I.J. Intelligent Systems and Applications, **2012, 2, 42-48**
Published Online March 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2012.02.05

Finding Within Cluster Dense Regions Using Distance Based Technique

Wesam Ashour
Computer Engineering Department, Islamic University of Gaza, Gaza, Palestine
Email: washour@iugaza.edu.ps

Motaz Murtaja
Computer Engineering Department, Islamic University of Gaza, Gaza, Palestine
Email: mou8taz@gmail.com

Abstract— One of the main categories in Data Clustering is density based clustering. Density based clustering techniques like DBSCAN are attractive because they can find arbitrary shaped clusters along with noisy outlier. The main weakness of the traditional density based algorithms like DBSCAN is clustering the different density level data sets. DBSCAN calculations done according to given parameters applied to all points in a data set, while densities of the data set clusters may be totally different. The proposed algorithm overcomes this weakness of the traditional density based algorithms. The algorithm starts with partitioning the data within a cluster to units based on a user parameter and compute the density for each unit separately. Consequently, the algorithm compares the results and merges neighboring units with closer approximate density values to become a new cluster. The experimental results of the simulation show that the proposed algorithm gives good results in finding clusters for different density cluster data set.

Index Terms—**data clustering, density-based, DBSCAN, glory point, different density.**

I. INTRODUCTION

Clustering is an important tool for data mining since it can identify major patterns of data without any supervisory information such as data labels [1]. It can be broadly defined as the art of finding groups of similar objects (database records, nodes in a graph, words, images, or any collection in which individuals are described by a set of features or distinguishing relationships) in large data sets without having specified these groups by means of explicit features and these groups represent a meaningful sub-population.

Finding clusters in a data set is not an easy process when the clusters are of widely different sizes, arbitrary shapes, different densities, and existence of noise data points. Many algorithms proposed to solve clustering problems. Clustering algorithms can be classified into five main categories: density-based, partitioning or distance-based, hierarchical, grid-based and model-based [1]. The two prominent categories are distance-based and density-based.

Partitioning algorithms finds k clusters for a given dataset with n data points that are mainly based on distances between data points so that data points in each cluster is close or relevant to other points in the same cluster. K-means is the most well known partitioning method. In k-means [1] the first step is to selects k locations randomly to represent initial centroids of the clusters. For each of the remaining objects, each object is assigned to the most similar cluster. This similarity is based on the distance between the object and the cluster mean (i.e. assigned to the nearest centroid). Afterwards, the algorithm computes the new mean for each cluster. This process iterates until the algorithm converges.

Density Based algorithm can classify data points in to clusters based on density notation. Where the cluster is continually growing while the number of data points in the neighborhood exceeds some threshold [1] [2]

DBSCAN is a typical density-based algorithm. DBSCAN tries to find clusters depending on user defined parameters for density threshold, neighborhood radius (*eps*) and threshold number of points in *eps*–neighborhood (*MinPts*), and then find the object that satisfy the *MinPts* threshold in the given radius *eps*.

K-means is fast, easy to implement, and converges to local optima almost surely, however, it is largely affected by noise [3]. On the other hand, while density-based clustering can find arbitrary shape clusters and handle noise well, it is also slow in comparison due to neighborhood search for each data point, faces difficulty in setting density threshold properly, and the main weakness is when the algorithm deals with different density level data set.

Clustering of the data has been applied in different fields. In engineering and computer science (machine learning, pattern recognition, web mining, spatial database analysis, information retrieval, network intrusion detection), in economics for applications in customer characteristics and purchasing (marketing, business), also it is widely used in medical sciences, astronomy and earth sciences[1][4].

This paper proposes a new algorithm that tries to find the dense region within a cluster by partitioning core points into units and computes the dense factor as an indicator to the unit density. The dense factor is the number of core points in the unit divided by the distance between the unit's mean and furthest core, then merging neighboring units with closer dense factor to produce a new cluster.

The rest of this paper is organized as follows: Section II introduces related work for this paper. Section III proposes and discusses the new algorithm. Section IV presents experimental results and analysis. Finally section V and Section VI are the conclusion and the future work.

II. RELATED WORK

Multi-target track initiation is primary problem of Clusters may be described as continuous regions of this space containing a relatively high density of points, separated from other such regions by regions containing a relatively low density of points; this is the fact that Density-based Clustering is found. DBSCAN [5] (Density Based Spatial Clustering of Applications with Noise) algorithm is a classic density-based clustering algorithm, The basic idea of DBSCAN algorithm involve a number of definitions:[1][2]

Definition 1(*eps*-neighborhood) The *eps*-neighborhood of a point x in data set D is

$$N_\varepsilon(x) = \{y \in D : d(x,y) \leq eps\}$$

Definition 2 (Core Point) If the *eps-neighborhood* of a point contains at least a minimum number, *MinPts*, of points, then the object is called a core point.

Definition 3 (Directly Density reachable) We say that a point x is directly density-reachable from point y if x is within the *eps-neighborhood* of y, and y is a core point. Directly density-reachable is symmetric for pairs of core points (points inside a cluster), but it is in general not symmetric in case one core point and one border point (a point on the border of a cluster) are involved

Definition 4 (Density-reachable). A point x is said to be density-reachable from a point y if there is a sequence of points $x=x_1$, x_2 ,..., x_i = y such that x_l is directly density-reachable from x_{l+1} for l = 1, 2,...,i−1.

Definition 5 (Density-connected). Two points x and y are said to be density-connected with respect to *eps* and *MinPts* in a set of objects D, if there exists a point z such that both x and y are density-reachable from z.

Then we can defined the cluster as a set of density-connected points that is maximal with respect to density-reachability. And mathematically : A cluster C with respect to to *eps* and *MinPts* in data set D, is a nonempty subset of D satisfying :

1. $\forall x, y \in D, if\ x \in C$ and y is density-reachable with respect to ε and *MinPts* then $y \in c$
2. $\forall x, y \in C,$ x and y are density-connected with respect to ε and *MinPts*.

The result of DBSCAN algorithm relies on the parameters that are set by the user (*eps* and *MinPts*). On the same database with different parameters, the algorithm can produce different clusters. A very high value of density threshold will produce many small and very dense clusters, while a very small value will generate few clusters or just one cluster consisting of all data points. Therefore, users have to choose the proper parameters to produce the best result. However, DBSCAN algorithm is capable of finding arbitrary shape clusters, and handling noise well, but is not suitable for discovering clusters with different densities.

To sweep over the limitations and the problems of DBSCAN many algorithms has been proposed such as DENCLUE and OPTICS…etc.

OPTICS [6] (Ordering Points to Identify the Clustering Structure) is an improved method upon DBSCAN, which is practically the same in runtime and process, but represents the clusters of objects by the ordering of objects in the database. Optics display the point-order information based on densities of the clusters and it do good in different density data sets, but OPTICS is weak at finding out information of clusters in sparse datasets though it is good at finding them in dense areas.

DENCLUE [7] (DENsity-based CLUstEring) introduces the idea of an influence function that describes the impact of a data point upon its neighborhood. DENCLUE based on a set of density distribution functions, uses grids which are very efficient since it only keeps information about grids that do actually contain data objects and manages these grids in a tree-based access structure. A limitation of the algorithm is that it also requires selection of parameters *eps* and *MinPts.*

Another recent research presents VDBSCAN [8]. The basic idea of VDBSCAN is that before adopting traditional DBSCAN algorithm, some methods are used to select several values of parameter *eps* for different densities according to a k-dist plot. Firstly, VDBSCAN calculates and stores k-dist for each project and partition k-dist plots. Secondly, the number of densities is given intuitively by k-dist plot. Thirdly, choose parameters *epsi* automatically for each density. Fourthly scan the dataset and cluster different densities using corresponding *epsi*. And finally, displays the valid clusters corresponding with varied densities.

Another optimization for VDBSCAN is k-VDBSCAN [9] in which it introduces a new method to find out the value of the parameter k automatically based on the characteristics of the datasets. This method considers spatial distance from a point to all others points in the data sets. Figure 1 shows sample k-dist plot.

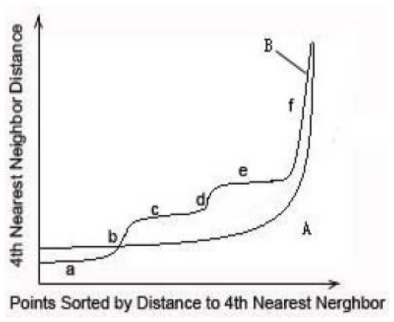

Figure 1 K-dist plot

BRIDGE [10] is new algorithm that merges DBSCAN and K-means algorithms; it is exploiting the advantages of one to counter the limitations of the other and vice versa. For density-based clustering – improving the speed and estimating a probable density threshold, and improves the quality of K-means clusters by removing the noisy points. BRIDGE is of multi-phase: first K-means is performed to partition the data into k number of clusters, and then density-based clustering is done on each partition to find dense clusters.

CCDD [4] is a new clustering algorithm based on characteristics of density distribution. It firstly divides data space into a number of grids. Secondly, it re-divides data space into many smaller partitions, according to each grid's one-dimensional or multi-dimensional characteristics of density distribution. And the final step uses an improved DBSCAN algorithm, which chooses different parameters according to each partition's local density, to cluster respectively.

III. PROPOSED SOLUTION

This section presents the proposed algorithm in two phases; partitioning phase and merging phase. The following definitions are used in this paper:

A. *Definitions*

Definition 1 (*NumCore*) It is the number of core points in each unit; it is user specified value.
Definition 2 (*Unit*) Each *NumCore* points will produce new unit.
Definition 3 (*unit mean*) The mean point between all core points belonging to one unit.
Definition 4 (*max*) It is the distance value from unit mean to the farthest core point in the same unit.
Definition 5 (*avg)* It is the average distance from mean to all cores points in the unit.
Definition 6 (*glory point*) Any point in the unit that fails between (*max- eps*) to max *distance*.
Definition 7 (*eps, MinPts*) The same parameters value used in original DBSCAN algorithm.

B. *Partitioning Phase*

The proposed solution depends on finding the density of a given area; the algorithm tries to split the given clusters resulting by DBSCAN into smaller units of **core regions** as illustrated in Figure 2. Figure 2 shows the partitioning of cluster C into smaller units. For each unit *dfactor* value is calculated by dividing *NumCore* by *max* value. The *dfactor* value will be used as an indication to the unit density. The *dfactor* values are calculated for all cluster units with the same process.

Another way to calculate the dfactor depends on estimating the volume of the unit (assume the shape to be circle, sphere, or hypersphere) with radius equal *max* or *avg*; then divide the volume by the *NumCore* as an indication to the unit density.

NumCore parameter value is too critical. Such that small values will produce smaller units and more computation with higher performance and precision and large values need less computation with degradation in performance.

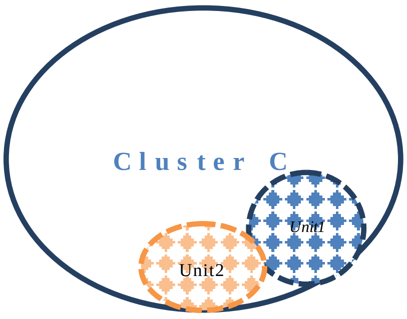

Figure 2 Partitioning Phase
partition cluster c into smaller units based on the number of core points (NumCore)

C. *Merging Phase*

Merging phase consists of two steps:

First Step: Determines neighboring units, the proposed algorithm considers two units to be neighbors if the number of links between the two units is at least equal to or greater than neighborhood threshold. Links between two units are increased by one unit for each two glory points belonging to different units, with in-between distance less than *eps*. Figure 3 shows glory points that fail between *max-eps* to *max* distance value. Determining neighborhood threshold value is done with relevance to *MinPts* parameter in the original DBSCAN algorithm, so that at minimum it is larger than *MinPts*.

Second Step: In this step, the *dfactor* value is calculated for each two units with some threshold, while checking the neighborhood between them. Afterwards the neighboring units are merged based on close *dfactor* values in the same cluster.

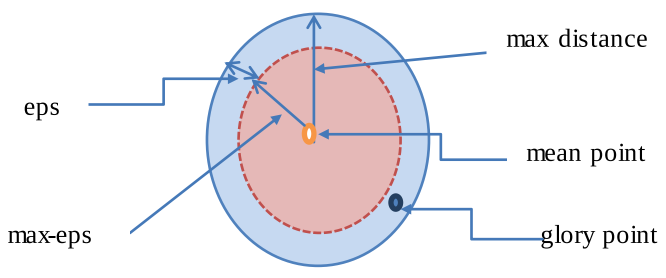

Figure 3 shows unit mean point , glory point lies between max distance and max-eps

In partitioning phase and as a modification, the same method of estimating the area of the cluster in [3] can be used. For a cluster, the total volume is the volume of the smallest size hyper-rectangle that circumscribes all points inside the cluster. The hyper-rectangle is determined by finding the minimum and the maximum of each attribute for the cluster. Dividing this volume by the number of cores in the cluster helps in determining the initial value for *NumCore* parameter and achieves better results per cluster.

Another important thing to say is that the distance based calculations were used in the previously proposed algorithm instead of depending on density for determining the local clusters. Partitioning phase is inspired from grid-based clustering with some differences. Grid-based clustering partitions the data into grids with the same parameters or dimensions. But in this case, the units take different shapes depending on the distribution of core points in that unit. Another difference is the regular shapes of grid (e.g. rectangular), and our proposed algorithm, the furthest point from the center or any other indications to this shape.

```
Proposed Algorithm
Input: Cluster C ,NumCore
Output: K clusters
Partitioning Phase:
1. Cnt=number of core points in c
2. Initiate Cnt/NumCore units
3. Fill each unit with NumCore of core points
4. For each unit
5.    calculate mean according to the equation ;
          $m_u = \frac{1}{n}\sum_{x_i \in u} x_i$
6.   find max distance to the farthest core  where
          max = $max_{x_i \in u}\ d(x_i, m_u)$
7.    find glory points ;
        if $eps < d(x_i, m_u) < max$ then
        add this point to List glory
8.    calculate dfactor ;
          dfactor= NumCore/max
end for
```

```
Merging Phase:
9. For each two units u1,u2
   10.Calculate distance from each point in u1.glory to each point in u2.glory
   11. If distance <eps then Links++
   12. If Links <= threshold then
   u1 and u2 is neighbors
end for

13. For each two units u1,u2
   14. If u1.dfactor ≅ u2.dfactor && u1.isNeighbor(u2)
   15.Put u1.points and u2.points in the same cluster
end for

16. Append each border point to closest cluster
```

Figure 4 Proposed Algorithm, consist of two phases: partitioning phase and merging phase

IV. Experimental Evaluation

Java language is used to implement the proposed algorithm. The first data set that has been used in Figure 5 and Figure 6 has been generated to contain 150 data point spread in three clusters. Applying DBSCAN to this data set with *eps=2* and *MinPts=4*, produces two clusters. One of them is one of the original 3 clusters, and the other one is a merged version of the other 2 original clusters. This merging is due to the different densities between the original 2 clusters as shown in Figure 5. Figure 5 shows result of DBSCAN, the first discovered cluster is shown in blue and the other shown in red.

Applying the proposed algorithm to each cluster resulted from DBSCAN with parameters *NumCore =10*, finds the other 2 clusters that DBSCAN fails to find. Figure 6 shows the result of the proposed algorithm when finding the 2 dense areas that represent the clusters. The discovered clusters are shown in red, blue and yellow.

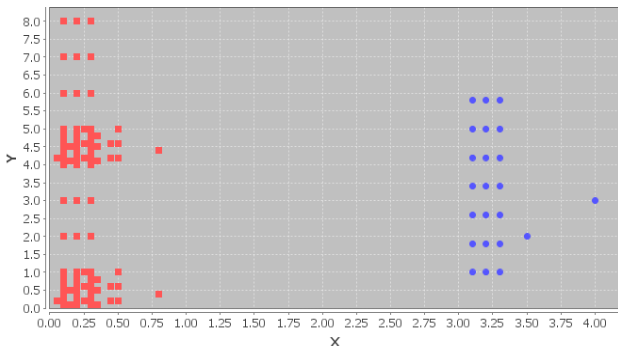

Figure 5 DBSCAN finds two clusters eps=2 and minpts=4

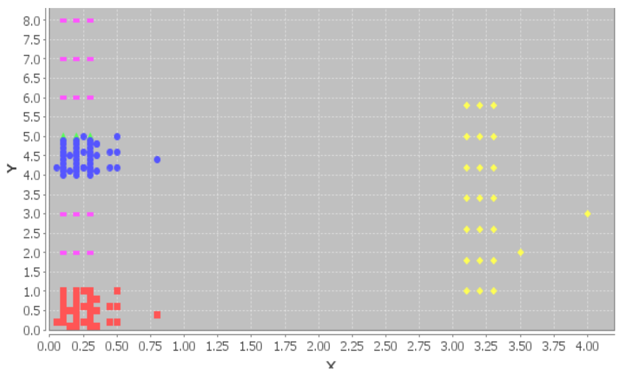

Figure 6 Our algorithm result when NumCore=10; it finds two clusters in cluster 1

The second data set contains 1200 point spread in 14 clusters. Applying DBSCAN to this data set with *eps=3.5* and *MinPts=5* produces 4 large clusters shown in Figure 7 by red, blue, green and light blue; none of these clusters are correct in relative to the original clusters. DBSCAN fails to find small dense areas that represent clusters because of difference in density level. Applying the proposed algorithm to each cluster resulted from DBSCAN with parameters *NumCore=30* and small neighborhood threshold, finds 8 clusters of which 5 are of the original clusters while the others are merged. The proposed algorithm is applied with different parameters *NumCore=20* and large neighborhood threshold, finds 9 of the original clusters while the other clusters are merged. Figure 8 shows the result of the proposed algorithm which finds 8 clusters with *NumCore=30*. Figure 9 also shows the result of the proposed algorithm that finds 9 correct clusters with *NumCore=20*.

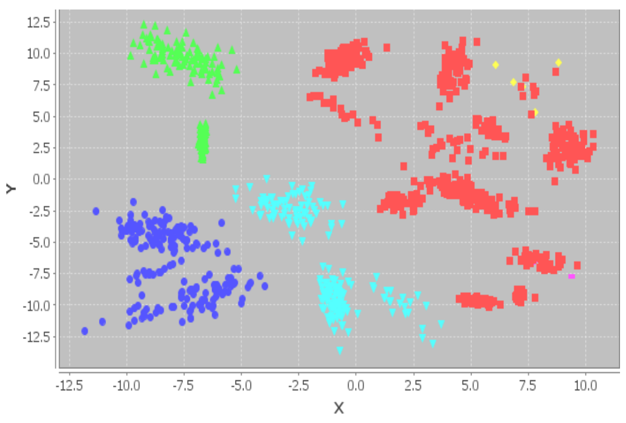

Figure 7 DBSCAN with eps=3.5 and minpts=5

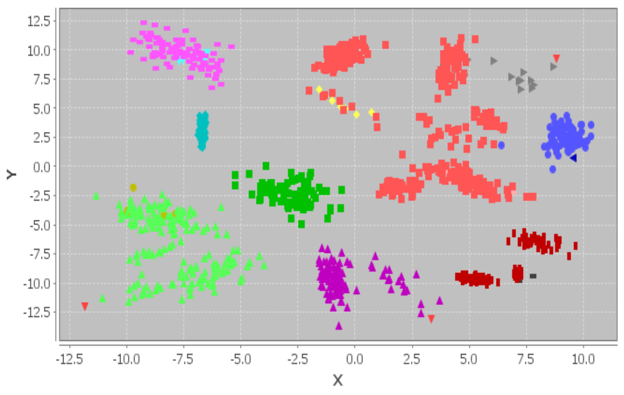

Figure 8 Our algorithm result with NumCore = 30

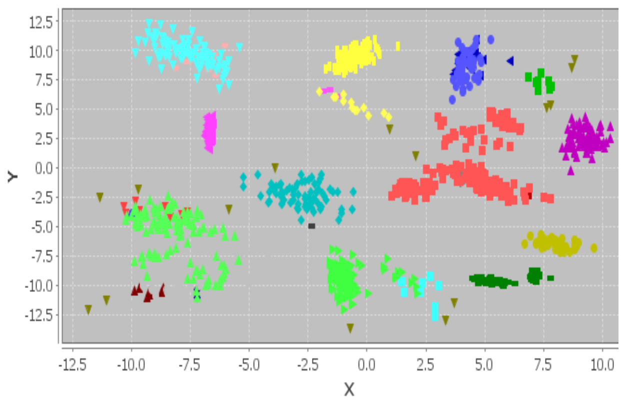

Figure 9 Our algorithm result with NumCore=20

The third data set tested with the proposed algorithm contains 3073 data point spread in ten clusters. The proposed algorithm shows better results than DBSCAN. DBSCAN finds 4 clusters with *eps = 2* and *MinPts*= 6. The result of DBSCAN is shown in Figure 10. Applying the proposed algorithm to each cluster resulted from DBSCAN with parameters *NumCore=30* and small neighborhood threshold, finds 8 clusters where 7 are of the original clusters while the remaining clusters are merged. As shown in figure 11 the red cluster is the merged one and other colors represent the other 7 clusters. Decreasing *NumCore* to 20, finds 9 clusters. 6 of them match the original clusters and one is separated into 2 clusters while the remaining clusters are merged into one cluster. Note that splitting and merging clusters is highly dependent on neighborhood threshold. In Figures 11 and 12, the 3 clusters represented by red are merged because the threshold between their units is low. On the other hand, splitting is caused by exceeding the assigned value of the neighborhood threshold.

Our experiments are continued by decreasing the value of *NumCore* to 10 and using high neighborhood threshold. This finds 10 clusters with 7 matching original clusters and 2 clusters are merged. The results are shown by Figure 13.

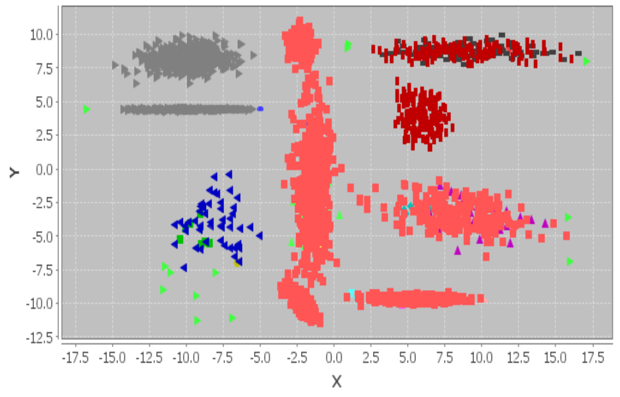

Figure 10 Results of DBSCAN algorithm with minpts =6 and eps=2, four different clusters is found by DBSCAN

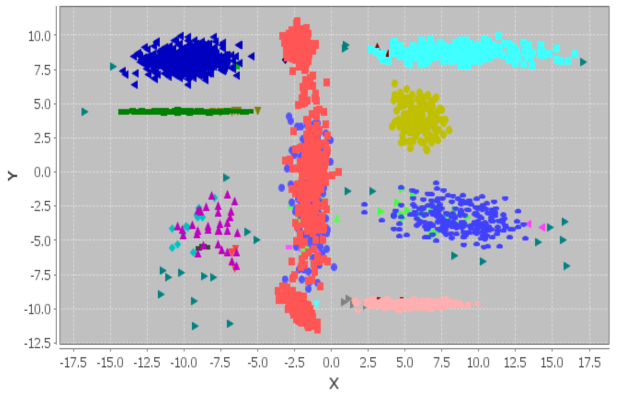

Figure 11 Our algorithm result with NumCore=30 and low neighborhood threshold, 7 matching clusters and 3 clusters merged clusters into one –red-

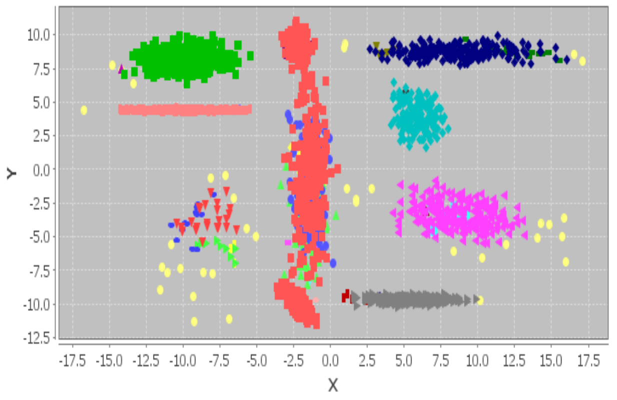

Figure 12 Our algorithm result with NumCore=20, 6 matching clusters, 3 clusters merged clusters into one, and one splitted into 2 clusters.

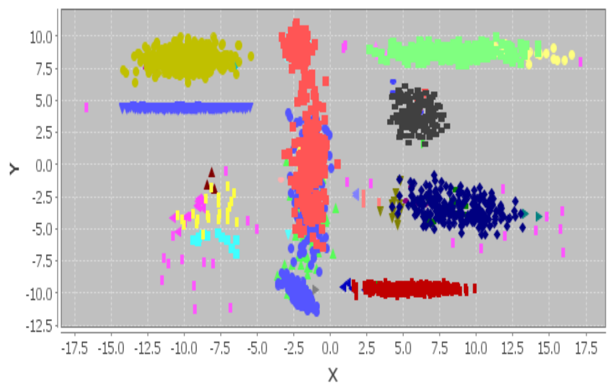

Figure 13 Our algorithm result with NumCore=10; 7 matching clusters, two merging into one -red-, and one cluster spiltted into two.

V. Conclusion

This paper proposed a new method for finding the within-clusters dense areas after running DBSCAN. The new algorithm tries to find the solution for the problem of different density level data sets. In the first stage the algorithm partitions the core points of a cluster to small units based on number of core in each and finds the *dfactor* which is an indication for unit density. The second phase of the algorithm is to merge the neighboring units with close *dfactor* values depending on the value of the neighborhood threshold. The algorithm is tested over many data sets and produced acceptable results compared to the original DBSCAN algorithm.

VI. Future Work

According to our observation the algorithm parameters are sensitive such that a small change in them leads to a large change in the results. Automatically choosing the parameter *NumCore* and *Neighborhood threshold* will be the future improvement for this algorithm to make it less sensitive to user choices. In addition to that more testing in large data sets is needed.

References

[1] J. Han and M. Kamber, Data Mining: Concepts and Techniques (2nd Edition), Morgan Kaufmann Publishers, 2006,

[2] G. Gan, C. Ma, and J. Wu, Data Clustering: Theory, Algorithms, and Applications, ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA, 2007.

[3] W. Zhenxing, Z. Liancheng, W. Qian, Clustering Algorithm Based on Characteristics of Density Distribution, Zheng Hua, IEEE 2010

[4] RUI XU and DONALD C. WUNSCH; "clustering" , JOHN WILEY & SONS, INC ,IEEE Press ,2009

[5] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In

Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD' 96), 1996.
[6] M. Ankerst, M. Bruenig, H.-P. Kreigel, and J. Sander. OPTICS:ordering points to identify the clustering structure. In Proceedings of ACM SIGMOD International Conference on Management of Data, June 1999
[7] A. Hinneburg and D. A. Keim, “An efficient approach to clustering in large multimedia databases with noise,” Proc. Fourth International Conference on Knowledge Discovery and Data Mining,1998
[8] Peng Liu, Dong Zhou, Naijun Wu, “Varied Density Based Spatial Clustering of Applications with Noise”, 2007, IEEE.
[9] E. Mollah , A. Rahman , R. Chowdhury; An Efficient Method for subjectively choosing parameter ‘k’ automatically in VDBSCAN , IEEE 2010
[10] M. Dash, H. Liu, X. Xu. “1+1>2“: Merging Distance and Density Based Clustering
[11] K. Jain and R. C. Dubes. Algorithm for Clustering Data, chapter Clustering Methods and Algorithms. Prentice-Hall Advanced Reference Series, 1988.
[12] H. Kriegel, P. Kroger, J. Sander,and A. Zimek, Density-based clustering, John Wiley & Sons, Inc. WIREs Data Mining Knowl Discov 2011
[13] R. Weber, H.-J Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Proceedings of the Intl. Conf. on Very Large Databases, 1998
[14] A. Hinneburg, D. Keim. Optimal Grid – Clustering : Toward Breaking the Curse if Dimensionality in High-Dimensional Clustering, IEEE
[15] P. Viswanath and V. Suresh Babu. Rough-DBSCAN: A fast hybrid density based clustering method for large data sets, Published in Pattern Recognition Letters 30 (2009).
[16] Sugato Basu, Ian Davidson, Kiri L. Wagstaff “Constrained Clustering Advances in Algorithms, Theory, and Applications”
[17] L. Kaufman and P. Rousseuw. “Finding Groups in Data - An Introduction to Cluster Analysis”. Wiley Series in Probability and Mathematical Statistics, 1990

Wesam Ashour has received his B.Sc. degree in Electrical and Computer Engineering in 2000 from the Islamic University of Gaza. He has completed his M.Sc. in Multimedia with Distinction in 2004 from the University of Birmingham, UK. He has got his PhD degree from the University of the West of Scotland in 2008. Dr. Ashour is currently a researcher at the Applied Computational Intelligence Research Unit in the University of the West of Scotland, UK since October, 2005. His research interests include data mining, artificial intelligence, reinforcement learning and neural networks.

Motaz Murtaja was born on August 30,1986. He received the B.Sc. degree in computer engineering from The Islamic University of Gaza (IUG) in 2009 In 2010 he was joined the M.Sc. program in the IUG.

From 2009 to 2010 he was worked as Teaching Assistant in Computer Engineering Department in the IUG and as a Teacher in the university collage in Gaza. Currently he works as computer engineering in Ministry of Health in Gaza Strip. Her research interests include classification data and clustering, and security attack detection.

How to cite this paper: Wesam Ashour,Motaz Murtaja,"Finding Within Cluster Dense Regions Using Distance Based Technique", International Journal of Intelligent Systems and Applications(IJISA), vol.4, no.2, pp.42-48, 2012. DOI: 10.5815/ijisa.2012.02.05