average Cluster Size, tuned appropriately by a certain factor. This factor is called the Merge Order and is user-specified.

The Inter-Cluster-Distance (ICD) between any two clusters A and B, is the minimum distance between any 2 points belonging to different clusters, and is computed as follows:

$$ICD(A,B) = \min\left(\sqrt{(a_x-b_x)^2+(a_y-b_y)^2}\right) \forall a \in A, b \in B \quad (3)$$

The Merging algorithm works as follows:
1. Determine ACS
2. For each cluster P, do the following:
   a. For each cluster Q, such that P≠Q, do the following:
      i. D = ICD(P,Q)
      ii. If D < (ACS * MERGE_ORDER) then Merge Clusters P and Q

2.4 Agglomeration

The clusters formed in the previous phase are good, but not accurate. The Agglomeration phase merges related clusters into still larger groups. For each cluster, its neighbours are determined, and the distances between them are computed. Usually, if this distance is more than the Average Inter-Cluster Distance, intuitively they should not be merged. However, in this case we are ignoring the fact regarding the density (or rather, rarity) of each cluster. Most other density-based clustering algorithms assume that all clusters will have more or less the same density, and hence, these algorithms fail to detect clusters with largely varying densities. To overcome this problem, this agglomeration step is essential.

The process of agglomeration merges 2 clusters A and B, if all the following conditions hold:

1. A and B are neighbors, i.e. NEIGHBOR(A,B)=TRUE
2. NEIGHBORHOOD(A,B) > AICD
3. NEIGHBORHOOD(A,B) < (RARITY(A) * CLUSTER_COUNT)
4. NEIGHBORHOOD(A,B) < (RARITY(B) * CLUSTER_COUNT)

The Agglomeration proceeds by first determining all Neighbors and their Neighborhoods within the dataset. Then, it iterates over each pair of clusters A and B, and determines their Rarity, followed by applying all the 4 rules on them. If all the conditions are satisfied, A & B are merged together. We now explain how neighbor and rarity are computed.

2.4.1 Neighbours and Neighbourhood of Clusters

Two Clusters A and B are neighbours if both the following conditions hold, for each point P in cluster A:

1. P is closest to Q (in cluster B) than to any other point in cluster B
2. There is no line RS (where R and S belong to a cluster C, such that A≠C and B≠C) intersecting the line PQ

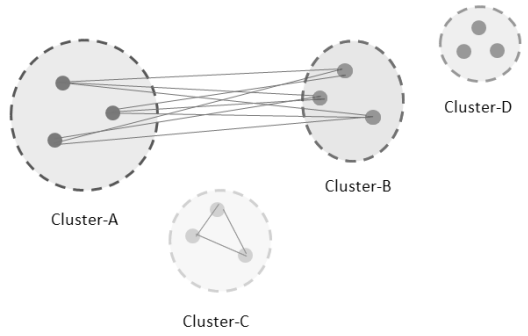

Fig.5. Determination of cluster neighbours

In Fig. 5, we see how neighbours are detected. To determine if A and B are neighbors, we create their Inter-Cluster Mesh by joining each point of A with each point of B. Next, we iterate through every other cluster, and for each such cluster, we create its Intra-Cluster Mesh by joining each point in that cluster with every other point in the same cluster. If none of these Intra-Cluster Meshes intersect with the Inter-Cluster Mesh of A & B, then A and B are said to be neighbors. In the figure, A and B are neighbors because the Intra-Cluster Meshes of C and D do not intersect with the Inter-Cluster Mesh of A & B. If A and B are found to be Neighbors, then their Inter-Cluster Distance is said to be the Neighborhood of A with respect to B, or vice-versa.

The Neighbor Determination algorithm works as follows:

1. For each cluster A, do the following:
   a. For each cluster B, such that A≠B, do the following:
      i. Set D = ICD(A,B) and N=TRUE
      ii. Initialize ICM(A,B) to ϕ
      iii. For each point P in A, do the following:
         1. For each point Q in A, such that P≠Q, add PQ to ICM(A,B)
      iv. For each cluster C, such that A≠C and B≠C, do the following:
         1. Initialize CM(C) to ϕ
         2. For each point P in C, do the following:
            a. For each point Q in C, such that P≠Q, add PQ to CM(C)
         3. For each line PQ in ICM(A,B), do the following:
            a. For each line MN in CM(C), do the following:
               i. If PQ and MN intersect, set N=FALSE and continue outer loop
      v. If N=TRUE, do the following:
         1. Set NEIGHBOR(A,B)=TRUE
         2. Set NEIGHBORHOOD(A,B)=D

2.4.2 Closest Neighbour Distance

The Closest Neighbor Distance of a point A in cluster C is the distance between A and B where B is the closest point to A, in cluster C. B is then said to be the neighbor of A. In other words, A and B are neighbors if the following conditions hold:

1. A and B both belong to the same cluster
2. There is no other point P, such that A is closer to P than to A

CND is calculated as follows:

$$CND(a,C) = \min\left(\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}\right) \forall b \in C, a \neq b \quad (4)$$

2.4.3 Rarity

The Rarity of a Cluster C is defined to be the maximum distance between any 2 member points of that cluster, who are neighbors themselves. Rarity of a cluster is computed as follows:

$$RARITY(C) = \max(CND(a,C)) \forall a \in C \quad (5)$$

2.4.4 Average Inter-Cluster Distance

The Average Inter-Cluster Distance (AICD) is the Average of the Inter-Cluster Distances of every pair of clusters. AICD is computed as follows:

$$AICD = \frac{1}{2N} \sum ICD(A,B) \forall A, B; A \neq B \quad (6)$$

III. Results and Discussion

The SIDNAC algorithm was implemented using Java and tested against some artificial data, the results of which are presented in Fig. 6 and Table 2. Similar datasets were fed into DBSCAN [9] and the results thus obtained are presented in Fig. 7.

From Fig. 6 and Fig. 7, we can see the clustering results of various datasets after having SIDNAC and DBSCAN applied to them respectively (for similar data). Each of the results shown, signify an important aspect of the proposed algorithm, and we now compare the results with those obtained using DBSCAN:

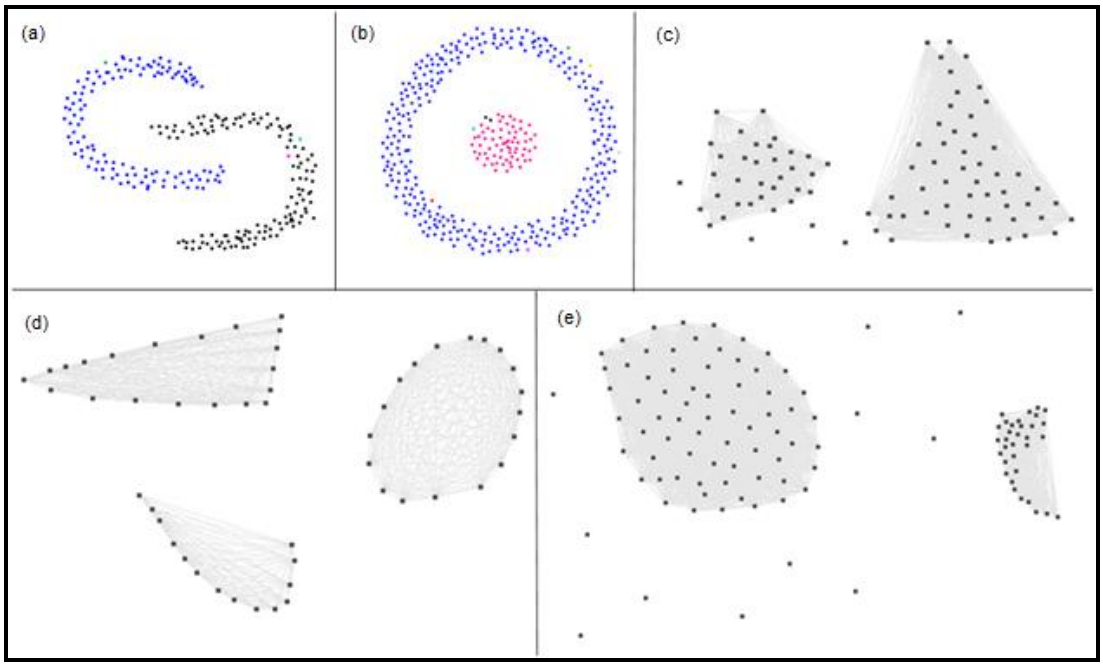

Fig.6. (a-e) Clustering results of SIDNAC

1. In Fig. 6(a), we can see how efficiently SIDNAC detects clusters that partially overlap. Similar capability is also demonstrated by DBSCAN as shown in Fig. 7(a).
2. In Fig. 6(b), we can see SIDNAC's ability to detect clusters even when one cluster is totally contained within another, and DBSCAN is also at par in this respect as shown in Fig. 7(b).
3. In Fig. 6(c), we show SIDNAC's ability to detect clusters of arbitrary shapes. However, as we can see in Fig. 7(c), DBSCAN fails to properly cluster them. The left group is clustered into 4 clusters (orange, green, red, and mauve) by DBSCAN apart from classifying some members as outliers (shown in blue). The right most group is also clustered into two separate groups.
4. In Fig. 6(d), we show how SIDNAC can detect clusters having no cores, and DBSCAN's ability is also at par with it as shown in Fig. 7(d).
5. In Fig. 6(e), we show the outlier detection capability as well as SIDNAC's ability to detect clusters with varying densities. The cluster on the right is far denser than the one on the left. In Fig. 7(e), as expected, we can see that DBSCAN fails to detect clusters having wide variation in density in the same dataset. Most of the points in the left most group has been misclassified as outliers.

Table 2. Classification accuracy comparison

Dataset	Accuracy	
	SIDNAC	DBSCAN
(a)	93.17%	100%
(b)	95.64%	100%
(c)	84.71%	73.29%
(d)	100%	100%
(e)	100%	43.12%

We can therefore conclude that SIDNAC is superior to DBSCAN in many respects. The primary disadvantages of SIDNAC however, lies in its speed, as it has a worst-case time complexity of $O(n^2)$ when the Agglomeration step is not performed, and $O(n^4)$ when all steps are performed. It also has high I/O costs as it requires multiple passes of the dataset. Finally, it is expensive to extend this algorithm to work with data having more than 3 dimensions. However, it has several advantages that are seldom absent in any given single clustering algorithm such as:

1. **Efficient Noise Detection:** K-Means and other algorithms don't have any noise/outlier detection capability. But, SIDNAC efficiently detects outliers, resulting in accurate clustering results. This is a very important feature as most real-world data contain noise.
2. **Highly Customizable:** SIDNAC has 4 user-specified parameters, that help tune the cluster-results as per requirements and pre-existing knowledge about the dataset. K-Means and some other algorithms take the number of clusters as the only parameter, which is not a very good tune-up factor.
3. **Intuitive Parameters with Default values:** All of SIDNAC's parameters are intuitive and do not adversely affect clustering results. E.g. Supplying K=3 in K-Means where there clearly are 4 clusters, results in 3 clusters being formed. However, this disadvantage is not present in SIDNAC, and every cluster result that it produces is accurate to some extent at a minimum. Further, all its parameters have default values, which means that unlike K-Means where failure to supply the value of K will not produce any result, in SIDNAC, failure to specify the value of any

parameter makes SIDNAC take that parameter's default value and proceed to clustering.

4. **Arbitrarily Shaped Cluster Detection:** Most algorithms like BIRCH, can detect clusters only when they are spherical in shape to some extent. This is because of their inherent use of the concept of Radius-neighborhoods. However, SIDNAC can detect clusters of any imaginable shape, which is a very important factor.
5. **Immune to high-variance in densities:** Most density-based algorithms like DBSCAN, fail to detect clusters having widely different densities, in the same dataset. However, SIDNAC is immune to this effect.
6. **Data-Order Insensitivity:** Most algorithms like CLARANS, produce clustering results which are highly dependent on the scan order of the data-points. If the relative order of the points are changed, the results vary. However, SIDNAC is insensitive to data-order, and produce deterministic results each time.
7. **Pre-knowledge of number of clusters not required:** Algorithms like K-Means and K-Medians require the user to supply the number of clusters. However, this number is seldom known. SIDNAC does not require such a parameter.
8. **Neighborhood and Cluster-Sizes not required:** Algorithms like DBSCAN and OPTICS require the neighborhood and minimum cluster size (in terms of number of points). However, SIDNAC does not require such difficult-to-determine parameters to be specified by the user.

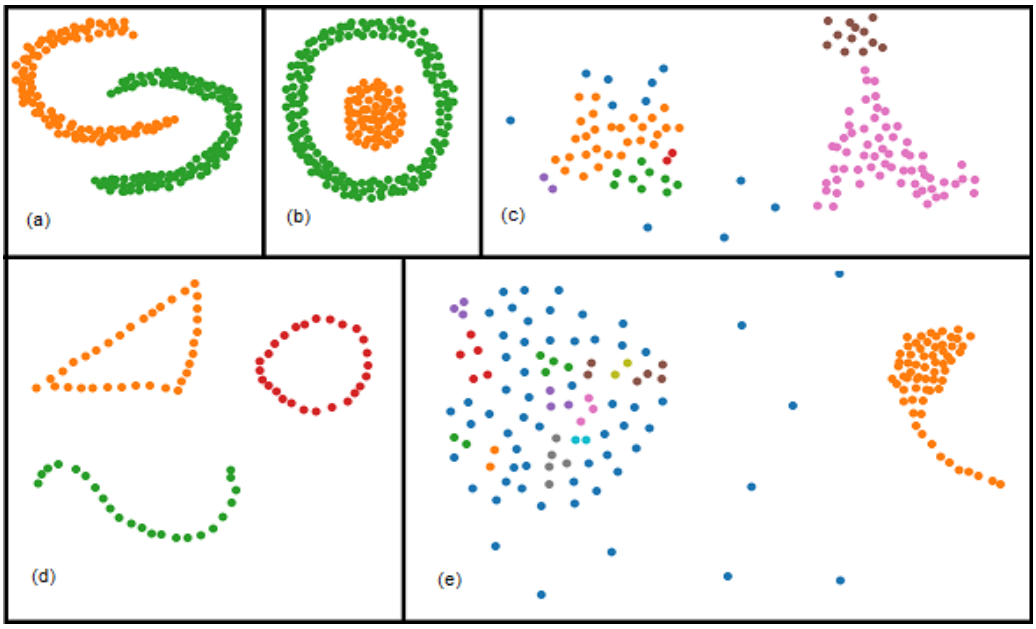

Fig.7. (a-e) Clustering results of DBSCAN

IV. Conclusion

The SIDNAC algorithm has been presented here in its most basic form. The time complexity of this algorithm can be improved using proper data-structures and indexing to store and organize the data. It must be assumed that the memory may not be able to store the entire dataset at once, and I/O costs are quite high if multiple scans are required for the entire dataset and it is not stored in memory. During the implementation of SIDNAC, our goals would be to eliminate these drawbacks. Extension of SIDNAC to work with high dimensional data, though important, is not so important in spatial clustering – the class to which SIDNAC belongs, and is not what we are trying to achieve with SIDNAC. However, if we need to deal with high-dimensional data, we can always use feature extraction, or any other dimensionality reduction techniques before passing the dataset to SIDNAC. The distance function of SIDNAC also lends itself to change, if so desired, replacing the Euclidean distance metric with some other measure.

We can state that SIDNAC can be used efficiently in situations where the following are of paramount importance: accuracy and, noise & outlier detection. SIDNAC is most applicable to spatial datasets but can be used successfully in other types of data sets as well. In comparison to the existing density/distance based algorithms SIDNAC's accuracy in detecting noise as well as detecting clusters exhibiting high density variances is unmatched.

References

[1] MacQueen, James. "Some methods for classification and analysis of multivariate observations." Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 14. 1967.

[2] Kaufman, Leonard, and Peter J. Rousseeuw. Finding groups in data: an introduction to cluster analysis. Vol. 344. John Wiley & Sons, 2009.

[3] Huang, Zhexue. "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining." DMKD. 1997.
[4] Ng, Raymond T., and Jiawei Han. "Efficient and Effective Clustering Methods for Spatial Data Mining." Proc. of. 1994.
[5] Schikuta, Erich. "Grid-clustering: An efficient hierarchical clustering method for very large data sets." Pattern Recognition, 1996., Proceedings of the 13th International Conference On. Vol. 2. IEEE, 1996.
[6] Schikuta, Erich, and Martin Erhart. "The BANG-clustering system: Grid-based data analysis." International Symposium on Intelligent Data Analysis. Springer Berlin Heidelberg, 1997.
[7] Guha, Sudipto, Rajeev Rastogi, and Kyuseok Shim. "CURE: an efficient clustering algorithm for large databases." ACM Sigmod Record. Vol. 27. No. 2. ACM, 1998.
[8] Sibson, Robin. "SLINK: an optimally efficient algorithm for the single-link cluster method." The computer journal 16.1 (1973): 30-34.
[9] Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd. Vol. 96. No. 34. 1996.
[10] Hinneburg, Alexander, and Daniel A. Keim. "An efficient approach to clustering in large multimedia databases with noise." KDD. Vol. 98. 1998.
[11] Agrawal, Rakesh, et al. Automatic subspace clustering of high dimensional data for data mining applications. Vol. 27. No. 2. ACM, 1998.
[12] Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases." ACM Sigmod Record. Vol. 25. No. 2. ACM, 1996.
[13] Fu, Limin, and Enzo Medico. "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data." BMC bioinformatics 8.1 (2007): 3.
[14] Ankerst, Mihael, et al. "OPTICS: ordering points to identify the clustering structure." ACM Sigmod record. Vol. 28. No. 2. ACM, 1999.

Authors' Profiles

Akash Nag completed his Bachelors in Computer Applications from the University of Burdwan, and his Masters in Computer Science from the University of Calcutta. He received his Ph.D. in Computer Science from the University of Burdwan. He is currently a faculty member in the Dept. of Computer Science at M.U.C. Women's College, Burdwan. His research interests include algorithms and bioinformatics.

Prof. Sunil Karforma completed his Bachelors in Computer Science & Engineering, and his Masters in Computer Science & Engineering, from Jadavpur University. He received his Ph. D. in Computer Science, and is presently Professor & Head of the Dept. of Computer Science at the University of Burdwan. His research interests include Network Security, E-Commerce, and Bioinformatics. He has published numerous papers in both national as well as international journals and conferences.

How to cite this paper: Akash Nag, Sunil Karforma, " An Efficient Clustering Algorithm for Spatial Datasets with Noise", International Journal of Modern Education and Computer Science(IJMECS), Vol.10, No.7, pp. 29-36, 2018.DOI: 10.5815/ijmecs.2018.07.03

