Active Semi-Supervised Clustering Algorithm for Multi-Density Datasets

Walid Atwa[1], Abdulwahab Ali Almazroi[2], Eman A. Aldhahr[3], Nourah Fahad Janbi[4]
Department of Information Technology-College of Computing and Information Technology at Khulais, University of Jeddah, Jeddah, Saudi Arabia[1, 2, 4]
Department of Computer Science and Artificial Intelligence-College of Computer Sciences and Engineering, University of Jeddah, Jeddah, Saudi Arabia[3]

***Abstract*—Semi-supervised clustering with pairwise constraints has been a hot topic among researchers and experts. However, the problem becomes quite difficult to manage using random constraints for clustering data when the clusters have different shapes, densities, and sizes. This research proposes an active semi-supervised density-based clustering algorithm, termed "ASS-DBSCAN," designed specifically for clustering multi-density data. By integrating active learning and semi-supervised techniques, ASS-DBSCAN enhances traditional clustering methods, allowing it to handle complex data distributions with varying densities more effectively. This research provides two major contributions. The first contribution of this research is to analyze how to link constraints (including that must be linked and ones that should not be linked) that will be utilized by the clustering algorithm. The second contribution made by this research is the ability to add multiple density levels to the dataset. We perform experiments over real datasets. The ASS-DBSCAN algorithm was evaluated against existing state-of-the-art system for various evaluation metrics in which it performed remarkably well.**

***Keywords*—Semi-supervised clustering; pairwise constraints; multi-density data; active learning**

I. Introduction

In data mining, clustering algorithms are used to divide data into multiple groups based on selected similarity metrics [1]. The rate at which Web data is increasing also called Big Data, clustering algorithms are expected to death with 1) scalability, 2) noise, 3) multidimensional data, 4) discovering clusters with arbitrary shapes, and 5) least dependency of domain knowledge to establish input parameters [2].

Clustering algorithms are categorized into several categories based on the requirements of the problem given. This includes 1) Partitional, 2) grid-based, 3) hierarchical and 4) density-based clustering algorithms [3]. From the list, density-based are suitable for finding clusters based on their unique size and shape. It does this by focusing on the density of the clusters in the region as opposed to clusters falling where the density is low [4-7].

The DBSCAN algorithm is one of the most popular used techniques in relation to this category of clustering algorithms that carries forward all advantages of density-based clustering family [8]. It works by calculating the total number of points (called *Eps*) around the point in a region. Points having density above the specified threshold value (called *MinPts*) are referred to as Core points, while others that do not meet these criteria are referred to as Noise points [9].

While this clustering technique is able to identify cluster regions having different shapes or sizes, it cannot handle data that contains clusters having points with different densities. This technique is only able to identify them as a cluster if the points are separated by sparse regions.

For example, Fig. 1(a) shows the dataset contains two cluster groups, sparse cluster C_1 (red points) and dense cluster C_2 (green points), and noise data (black points). When using large *Eps*-distance, DBSCAN algorithm can find a sparse cluster successfully. However, as shown in Fig. 1(b), the algorithm merges the dense cluster with the adjacent noise points. While using small *Eps*-distance, as shown in Fig. 1(c), the DBSCAN algorithm can assist in finding dense cluster C_2. On the other hand, it incorrectly marks the sparse clusters are noise points instead due to the core object condition. As a result of its reliance on set global parameters, DBSCAN is unable to discover clusters with variable densities.

Moreover, most existing density-based algorithms are unsupervised learning methods that can't utilise the available label information such as background knowledge [10] [11]. However, semi-supervised clustering algorithms can be considered as they utilize existing knowledge to enhance the clustering process. However, if these constraints are improperly selected the clustering performance will be affected [12]. Active learning was originally proposed to select the most important labeled data to generate pairwise constraints. However, most of the data in real-world applications is unlabeled [13-15].

To address the issue highlighted above, this is an enhanced version of DBSCAN algorithm that follows an active semi-supervised clustering approach. This new algorithm is named as ASS-DBSCAN which is capable of clustering multi-density data with active pairwise constraints. By quantitively analyzing the data using statistical techniques, the research was able to split the dataset into various density groups. Then, generate a set of active pairwise constrints from existing groups and compute their *Eps*. Finally, expand the clusters using selected active pairwise constraints.

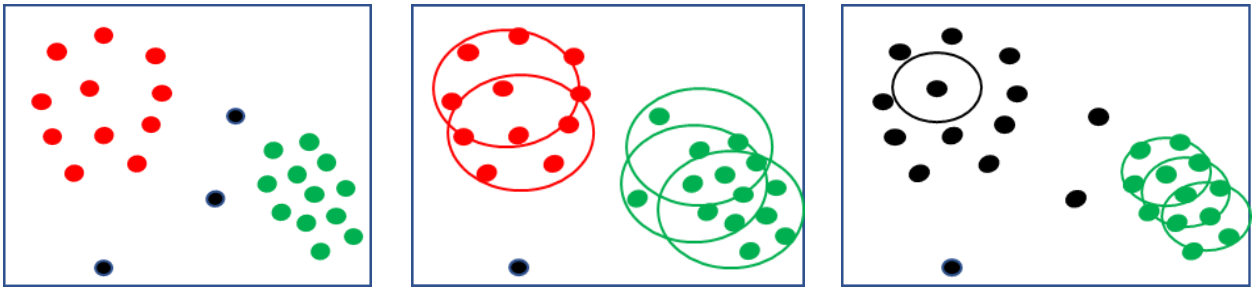


Fig. 1. Finding clusters using the DBSCAN algorithm from multi-density data (MinPt s = 5).

The paper is organized as follows. In Section II, existing literature on the given research is highlighted and critically discussed. In Section III, the research discusses the proposed algorithm and its working. In Section IV, the dataset and evaluation metrics are briefly discussed before highlighting the results of the search and its comparison to existing state-of-the-art systems. In Section V, the research is concluded along with directions for future research.

II. Related Work

In this section, existing literature on clustering machine learning algorithms is reviewed. Since the scope of this research is limited towards density-based algorithms thus it will focus on that as well as active learning methods.

In a study by Ankerst et al., an algorithm named OPTICS was proposed for the sole purpose of clustering analysis [1]. This algorithm generates two key parameters for clustering multi-density data including 1) core distance and 2) reachability. This additional information is added to the dataset to enable other algorithms to use the information for performing density-based clustering. The evaluation results showed that the algorithm was more efficient using these newly generated parameters over using existing information given in the dataset.

In another research, authors presented a clustering algorithm named DBSCAN, which is capable of finding clusters having various shapes as well as considering multiple densities [8]. However, DBSCAN algorithm faces limitation in finding clusters as they are reliant on *Eps* and *MinPts* parameters, where these values are incorrectly applied. The two studies by Jahirabadkar et al. and Liu et al. also utilize k-nearest neighbor of all selected objects so as to get the density for a given dataset. The authors further utilize various *Eps* values for sparse and dense clusters [2-3].

To address the issue of finding clusters using a dataset containing high dimensional data in distinct sizes, shapes, and density, Ertoz et al. suggested to check the points in the neighborhood of the clusters and use that information to check the similarity between the points [5]. Using this, the algorithm is able to define clusters using the defined points for those regions.

With the aim of detecting clusters that may have a unique shape or size, Liu et al. provided a variant of the DBSCAN algorithm. This algorithm as named as Entropy and Probability Distribution (EPDCA) [9]. The evaluation results conducted on benchmark datasets for the EPDCA algorithm show improved performance over other algorithms that were evaluated on the same dataset. This work was improved further by providing an optimized combinatorial clustering algorithm specifically designed for noisy performance. This algorithm is vital, particularly for data that is large with random sampling [16]. The result shows that the proposed clustering algorithm outperformance various traditional approaches that are compared with.

Kim et al. improved their work further by proposing an approximate adaptive AA-DBSCAN algorithm. This algorithm improved efficiency by reducing the time taken to calculate the parameters required to find clusters having multiple density points. The algorithm uses a density layer tree to distinguish between sparse and dense regions [10]. Kim et al. enhanced AA-DBSCAN further with *k*AA-DBSCAN that uses k^{th} nearest neighbors technique. The newer algorithm showed improvement in approximating the Eps distance but requires a longer running time [10].

In another research, authors proposed the GCMDDBSCAN algorithm, which is based on the DBSCAN algorithm. The authors proposed this algorithm to allow better performance when dealing with large databases [11]. The evaluation results showed improved performance on large databases over other algorithms evaluated on the same databases.

Constraint-based algorithms have emerged as extensions to traditional unsupervised clustering algorithms [17-19]. These methods incorporate constraints into the clustering process to enhance the learning of similarity metrics. Recently, several constraint-based clustering algorithms have been developed. For instance, Ruiz et al. introduced a pairwise-constrained clustering technique called C-DBSCAN, which leverages pairwise restrictions to enhance clustering accuracy [17]. The algorithm does this by determining the points that should be linked and the ones that should be avoided. This allows the algorithm to form clusters having unique shapes and sizes using the available constraints. In another study, a semi-supervised clustering algorithm (SSDBSCAN) is proposed that uses information provided in the dataset for evaluating density parameters [18]. The evaluation results showed better performance in comparison to other algorithms. However, SSDBSCAN can only be applied on smaller datasets.

Active learning in supervised machine learning has been a topic of interest for several decades when it comes to solving classification problems [20-22]. However, active learning for

clustering has acquired significant attention in recent years as it enhances the efficiency and accuracy of unsupervised learning algorithms [23]. Researchers have integrated active learning methods with different clustering algorithms. Active learning methods focused on uncertainty sampling, querying data points near cluster boundaries or with high uncertainty to improve cluster performance. Recently active semi-supervised clustering uses pairwise constraints to improve the clustering performance, especially for noisy, unbalanced or high-dimensional data. Thus, active learning methods can enhance clustering performance while reducing labeling costs [24-26].

III. ASS-DBSCAN ALGORITHM

This section covers ASS-DBSCAN that can rapidly and efficiently identify clusters for multi-density data having different shapes and sizes, with the knowledge of dealing with points that do not fall within the cluster. This algorithm comprises two main steps. 1) Partitioning dataset: divide the dataset into multiple density levels and generate pairwise constraints for linking and not linking points 2) Clusters are formed from the pairwise constraint information generated in step 1.

A. Generating Density levels

Let us suppose, that $X = \{x_1, x_2, \ldots, x_n\}$ represent data points where the i-th point can show nominated as $x_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$. We start by computing the local density function for each data point. As described in Eq. (1), the local density is calculated by summing the distances between a point and its nearest neighbors. Next, the data points are ranked in descending order according to their local density values. This will allow the algorithm to calculate the difference in density between the point and its adjacent point x_i and x_{i+1} denoted by *DENVAR* (x_i, x_{i+1}) as described in Eq. (3).

$$DEN(x) = \sum_{i=1}^{k} D(x, y_i) \quad (1)$$

$$D(x, y_i) = \sqrt{\sum_{j=1}^{m} (x_j - y_j)^2} \quad (2)$$

Here $D(x, y_i)$ shows the Euclidean distance between point x and its k-nearest neighbors y_i.

$$DENVAR(x_i, x_{i+1}) = \frac{|DEN(x_{i+1}) - DEN(x_i)|}{DEN(x_i)} \quad (3)$$

After getting the density variation between data points we will notice that the variation for points that are within the proximity of the density is small and there is some distinct variation between the different densities. Thus, all density level sets (*DLS*) are collected by computing the unique density variations.

To simplify the explanation of our algorithm, we provide an example using a dataset that contains four clusters with varying densities, as explained in Fig. 2(a). We compute the density for each point of this dataset and sort them in descending order to get the results in Fig. 2(b). Fig. 2(b) contains four relatively smooth lines which accordingly represent four density levels and some sharp dips. After computing distinct variation, it can be seen that the smooth lines and sharp waves in Fig. 2(b) correspond to the density level sets and sharp change of two density levels that is highlighted in Fig. 2(c) respectively.

Each smooth line needs to be extracted to acquire the density level for that set. This can be achieved by using a density variation threshold τ. This can allow a dataset having multi-density data to be converted into a dataset containing multiple density level sets. Each Density level set should have data points with approximate densities. Points x_i and x_j are considered to be in the same density level set if the following criteria is met:

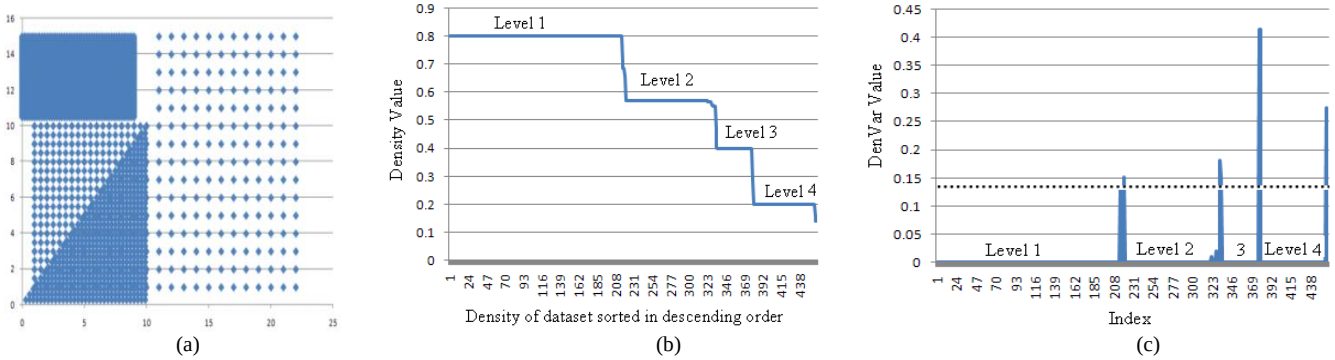
$$x_i, x_j \in DLS \quad if\ DENVAR(x_i, x_j) \leq \tau \quad (4)$$

The τ is calculated using the density variation values (*DVList*) as follows:

$$\tau = E(DVList) + \sigma(DVList)$$

where, σ is standard deviation of *DVList* and E is the mathematical expectation.

DVList values indicate that there are only a handful of points that have large DENVAR values. These points can be used to divide the dataset and transform into one having multiple density levels set.



(a) (b) (c)

Fig. 2. Density and density variation for multi-density dataset.

After splitting the dataset into various density level sets, we generate a set pairwise constraints for performing clustering. The points that must be linked should be picked from the same density level and vice-versa Thus, we need to select the most informative objects from each density level set which have the highest neighborhood density. Now, we can generate a set of not to be linked constraints between the selected objects from different density level sets. Also, generate a set of constraints

for the points that must be linked between the most informative objects and their *k*-nearest neighbors objects in the same density level set. We summarize our strategy for selecting active pairwise constraints in Algorithm 1.

Algorithm 1. Active Pairwise Constraints

1. Initialization: $N_1 = \{x\}$, where x is a random point in X; $N = N_1$; $l = 1$; $q = 0$;
2. **while** $q<Q$
3. Select the most informative point x^* with the highest neighborhood density.
4. **for** each neighborhood $N_i \in N$
5. Query x^* against any data point $x_i \in N_i$;
6. q++;
7. **if** a must-link established between x^* and x_i
8. Add x^* to neighborhood N_i;
9. **end if**
10. **end for**
11. **if** no must-link is achieved
12. Add cannot-link between x^* and all points in $\{N_i\}_{i=1}^{l}$;
13. l++; create new neighborhood $N_l = \{ x^*\}$; $N = N \cup N_l$
14. **end if**
15. **end while**
16. **return** set of active pairwise constraints

First, we select point having the most information, which has highest neighborhood density and query it against the points in each neighborhood. If a point is returned that must be linked, then this operation needs to be performed again. This process is repeated until a must does not link point is returned.

We can generate more constraints using the transitive closure between the objects in pairwise constraints. For example, if we have must-link between the two objects x_i and x_j and must-link between the two objects x_i and x_k, then we get must-link between the two objects x_j and x_k. Also, if we have must-link between the two objects x_i and x_j and cannot-link between the two objects x_i and x_k, then we get cannot-link between the two objects x_j and x_k. The partitioning dataset into different density level sets and generating the pairwise constraints are presented in Algorithm 2.

Algorithm 2: Generating density level sets with pairwise constraints

1. Calculate the density function for every point in the dataset.
2. Sort the points based on their density in descending order.
3. Determine the density variation between each pair of consecutive data points.
4. Group the dataset into different Density Level Sets (*DLS*) based on the computed density variations.
5. Create a cannot-link constraint between data points in different density level;
6. **For** each density level set (DLS_i)
7. Call Active Pairwise Constraints algorithm.

B. Expanding Clusters

When the *Eps* value is the same, clusters with different densities cannot be found. For every density level set, the *Eps* the value is approximated. The associated *Eps* for a particular density level set (*DLS*) can be boosted by setting the maximum *DEN* value. Some of the points selected may be classified as border objects or noise, which could affect the *Eps* value. Therefore, this issue is resolved by calculating Eps_i for DLS_i as follows:

$$Eps_i = \mathrm{max}DEN(DLS_i) \cdot \sqrt{\frac{medianDEN(DLS_i)}{meanDEN(DLS_i)}} \qquad (5)$$

where, meanDEN, medianDEN and maxDEN, are the mean, median and maximum density of DLS_i respectively.

Now, we explain an example to demonstrate the effect of the proposed algorithm on a dataset with different densities. First, we assume that the parameter MinPts = 5 and compute the parameter Eps for each density according to Eq. (5). Next, as shown in Fig. 3(a), we have two different densities and set of noise points. Thus, we have two values for the Eps parameter (1.7 and 0.9) for the sparse and dense density respectively as shown in Fig. 3(b) and Fig. 3(c).

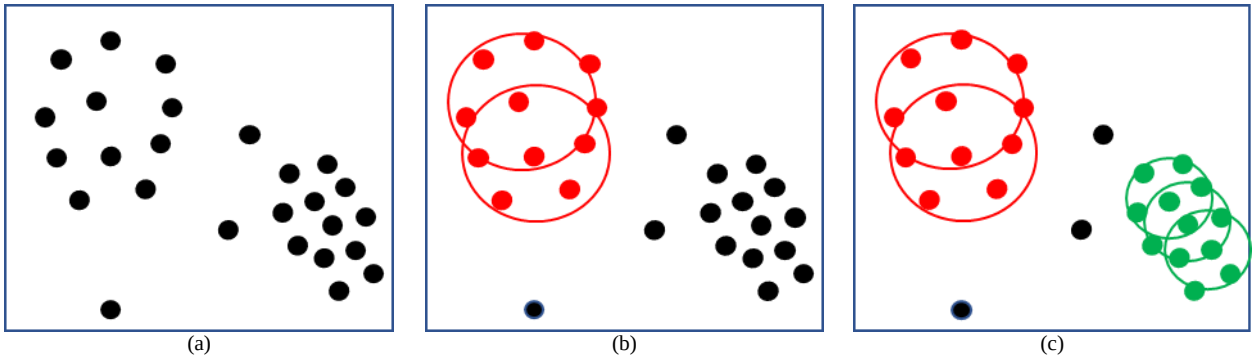
With density levels partitioning and parameters estimation done, pairwise constraints are used for each density level cluster to increase the cluster size if necessary as shown in Algorithm 3:

Algorithm 3. ASS-DBSCAN

1. Partitioning the dataset and generating pairwise constraints using algorithm 1;
2. **For** each density level set (DLS_i)
3. Estimate the parameter Eps_i ;
4. Initialize all data points in DLS_i as *UNCLUSTERED*;
5. *ClusterNum* = 0;
6. **For** each $x \in DLS_i$
7. **If** data *x not CLUSTERED* then
   Count the number of data points in *x's Eps*-neighborhood;
   a. **If** the number of data points in *neighborhood* < *MinPts*
      add x to NOISE set;
      Else
      Add x to the current cluster ClusterNum;
   b. **If** there are must-link constraints between *x* and *y*
      For each point *y* in *ML*(*x*, *y*)
      Add y to the current cluster ClusterNum;
   c. **For** each point *z* in neighborhood **do**
      If *z* is not C*LUSTERED* and does not violated cannot-link constraints then
      Add z to the current cluster ClusterNum;
8. *ClusterNum* = *ClusterNum* +1;
9. Return the set of clusters.

- In Step 7(a), after accurately estimating the *Eps* parameter, we proceed to compute the *Eps*-neighborhood for each unclustered point. We then compare the number of data points within this neighborhood to the specified *MinPts* parameter to decide regarding the classification of the data point as part of a cluster or as noise. If the count of points in the *Eps*-neighborhood is less than *MinPts*, we classify the data point as noise. Conversely, if the number of points meets or exceeds *MinPts*, we incorporate the data point into the current cluster, thereby refining the clustering process.

- In Step 7(b), we address the Must-link constraints. If a point has any Must-link constraints with other points, all of these points are assigned to the current cluster.



(a) (b) (c)

Fig. 3. Example of different *Eps* values with varying densities (MinPt s = 5).

- In Step 7(c), we consider the Cannot-link constraints. For each unclustered point within the current neighborhood that does not violate any Cannot-link constraints with the existing points in the current cluster, these points will also be included in the current cluster.

C. Time Complexity

This is divided into two parts. The first part is the time for dividing the dataset into different densities by calculating the density values. As per our calculations it leads to a time complexity of O($n$log$n$); the ordering of density values consumes O($n$log$n$); the other processes have the same runtime complexity of O(n). The second part is the time for expanding clusters, where the time required for a neighborhood query of a single object is O(n) and neighborhood query is is executed for every n points present in the dataset. The time of this part is O(m+n^2); where m is the number of pairwise constraints. However, using an index structure like R*-tree, minimize the time required for a neighborhood query of a single object to O($logn$). Thus the time required for this part is O($m + n\ log\ n$). The total runtime complexity of the proposed algorithm is O($m + n\ log\ n$).

IV. Experimental Results

This section evaluates the proposed algorithm against other baseline algorithms on a real dataset from the UCI repository. Each dataset was labeled with the number of data points, features, and cluster labels respectively as follows: Class (214,10,6), Ecoli (336,8,8), Ionosphere (351,34,2), Liver (345,6,2), Breast (683,9,2), Yeast (1484,8,10), Waveform (5000,21,3), Segment (2310,19,7) and Magic (19020,10,2). The evaluation of these algorithms is conducted using Normalized Mutual Information (*NMI*) as the clustering validation metric that is defined as follows:

$$NMI = \frac{I(X;Y)}{\big(H(X) + H(Y)\big)/2}$$

A. Clustering Performance

In this subsection, we explain the clustering performances of the algorithms C-DBSCAN, SSDBSCAN, AA-DBSCAN, and ASS-DBSCAN across the nine real-world datasets mentioned earlier. Table I presents the parameters values of each algorithm. As the parameter settings will be different according to each dataset, we employ a range of values for each parameter as detailed in Table I.

TABLE I. Parameters Values of Each Algorithm

Algorithm	Parameters
C-DBSCAN	*Eps* ∈ [0.1, 0.5] and *MinPts* ∈ {3, 4, . . . , 10}
SSDBSCAN	*Eps* ∈ [0.1, 0.5] and *MinPts* ∈ {3, 4, . . . , 10}
AA-DBSCAN	*MinPts* ∈ {3, 4, . . . , 10}
ASS-DBSCAN	*MinPts* ∈ {3, 4, . . . , 10}

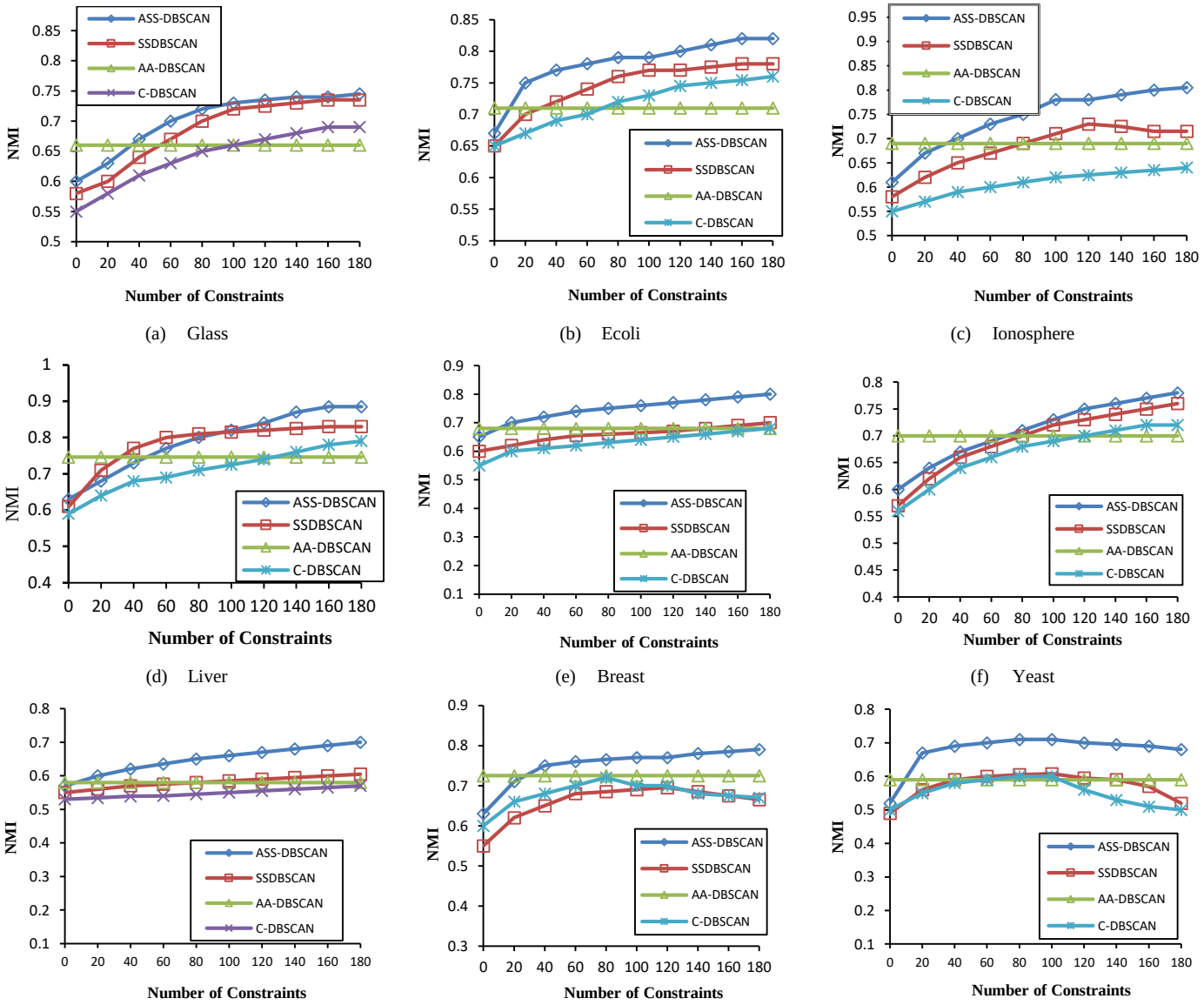
We present the clustering performance result based on NMI of ASS-DBSCAN in comparison to compared algorithms with varied constraints on the real datasets. Each algorithm is shown with its own unique color in Fig. 4. For each dataset, a number of constraints were utilized with respect to the NMI. The result demonstrates that ASS-DBSCAN performance better than the compared algorithms in general. For example, as illustrated in Fig. 4(a), when utilizing 20 constraints, ASS-DBSCAN achieves a performance level exceeding 0.6 NMI, and with 80 constraints, ASS-DBSCAN achieves NMI of more than 0.7, respectively.

It can be observed that SSDBSCAN comes second in terms of effectiveness, followed by C-DBSCAN. Even though AA-DBSCAN has a higher performance in the early stages. However, with the increased number of pairwise constraints, AA-DBSCAN has consistently shown in all the datasets its static performance with stable and non-improved result. This can be explained since algorithms are an unsupervised clustering algorithm. Hence, it is not effective with pairwise constraints. Also, we can observe that the performance of ASS-DBSCAN can slowly increases with increased number of pairwise constraints as shown for Glass and Breast datasets. However, its performance surpasses the other algorithms.

B. Running Time Evaluation

This section discusses the running time of the ASS-DBSCAN and other algorithms that have been evaluated. Fig. 5 shows the execution times on the six datasets (containing pairwise constraints) used for evaluating. Lower execution time is considered better and vice versa. From Fig. 5, SSDBSCAN achieved the lowest performance of all pairwise constraints. AA-DBSCAN also has static value as shown in Fig. 5. These static values manifest as the number of

constraints are increased. We observed that the ASS-DBSCAN on most datasets achieve relatively high execution time with low constraints, however, overall, the execution time drops, and better performance was recorded for ASS-DBSCAN. Looking at all the experiments conducted on the other datasets, we can see that consistently ASS-DBSCAN performs better by achieving lower execution times in comparison to other algorithms.

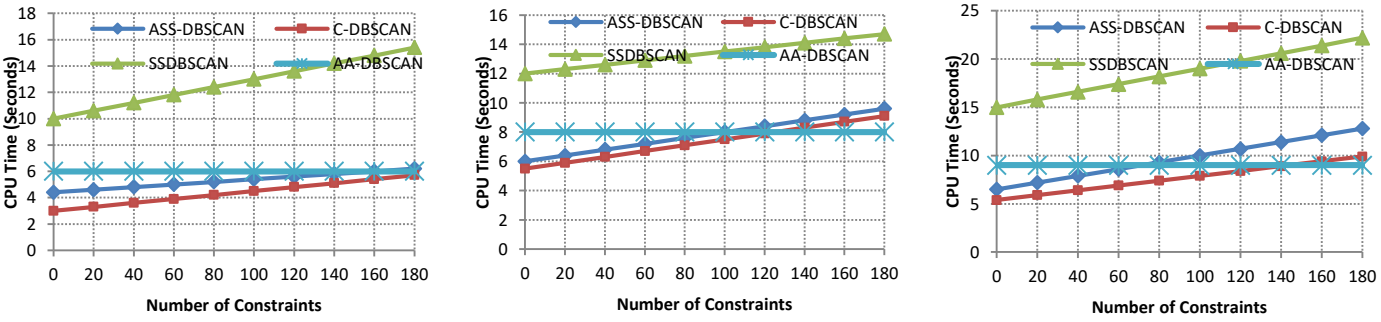


(a) Glass (b) Ecoli (c) Ionosphere

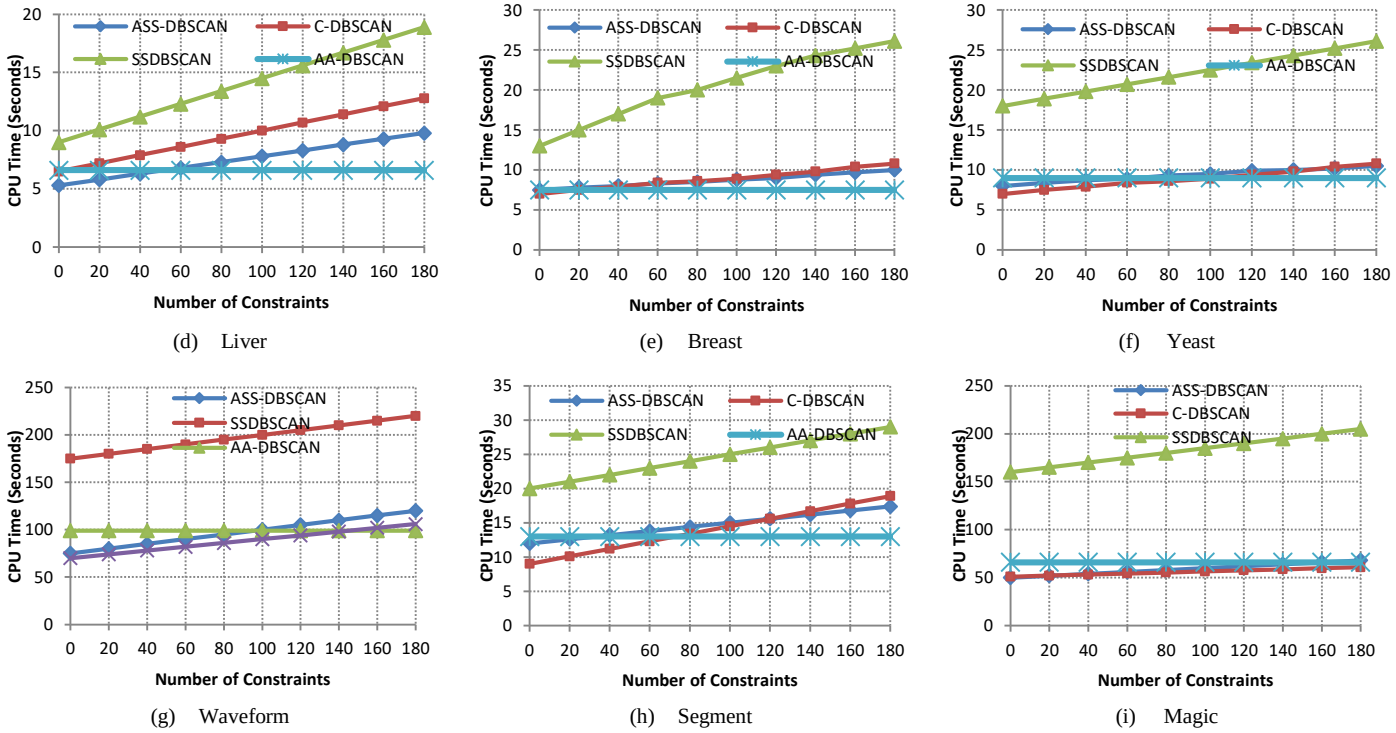
(d) Liver (e) Breast (f) Yeast

(g) Waveform (h) Segment (i) Magic

Fig. 4. Clustering performance based on NMI with different numbers of pairwise constraints.



(a) Glass (b) Ecoli (c) Ionosphere



(d) Liver (e) Breast (f) Yeast

(g) Waveform (h) Segment (i) Magic

Fig. 5. Execution time with different numbers of pairwise constraints.

V. Conclusions and Future Work

In this paper, we propose ASS-DBSCAN for clustering different density data with a set of active pairwise constrints. By examining the statistical properties of the dataset's density variation, the suggested algorithm divides it into multiple density level sets, which are subsequently expanded using active pairwise constraints. The algorithm was evaluated for performance and execution time on the real datasets again other algorithms. The evaluations results showed that the algorithm not only performed better in achieving its goals but also took less time in order to do so. Future work includes the extension the current work to cluster more complex data from real-life streaming applications. In addition, we aim to develop a comprehensive system that integrates active learning and semi-supervised learning techniques to be applied on different applications.

Acknowledgment

This work was funded by the University of Jeddah, Saudi Arabia, under grant No. (UJ-21-DR-134). The authors, therefore, acknowledge with thanks the university's technical and financial support.

References

[1] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," *ACM Sigmod Rec.*, vol. 28, no. 2, pp. 49–60, 1999.

[2] P. Liu, D. Zhou, and N. Wu, "VDBSCAN: varied density based spatial clustering of applications with noise," in *2007 International conference on service systems and service management*, 2007, pp. 1–4.

[3] X. Zhang, and Z. Shibo "WOA-DBSCAN: Application of Whale Optimization Algorithm in DBSCAN Parameter Adaption," IEEE Access, 2023.

[4] A. A. Almazroi, and W. Atwa, "An improved clustering algorithm for multi-density data," Axioms vol 11, no 8, 2022.

[5] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the 2003 SIAM international conference on data mining*, 2003, pp. 47–58.

[6] A. Fahim, "Adaptive Density-Based Spatial Clustering of Applications with Noise (ADBSCAN) for Clusters of Different Densities," Computers, Materials & Continua vol 75, no 2, 2023.

[7] W. Atwa, and K. Li. "Constraint-based clustering algorithm for multi-density data and arbitrary shapes." *In Industrial Conference on Data Mining*, pp. 78-92. Springer, Cham, 2017.

[8] 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 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[9] X. Liu, Q. Yang, and L. He, "A novel DBSCAN with entropy and probability for mixed data," *Cluster Comput.*, vol. 20, no. 2, pp. 1313–1323, 2017.

[10] J.-H. Kim, J.-H. Choi, K.-H. Yoo, and A. Nasridinov, "AA-DBSCAN: an approximate adaptive DBSCAN for finding clusters with varying densities," *J. Supercomput.*, vol. 75, no. 1, pp. 142–169, 2019.

[11] L. Zhang, Z. Xu, and F. Si, "GCMDDBSCAN: multi-density DBSCAN based on grid and contribution," in *2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing*, 2013, pp. 502–507.

[12] M. A. Masud, J. Z. Huang, M. Zhong, and X. Fu, "Generate pairwise constraints from unlabeled data for semi-supervised clustering," *Data Knowl. Eng.*, vol. 123, p. 101715, 2019.

[13] W. Atwa, and K. Li. "Active query selection for constraint-based clustering algorithms." In *International Conference on Database and Expert Systems Applications*, pp. 438-445. Springer, Cham, 2014.

[14] W. Atwa, and M. Emam. "Improving Semi-Supervised Clustering Algorithms with Active Query Selection." *Advances in Systems Science and Applications* 19, no. 4 (2019): 25-44.

[15] W. Atwa, "A Supervised Feature Selection Method with Active Pairwise Constraints." In *Proceedings of the 11th International Conference on Informatics & Systems (INFOS 2018)*. 2018.

[16] J. Kim, W. Lee, J. J. Song, and S.-B. Lee, "Optimized combinatorial clustering for stochastic processes," *Cluster Comput.*, vol. 20, no. 2, pp. 1135–1148, 2017.

[17] C. Ruiz, M. Spiliopoulou and E. Menasalvas. C-DBSCAN: Density-Based Clustering with Constraints. In Proceedings of the International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, pages 216- 223, 2007.

[18] L. Lelis and J. Sander. Semi-Supervised Density-Based Clustering. In Proceeding of the Ninth IEEE International Conference on Data Mining, 842-847, 2009.

[19] W. Atwa, and K. Li. "Semi-supervised Clustering Method for Multi-density Data." *In International Conference on Database Systems for Advanced Applications*, pp. 313-319. Springer, Cham, 2015.

[20] P. Zhou, S. Bicheng, L. Xinwang, D. Liang, and L. Xuejun, "Active Clustering Ensemble With Self-Paced Learning," IEEE Transactions on Neural Networks and Learning Systems, 2023.

[21] P. Kumar, and G. Atul, "Active learning query strategies for classification, regression, and clustering: a survey," Journal of Computer Science and Technology, vol 35, pp 913-945, 2020.

[22] A. Hussein, A. A. Almazroi, "Multiclass Classification for Cyber Threats Detection on Twitter", CMC-COMPUTERS MATERIALS & CONTINUA, 77(3), 2023.

[23] W. Yu, L. Xing, F. Nie, and X. Li, "An efficient semi-supervised balanced cut with hard pairwise constraints and partial labels," Knowledge-Based Systems, vol 276, 2023.

[24] J. Cai, J. Hao, H. Yang, X. Zhao, and Y. Yang, "A review on semi-supervised clustering," Information Sciences, 2023.

[25] W. Atwa, A. A. Almazroi, "Active selection constraints for semi-supervised clustering algorithms", . Int. J. Inf. Technol. Comput. Sci, 2020.

[26] A. A. Almazroi, and W. Atwa, "Semi-Supervised Clustering Algorithms Through Active Constraints", *International Journal of Advanced Computer Science & Applications*, *15*(7), 2024.