International Journal of Machine Learning and Cybernetics (2020) 11:675–714
https://doi.org/10.1007/s13042-019-01027-z

ORIGINAL ARTICLE

An efficient clustering algorithm based on the *k*-nearest neighbors with an indexing ratio

Raneem Qaddoura[1] · **Hossam Faris**[1] · **Ibrahim Aljarah**[1]

Received: 20 February 2019 / Accepted: 28 October 2019 / Published online: 18 November 2019

Abstract
Clustering is a challenging problem that is commonly used for many applications. It aims at finding the similarity between data points and grouping similar ones into the same cluster. In this paper, we introduce a new clustering algorithm named Nearest Point with Indexing Ratio (NPIR). The algorithm tries to solve the clustering problem based on the nearest neighbor search technique by finding the nearest neighbors for the points that are already clustered based on the distance between them and cluster them accordingly. The algorithm does not consider all the nearest points at once to cluster a single point but iteratively considers only one nearest point based on an election operation using a distance vector. NPIR tries to solve some limitations of other clustering algorithms. It tries to cluster arbitrary shapes which have non-spherical clusters, clusters with unusual shapes, or clusters with different densities. NPIR is evaluated using 20 real and artificial data sets of different levels of complexity with different number of clusters and points. Results are compared with those obtained for other well-known and common clustering algorithms. The comparative study demonstrates that NPIR outperforms the other algorithms for the majority of the data sets in terms of different evaluation measures including Homogeneity Score, Completeness Score, V-measure, Adjusted Mutual Information, and Adjusted Rand Index. Furthermore, NPIR is experimented on a real-life application for segmenting mall customers for effective decision making. The source code of NPIR is available at http://evo-ml.com/2019/10/28/npir/.

Keywords Clustering · Data mining · Unsupervised learning · Cluster analysis · Nearest point · Indexing ratio · Nearest neighbor search technique · Nearest point with indexing ratio algorithm · NPIR

1 Introduction

Clustering is a technique for data analysis which shows the structure of data points and explores useful patterns from these points. Clustering is used to group data points so that each group contains similar points which are dissimilar to others in other groups [1]. Clustering observes the features of each point to predict the group in which each point belongs to.

Clustering algorithms have been used in wide range of applications, for example, it is used for bioinformatics [2], image processing [3–7], pattern recognition [8–10], document categorization [11, 12], financial risk analysis [13], cancerous data [14, 15], search engines [16], academics [17], drug activity prediction [18], and much more.

Many algorithms were proposed for clustering in the literature, but still there is not such an algorithm that can fit for all types of data sets. That is, an algorithm might perform better than other algorithms for a specific data set but not for another one. Algorithms might perform differently for different sizes of data sets, different dimensionality, or different densities. In addition, some of the algorithms are more simple and easier to understand than others [19].

One of the most popular algorithms for clustering is the *k*-means algorithm which was proposed over 60 years ago, and it is still widely used [20]. The *k*-means algorithm has simple implementation and fast execution which makes it more popular. However, *k*-means assumes a known number of clusters, a spherical distribution of points within the cluster, and same clusters size or density [21] which might not be true for all types of data sets. *k*-means also might fall in local optima and fixed shaped clustering depending on

✉ Ibrahim Aljarah
i.aljarah@ju.edu.jo

Raneem Qaddoura
rqaddoura@philadelphia.edu.jo

Hossam Faris
hossam.faris@ju.edu.jo

[1] King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

the initial assignments of the centroids. A popular variation of *k*-means is *k*-means++ which works similar to *k*-means but chooses the initial assignments of the centroids [22]. Other popular clustering algorithms include Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [23], density-based spatial clustering of applications with noise (DBSCAN), hierarchical DBSCAN (HDBSCAN) [24], Expectation-Maximization (EM) [1], Minimum Spanning Tree (MST) [25, 26], single linkage hierarchical clustering (HC-SL) [25, 27, 28], complete linkage hierarchical clustering (HC-CL) [28, 29], and nature-inspired algorithms such as genetic algorithm (GA) [30], Particle Swarm Optimization (PSO) [31–33], and static and dynamic clustering based multi-verse optimizer (SCMVO and DCMVO) [34].

In this paper, we propose a new clustering algorithm with the aim of clustering different types of data sets including challenging ones in the literature which have different cluster densities, points distribution, and different sizes. We also try to overcome the fixed shaped clustering which is an observed problem in other clustering algorithms. The new proposed algorithm is referred to as Nearest Point with Indexing Ratio (NPIR). The algorithm tries to solve the clustering problem based on the nearest neighbor search technique by finding the nearest neighbors for the points that are already clustered based on the distance between them and cluster them accordingly. The algorithm does not consider all the nearest points at once to cluster a single point but iteratively considers only one nearest point based on several operations using a distance vector. It also includes settings of two parameters which are the Indexing Ratio (*IR*) and the iterations (*i*) to optimize the generated results more further. In general, NPIR differs from other clustering algorithms in that it tries to partition data of arbitrary shapes, non-spherical clusters, or clusters having different densities.

Twenty data sets are used for experiments which are Aggregation, Aniso, Appendicitis, Blobs, Circles, Diagnosis II, Flame, Glass, Iris, Iris 2D, Jain, Moons, Mouse, Pathbased, Seeds, Smiley, Varied, Vary density, WDBC, and Wine. These data sets have different number of points, features, and clusters. Some of them are considered challenging in the literature as they have complex shapes including circular patterns and interleaving clusters. Others have visual separated parts but are computationally difficult to partition. Furthermore, the proposed NPIR gives enhanced results compared to the other well-known clustering algorithms. NPIR iteratively considers correcting wrongly clustered points which produces better results and avoids the termination of cluster shaping process at an early stage.

The proposed NPIR is evaluated in terms of Homogeneity Score, Completeness Score, V-measure, Adjusted Mutual Information, and Adjusted Rand Index. Results are compared with *k*-means++, BIRCH, HDBSCAN, EM, MST, HC-SL, HC-CL, GA, PSO, and DCMVO. The experiments show that NPIR outperforms the other algorithms in the majority of the data sets, and it is promising for many applications. Mall customer segmentation application is also experimented and analyzed.

The remainder of the paper is organized as follows: The next section presents the most common algorithms and the recent work on clustering. Section 3 describes in details the proposed algorithm. Section 4 discusses the experiments and results along with sensitivity analysis for the NPIR parameters. The last section concludes the work.

2 Related work

Clustering is a common task in machine learning which can be used for finding the characteristics of each group of clusters. Numerous previous works discussed clustering techniques and applications.

k-means is one of the earliest algorithms that was recognized in several applications as a useful algorithm for clustering. Since then, the importance of the clustering techniques and their positive effect on real life applications has been recognized. Clustering algorithms can be classified into partitioning algorithms, hierarchical algorithms, and density-based algorithms [1, 35, 36].

Partitioning algorithms are the simplest and the most basic algorithms for clustering. Some popular algorithms in the partitioning category include *k*-means [20], *k*-means++ [22], and EM [1]. Algorithms of this category relatively require less computational time compared to the other algorithms in other categories. They also consider initial clustering results with iterative correction of wrongly clustered data points. They also have random behavior as they consider different clustering results for different executions which increases the possibility of finding correct clustering results and enhancing the quality of these results. However, they do not work well for non-spherical shapes [35] and they usually fall in local optima.

Hierarchical algorithms such as BIRCH algorithm [23] group data points into a hierarchy of clusters. The strength of such algorithms is that they can detect arbitrary and non-spherical shapes that cannot be detected by partitional algorithms. However, they are unable to correct wrongly clustered data points as they cannot undo the clustering of the data points [1]. They also consume much time and space [36].

Density-based algorithms such as DBSCAN [37] and OPTICS [38] can also work well with non-spherical shapes but they fail to detect shapes of different densities [35, 36]. DBSCAN also requires careful selection of its parameters [35]. Thus, HDBSCAN [24] was recently introduced to solve the parametric and border points problems by revising the DBSCAN and OPTICS algorithms and generating a refined

version of the two algorithms considering both density and hierarchy clustering.

Variations of the aforementioned algorithms are observed in the literature and several recent approaches were proposed. The work of [39] considers the MinMax *k*-means algorithm as a weighted variation of the *k*-means which assigns weights to the clusters according to the clusters variance. Site rates clustering with an automatic selection of partitioning schemes using iterative *k*-means is presented in [40]. Authors in [41] proposed a variant method for finding initial centroids using entropy-based farthest neighbor approach. An algorithm that works as a mask for the EM algorithm which is suitable for large and high dimensional data sets is presented in [42]. Parallel implementation for large scale data sets are also exist in literature [43–46].

Nature-inspired algorithms are used for clustering such as GA [30], PSO [31–33], MVO [47, 48], and GWO [49, 50]. Variations of these algorithms can be found in the literature: A genetic algorithm-based clustering was proposed by [51] for constrained networks. Automatic clustering for finding the right number of clusters using genetic algorithm based clustering algorithm can be found in the work of [52]. Traveling salesman problem is also considered by [53] by proposing a new initial population strategy for the genetic algorithm. Adaptive PSO based on clustering (APSO-C) can be found in the work of [54]. A combination of fuzzy clustering pre-processing and PSO for clustering can also be found in the work of [55]. Authors in [56] proposed a hybrid approach combining *k*-means algorithm with Particle Swarm Optimization for clustering Arabic documents. In addition, authors in [34] proposed a clustering algorithm using multi-verse optimizer with two modes: static clustering with predefined number of clusters (SCMVO) and dynamic clustering with automatic detection of the number of clusters (DCMVO).

Under this view, NPIR is introduced and discussed in this work. NPIR takes advantage of both partitional and density-based algorithms to detect data sets of arbitrary and non-spherical shapes while it also considers iterations to correct wrongly clustered data points. It is based on the nearest neighbor search technique and a random and iterative behavior of the partitional clustering algorithms to give quality clustering results. It does not depend on cluster centers to predict clusters as in partitional clustering algorithms but instead considers cluster density as in density-based clustering algorithms. However, it also does not consider a user to define the density as in DBSCAN but it rather automatically adjust different densities and sizes of clusters. NPIR is discussed in details in the next section.

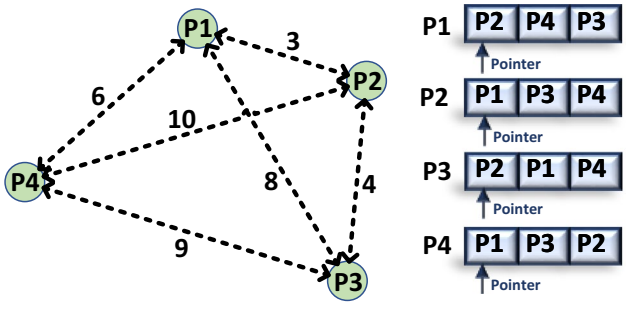


Fig. 1 Data points and the distance vector for each based on the distance between each pair of point

3 Nearest Point with Indexing Ratio (NPIR)

NPIR is a clustering algorithm which explores the characteristics of the data points to group similar data points into the same clusters and dissimilar data points into different clusters.

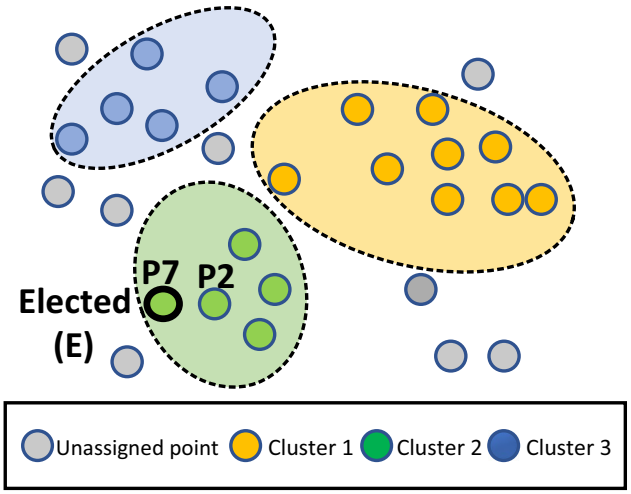


Fig. 2 Election operation which randomly elects one of the points that are already assigned to a cluster

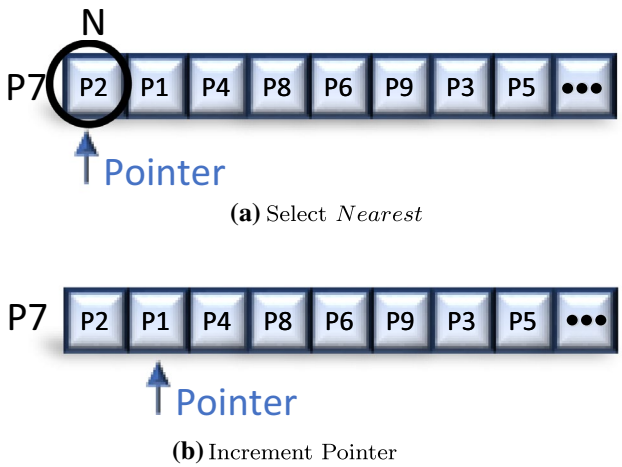


Fig. 3 Selection operation. **a** Selection of P2 as the *Nearest* point for point P7 which is the point with an index at the pointer of the distance vector of the *Elected*; **b** increment pointer of the distance vector of the *Elected*

It is based on the nearest neighbor search technique in finding a k-nearest neighbor to a certain point. The algorithm iterates to assign data points to the most suitable clusters. It performs Election, Selection, and Assignment operations to assign data points to appropriate clusters. In this section, we discuss the algorithm details, operations, and pseudocode. A simple example is also presented to illustrate the algorithm more further. Furthermore, the complexity of the algorithm is discussed.

3.1 Preliminaries

NPIR clustering algorithm is based on the nearest neighbor search technique where the k-nearest neighbor of a certain point is considered for assignment to the cluster of that point. Searching for the k-nearest neighbor is considered as a nearest neighbor search (NNS) problem. NNS problem can be defined as follows [57]:

Definition 1 Given a set of N points $P = \{p_1, p_2 \dots p_N\}$ in space, find the k-nearest neighbors set to a certain point p_i where $p_i \in \{p_1, p_2 \dots p_N\}$.

NPIR does not only consider a single point as the nearest neighbor for a given point but it also considers different k-nearest neighbors (k-NN) at different iterations of the algorithm. Thus, k-NN can be defined as follows [36]:

Definition 2 Given a set of N points $P = \{p_1, p_2 \dots p_N\}$ in space, the k-NN$(p_i) = \{nn_1, nn_2 \dots nn_k\}$ represents the k-nearest neighbors set of a certain point p_i where $k = |\text{k-NN}(p_i)|$ and $k < N$, a nearest neighbor $nn_j \in \{nn_1, nn_2 \dots nn_k\}$.

Furthermore, the nearest neighbor is discovered using the distance between points. In this study, we use the euclidean distance between point p_i and nn_j which can be defined as follows [58, 59]:

$$dist(p_i, nn_j) = \sqrt{\sum_{r=1}^{d} (p_{ir} - nn_{jr})^2} \tag{1}$$

where d is the dimension or the number of features.

3.2 NPIR description and steps

The main idea of NPIR is to find the nearest neighbors for the points that are already clustered based on the distance between them and cluster them accordingly. In general, NPIR names the nearest neighbors for the points that are already clustered as *Nearest* points. Thus, NPIR finds the *Nearest* point for an already assigned point and then clusters the *Nearest* point to the same cluster of the assigned point under some conditions which are discussed in this section.

The algorithm takes as an argument a data set containing the points that need to be clustered along with three parameters that need to be determined before running the algorithm, the parameters are:

- The number of clusters (k). The value of k can be any integer larger than 1.
- The indexing ratio (IR). This parameter controls the amount of possible reassignment of points. That is, higher IR value means that the assigned points have more possibility for reassignment. In contrast, lower IR value means that the assigned points have less possibility for reassignment. The value of IR should be between 0 and 1.
- The number of iterations (i). It determines the number of times the algorithm repeats. Therefore, the algorithm terminates when the maximum number of iterations i is reached. The value of i can be any integer larger than 0.

At a preparatory stage, the algorithm generates the distance vector for each point. The distance vector of a point is a vector that contains all the other points in the data set sorted in ascending order according to the distance between them and the corresponding point. In addition, a pointer is defined for each distance vector which indicates the current index of the vector. At this stage, the current index of the pointer for all vectors refers to the first index of the vector. In practice, K-dimensional tree [60–62] can be used instead of the distance vectors for large data sets to calculate the distance between each pair of points which is discussed more further at the end of this section. However, we use the distance vector data structure for small data sets and we consider it in our discussion for simplicity.

Furthermore, suppose we have 4 points in space with defined distances between them as shown in Fig. 1. Each point has its own corresponding distance vector and a pointer at the first index of the vector. For example, a distance vector for a given point P1 consists of points P2, P4, and P3 which are sorted according to the distance between them and P1 with distance values of 3, 6, and 8, respectively. In addition, the current index of the distance vector for P1 is the index of P2 which is defined using the pointer of the distance vector of P1. The figure also shows the distance vector and its corresponding pointer for points P2, P3, and P4. The distance vector and its pointer are used later to select the points that are considered for clustering.

The algorithm starts by randomly assigning an initial point for each cluster resulting in k initial points. The initial points are used to assign other points to the clusters. The algorithm then iterates for a predefined number of iterations according to the i parameter discussed earlier. For each iteration in i, inner iterations are also executed to allow for assignment for the unassigned points and reassignment for the already assigned points. For each inner iteration, one point is considered for assignment using the following operations which are illustrated in an example of 28 points:

- *Election*: It randomly elects one of the points that are already assigned to a cluster and names it as '*Elected*' or shortly as '*E*' which is illustrated in Fig. 2. Upon the assignment of the initial points, the election is done for one of the initial points. In contrast, the remaining elections are done for one of the already assigned points. The election operation is random and is conducted in no predefined order. Furthermore, a point might be elected several time while other points are not elected at all. The randomness behavior of such operation gives advantage of defining different shapes for clustering and enhancing the diversity of searching for data points in space.
- *Selection*: The point at the pointer of the distance vector of the *Elected* is selected and is named '*Nearest*' or shortly as '*N*' and the pointer is incremented by one for possible next selection. Figure 3 illustrates the selection process for the *Nearest*; Fig. 3a illustrates the selection of the *Nearest* and Fig. 3b illustrates incrementing the pointer.
- *Assignment*: To assign the *Nearest* to the cluster of the *Elected*, a check must be made in order to decide if the *Nearest* should be assigned to the same cluster as the *Elected*. The following are possible conditions for the *Nearest*:
  - *Case 1*: The *Nearest* is not yet assigned to a cluster. In this case, the *Nearest* is directly assigned to the cluster of the *Elected* and the *Elected* is marked as the '*Assigner*' or shortly as '*A*' for the *Nearest*. Therefore, the *Assigner* is the point that has been *Elected* and has successfully assigned the point to its cluster (see Fig. 4a). In addition, the *Nearest* is added as a descendant to the *Assigner*. A tree data structure with hash table is used for marking preceding parent and descendants.
  - *Case 2*: The *Nearest* is already assigned to the same cluster of the *Elected*. Therefore, no reassignment is made (see Fig. 4b). As a complementary step, the *Elected* might become the new *Assigner* for the *Nearest* if it is closer to the *Nearest* than its old *Assigner*.
  - *Case 3*: The *Nearest* is already assigned to a different cluster than the *Elected*. In this case, the algorithm checks if the *Nearest* should move from its cluster to the cluster of the *Elected* by measuring the distance between the *Nearest* and its *Assigner*, and the distance between the *Nearest* and the *Elected*. Then it compares the two calculated distances. The following apply:
    - *Case 3a*: If the distance between the *Nearest* and the *Assigner* (d1) is less than or equal to the distance between the *Nearest* and the *Elected* (d2), then the *Nearest* does not move to the cluster of the *Elected* (see Fig. 4c).
    - *Case 3b*: If the distance between the *Nearest* and the *Assigner* (d1) is greater than the distance between the *Nearest* and the *Elected* (d2), the *Nearest* moves to the cluster of the *Elected* (see Fig. 4d) and the *Elected* is marked as the *Assigner* for the *Nearest*. This can avoid the termination of cluster shaping process at early stage. It is highly probable that the points on the boundaries will be finally reassigned according to the corresponding elected point. Unlike partitional clustering algorithms like k-means which usually fall in local optima and get stuck into fixed shapes of clusters, NPIR explores more shapes by reassigning points over the course of iterations and can handle arbitrary shaped clusters.

The inner iterations terminate when all points are assigned and *TotalIndex* value is reached. The *TotalIndex* value is calculated using the following formula:

$$TotalIndex = Round(IR * n^2) \tag{2}$$

where IR is the indexing ratio, and n is the number of points. *TotalIndex* value represents the amount of possible reassignment of points. n^2 represents the size of all the distance vectors since n points have n number of elements in the distance vector. Therefore, *TotalIndex* represents the total indices reached for all the points which is the amount of possible reassignment of points since IR value falls between 0 and 1. This allows for more possible reassignment of points depending on the IR value even after all points are already assigned.

Furthermore, when the inner iterations terminate, pointers are reset and points to the first element of the distance vector for all the points which allows for more possible reassignment of points for the next iteration in i.

NPIR is described by the pseudocode given in Algorithm 1. As shown from the pseudo code, The algorithms

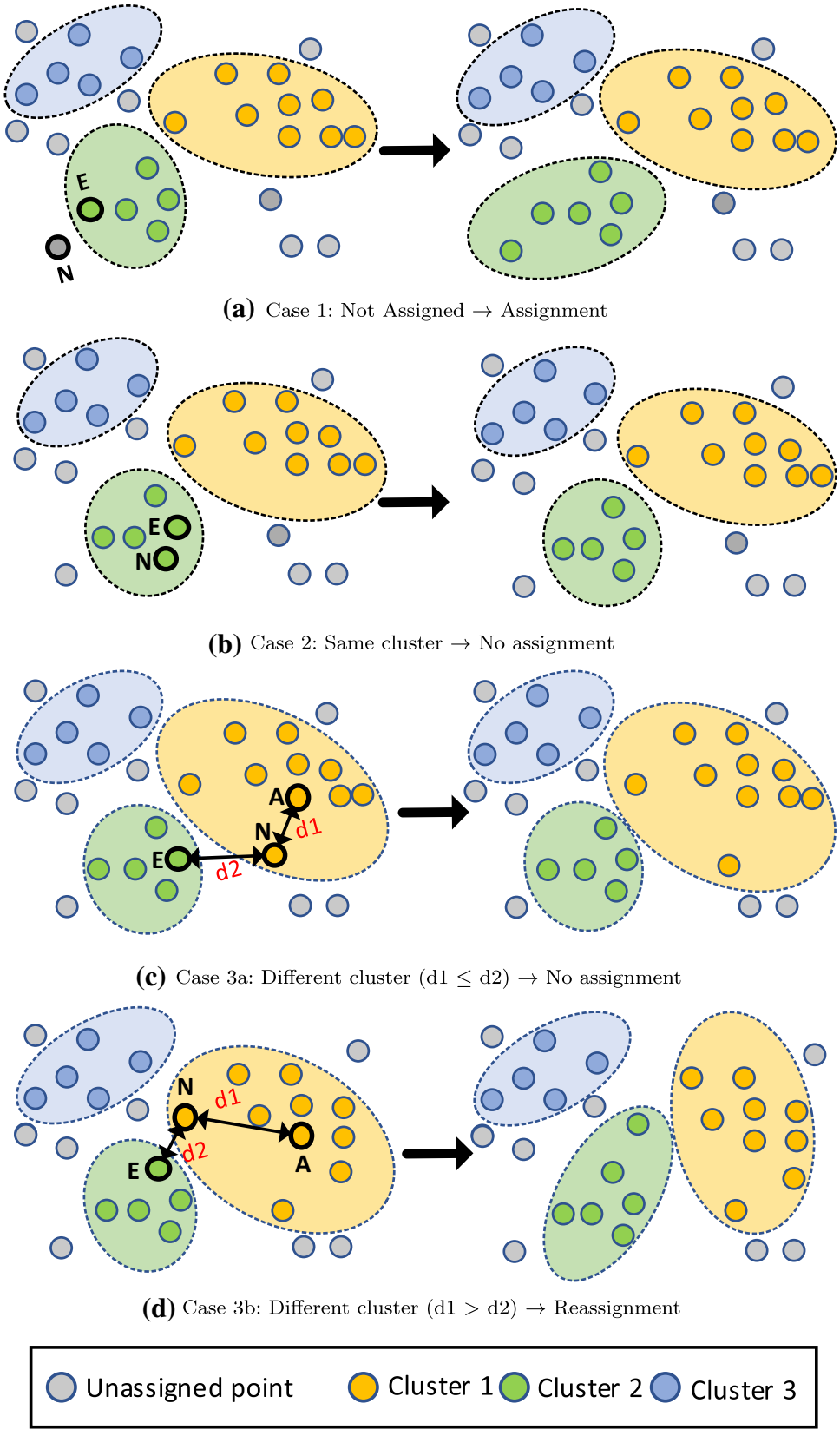


Fig. 4 Assignment operation. **a** Case1: Unassigned *Nearest* is assigned to the cluster of the *Elected*; **b** Case2: No assignment for the *Nearest* if it is assigned to the same cluster of the *Elected*; **c** Case 3a: No assignment for the *Nearest* if it is assigned to a different cluster than the *Elected* and d1 <= d2; **d** Case 3b: Reassignment of the *Nearest* to the cluster of the *Elected* if it is assigned to a different cluster than the *Elected* and d1 > d2

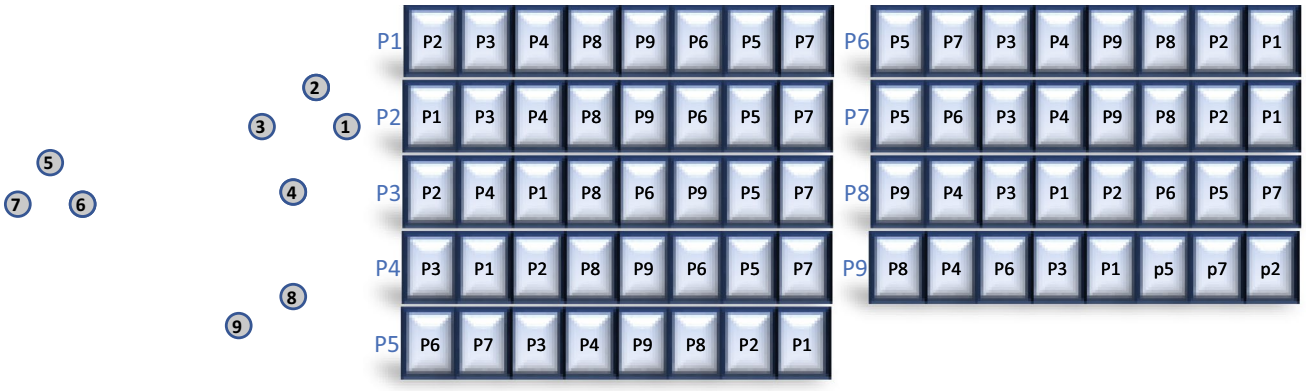


Fig. 5 A simple example of nine points in two-dimensional space and the distance vectors for all the points

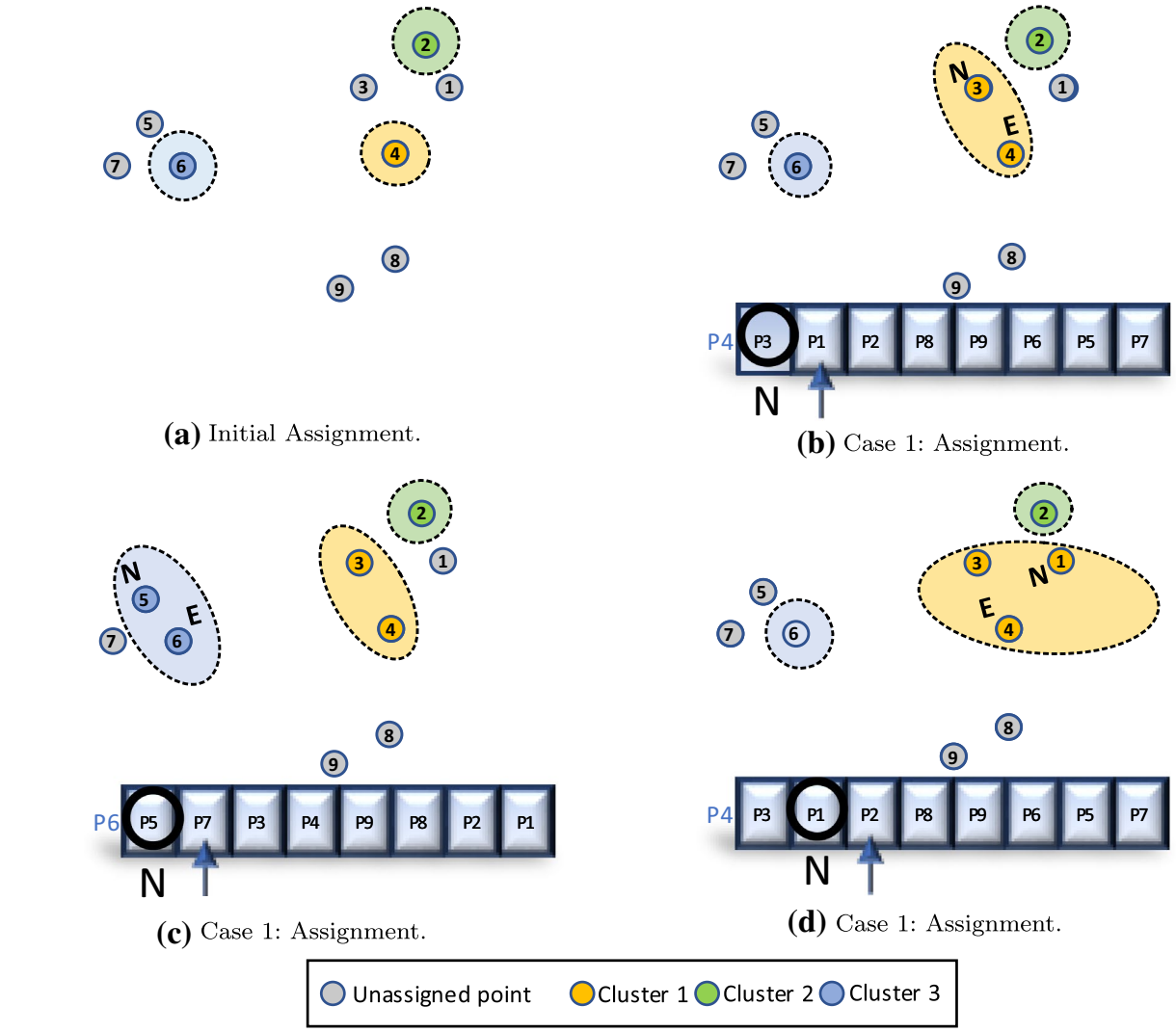
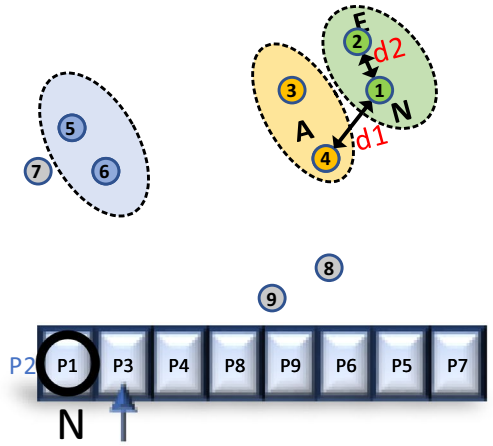
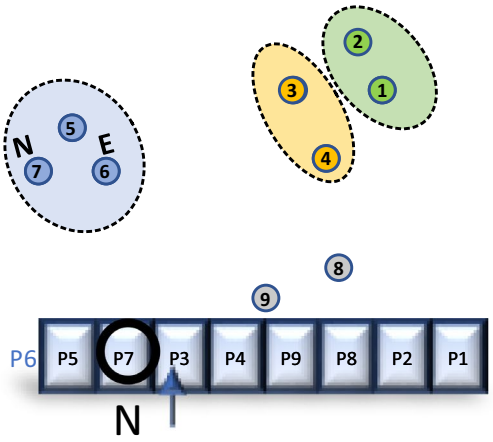


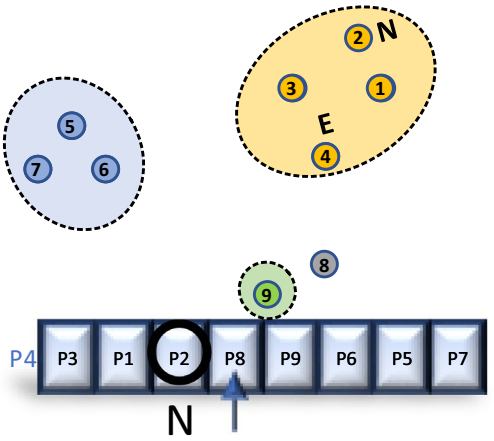
Fig. 6 NPIR applied on a the simple example. **a** Assignment of the initial points; **b** assignment of point 3 based on case 1; **c** assignment of point 5 based on case 1; **d** assignment of point 1 based on case 1; **e** reassignment of point 1 based on case 3b; **f** assignment of point 7 based on case 1; **g** reassignment of point 2&1 and initial assignment for point 9 based on case 3b; **h** no assignment of point 3 based on case 2; **i** assignment of point 8 based on case 1; **j** no reassignment of point 8 based on case 3a



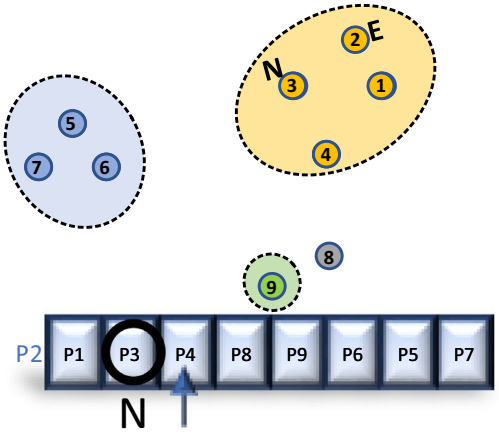
(e) Case 3b: Reassignment.



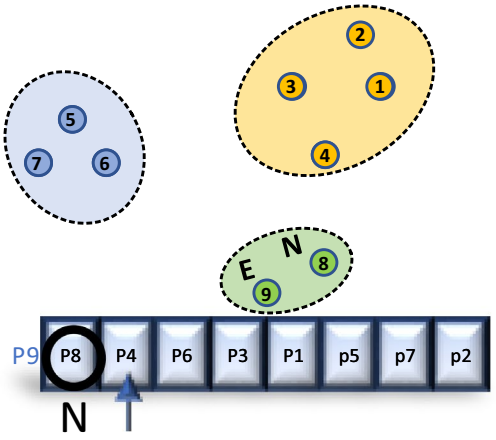
(f) Case 1: Assignment.



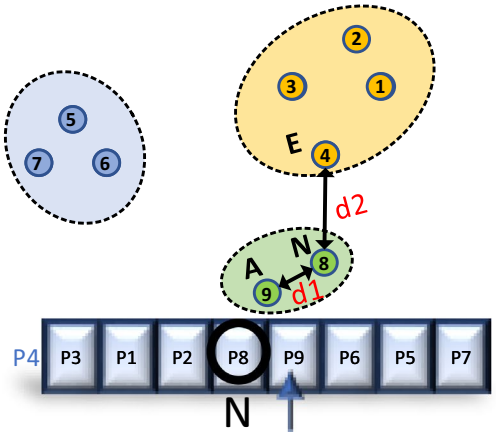
(g) Case 3b: Reassignment.



(h) Case 2: No Assignment.



(i) Case 1: Assignment.



(j) Case 3a: No Reassignment.

Fig. 6 (continued)

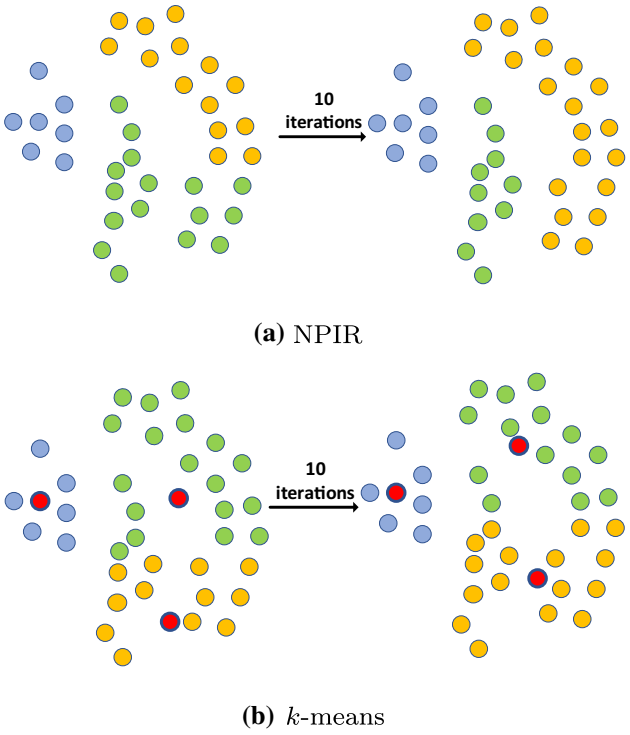


Fig. 7 Detecting arbitrary shaped clusters using NPIR vs. *k*-means

Algorithm 1 NPIR Pseudo Code

Input: Points, K, IR, i
Output: The predicted assignments

```
1: procedure NPIR
2:     Initialize index, iterations
3:     create a distance vector for each point containing
       sorted distances between this point and all the other
       points
4:     TotalIndex =Round(IR * (#Points)^2)
5:     Select k random points as the initial points for the
       clusters
6:     repeat
7:         repeat
8:             Randomly elect an assigned point and mark it
               as Elected (E)
9:             Select the point at the pointer of the distance
               vector of the Elected
10:            Mark the selected points as Nearest (N)
11:            Increment Elected pointer by 1
12:            if Nearest is not assigned yet to a cluster or
                 (Nearest is assigned to a different cluster
                 than the Elected and
                 distance(N,E)<distance(N,A))                then
13:              Assign the Nearest and its descendants to
                 the cluster of the Elected
14:              Mark the Elected as the Assigner (A) for
                 the Nearest
15:              Add the Nearest as a descendant to the
                 Assigner
16:              if Old cluster of the Nearest becomes empty then
17:                  Assign a random point to the old empty
                     cluster of the Nearest
18:              end if
19:            end if
20:        until All points are clustered and index++<TotalIndex
21:        Set the pointer to the first element of the distance
           vector for all points
22:    until iterations++<i
23:    return the predicted assignments
24: end procedure
```

accepts 3 parameters; *k*, *IR*, and *i* along with the set of points that are considered for clustering. The preparatory steps are performed in lines 2–4 which creates the distance vector and calculates the *TotalIndex* value. The algorithm starts in line 5 upon selecting the initial points. *i* iterations are performed in lines 6–22 where it terminates when the value of *i* is reached. For each iteration in *i*, inner iterations are performed to assign points to clusters which are performed in lines 7–20 where it terminates when all points are clustered and *TotalIndex* value is reached. For each inner iteration, the *election*, *selection*, and *assignment* operations are performed in lines 8, 9–11, and 12–19, respectively. Case 1, 2, and 3 of the *assignment* operation are shown in the if statement in line 12. Each iteration in *i* terminates by resetting the pointer for all the distance vectors as shown in line 21. the algorithm then returns the predicted assignment for each point.

The following notes are applied in the algorithm:

– Distance vectors construction can be replaced by K-dimensional tree data structure [60, 61] for faster execution. K-dimensional tree is a binary search tree used with nearest neighbor search tasks to organize data points into K-dimensional space with the aim of finding a nearest point to some other point in the fastest possible way. It can be used in NPIR for large data sets to fasten the process of constructing the nearest neighbors of every data point in space. Thus, we can construct the K-dimensional tree at the preparatory stage and use it in later stages for finding the *Nearest* of an *Elected* in the *Selection* operation. The complex-

ity analysis for both data structures is further explained in Sect. 3.4.

- When an assigned point is selected as the *Nearest* from an *Elected* point and is reassigned to the cluster of the *Elected*, all the descendants of the *Nearest* point are also reassigned to the cluster of the *Elected*. Descendants are the points that have the *Nearest* or any other descendant of the *Nearest* as their *Assigner*.
- If the *IR* value is zero which results in zero value of *TotalIndex*, then the inner iteration terminates just upon assignment of all points. That is, the algorithm executes in its simplest form where an iteration terminates once all points are assigned. Also, if the *IR* value is 1 which is the maximum value for the *IR*, then all the points are marked as *Nearest* by every other point marked as *Elected*. In other words, the pointer for all the points reach the last index for the distance vector for each iteration in *i*. This adds unnecessary complexity having more elections of points with more selections of far points with no reassignments, which is not recommended.
- Upon moving the *Nearest* to the cluster of the *Elected* and if the cluster of the *Nearest* contains the *Nearest* and its descendants only, then a new random point is selected to be clustered to the cluster of the *Nearest* so that the cluster is not left empty and the *k* value is preserved. The random point is selected from the pool of points that are not yet assigned. However, if all the points are already assigned, then the random point is selected from the pool of points that are already assigned and is assigned to the cluster of the *Nearest* along with its descendants.
- An *Elected* point might be elected more than once allowing for the selection of a different *Nearest* by the *Elected* which makes the detection of dense clusters more possible and more reassignments are also possible.
- If case 3 is applicable for the *Nearest* but the *Nearest* does not have an *Assigner* which is the case if it was selected as an initial point, then the decision is made for case 3b and the *Nearest* is reassigned to the cluster of the *Elected* to allow for less identification of null *Assigners* and less dependency on the randomness selection of the initial points which can avoid fixed shaped clustering at early iterations.

3.3 A simple example

To illustrate the algorithm more further, a simple example of nine points is discussed in this section as shown in Figs. 5 and 6. For simplicity, the example considers a value of 1 for the parameter *i* which means that only one iteration is performed.

The algorithm at the preparatory stage calculates the distance between each two points which generates nine vectors where each vector is related to a single point as shown in Fig. 5. For example, point 2, 3, 4, 8, 9, 6, 5, and 7 are sorted according to their distance with point 1 in the first vector having point 2 as the *Nearest* and point 7 as the farthest. Then initial points are randomly set for each cluster as shown in Fig. 6a and the pointers are already reset to the first element of the vectors. The remaining operations are then performed for each sub figure in Fig. 6 as follows:

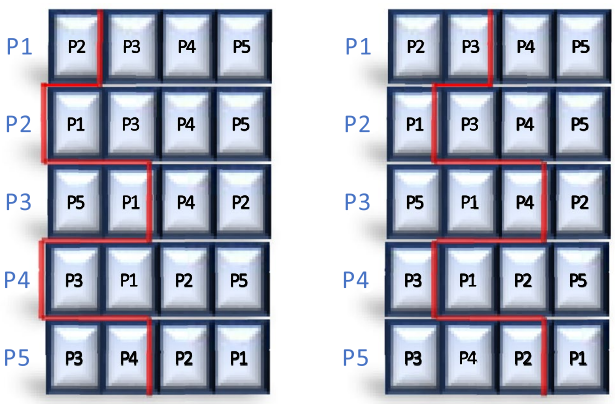
- Figure 6b illustrates the first *election* of random point from the set of assigned points which is point 4. According to the distance vector of point 4, the *Nearest* is *selected* which is point 3. As a result, the *assignment* operation is performed and point 3 is clustered to the same cluster of point 4 according to case 1 discussed in the algorithm. Point 4 becomes the *Assigner* for point 3 and point 3 becomes a descendant of point 4 accordingly. In addition, the pointer of the distance vector for point 4 is incremented by 1 to allow for possible next *election* for point 4 and *selection* for the next *Nearest* which is point 1 in later iterations.
- Figures 6c, d illustrate the next *elections*, *selections*, and *assignments* where point 5 and point 1 are clustered to the cluster of point 6 and point 4 respectively according to case 1 discussed in the algorithm. It is noted in Fig. 6d that the randomness of the election operation in which point 4 is selected again makes possible different clustering alternatives and defines different shapes for clustering.
- Figure 6e illustrates the next *election* of random point from the set of assigned points which is point 2. According to the distance vector of point 2, the nearest point is *selected* which is point 1. As a result, the *assignment* operation is performed and point 1 is reassigned to the cluster of point 2 according to case 3b discussed in the algorithm because the distance d1 between the *Assigner* (point 4) and the *Nearest* (point 1) is greater than the distance d2 between the *Elected* (point 2) and the *Nearest* (point 1). In this case, point 2 becomes the new *Assigner* for point 1. In addition, the pointer of the distance vector for point 2 is incremented by 1 to allow for possible next *election* for point 2 and *selection* for the next *Nearest* which is point 3 in later iterations.
- Figure 6f illustrates the next *election* of random point from the set of assigned points which is point 6. According to the distance vector of point 6, the nearest point is *selected* which is point 7 and the pointer of the distance vector for point 6 is incremented by 1. As a result, the *assignment* operation is performed and Point 7 is clus-

Table 1 Data sets properties which show the name, number of clusters, number of points, number of features, *IR* value, *i* value, and data set type

ID	Data set	k	#Points	#Features	IR	i	Type
1	Aggregation	7	788	2	0.05	50	Artificial
2	Aniso	3	1500	2	0.15	100	Artificial
3	Appendicitis	2	106	7	0.15	100	Real
4	Blobs	3	1500	2	0.2	5	Artificial
5	Circles	2	1500	2	0.05	30	Artificial
6	Diagnosis II	2	120	6	0.2	30	Real
7	Flame	2	240	2	0.2	100	Artificial
8	Glass	6	214	9	0.05	30	Real
9	Iris	3	150	4	0.1	30	Real
10	Iris 2D	3	150	2	0.1	50	Real
11	Jain	2	373	2	0.2	50	Artificial
12	Moons	2	1500	2	0.15	100	Artificial
13	Mouse	3	490	2	0.1	50	Artificial
14	Pathbased	3	300	2	0.1	100	Artificial
15	Seeds	3	210	7	0.05	100	Real
16	Smiley	4	500	2	0.1	50	Artificial
17	Varied	3	1500	2	0.1	100	Artificial
18	Vary density	3	150	2	0.15	50	Artificial
19	WDBC	2	569	30	0.1	50	Real
20	Wine	3	178	13	0.15	50	Real

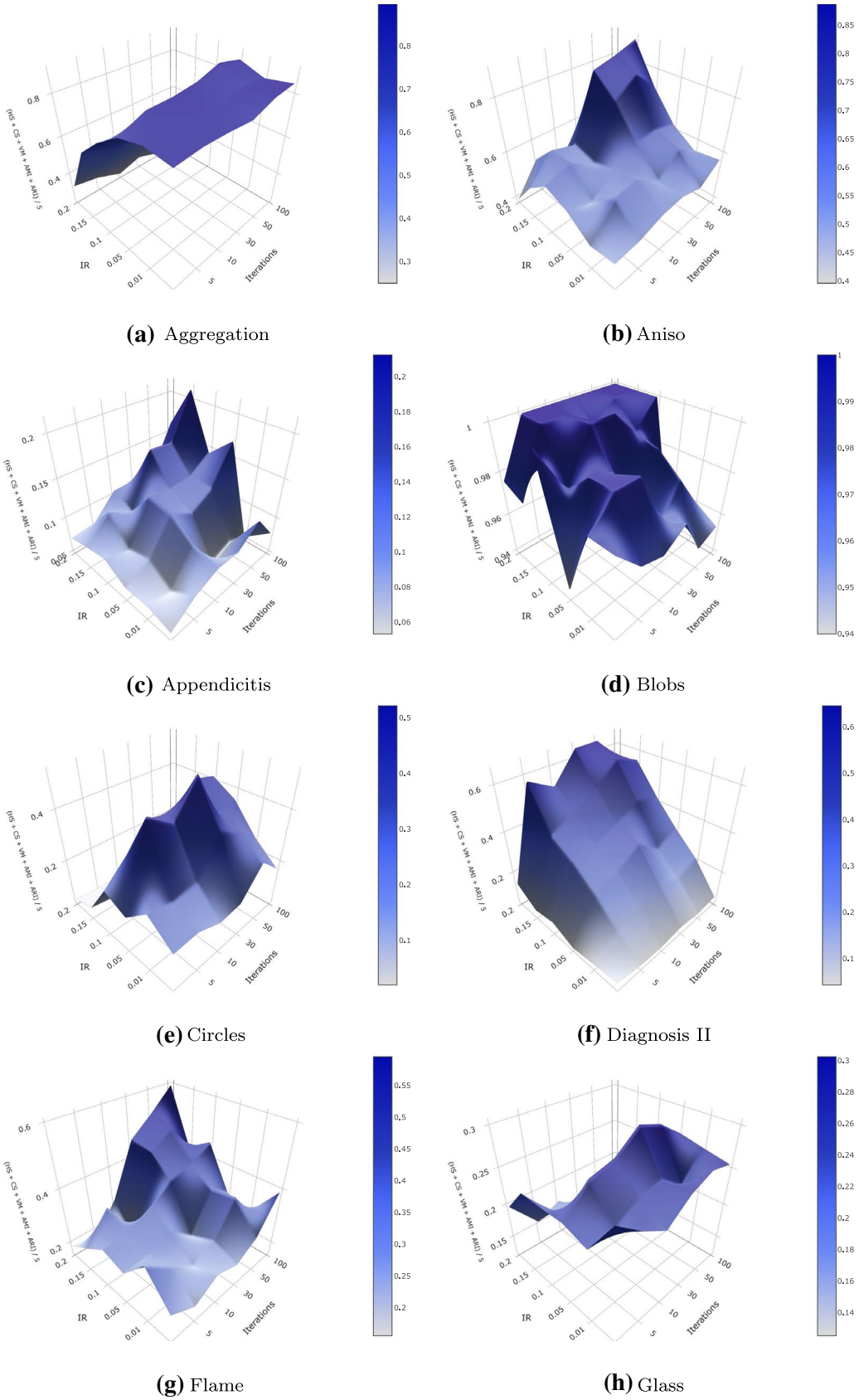
tered to the same cluster of point 6 according to case 1 discussed in the algorithm.

- Figure 6g illustrates the next *election* of random point from the set of assigned points which is point 4. According to the distance vector of point 4, the nearest point is *selected* which is point 2 and the pointer of the distance vector for point 4 is incremented by 1. As a result, the *assignment* operation is performed and point 2 is reassigned to the same cluster of point 4 according to case 3b discussed in the algorithm because the *Nearest* does not have an *Assigner* so the decision is made for case 3b and the *Nearest* is reassigned to the cluster of the *Elected* along with its descendants which is point 1 (point 2 is the *Assigner* to point 1). In this case, point 4 becomes the new *Assigner* for point 2. Furthermore, a random point is selected from the pool of the unassigned points which is point 9 and is assigned to the empty cluster which preserves the *k* value of the clustering. This shows how the algorithm can avoid the termination of cluster shaping process at early stage and can detect arbitrary shapes.
- Figure 6h illustrates the next *election* of random point from the set of assigned points which is point 2. According to the distance vector of point 2, the nearest point is *selected* which is point 3 and the pointer of the distance vector for point 2 is incremented by 1. As a result, the *assignment* operation is performed and no assignment is made for point 3 because it is assigned to the same cluster of point 2 according to case 2 discussed in the algorithm. Furthermore, point 2 become the new *Assigner* for point 3 because it is closer to point 3 than point 4 which is the old *Assigner*.
- Figure 6i illustrates the next *election*, *selection*, and *assignment* where point 8 is clustered to the cluster of point 9 according to case 1 discussed in the algorithm. At this stage all points are successfully assigned.
- Figure 6j illustrates a continued step if the value of the *TotalIndex* is not reached. In this case, the next *election* of random point from the set of assigned points is per-



(a) $TotalIndex = 5$ $IR = 0.2$ **(b)** $TotalIndex = 10$ $IR = 0.4$

Fig. 8 *IR* effect on the election of 5 points for *IR* values of 0.2 and 0.4. The red line shows the index reached for each distance vector. **a** IR = 0.2 and *TotalIndex* = 5; **b** IR = 0.4 and *TotalIndex* = 10



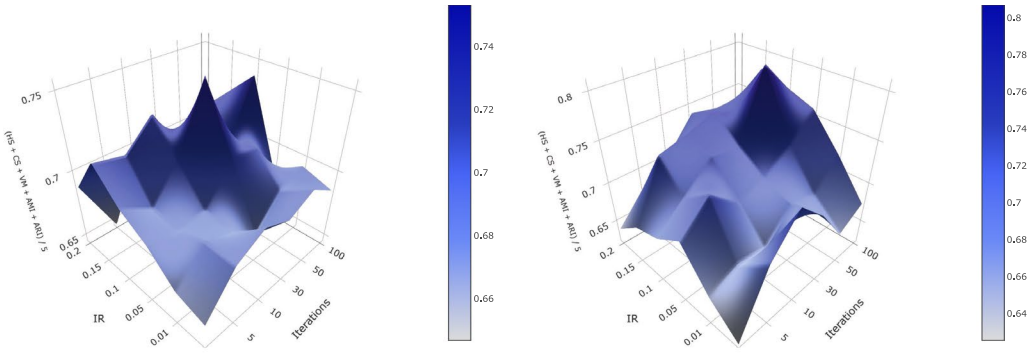
(a) Aggregation **(b)** Aniso

(c) Appendicitis **(d)** Blobs

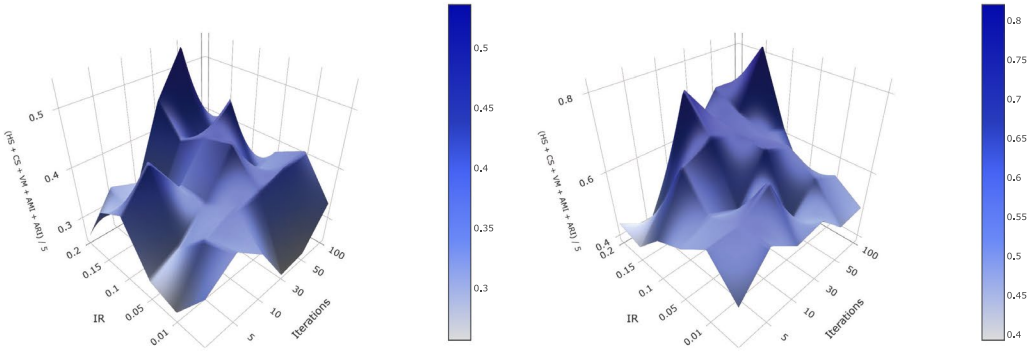
(e) Circles **(f)** Diagnosis II

(g) Flame **(h)** Glass

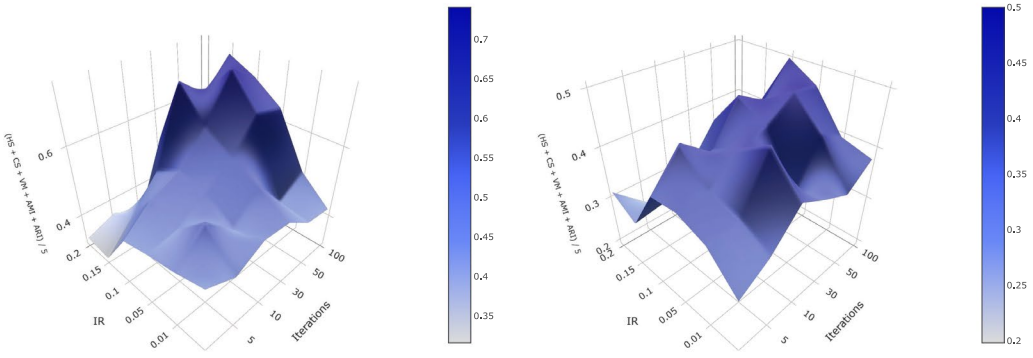
Fig. 9 Sensitivity analysis for *IR* and *i* parameters for each data set for the average results using the aggregation formula



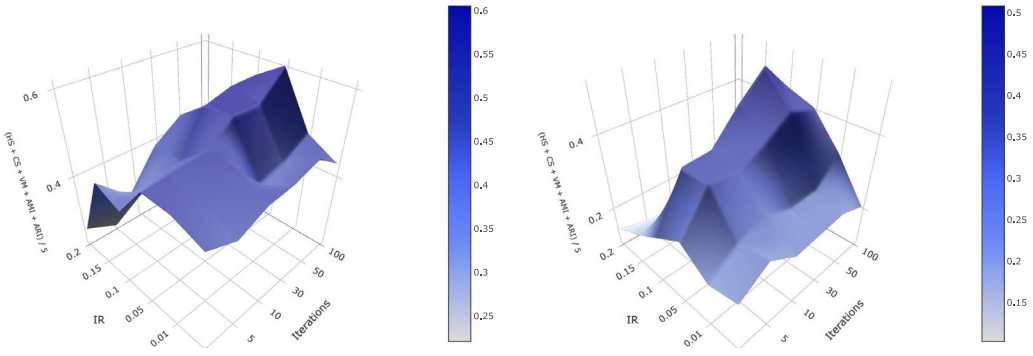
(i) Iris **(j)** Iris 2D



(k) Jain **(l)** Moons



(m) Mouse **(n)** Pathbased



(o) Seeds **(p)** Smiley

Fig. 9 (continued)

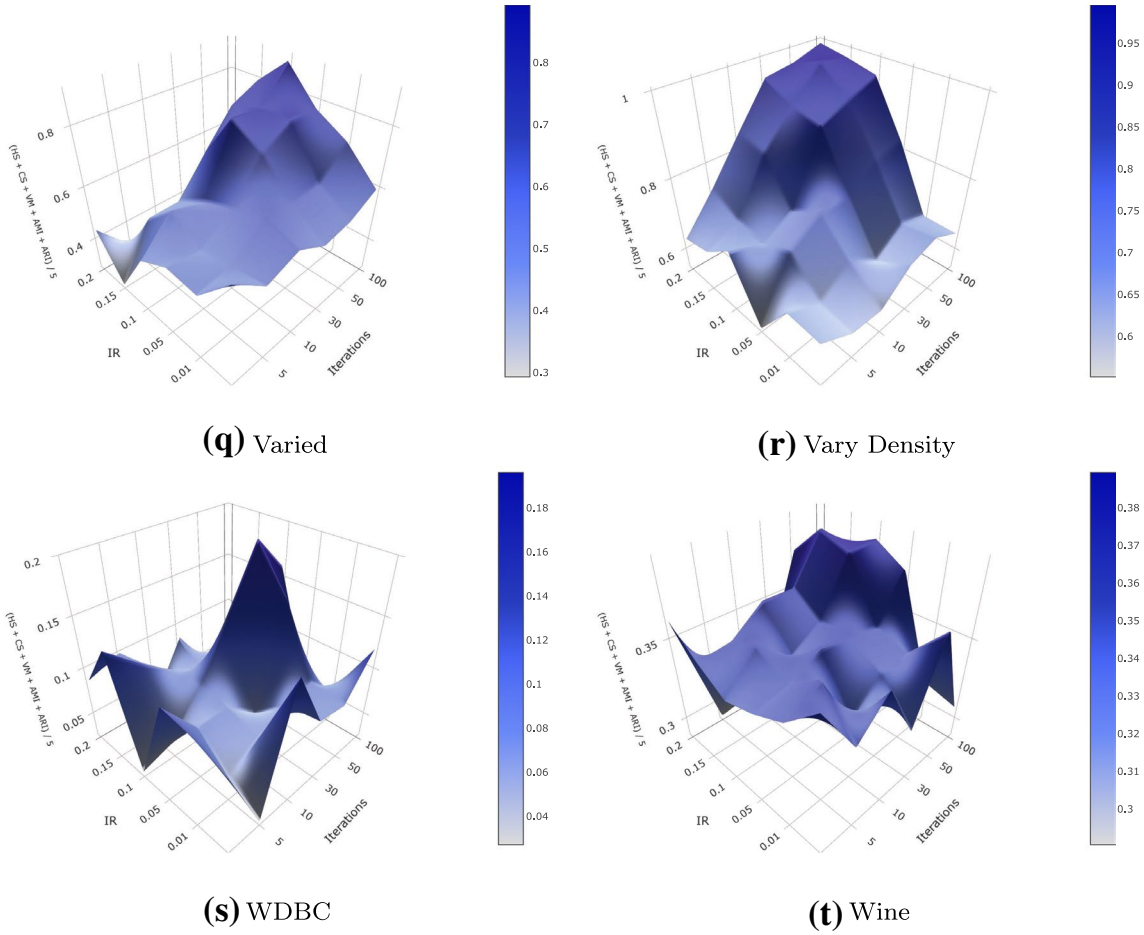


Fig. 9 (continued)

Table 2 Parameters setting

Algorithm	Parameter	Value
k-means++	Iterations	100
BIRCH	Iterations	100
EM	Iterations	100
DCMVO	Iterations	150
	#Search agents	50
GA	Crossover probability	0.8
	Mutation probability	0.001
	#Generations	50
	#Chromosomes	20
PSO	#Generations	50
	#Chromosomes	20

formed which is point 4. According to the distance vector of point 4, the nearest point is *selected* which is point 8 and the pointer of the distance vector for point 4 is incremented by 1. As a result, the *assignment* operation is performed and no reassignment is made for point 8 because the distance d1 between the *Assigner* (point 9) and the *Nearest* (point 8) is less than the distance d2 between the *Elected* (point 4) and the *Nearest* (point 8) according to case 3a discussed in the algorithm. In this case, point 9 remains as an *Assigner* for point 8.

– The algorithm then continues and reassignments are done until the *TotalIndex* value is reached.

As observed from the simple example, NPIR can detect data sets of arbitrary shapes having different cluster densities, points distribution, and different sizes. Unlike the other partitional clustering algorithms such as *k*-means which get stuck into fixed spherical shaped clusters, NPIR explores more shapes over the course of iterations. It is highly probable that the points on the boundaries will be finally reassigned according to the corresponding elected point which can be found in case 3b.

Figure 7 illustrates how different clustering results are achieved using NPIR compared to *k*-means for arbitrary shaped clusters after a course of iterations. It is observed from the figure that *k*-means trapped in fixed shaped clusters

after few iterations and is unable to detect arbitrary shapes because it is a centroid-based clustering algorithm where the red points indicates the centroids of the clusters. Some green and yellow points that are adjacent to the centroids were selected by *k*-means although they belong to different clusters. In each iteration, the assignments of the centroids will be slightly changed resulting in fixed shaping of the clusters around the centroids that were selected. In contrast, NPIR can detect arbitrary shapes by avoiding the termination of the cluster shaping process at early stages and exploring more shapes over the course of iterations due to the reassignment behavior of the points at the boundaries. Some green points of the clusters achieved in early iterations are more adjacent to some other yellow points in different clusters which can be reassigned in later iterations to the cluster of the yellow points through the reassignments of points. This results in exploring more shapes over the course of iterations and finding arbitrary shaped clusters. Section 4 gives more examples of arbitrary shapes detection by NPIR.

3.4 Complexity analysis

As mentioned earlier, two data structures can be used for the construction of the distances between data points and searching for the nearest neighbors. In this section, we discuss time and space complexity of using vectors and the K-dimensional tree [60, 61] as two data structures for implementing the algorithm:

Vectors: The time complexity of the algorithm using the distance vectors can be analyzed as follows:

- NPIP algorithm starts with the preparatory stage which creates the distance vectors by calculating the distance between each point with all the other points taking into consideration the points' dimensionality. This takes $O(n^2d)$ where n is the number of points and d is the number of dimensions. Sorting the distance vector takes $O(nlogn)$.
- The selection of the initial points takes $O(k)$ where k is the number of clusters.
- The *Election* operation takes $O(\frac{in^2}{r})$ where n is the number of points to assign, i is the number of iterations, r is the inverted number of *IR* parameter. Since *IR* is recommended to have small values which should not go beyond the value of 0.2 which is discussed in Sect. 4.3, the *Election* operation can be considered of time $O(\frac{in^2}{5})$.
- The *Selection* operation of pointing to the *Nearest* point takes $O(1)$ since the distance vector of the *Elected* is already sorted.

Table 3 Performance comparison of the average values of Homogeneity score for different algorithms in the form of avg(rank) which indicates the average, ranking, and standard deviation of 30 independent runs, respectively

Homogeneity Score											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.88 (6)	0.9 (4)	0.95 (1)	0.87 (7)	0.93 (3)	0.72 (10)	0.8 (9)	0.95 (1)	0.86 (8)	0.89 (5)	0.36 (11)
Aniso	0.89 (2)	0.61 (7)	0.31 (9)	0.95 (1)	0.66 (3)	N/A (11)[a]	0 (10)	0.43 (8)	0.64 (6)	0.66 (3)	0.66 (3)
Appendicitis	0.17 (3)	0.02 (6)	0 (8)	0 (8)	0.02 (6)	N/A (11)[a]	0 (8)	0.05 (5)	0.08 (4)	0.22 (2)	0.25 (1)
Blobs	1 (1)	0.99 (8)	1 (1)	0.95 (10)	1 (1)	1 (1)	1 (1)	1 (1)	0.98 (9)	0.94 (11)	1 (1)
Circles	0.51 (3)	0.02 (5)	N/A (11)[a]	0.09 (4)	0.02 (5)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	0.63 (4)	0.24 (8)	0.4 (5)	1 (1)	0.14 (11)	1 (1)	1 (1)	0.4 (5)	0.17 (9)	0.15 (10)	0.4 (5)
Flame	0.6 (1)	0.41 (5)	0.21 (7)	0.09 (8)	0.4 (6)	N/A (11)[a]	0.01 (10)	0.06 (9)	0.45 (2)	0.44 (3)	0.44 (3)
Glass	0.28 (6)	0.35 (1)	0.34 (2)	0.3 (3)	0.26 (7)	N/A (11)[a]	0.04 (10)	0.3 (3)	0.25 (8)	0.29 (5)	0.16 (9)
Iris	0.72 (5)	0.74 (3)	0.63 (8)	0.77 (1)	0.73 (4)	0.58 (10)	0.59 (9)	0.7 (7)	0.72 (5)	0.76 (2)	0.43 (11)
Iris2D	0.79 (7)	0.79 (7)	0.84 (3)	0.87 (1)	0.74 (9)	0.58 (10)	0.58 (10)	0.87 (1)	0.83 (6)	0.84 (3)	0.84 (3)
Jain	0.57 (3)	0.39 (4)	0.23 (8)	0.67 (1)	0.21 (9)	N/A (11)[a]	0.18 (10)	0.66 (2)	0.36 (7)	0.37 (5)	0.37 (5)
Moons	0.82 (3)	0.2 (10)	0.22 (6)	0.27(4)	0.05 (11)	1 (1)	1 (1)	0.26 (5)	0.21 (9)	0.22 (6)	0.22 (6)
Mouse	0.78 (1)	N/A (8)[a]	N/A (8)[a]	0.44 (5)	N/A (8)[a]	N/A (8)[a]	0.01 (7)	0.47 (4)	0.68 (3)	0.7 (2)	0.39 (6)
Pathbased	0.48 (5)	0.51 (2)	0.38 (8)	0.71 (1)	0.49 (4)	N/A (11)[a]	0.01 (10)	0.38 (8)	0.47 (6)	0.51 (2)	0.4 (7)
seeds	0.6 (6)	0.69 (1)	0.6 (6)	0.47 (9)	0.54 (8)	N/A (11)[a]	0.04 (10)	0.61 (5)	0.65 (3)	0.69 (1)	0.65 (3)
Smiley	0.68 (3)	0.33 (7)	0.4 (4)	1 (1)	0.36 (5)	0.27 (10)	1 (1)	0.3 (9)	0.33 (7)	0.35 (6)	0.21 (11)
Varied	0.89 (3)	0.8 (6)	0.64 (8)	0.92 (2)	0.93 (1)	N/A (11)[a]	0 (10)	0.52 (9)	0.79 (7)	0.83 (4)	0.83 (4)
VaryDensity	0.99 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0.01 (6)	0.58 (5)	0.74 (4)	0.86 (2)	0.86 (2)
WDBC	0.18 (2)	0 (8)	0 (8)	0.03 (4)	0 (8)	N/A (11)[a]	0.01 (7)	0.1 (3)	0.02 (5)	0.02 (5)	0.47 (1)
Wine	0.39 (7)	0.42 (4)	0.42 (4)	0.4 (6)	0.33 (8)	N/A (11)[a]	0.04 (10)	0.44 (1)	0.43 (2)	0.43 (2)	0.13 (9)
Rank	**65**	**107**	**118**	**78**	**116**	**158**	**131**	**97**	**115**	**84**	**99**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

– The *Assignment* operation is of constant time and can be considered as O(1).

In total, The overall complexity of the algorithm using distance vectors is $O(n^2d+nlogn+k+\frac{in^2}{5})$ which equals $O(n^2d+\frac{in^2}{5})$. The best case complexity of the NPIP algorithm is $O(n^2d+n)$ when i equals 1 and there are no reassignments to be considered. For all cases, the separation between the complexity of the iterative behavior of NPIR algorithm which is $O(\frac{in^2}{5})$ and the complexity of the data sets dimensionality and the number of points which is $O(n^2d)$ can cause a recognizable drop in the running time of the NPIR algorithm compared to other clustering algorithms where these values interchangeably effect the algorithm complexity.

On the other hand, the space complexity of the algorithm using distance vectors is $O(n^2)$ for storing n distance vectors of size n and storing parents-descendents points in a tree of size n.

K-dimensional tree: The time complexity of the algorithm using K-dimensional tree can be analyzed as follows:

– NPIR algorithm starts with the preparatory stage which creates the K-dimensional tree for all data points. This takes $O(dnlogn)$ where n is the number of points and d is the dimension of the tree [60].
– The selection of the initial points takes $O(k)$ where k is the number of clusters.
– The *Election* operation takes $O(\frac{in^2}{r})$ where n is the number of points to assign, i is the number of iterations, r is the inverted number of *IR* parameter. Since *IR* is recommended to have small values which should not go beyond the value of 0.2 which is discussed in Sect. 4.3, the *Election* operation can be considered of time $O(\frac{in^2}{5})$.
– The *Selection* operation of searching for the *Nearest* point takes $O(ilogn)$ in the average case and $O(in^{1-1/d})$ in the worst case [60] which cannot be larger than $O(in)$ where n is the number of points, d is the dimension of the tree, and i is the number of iterations.
– The *Assignment* operation is of constant time and can be considered as O(1).

In total, the overall complexity of the algorithm using K-dimensional tree in the average case is $O(dnlogn+k+\frac{in^2}{5}+ilogn)$ which equals $O(dnlogn+\frac{in^2}{5})$. The best case complexity of the NPIR algorithm is $O(dnlogn+k+n)$ which equals $O(dnlogn)$ when i equals 1 and there are no reassignments to be considered. For all cases, the separation between the complexity of the iterative behavior of NPIR algorithm which is $O(\frac{in^2}{5})$ and the complexity of the data sets dimensionality and the number

Table 4 Performance comparison of the average values of Completeness Score for different algorithms in the form of avg(rank) which indicates the average, ranking, and standard deviation of 30 independent runs, respectively

Completeness Score											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.95 (3)	0.81 (9)	0.86 (6)	0.94 (5)	0.86 (6)	1 (1)	0.98 (2)	0.86 (6)	0.78 (11)	0.81 (9)	0.95 (3)
Aniso	0.9 (1)	0.62 (6)	0.41 (8)	0.37 (9)	0.67 (2)	N/A (11)[a]	0.13 (10)	0.46 (7)	0.66 (3)	0.66 (3)	0.66 (3)
Appendicitis	0.23 (3)	0.22 (4)	1 (1)	1 (1)	0.22 (4)	N/A (11)[a]	0.04 (10)	0.05 (9)	0.08 (8)	0.19 (7)	0.22 (4)
Blobs	1 (1)	0.99 (8)	1 (1)	0.29 (11)	1 (1)	1 (1)	1 (1)	1 (1)	0.99 (8)	0.97 (10)	1 (1)
Circles	0.54 (3)	0.15 (4)	N/A (11)[a]	0.15 (4)	0.15 (4)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	0.68 (3)	0.27 (8)	0.49 (4)	0.31 (7)	0.16 (11)	1 (1)	1 (1)	0.49 (4)	0.18 (9)	0.17 (10)	0.49 (4)
Flame	0.6 (1)	0.39 (5)	0.2 (7)	0.08 (10)	0.38 (6)	N/A (11)[a]	0.18 (8)	0.1 (9)	0.43 (2)	0.42 (3)	0.42 (3)
Glass	0.43 (5)	0.39 (6)	0.55 (1)	0.23 (9)	0.35 (7)	N/A (11)[a]	0.35 (7)	0.52 (3)	0.5 (4)	0.55 (1)	0.22 (10)
Iris	0.85 (3)	0.76 (7)	0.79 (4)	0.39 (11)	0.78 (5)	1 (1)	0.92 (2)	0.75 (8)	0.75 (8)	0.77 (6)	0.7 (10)
Iris2D	0.86 (4)	0.81 (9)	0.84 (7)	0.57 (11)	0.78 (10)	1 (1)	0.95 (2)	0.87 (3)	0.84 (7)	0.85 (5)	0.85 (5)
Jain	0.55 (2)	0.32 (4)	0.21 (8)	0.16 (10)	0.19 (9)	N/A (11)[a]	0.4 (3)	0.74 (1)	0.3 (5)	0.3 (5)	0.3 (5)
Moons	0.83 (3)	0.25 (6)	0.26 (4)	0.2 (10)	0.17 (11)	1 (1)	1 (1)	0.26 (4)	0.22 (7)	0.22 (7)	0.22 (7)
Mouse	0.73 (1)	N/A (8)[a]	N/A (8)[a]	0.42 (5)	N/A (8)[a]	N/A (8)[a]	0.09 (7)	0.41 (6)	0.51 (4)	0.53 (3)	0.65 (2)
Pathbased	0.58 (2)	0.58 (2)	0.44 (9)	0.5 (7)	0.58 (2)	N/A (11)[a]	0.17 (10)	0.45 (8)	0.52 (6)	0.58 (2)	0.63 (1)
seeds	0.64 (5)	0.69 (2)	0.62 (6)	0.29 (9)	0.56 (8)	N/A (11)[a]	0.23 (10)	0.62 (6)	0.66 (3)	0.7 (1)	0.66 (3)
Smiley	0.49 (4)	0.18 (10)	0.22 (6)	1 (1)	0.21 (7)	1 (1)	1 (1)	0.17 (11)	0.19 (9)	0.2 (8)	0.23 (5)
Varied	0.89 (2)	0.81 (5)	0.68 (7)	0.3 (9)	0.93 (1)	N/A (11)[a]	0.14 (10)	0.64 (8)	0.8 (6)	0.83 (3)	0.83 (3)
VaryDensity	0.99 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0.19 (6)	0.84 (4)	0.79 (5)	0.86 (2)	0.86 (2)
WDBC	0.25 (4)	0.12 (9)	0.13 (7)	0.02 (10)	0.13 (7)	N/A (11)[a]	0.15 (6)	0.28 (2)	0.27 (3)	0.17 (5)	0.48 (1)
Wine	0.42 (4)	0.43 (2)	0.42 (4)	0.37 (8)	0.4 (7)	N/A (11)[a]	0.24 (9)	0.45 (1)	0.43 (2)	0.42 (4)	0.24 (9)
Rank	**51**	**119**	**112**	**146**	**116**	**122**	**98**	**107**	**115**	**97**	**79**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

Table 5 Performance comparison of the average values of Vmeasure for different algorithms in the form of avg(rank) which indicates the average, ranking, and standard deviation of 30 independent runs, respectively

Vmeasure											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.91 (1)	0.85 (7)	0.9 (3)	0.9 (3)	0.89 (5)	0.84 (9)	0.88 (6)	0.91 (1)	0.81 (9)	0.85 (7)	0.52 (11)
Aniso	0.89 (1)	0.62 (6)	0.35 (9)	0.54 (7)	0.67 (2)	N/A (11)[a]	0 (10)	0.45 (8)	0.65 (11)	0.66 (3)	0.66 (3)
Appendicitis	0.19 (3)	0.05 (5)	0 (9)	0 (9)	0.05 (5)	N/A (11)[a]	0.01 (8)	0.05 (5)	0.08 (11)	0.2 (2)	0.23 (1)
Blobs	1 (1)	0.99 (8)	1 (1)	0.44 (11)	1 (1)	1 (1)	1 (1)	1 (1)	0.98 (1)	0.96 (10)	1 (1)
Circles	0.52 (3)	0.04 (5)	N/A (11)[a]	0.11 (4)	0.04 (5)	1 (1)	1 (1)	0 (7)	0 (1)	0 (7)	0 (7)
Diagnosis II	0.65 (3)	0.25 (8)	0.44 (5)	0.48 (4)	0.15 (11)	1 (1)	1 (1)	0.44 (5)	0.17 (1)	0.16 (10)	0.44 (5)
Flame	0.6 (1)	0.4 (5)	0.21 (7)	0.08 (8)	0.39 (6)	N/A (11)[a]	0.02 (10)	0.07 (9)	0.44 (11)	0.43 (3)	0.43 (3)
Glass	0.34 (5)	0.36 (4)	0.42 (1)	0.26 (8)	0.3 (7)	N/A (11)[a]	0.07 (10)	0.38 (2)	0.33 (11)	0.38 (2)	0.18 (9)
Iris	0.77 (1)	0.75 (3)	0.7 (9)	0.51 (11)	0.75 (3)	0.73 (5)	0.72 (7)	0.72 (7)	0.73 (5)	0.77 (1)	0.54 (10)
Iris2D	0.82 (6)	0.8 (7)	0.84 (2)	0.69 (11)	0.76 (8)	0.73 (9)	0.72 (10)	0.87 (1)	0.83 (9)	0.84 (2)	0.84 (2)
Jain	0.55 (2)	0.35 (3)	0.22 (9)	0.26 (7)	0.2 (10)	N/A (11)[a]	0.25 (8)	0.7 (1)	0.32 (11)	0.33 (4)	0.33 (4)
Moons	0.82 (3)	0.22 (7)	0.24 (5)	0.23 (6)	0.07 (11)	1 (1)	1 (1)	0.26 (4)	0.21 (1)	0.22 (7)	0.22 (7)
Mouse	0.75 (1)	N/A (8)[a]	N/A (8)[a]	0.43 (6)	N/A (8)[a]	N/A (8)[a]	0.01 (7)	0.44 (5)	0.58 (8)	0.61 (2)	0.49 (4)
Pathbased	0.52 (5)	0.55 (2)	0.41 (8)	0.59 (1)	0.53 (4)	N/A (11)[a]	0.01 (10)	0.41 (8)	0.49 (11)	0.54 (3)	0.49 (6)
seeds	0.62 (5)	0.69 (1)	0.61 (7)	0.36 (9)	0.55 (8)	N/A (11)[a]	0.07 (10)	0.62 (5)	0.65 (11)	0.69 (1)	0.66 (3)
Smiley	0.56 (3)	0.23 (9)	0.29 (5)	1 (1)	0.27 (6)	0.42 (4)	1 (1)	0.21 (11)	0.24 (4)	0.25 (7)	0.22 (10)
Varied	0.89 (2)	0.8 (5)	0.66 (7)	0.45 (9)	0.93 (1)	N/A (11)[a]	0.01 (10)	0.57 (8)	0.79 (11)	0.83 (3)	0.83 (3)
VaryDensity	0.99 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0.03 (6)	0.69 (5)	0.76 (7)	0.86 (2)	0.86 (2)
WDBC	0.18 (2)	0.01 (7)	0.01 (7)	0.02 (6)	0.01 (7)	N/A (11)[a]	0.01 (7)	0.14 (3)	0.04 (11)	0.04 (4)	0.47 (1)
Wine	0.4 (6)	0.43 (2)	0.42 (5)	0.38 (7)	0.36 (8)	N/A (11)[a]	0.06 (10)	0.44 (1)	0.43 (11)	0.43 (2)	0.17 (9)
Rank	**49**	**107**	**120**	**128**	**115**	**145**	**124**	**96**	**113**	**80**	**92**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

of points which is O(*dnlogn*) can cause a recognizable drop in the running time of the NPIR algorithm.

On the other hand, the space complexity of the algorithm is O(n) for storing the data points in the K-dimensional tree and the parents-descendants tree.

Looking into the time complexity for both data structures, distance vectors can be used for small data sets where time and space complexity is not an issue. For large data sets, we recommend using the K-dimensional tree for faster execution and less consuming of space.

4 Experiments and results

This section presents the experiments that are conducted to investigate the efficiency of the proposed NPIR clustering algorithm. NPIR is evaluated based on well-known challenging data sets, which are widely used for the performance tests of clustering algorithms. Furthermore, the performance of NPIR is compared to other well-known clustering algorithms which are *k*-means++ [22], BIRCH [23], HDBSCAN [24], EM [1], MST [25, 26], HC-SL [25, 27, 28], HC-CL [28, 29], GA [30], PSO [31–33], and DCMVO [34]. In addition, NPIR is evaluated using different measures which are Homogeneity Score (HS), Completeness Score (CS), Vmeasure (VM), Adjusted Mutual Information (AMI), and Adjusted Rand Index (ARI) [63, 64].

We used python 3.7 to implement NPIR and Scikit learn python library [65] to evaluate the algorithm using the evaluation measures. We also used scikit Learn[1], Python Package Index (PyPI)[2], Yarpiz packages[3], and MATLAB[4] for the other algorithms in comparison. All experiments were conducted on a personal computer with Intel core i7-5500U CPU @ 2.40 GHz/8 GB RAM.

This section presents the evaluation measures and the data sets used in the experiments. Sensitivity analysis of the algorithm parameters and a quantitative and qualitative analysis of the results are also discussed. Computation time of the algorithm is also provided. In addition, mall customer segmentation case study is discussed and analyzed.

[1] https://scikit-learn.org/stable/modules/clustering.html.
[2] https://pypi.org/.
[3] http://yarpiz.com/64/ypml101-evolutionary-clustering.
[4] https://www.mathworks.com/products/matlab.html.

Table 6 Performance comparison of the average values of AMI for different algorithms in the form of avg(rank) which indicates the average, ranking, and standard deviation of 30 independent runs, respectively

Adjusted Mutual Information											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.87 (1)	0.81 (6)	0.85 (5)	0.86 (2)	0.86 (2)	0.72 (10)	0.8 (8)	0.86 (2)	0.77 (9)	0.81 (6)	0.36 (11)
Aniso	0.89 (1)	0.61 (6)	0.31 (9)	0.37 (8)	0.66 (2)	N/A (11)[a]	0 (10)	0.43 (7)	0.64 (5)	0.66 (2)	0.66 (2)
Appendicitis	0.16 (3)	0.01 (6)	0 (8)	0 (8)	0.01 (6)	N/A (11)[a]	−0.01(10)	0.04 (5)	0.06 (4)	0.18 (2)	0.21 (1)
Blobs	1 (1)	0.99 (8)	1 (1)	0.28 (11)	1 (1)	1 (1)	1 (1)	1 (1)	0.98 (9)	0.94 (10)	1 (1)
Circles	0.51 (3)	0.02 (5)	N/A (11)[a]	0.09 (4)	0.02 (5)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	0.63 (3)	0.23 (8)	0.39 (4)	0.3 (7)	0.13 (11)	1 (1)	1 (1)	0.39 (4)	0.16 (9)	0.15 (10)	0.39 (4)
Flame	0.59 (1)	0.38 (5)	0.2 (7)	0.07 (8)	0.38 (5)	N/A (11)[a]	0.01 (10)	0.06 (9)	0.43 (2)	0.41 (3)	0.41 (3)
Glass	0.26 (5)	0.32 (1)	0.32 (1)	0.19 (8)	0.23 (6)	N/A (11)[a]	0.02 (10)	0.27 (3)	0.23 (6)	0.27 (3)	0.14 (9)
Iris	0.72 (4)	0.74 (2)	0.63 (7)	0.37 (11)	0.73 (3)	0.58 (8)	0.58 (8)	0.7 (6)	0.72 (4)	0.76 (1)	0.43 (10)
Iris2D	0.79 (6)	0.79 (6)	0.83 (4)	0.56 (11)	0.74 (8)	0.58 (9)	0.58 (9)	0.87 (1)	0.83 (4)	0.84 (2)	0.84 (2)
Jain	0.54 (2)	0.32 (3)	0.21 (7)	0.16 (10)	0.19 (8)	N/A (11)[a]	0.18 (9)	0.66 (1)	0.3 (4)	0.3 (4)	0.3 (4)
Moons	0.82 (3)	0.2 (9)	0.22 (5)	0.2 (9)	0.05 (11)	1 (1)	1 (1)	0.26 (4)	0.21 (8)	0.22 (5)	0.22 (5)
Mouse	0.73 (1)	N/A (8)[a]	N/A (8)[a]	0.41 (4)	N/A (8)[a]	N/A (8)[a]	0 (7)	0.41 (4)	0.51 (3)	0.53 (2)	0.39 (6)
Pathbased	0.48 (4)	0.51 (1)	0.37 (9)	0.5 (3)	0.48 (4)	N/A (11)[a]	0 (10)	0.38 (8)	0.47 (6)	0.51 (1)	0.4 (7)
seeds	0.59 (6)	0.68 (2)	0.59 (6)	0.27 (9)	0.54 (8)	N/A (11)[a]	0.03 (10)	0.6 (5)	0.64 (4)	0.69 (1)	0.65 (3)
Smiley	0.47 (3)	0.18 (9)	0.22 (5)	1 (1)	0.2 (7)	0.27 (4)	1 (1)	0.16 (11)	0.18 (9)	0.19 (8)	0.21 (6)
Varied	0.89 (2)	0.8 (5)	0.64 (7)	0.29 (9)	0.93 (1)	N/A (11)[a]	0 (10)	0.52 (8)	0.79 (6)	0.83 (3)	0.83 (3)
VaryDensity	0.99 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0 (6)	0.57 (5)	0.74 (4)	0.85 (2)	0.85 (2)
WDBC	0.17 (2)	0 (7)	0 (7)	0.02 (4)	0 (7)	N/A (11)[a]	0 (7)	0.1 (3)	0.02 (4)	0.02 (4)	0.47 (1)
Wine	0.38 (6)	0.41 (4)	0.41 (4)	0.35 (7)	0.32 (8)	N/A (11)[a]	0.02 (10)	0.43 (1)	0.42 (2)	0.42 (2)	0.13 (9)
Rank	**52**	**104**	**118**	**134**	**110**	**149**	**129**	**94**	**107**	**76**	**87**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

4.1 Evaluation measures

The evaluation of a clustering algorithm is not simply specified by the correctness of the predicted clusters against the true classes. Other considerations should be taken into account such as the separateness and cohesion of the clusters or the similarity between the data points in the same predicted cluster and the dissimilarity between data points in different clusters. With this regards, several measures can be considered for the evaluation as follows:

Given a set S of N points, C a true partitioning assignment of S, K a predicted partitioning assignment of S.

– *Homogeneity Score:* For all clusters, it measures if all data points of a specific cluster are in the same class [63].

Given $H(C)$ which is the classes entropy [63]:

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{N} \cdot \log\left(\frac{n_c}{N}\right) \tag{3}$$

where n_c is the number of data points of class c.

Furthermore, given the assignments of the clusters K, $H(C|K)$ is the classes conditional entropy [63]:

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|} \frac{n_{kc}}{N} \cdot \log\left(\frac{n_{kc}}{n_k}\right) \tag{4}$$

where n_{kc} is the number of data points from class c assigned to cluster k and n_k is the number of data points of cluster k.

The Homogeneity Score is then given by [63]:

$$\text{HS} = 1 - \frac{H(C|K)}{H(C)}. \tag{5}$$

– *Completeness Score:* For all classes, it measures if all data points of a specific class are clustered in the same cluster [63]

Given $H(K)$ which is the clusters entropy [63]:

$$H(K) = -\sum_{k=1}^{|K|} \frac{n_k}{N} \cdot \log\left(\frac{n_k}{N}\right). \tag{6}$$

Furthermore, given the data points of the classes C, $H(K|C)$ is the clusters conditional entropy [63]:

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{kc}}{N} \cdot \log\left(\frac{n_{kc}}{n_c}\right). \tag{7}$$

Table 7 Performance comparison of the average values of ARI Score for different algorithms in the form of avg(rank) which indicates the average, ranking, and standard deviation of 30 independent runs, respectively

Adjusted Rand Index											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.86 (1)	0.73 (7)	0.78 (6)	0.85 (2)	0.8 (3)	0.73 (7)	0.8 (3)	0.79 (5)	0.68 (10)	0.72 (9)	0.36 (11)
Aniso	0.87 (1)	0.59 (6)	0.18 (9)	0.3 (8)	0.66 (2)	N/A (11)[a]	0 (10)	0.36 (7)	0.63 (5)	0.66 (2)	0.66 (2)
Appendicitis	0.31 (2)	0.04 (5)	0 (7)	0 (7)	0.04 (5)	N/A (11)[a]	−0.01(9)	−0.09(10)	0.05 (4)	0.3 (3)	0.38 (1)
Blobs	1 (1)	0.98 (9)	1 (1)	0.12 (11)	1 (1)	1 (1)	1 (1)	1 (1)	0.99 (8)	0.93 (10)	1 (1)
Circles	0.52 (3)	0 (5)	N/A (11)[a]	0.02 (4)	0 (5)	1 (1)	1 (1)	0 (5)	0 (5)	0 (5)	0 (5)
Diagnosis II	0.63 (3)	0.22 (8)	0.41 (4)	0.23 (7)	0.14 (10)	1 (1)	1 (1)	0.41 (4)	0.15 (9)	0.11 (11)	0.41 (4)
Flame	0.59 (1)	0.44 (5)	0.2 (7)	−0.03(9)	0.39 (6)	N/A (11)[a]	0.01 (8)	−0.04(10)	0.48 (2)	0.46 (3)	0.46 (3)
Glass	0.21 (4)	0.2 (6)	0.25 (1)	0.07 (9)	0.18 (7)	N/A (11)[a]	0.01 (10)	0.23 (3)	0.21 (4)	0.24 (2)	0.12 (8)
Iris	0.7 (3)	0.72 (2)	0.57 (7)	0.31 (11)	0.66 (5)	0.57 (7)	0.56 (9)	0.64 (6)	0.69 (4)	0.75 (1)	0.47 (10)
Iris2D	0.78 (6)	0.77 (7)	0.85 (2)	0.64 (9)	0.67 (8)	0.57 (10)	0.57 (10)	0.89 (1)	0.83 (5)	0.85 (2)	0.85 (2)
Jain	0.48 (2)	0.3 (3)	0.02 (9)	0.04 (8)	−0.01(10)	N/A (11)[a]	0.26 (4)	0.78 (1)	0.24 (7)	0.26 (4)	0.26 (4)
Moons	0.82 (3)	0.19 (9)	0.21 (8)	0.14 (10)	0.01 (11)	1 (1)	1 (1)	0.33 (4)	0.28 (7)	0.29 (5)	0.29 (5)
Mouse	0.72 (1)	N/A (8)[a]	N/A (8)[a]	0.24 (5)	N/A (8)[a]	N/A (8)[a]	−0.01(7)	0.24 (5)	0.43 (3)	0.46 (2)	0.39 (4)
Pathbased	0.44 (4)	0.46 (2)	0.33 (9)	0.56 (1)	0.44 (4)	N/A (11)[a]	0 (10)	0.35 (8)	0.42 (6)	0.46 (2)	0.4 (7)
seeds	0.58 (5)	0.7 (2)	0.57 (6)	0.13 (9)	0.53 (8)	N/A (11)[a]	0 (10)	0.55 (7)	0.66 (3)	0.71 (1)	0.63 (4)
Smiley	0.34 (3)	0.03 (7)	0.04 (6)	1 (1)	0.02 (11)	0.29 (4)	1 (1)	0.03 (7)	0.03 (7)	0.03 (7)	0.05 (5)
Varied	0.91 (2)	0.81 (5)	0.61 (7)	0.2 (9)	0.96 (1)	N/A (11)[a]	0 (10)	0.51 (8)	0.8 (6)	0.85 (3)	0.85 (3)
VaryDensity	1 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0 (6)	0.51 (5)	0.71 (4)	0.87 (2)	0.87 (2)
WDBC	0.2 (2)	0 (6)	0 (6)	−0.01(10)	0 (6)	N/A (11)[a]	0 (6)	0.1 (3)	0.02 (4)	0.02 (4)	0.6 (1)
Wine	0.35 (6)	0.36 (5)	0.37 (2)	0.28 (7)	0.27 (8)	N/A (11)[a]	0.01 (10)	0.37 (2)	0.37 (2)	0.38 (1)	0.17 (9)
Rank	**48**	**109**	**121**	**137**	**118**	**146**	**117**	**100**	**103**	**78**	**82**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

The Completeness Score is then given by [63]:

$$\mathrm{CS} = 1 - \frac{H(K|C)}{H(K)}. \tag{8}$$

– *V-measure:* V-measure measures the agreement between the true partitioning assignments and the predicted partitioning assignments. V-measure is a harmonic mean between homogeneity and completeness as defined by Rosenberg and Hirschberg [63]:

$$VM = 2 \cdot \frac{HS \cdot CS}{HS + CS}. \tag{9}$$

– *Adjusted Rand Index (ARI):* ARI considers pair counting to measure the similarity between two assignments [64] which is the true and the predicted assignments. The Rand index is calculated using the following formula [66, 67]:

$$\mathrm{RI} = \frac{a+b}{\binom{N}{2}} = \frac{\sum_{k,c}\binom{n_{kc}}{2}}{\binom{N}{2}} \tag{10}$$

Where a is the number of pairs of points that are at the same class of C and at the same cluster of K, b is the number of pairs of points that are in different class in C and in different cluster in K, and $\binom{N}{2}$ is the total number of pairs in S.

Furthermore, E[RI], max[RI] are the expected RI and maximum RI which are given by [67]:

$$\mathrm{E(RI)} = E\left(\sum_{k,c}\binom{n_{kc}}{2}\right) = \frac{\sum_{k=1}^{|K|}\binom{n_k}{2}\sum_{c=1}^{|C|}\binom{n_c}{2}}{\binom{N}{2}} \tag{11}$$

$$\max[\mathrm{RI}] = \frac{1}{2}\left[\sum_{k=1}^{|K|}\binom{n_k}{2} + \sum_{c=1}^{|C|}\binom{n_c}{2}\right]. \tag{12}$$

Correction by chance is achieved for the rand index and is called the Adjusted Rand Index which is given by [67]:

$$\mathrm{ARI} = \frac{\mathrm{RI} - E[\mathrm{RI}]}{\max(\mathrm{RI}) - E[\mathrm{RI}]}. \tag{13}$$

Table 8 Performance comparison of the average values of grand average score for different algorithms in the form of avg(rank) which indicates the average, ranking, and standard deviation of 30 independent runs, respectively

Grand Average Score											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.9 (1)	0.82 (7)	0.868 (4)	0.884 (2)	0.868 (4)	0.8 (9)	0.85 (6)	0.87 (3)	0.779 (10)	0.816 (8)	0.511 (11)
Aniso	0.89 (1)	0.61 (6)	0.312 (9)	0.506 (7)	0.664 (2)	N/A (11)[a]	0.03 (10)	0.43 (8)	0.646 (5)	0.661 (3)	0.66 (4)
Appendicitis	0.21 (3)	0.07 (6)	0.2 (4)	0.2 (4)	0.07 (6)	N/A (11)[a]	0.01 (10)	0.02 (9)	0.07 (6)	0.22 (2)	0.26 (1)
Blobs	1 (1)	0.988 (8)	1 (1)	0.416 (11)	1 (1)	1 (1)	1 (1)	1 (1)	0.985 (9)	0.949 (10)	1 (1)
Circles	0.52 (3)	0.046 (5)	N/A (11)[a]	0.092 (4)	0.046 (5)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	0.65 (3)	0.242 (8)	0.426 (6)	0.464 (4)	0.144 (11)	1 (1)	1 (1)	0.43 (5)	0.167 (9)	0.151 (10)	0.426 (7)
Flame	0.6 (1)	0.404 (5)	0.204 (7)	0.058 (8)	0.388 (6)	N/A (11)[a]	0.05 (9)	0.05 (9)	0.447 (2)	0.432 (3)	0.432(3)
Glass	0.3 (6)	0.324 (4)	0.376 (1)	0.21 (8)	0.264 (7)	N/A (11)[a]	0.1 (10)	0.34 (3)	0.304 (5)	0.345 (2)	0.163 (9)
Iris	0.75 (2)	0.742 (3)	0.664 (9)	0.47 (11)	0.73 (4)	0.69 (7)	0.67 (8)	0.7 (6)	0.723 (5)	0.761 (1)	0.511 (10)
Iris2D	0.81 (6)	0.792 (7)	0.84 (4)	0.666 (11)	0.738 (8)	0.69 (9)	0.68 (10)	0.87 (1)	0.83 (5)	0.846 (2)	0.844 (3)
Jain	0.54 (2)	0.336 (3)	0.178 (9)	0.258 (7)	0.156 (10)	N/A (11)[a]	0.25 (8)	0.71 (1)	0.302 (6)	0.312 (4)	0.312 (4)
Moons	0.82 (3)	0.212 (9)	0.23 (7)	0.208 (10)	0.07 (11)	1 (1)	1 (1)	0.27 (4)	0.228 (8)	0.234 (5)	0.234 (5)
Mouse	0.74 (1)	N/A (8)[a]	N/A (8)[a]	0.388 (6)	N/A (8)[a]	N/A (8)[a]	0.02 (7)	0.39 (5)	0.544 (3)	0.565 (2)	0.462 (4)
Pathbased	0.5 (5)	0.522 (2)	0.386 (9)	0.572 (1)	0.504 (4)	N/A (11)[a]	0.04 (10)	0.39 (8)	0.474 (6)	0.52 (3)	0.464 (7)
seeds	0.61 (5)	0.69 (2)	0.598 (7)	0.304 (9)	0.544 (8)	N/A (11)[a]	0.07 (10)	0.6 (6)	0.651 (4)	0.698 (1)	0.653 (3)
Smiley	0.51 (3)	0.19 (9)	0.234 (5)	1 (1)	0.212 (6)	0.45 (4)	1 (1)	0.17 (11)	0.192 (8)	0.202 (7)	0.184 (10)
Varied	0.89 (2)	0.804 (5)	0.646 (7)	0.432 (9)	0.936 (1)	N/A (11)[a]	0.03 (10)	0.55 (8)	0.792 (6)	0.834 (3)	0.834 (3)
VaryDensity	0.99 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0.05 (6)	0.64 (5)	0.748 (4)	0.86 (2)	0.86 (2)
WDBC	0.2 (2)	0.026 (9)	0.028 (7)	0.016 (10)	0.028 (7)	N/A (11)[a]	0.03 (6)	0.14 (3)	0.076 (4)	0.054 (5)	0.497 (1)
Wine	0.39 (6)	0.41 (4)	0.408 (5)	0.356 (7)	0.336 (8)	N/A (11)[a]	0.07 (10)	0.43 (1)	0.415 (3)	0.417 (2)	0.168 (9)
Rank	**51**	**113**	**122**	**130**	**116**	**147**	**125**	**103**	**112**	**80**	**95**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

– *Adjusted Mutual Information (AMI):* AMI considers Shannon information theory to measures the similarity between two partitioning assignments [64] which is the true and the predicted assignments. The Mutual index is calculated using the following formula [64]:

$$\mathrm{MI} = \sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{n_{kc}}{N}\log\left(\frac{\frac{n_{kc}}{N}}{\frac{n_k}{N}\cdot\frac{n_c}{N}}\right). \tag{14}$$

Furthermore, E[MI] is the expected MI which is given by [64, 68]:

$$\mathrm{E(MI)} = \sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\sum_{n_{kc}=max(0,n_k+n_c-N)}^{min(n_k,n_c)}\frac{n_{kc}}{N}\log\left(\frac{N.n_{kc}}{n_k n_c}\right) \times\left(\frac{n_k!n_c!(N-n_k)!(N-n_c)!}{N!n_{kc}!(n_k-n_{kc})!(n_c-n_{kc})!(N-n_k-n_c+n_{kc})!}\right). \tag{15}$$

Given H (C) and H (K) as the entropy of the partitioning assignment from formulas 3 and 6, correction by chance is achieved for the mutual information and is called the Adjusted Mutual Information which is given by [64]:

$$\mathrm{AMI} = \frac{\mathrm{MI} - E[\mathrm{MI}]}{\max(H(K), H(C)) - E[\mathrm{MI}]}. \tag{16}$$

4.2 Data sets

Twenty data sets are selected with different number of features, points, and clusters which reflects different levels of complexity. These data sets are gathered from scikit learn[5], UCI machine learning repository[6] [69], School of Computing at University of Eastern Finland[7], ELKI[8], KEEL[9], and Naftali Harris Blog[10].

Table 1 shows the name and type of these data sets along with the number of clusters (k), features, and points. The values of IR and i parameters that are used in the experiments are also observed for each data set. Moreover, a brief description of these data sets are listed below:

– **A***ggregation*[7]: A two-dimensional data set consists of seven clusters of different shapes and sizes having some connections between some of them.

[5] http://scikit-learn.org/stable/datasets/index.html.

[6] https://archive.ics.uci.edu/ml/.

[7] http://cs.uef.fi/sipu/datasets/.

[8] https://elki-project.github.io/datasets/.

[9] https://sci2s.ugr.es/keel/datasets.php.

[10] https://www.naftaliharris.com/blog/visualizing-K-means-clustering/.

Table 9 Performance comparison of the best values of Homogeneity Score for different algorithms in the form of best(rank) which indicates the best and ranking of 30 independent runs

Homogeneity Score											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.98 (2)	0.93 (5)	0.95 (3)	0.87 (8)	1 (1)	0.72 (10)	0.8 (9)	0.95 (3)	0.92 (7)	0.93 (5)	0.36 (11)
Aniso	1 (1)	0.68 (5)	0.31 (9)	0.95 (2)	0.77 (4)	N/A (11)[a]	0 (10)	0.43 (8)	0.92 (3)	0.67 (6)	0.66 (7)
Appendicitis	0.36 (1)	0.03 (6)	0 (8)	0 (8)	0.03 (6)	N/A (11)[a]	0 (8)	0.05 (5)	0.28 (3)	0.29 (2)	0.25 (4)
Blobs	1 (1)	1 (1)	1 (1)	0.95 (11)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
Circles	1 (1)	0.02 (5)	N/A (11)[a]	0.09 (4)	0.02 (5)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	1 (1)	0.49 (7)	0.4 (9)	1 (1)	0.49 (7)	1 (1)	1 (1)	0.4 (9)	1 (1)	1 (1)	0.4 (9)
Flame	0.97 (1)	0.49 (3)	0.21 (7)	0.09 (8)	0.44 (4)	N/A (11)[a]	0.01 (10)	0.06 (9)	0.51 (2)	0.44 (4)	0.44 (4)
Glass	0.41 (1)	0.39 (2)	0.34 (3)	0.3 (7)	0.31 (6)	N/A (11)[a]	0.04 (10)	0.3 (7)	0.32 (5)	0.34 (3)	0.18 (9)
Iris	0.87 (1)	0.78 (4)	0.63 (8)	0.77 (5)	0.74 (6)	0.58 (10)	0.59 (9)	0.7 (7)	0.8 (2)	0.79 (3)	0.45 (11)
Iris2D	0.9 (1)	0.84 (7)	0.84 (7)	0.87 (3)	0.74 (9)	0.58 (10)	0.58 (10)	0.87 (3)	0.9 (1)	0.86 (5)	0.86 (5)
Jain	1 (1)	0.42 (5)	0.23 (8)	0.67 (2)	0.21 (9)	N/A (11)[a]	0.18 (10)	0.66 (3)	0.43 (4)	0.37 (6)	0.37 (6)
Moons	1 (1)	0.22 (7)	0.22 (7)	0.27 (4)	0.05 (11)	1 (1)	1 (1)	0.26 (6)	0.27 (4)	0.22 (7)	0.22 (7)
Mouse	0.97 (1)	N/A (8)[a]	N/A (8)[a]	0.44 (5)	N/A (8)[a]	N/A (8)[a]	0.01 (7)	0.47 (4)	0.77 (2)	0.75 (3)	0.4 (6)
Pathbased	0.74 (1)	0.52 (3)	0.38 (8)	0.71 (2)	0.49 (6)	N/A (11)[a]	0.01 (10)	0.38 (8)	0.52 (3)	0.51 (5)	0.4 (7)
seeds	0.7 (4)	0.73 (1)	0.6 (7)	0.47 (9)	0.56 (8)	N/A (11)[a]	0.04 (10)	0.61 (6)	0.72 (2)	0.72 (2)	0.69 (5)
Smiley	1 (1)	0.35 (8)	0.4 (4)	1 (1)	0.37 (5)	0.27 (10)	1 (1)	0.3 (9)	0.37 (5)	0.36 (7)	0.21 (11)
Varied	0.94 (1)	0.8 (7)	0.64 (8)	0.92 (3)	0.93 (2)	N/A (11)[a]	0 (10)	0.52 (9)	0.83 (4)	0.83 (4)	0.83 (4)
VaryDensity	1 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0.01 (6)	0.58 (5)	0.88 (2)	0.86 (3)	0.86 (3)
WDBC	0.68 (1)	0 (8)	0 (8)	0.03 (6)	0 (8)	N/A (11)[a]	0.01 (7)	0.1 (4)	0.15 (3)	0.05 (5)	0.47 (2)
Wine	0.43 (4)	0.43 (4)	0.42 (6)	0.4 (8)	0.42 (6)	N/A (11)[a]	0.04 (10)	0.44 (1)	0.44 (1)	0.44 (1)	0.13 (9)
Rank	**23**	**99**	**131**	**96**	**113**	**158**	**131**	**113**	**61**	**79**	**119**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

- **A***niso*[5]: A two-dimensional data set consists of 3 unconnected isotropic Gaussian blobs with similar densities. They interleave at some points in the *X* and *Y* axis.
- **A***ppendicitis*[9]: Represents the medical status of 106 patients for having appendicitis based on 7 measures.
- **B***lobs*[5]: A two-dimensional data set consists of 3 unconnected isotropic Gaussian blobs with similar densities forming spherical shapes. It is usually used to show and prove clustering as the cluster centers and standard deviation is easily controlled. That is, a clustering algorithm should at least satisfy predicting such data set.
- **C***ircles*[5]: A two-dimensional data set consists of 2 clusters of different sizes forming 2 circles that are not connected to each other. One circle is surrounded with the other one. The two circles need to be visualized with a spherical boundary for binary clustering.
- **D***iagnosisII*[6]: Represents the medical status of 120 patients for having a nephritis of renal pelvis origin based on 6 symptoms.
- **F***lame*[7]: A two-dimensional data set consists of two connected clusters forming the shape of a flower.
- **G***lass*[6]: Represents the types of glass for 214 items based on their chemical elements.
- **I***ris*[6]: It consists of 4 characteristics of 150 iris plants to detect the type of each iris plant. The characteristics are the petal length, the petal width, the sepal length, and the sepal width in centimeters.
- **I***ris2D*[6]: A modification of the previous data set found in Weka tool [70] considering only two characteristics of the iris plants which are the petal length and width.
- **J***ain*[7]: A two-dimensional data set consists of two half circles of different densities that are not connected with each other but interleave in the x and y axis for some points.
- **M***oons*[5]: A two-dimensional data set consists of 2 interleaving half circles with similar densities. The two half circles need to be visualized with a spherical boundary for binary clustering. They are not connected to each other but they interleave at some points in the *X* and *Y* axis.
- **M***ouse*[8]: A two-dimensional data set consists of three connected clusters of different sizes that form the ears and face of Mickey Mouse which is a popular cartoon character.
- **P***athbased*[7]: A two-dimensional data set consists of 3 unconnected clusters forming 2 spherical shape and an

Table 10 Performance comparison of the best values of Completeness Score for different algorithms in the form of best(rank) which indicates the best and ranking of 30 independent runs

Completeness Score											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.98 (3)	0.84 (10)	0.86 (7)	0.94 (6)	1 (1)	1 (1)	0.98 (3)	0.86 (7)	0.84 (10)	0.86 (7)	0.96 (5)
Aniso	1 (1)	0.68 (4)	0.41 (8)	0.37 (9)	0.77(3)	N/A (11)[a]	0.13 (10)	0.46 (7)	0.92 (2)	0.67 (5)	0.66 (6)
Appendicitis	0.39 (3)	0.29 (4)	1 (1)	1 (1)	0.29(4)	N/A (11)[a]	0.04 (10)	0.05 (9)	0.25 (7)	0.29 (4)	0.22 (8)
Blobs	1 (1)	1 (1)	1 (1)	0.29 (11)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
Circles	1 (1)	0.15 (4)	N/A (11)[a]	0.15 (4)	0.15(4)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	1 (1)	0.49 (6)	0.49 (6)	0.31 (11)	0.49(6)	1 (1)	1 (1)	0.49 (6)	1 (1)	1 (1)	0.49 (6)
Flame	0.96 (1)	0.46 (3)	0.2 (7)	0.08 (10)	0.41(6)	N/A (11)[a]	0.18 (8)	0.1 (9)	0.48 (2)	0.42 (4)	0.42 (4)
Glass	0.61 (1)	0.47 (6)	0.55 (4)	0.23 (9)	0.45(7)	N/A (11)[a]	0.35 (8)	0.52 (5)	0.59 (3)	0.6 (2)	0.22 (10)
Iris	0.95 (2)	0.79 (6)	0.79 (6)	0.39 (11)	0.78(8)	1 (1)	0.92 (3)	0.75 (9)	0.81 (4)	0.8 (5)	0.73 (10)
Iris2D	0.92 (3)	0.84 (8)	0.84 (8)	0.57 (11)	0.78(10)	1 (1)	0.95 (2)	0.87 (5)	0.9 (4)	0.86 (6)	0.86 (6)
Jain	1 (1)	0.35 (4)	0.21 (8)	0.16 (10)	0.19(9)	N/A (11)[a]	0.4 (3)	0.74 (2)	0.35 (4)	0.3 (6)	0.3 (6)
Moons	1 (1)	0.26 (5)	0.26 (5)	0.2 (10)	0.18(11)	1 (1)	1 (1)	0.26 (5)	0.27 (4)	0.22 (8)	0.22 (8)
Mouse	0.98 (1)	N/A (8)[a]	N/A (8)[a]	0.42 (5)	N/A (8)[a]	N/A (8)[a]	0.09 (7)	0.41 (6)	0.59 (3)	0.56 (4)	0.67 (2)
Pathbased	0.76 (1)	0.59 (3)	0.44 (9)	0.5 (7)	0.58 (5)	N/A (11)[a]	0.17 (10)	0.45 (8)	0.59 (3)	0.58 (5)	0.63 (2)
seeds	0.76 (1)	0.73 (2)	0.62 (6)	0.29 (9)	0.57 (8)	N/A (11)[a]	0.23 (10)	0.62 (6)	0.73 (2)	0.72 (4)	0.69 (5)
Smiley	1 (1)	0.2 (10)	0.22 (6)	1 (1)	0.21 (7)	1 (1)	1 (1)	0.17 (11)	0.21 (7)	0.21 (7)	0.23 (5)
Varied	0.94 (1)	0.81 (6)	0.68 (7)	0.3 (9)	0.93 (2)	N/A (11)[a]	0.14 (10)	0.64 (8)	0.84 (3)	0.84 (3)	0.83 (5)
VaryDensity	1 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0.19 (6)	0.84 (5)	0.89 (2)	0.86 (3)	0.86 (3)
WDBC	0.7 (2)	0.13 (7)	0.13 (7)	0.02 (10)	0.13 (7)	N/A (11)[a]	0.15 (6)	0.28 (4)	1 (1)	0.24 (5)	0.48 (3)
Wine	0.56 (1)	0.45 (3)	0.42 (7)	0.37 (8)	0.49 (2)	N/A (11)[a]	0.24 (9)	0.45 (3)	0.45 (3)	0.43 (6)	0.24 (9)
Rank	**27**	**104**	**122**	**151**	**114**	**122**	**101**	**120**	**70**	**87**	**102**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

uncompleted circle. The two spherical shapes interleave with the uncompleted circles in the *X* and *Y* axis.

- **S***eeds*[6]: Represents 3 different varieties of wheat for 210 elements based on the geometric parameters of wheat kernels.
- **S***miley*[10]: A two-dimensional data set which forms an outline of the face having circular boundary, two eyes, and a mouth. The face boundary interleaves in the *X* and *Y* axis with the eyes and mouth.
- **V***aried*[5]: A two-dimensional data set consists of 3 isotropic Gaussian blobs connected with each other. The blobs are of different densities having spherical shapes.
- **V***aryDensity*[8]: A two-dimensional data set consists of 3 gaussian blobs with variable densities.
- **W***DBC*[6]: Presents a cell nuclei digitized image characteristics of 569 breast mass. Binary clustering is performed to detect if the breast mass is malignant or benign.
- **W***ine*[6]: Represents 13 constituents quantities for each instance to detect one of the three types of wine.

4.3 Sensitivity analysis

As stated earlier, the algorithm has two parameters; *IR* and *i*. The *IR* controls the amount of possible reassignment of points and *i* determines the number of times the algorithm repeats. The *IR* values of 0, 0.01, 0.05, 0.1, 0.15 and 0.2 and the *i* values of 1, 5, 10, 30, 50, and 100 are used for the experiments. The best *IR* and *i* values for the average results are selected for each data set as observed from Table 1.

The effect of the *IR* parameter can be observed from Fig. 8. The figure shows two values of *IR* which are 0.2 and 0.4 for a data set of 5 points. When *IR* value equals 0.2, the *TotalIndex* value is 5 according to Eq. (2). This is observed from Fig. 8a which shows that 5 different considerations of assignments and reassignments are taken place having 5 different election, selection, and assignment operations for each iteration in *i*. The red line in the figure shows possible index reached for each distance vector of each point. For example, point 3 and 5 have been elected 2 times having the *Nearest* as point 5, 1, 3, and 4, respectively. Point 1 has been elected only once, point 2 and 4 have never been elected. In

Table 11 Performance comparison of the best values of Vmeasure for different algorithms in the form of best(rank) which indicates the best and ranking of 30 independent runs

Vmeasure											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.98 (2)	0.88 (7)	0.9 (4)	0.9 (4)	1 (1)	0.84 (10)	0.88 (7)	0.91 (3)	0.87 (9)	0.89 (6)	0.52 (11)
Aniso	1 (1)	0.68 (4)	0.35 (9)	0.54 (7)	0.77 (3)	N/A (11)[a]	0 (10)	0.45 (8)	0.92 (2)	0.67 (5)	0.66 (6)
Appendicitis	0.37 (1)	0.06 (5)	0 (9)	0 (9)	0.06 (5)	N/A (11)[a]	0.01 (8)	0.05 (7)	0.26 (3)	0.29 (2)	0.23 (4)
Blobs	1 (1)	1 (1)	1 (1)	0.44 (11)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
Circles	1 (1)	0.04 (5)	N/A (11)[a]	0.11 (4)	0.04 (5)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	1 (1)	0.49 (6)	0.44 (9)	0.48 (8)	0.49 (6)	1 (1)	1 (1)	0.44 (9)	1 (1)	1 (1)	0.44 (9)
Flame	0.96 (1)	0.47 (3)	0.21 (7)	0.08 (8)	0.42 (6)	N/A (11)[a]	0.02 (10)	0.07 (9)	0.5 (2)	0.43 (4)	0.43 (4)
Glass	0.45 (1)	0.43 (2)	0.42 (4)	0.26 (8)	0.36 (7)	N/A (11)[a]	0.07 (10)	0.38 (6)	0.4 (5)	0.43 (2)	0.19 (9)
Iris	0.87 (1)	0.79 (4)	0.7 (9)	0.51 (11)	0.76 (5)	0.73 (6)	0.72 (7)	0.72 (7)	0.8 (2)	0.8 (2)	0.56 (10)
Iris2D	0.9 (1)	0.84 (6)	0.84 (6)	0.69 (11)	0.76 (8)	0.73 (9)	0.72 (10)	0.87 (3)	0.9 (1)	0.86 (4)	0.86 (4)
Jain	1 (1)	0.38 (4)	0.22 (9)	0.26 (7)	0.2 (10)	N/A (11)[a]	0.25 (8)	0.7 (2)	0.39 (3)	0.33 (5)	0.33 (5)
Moons	1 (1)	0.24 (6)	0.24 (6)	0.23 (8)	0.07 (11)	1 (1)	1 (1)	0.26 (5)	0.27 (4)	0.22 (9)	0.22 (9)
Mouse	0.98 (1)	N/A (8)[a]	N/A (8)[a]	0.43 (6)	N/A (8)[a]	N/A (8)[a]	0.01 (7)	0.44 (5)	0.67 (2)	0.64 (3)	0.5 (4)
Pathbased	0.75 (1)	0.55 (3)	0.41 (8)	0.59 (2)	0.53 (6)	N/A (11)[a]	0.01 (10)	0.41 (8)	0.55 (3)	0.54 (5)	0.49 (7)
seeds	0.7 (4)	0.73 (1)	0.61 (7)	0.36 (9)	0.56 (8)	N/A (11)[a]	0.07 (10)	0.62 (6)	0.72 (2)	0.72 (2)	0.69 (5)
Smiley	1 (1)	0.25 (9)	0.29 (5)	1 (1)	0.27 (6)	0.42 (4)	1 (1)	0.21 (11)	0.27 (6)	0.27 (6)	0.22 (10)
Varied	0.94 (1)	0.8 (6)	0.66 (7)	0.45 (9)	0.93 (2)	N/A (11)[a]	0.01 (10)	0.57 (8)	0.84 (3)	0.83 (4)	0.83 (4)
VaryDensity	1 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0.03 (6)	0.69 (5)	0.88 (2)	0.86 (3)	0.86 (3)
WDBC	0.69 (1)	0.01 (7)	0.01 (7)	0.02 (6)	0.01 (7)	N/A (11)[a]	0.01 (7)	0.14 (4)	0.2 (3)	0.09 (5)	0.47 (2)
Wine	0.46 (1)	0.43 (5)	0.42 (7)	0.38 (8)	0.45 (2)	N/A (11)[a]	0.06 (10)	0.44 (3)	0.44 (3)	0.43 (5)	0.17 (9)
Rank	**23**	**94**	**133**	**136**	**112**	**147**	**125**	**114**	**61**	**76**	**114**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

contrast, when the *IR* value equals 0.4, the *TotalIndex* value is 10 according to Eq. (2). This is observed from Fig. 8b which shows that when *IR* value becomes larger, more possible reassignments of points are considered by comparing Fig. 8a with b.

Since *IR* parameter is considered after all points are clustered, it allows for more possible selection of different elected points and more possible reassignment of points which might correct wrong clustered points. This does not effect the right assignments of points as per case 3 in the Assignment operation discussed previously.

On the other hand, the effect of the *i* parameter results into iterative resetting of the pointers of all the distance vectors. This is also useful by allowing for corrections of wrong clustered points when the same point has been elected many times beyond what is required for a specific iteration in *i*. That is, when the pointers of all distance vectors are reset then more possible selections of different elected points are considered and reassignments of points are done to correct wrong clustered points. This also does not effect the right assignments of points as per case 3 in the assignment operation discussed previously.

IR and *i* parameters also help in avoiding the termination of cluster shaping process at early stage through the iterations while exploring different shapes of clusters because more possible selection of different elected points and more possible reassignment of points are considered. Thus, exploration of different clusters can be achieved which can help in detecting arbitrary shaped clusters. However, we don't recommend large value of *IR* and *i* which is beyond what is required because it may result in unnecessary complexity having more elections of points with more selections of far points with no reassignments. *IR* values beyond 0.2 and *i* values beyond 100 is not recommended for NPIR.

Since the algorithm is evaluated using multiple measures which are: HS, CS, VM, AMI, and ARI, then the decision of selecting the best values of *IR* and *i* is considered a multi-objective problem which are solved using aggregating, population-based non Pareto and Pareto-based techniques [71]. We use the conventional weighted aggregation formula as the Evaluation (*E*) for the purpose of analyzing the sensitivity of the *IR* and *i* values which is the most common approach for coping with multi-objective problems [72]. The weighted aggregation works by summing up all the objective measures to a weighted combination according to the following formula [72]:

Table 12 Performance comparison of the best values of AMI for different algorithms in the form of best(rank) which indicates the best and ranking of 30 independent runs

Adjusted Mutual Information											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.98 (2)	0.84 (7)	0.85 (5)	0.86 (3)	1 (1)	0.72 (10)	0.8 (9)	0.86 (3)	0.83 (8)	0.85 (5)	0.36 (11)
Aniso	1 (1)	0.68 (4)	0.31 (9)	0.37 (8)	0.77 (3)	N/A (11)[a]	0 (10)	0.43 (7)	0.92 (2)	0.67 (5)	0.66 (6)
Appendicitis	0.35 (1)	0.02 (6)	0 (8)	0 (8)	0.02 (6)	N/A (11)[a]	−0.01(10)	0.04 (5)	0.24 (3)	0.28 (2)	0.21 (4)
Blobs	1 (1)	1 (1)	1 (1)	0.28 (11)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
Circles	1 (1)	0.02 (5)	N/A (11)[a]	0.09 (4)	0.02 (5)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	1 (1)	0.48 (6)	0.39 (8)	0.3 (11)	0.48 (6)	1 (1)	1 (1)	0.39 (8)	1 (1)	1 (1)	0.39 (8)
Flame	0.96 (1)	0.46 (3)	0.2 (7)	0.07 (8)	0.41 (4)	N/A (11)[a]	0.01 (10)	0.06 (9)	0.48 (2)	0.41 (4)	0.41 (4)
Glass	0.39 (1)	0.36 (2)	0.32 (3)	0.19 (8)	0.28 (6)	N/A (11)[a]	0.02 (10)	0.27 (7)	0.3 (5)	0.32 (3)	0.16 (9)
Iris	0.87 (1)	0.78 (4)	0.63 (7)	0.37 (11)	0.74 (5)	0.58 (8)	0.58 (8)	0.7 (6)	0.8 (2)	0.79 (3)	0.44 (10)
Iris2D	0.9 (1)	0.83 (6)	0.83 (6)	0.56 (11)	0.74 (8)	0.58 (9)	0.58 (9)	0.87 (3)	0.9 (1)	0.86 (4)	0.86 (4)
Jain	1 (1)	0.35 (3)	0.21 (7)	0.16 (10)	0.19 (8)	N/A (11)[a]	0.18 (9)	0.66 (2)	0.35 (3)	0.3 (5)	0.3 (5)
Moons	1 (1)	0.22 (6)	0.22 (6)	0.2 (10)	0.05 (11)	1 (1)	1 (1)	0.26 (5)	0.27 (4)	0.22 (6)	0.22 (6)
Mouse	0.97 (1)	N/A (8)[a]	N/A (8)[a]	0.41 (4)	N/A (8)[a]	N/A (8)[a]	0 (7)	0.41 (4)	0.59 (2)	0.56 (3)	0.4 (6)
Pathbased	0.74 (1)	0.51 (2)	0.37 (9)	0.5 (5)	0.48 (6)	N/A (11)[a]	0 (10)	0.38 (8)	0.51 (2)	0.51 (2)	0.4 (7)
seeds	0.7 (4)	0.73 (1)	0.59 (7)	0.27 (9)	0.55 (8)	N/A (11)[a]	0.03 (10)	0.6 (6)	0.72 (2)	0.71 (3)	0.68 (5)
Smiley	1 (1)	0.19 (10)	0.22 (5)	1 (1)	0.21 (6)	0.27 (4)	1 (1)	0.16 (11)	0.21 (6)	0.2 (9)	0.21 (6)
Varied	0.94 (1)	0.8 (6)	0.64 (7)	0.29(9)	0.93 (2)	N/A (11)[a]	0 (10)	0.52 (8)	0.83 (3)	0.83 (3)	0.83 (3)
VaryDensity	1 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0 (6)	0.57 (5)	0.88 (2)	0.85 (3)	0.85 (3)
WDBC	0.68 (1)	0 (7)	0 (7)	0.02 (6)	0 (7)	N/A (11)[a]	0 (7)	0.1 (4)	0.15 (3)	0.05 (5)	0.47 (2)
Wine	0.43 (1)	0.42 (5)	0.41 (6)	0.35 (8)	0.41 (6)	N/A (11)[a]	0.02 (10)	0.43 (1)	0.43 (1)	0.43 (1)	0.13 (9)
Rank	**23**	**94**	**128**	**144**	**108**	**149**	**130**	**109**	**59**	**74**	**107**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

$$E = \sum_{i=1}^{m} w_i f_i(x) \tag{17}$$

where m is the number of objective measures and w_i is a non negative weight for each objective measure. The sum of 1 for the weights ($\sum_{i=1}^{m} w_i = 1$) should be considered.

For simplicity and for the purpose of the sensitivity analysis we are conducting, the Evaluation of each combination of *IR* and *i* can be obtained by considering an equal weight for each measure. This forms the average values of the objective measures as shown by the following formula [73]:

$$E = \sum_{i=1}^{m} \frac{1}{m} f_i(x). \tag{18}$$

The sensitivity analysis for each data set for the average results using the average formula are observed in Fig. 9. It is observed from the sensitivity graphs that NPIR performs differently for different values of *IR* and *i* for the majority of the data sets. In contrast, it performs approximately equal for all values of *i* for the Aggregation and Blobs data sets. Circles, Glass, and Iris data sets need middle values of *IR* and *i* to give good results while larger values of *IR* and *i* are needed for the Flame data set. However, some data sets perform better for large values of *IR* such as Diagnosis II and Jain or large values of *i* such as Aniso, Appendicitis, Moons, Pathbased, and Varied. Finally, WDBC, Appendicitis, Flame, and Jain data set has a recognizable enhanced results for specific values of *IR* and *i*.

4.4 Results and discussion

To evaluate the proposed algorithm, the experiments are conducted on each data set for 30 independent runs. The best and average results are listed against the other well-known clustering algorithm which are *k*-means++, BIRCH, HDBSCAN, EM, MST, HC-SL, HC-CL, GA, PSO, and DCMVO using the evaluation measures listed earlier. For NPIR, the *IR* values of 0, 0.01, 0.05, 0.1, 0.15 and 0.2 and the *i* values of 1, 5, 10, 30, 50, and 100 are experimented as discussed in Sect. 4.3 and the best values among these are selected.

For the other clustering algorithms, the value of 100 is considered as the maximum number of iterations for *k*-means++, BIRCH, and EM which is the largest iteration value selected for NPIR. A value of 150 for the iterations and the value of 50 for the search agents are considered for

Table 13 Performance comparison of the best values of ARI for different algorithms in the form of best(rank) which indicates the best and ranking of 30 independent runs

Adjusted Rand Index											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.99 (2)	0.77 (8)	0.78 (7)	0.85 (3)	1 (1)	0.73 (10)	0.8 (5)	0.79 (6)	0.83 (4)	0.77 (8)	0.36 (11)
Aniso	1 (1)	0.69 (4)	0.18 (9)	0.3 (8)	0.8 (3)	N/A (11)[a]	0 (10)	0.36 (7)	0.95 (2)	0.66 (5)	0.66 (5)
Appendicitis	0.56 (1)	0.06 (5)	0 (7)	0 (7)	0.06 (5)	N/A (11)[a]	−0.01(9)	−0.09(10)	0.43 (3)	0.47 (2)	0.38 (4)
Blobs	1 (1)	1 (1)	1 (1)	0.12 (11)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
Circles	1 (1)	0 (5)	N/A (11)[a]	0.02 (4)	0 (5)	1 (1)	1 (1)	0 (5)	0 (5)	0 (5)	0 (5)
Diagnosis II	1 (1)	0.44 (6)	0.41 (8)	0.23 (11)	0.44 (6)	1 (1)	1 (1)	0.41 (8)	1 (1)	1 (1)	0.41 (8)
Flame	0.98 (1)	0.54 (3)	0.2 (7)	−0.03(9)	0.45 (6)	N/A (11)[a]	0.01 (8)	−0.04(10)	0.55 (2)	0.46 (4)	0.46 (4)
Glass	0.3 (1)	0.26 (3)	0.25 (4)	0.07 (9)	0.23 (6)	N/A (11)[a]	0.01 (10)	0.23 (6)	0.25 (4)	0.28 (2)	0.12 (8)
Iris	0.89 (1)	0.79 (4)	0.57 (7)	0.31 (11)	0.67 (5)	0.57 (7)	0.56 (9)	0.64 (6)	0.82 (2)	0.8 (3)	0.49 (10)
Iris2D	0.92 (1)	0.85 (6)	0.85 (6)	0.64 (9)	0.67 (8)	0.57 (10)	0.57 (10)	0.89 (3)	0.92 (1)	0.89 (3)	0.89 (3)
Jain	1 (1)	0.35 (3)	0.02 (9)	0.04 (8)	−0.01(10)	N/A (11)[a]	0.26 (5)	0.78 (2)	0.35 (3)	0.26 (5)	0.26 (5)
Moons	1 (1)	0.21 (8)	0.21 (8)	0.14 (10)	0.01 (11)	1 (1)	1 (1)	0.33 (5)	0.34 (4)	0.29 (6)	0.29 (6)
Mouse	0.99 (1)	N/A (8)[a]	N/A (8)[a]	0.24 (5)	N/A (8)[a]	N/A (8)[a]	−0.01(7)	0.24 (5)	0.54 (2)	0.5 (3)	0.41 (4)
Pathbased	0.69 (1)	0.47 (3)	0.33 (9)	0.56 (2)	0.44 (6)	N/A (11)[a]	0 (10)	0.35 (8)	0.46 (4)	0.46 (4)	0.4 (7)
seeds	0.74 (2)	0.75 (1)	0.57 (6)	0.13 (9)	0.54 (8)	N/A (11)[a]	0 (10)	0.55 (7)	0.74 (2)	0.72 (4)	0.68 (5)
Smiley	1 (1)	0.04 (7)	0.04 (7)	1 (1)	0.02 (11)	0.29 (4)	1 (1)	0.03 (9)	0.05 (5)	0.03 (9)	0.05 (5)
Varied	0.96 (1)	0.81 (6)	0.61 (7)	0.2 (9)	0.96 (1)	N/A (11)[a]	0 (10)	0.51 (8)	0.86 (3)	0.85 (4)	0.85 (4)
VaryDensity	1 (1)	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	N/A (7)[a]	0 (6)	0.51 (5)	0.9 (2)	0.87 (3)	0.87 (3)
WDBC	0.79 (1)	0 (6)	0 (6)	−0.01(10)	0 (6)	N/A (11)[a]	0 (6)	0.1 (4)	0.15 (3)	0.06 (5)	0.6 (2)
Wine	0.46 (1)	0.37 (4)	0.37 (4)	0.28 (8)	0.37 (4)	N/A (11)[a]	0.01 (10)	0.37 (4)	0.4 (2)	0.39 (3)	0.17 (9)
Rank	**21**	**94**	**134**	**143**	**114**	**149**	**120**	**115**	**53**	**77**	**100**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

DCMVO as per the original paper presenting the algorithm [34]. Furthermore, the crossover probability of 0.8 and a mutation probability of 0.001 are considered for GA. In addition, 50 generations having 20 chromosomes for each generation are considered for GA and PSO. The values selected for the parameters of GA and PSO can be found extensively in the literature for data sets of similar sizes to get satisfactory results [74–78]. Table 2 summarizes the choice of parameters for these algorithms.

The analysis of the results are studied from two aspects: qualitative and quantitative analysis.

4.4.1 Quantitative analysis

This section discuss the performance of NPIR against other clustering algorithms which are *k*-means++, BIRCH, HDBSCAN, EM, MST, HC-SL, HC-CL, GA, PSO, and DCMVO. Tables 3, 4, 5, 6 and 7 show the performance of the average results of NPIR against these algorithms for each data set in terms of HS, CS, VM, AMI, and ARI, respectively. Grand average results which is calculated according to Eq. (18) are also presented by Table 8. In addition, the performance of the best results of NPIR against the other algorithms are presented by Tables 9, 10, 11, 12 and 13 and the average of best results are presented by Table 14. The average results are shown in the form of avg(rank) which indicates the average and the ranking for each value of the table of 30 independent runs, respectively. In contrast, the best results are shown in the form of best(rank) which indicates the best and the ranking for each value of the table of 30 independent runs, respectively. For each algorithm, a total ranking is calculated as a sum for all the ranks and is observed at the last row of each table for each evaluation measure. The lowest ranking is considered for better algorithms while the higher ranking indicates bad clustering for the data sets selected. The performance evaluation of these tables are discussed in more details as follows:

In general, the average results for each evaluation measure show that NPIR outperforms the other algorithms for the majority of the data sets. In addition, NPIR has the minimum value for the total ranking which indicates that NPIR is better than the other algorithms for the selected data sets and its ranking is satisfying for the majority of the data sets.

Furthermore, NPIR has a recognizable better performance compared to most of the other algorithms for Aniso, Flame, Jain, Mouse, Smiley, and Vary density data sets due to several reasons. Specifically, Vary density data set can be

Table 14 Performance comparison of the best values of average of bests score for different algorithms in the form of best(rank) which indicates the best and ranking of 30 independent runs

AVG of Bests Score											
Data set	NPIR	k-means++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	0.98 (2)	0.852 (7)	0.868 (5)	0.884 (3)	1 (1)	0.8 (10)	0.85 (8)	0.87 (4)	0.85 (8)	0.86 (6)	0.512 (11)
Aniso	1 (1)	0.682 (4)	0.312 (9)	0.506 (7)	0.776 (3)	$*N/A(11)$	0.03 (10)	0.43 (8)	0.926 (2)	0.668 (5)	0.66 (6)
Appendicitis	0.41 (1)	0.09 (7)	0.2 (5)	0.2 (5)	0.09 (7)	$*N/A(11)$	0.01 (10)	0.02 (9)	0.29 (3)	0.32 (2)	0.26 (4)
Blobs	1 (1)	1 (1)	1 (1)	0.416 (11)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
Circles	1 (1)	0.046 (5)	$*N/A(11)$	0.092 (4)	0.046 (5)	1 (1)	1 (1)	0 (7)	0 (7)	0 (7)	0 (7)
Diagnosis II	1 (1)	0.478 (6)	0.426 (10)	0.464 (8)	0.478 (6)	1 (1)	1 (1)	0.43 (9)	1 (1)	1 (1)	0.426 (10)
Flame	0.97 (1)	0.484 (3)	0.204 (7)	0.058 (8)	0.426 (6)	$*N/A(11)$	0.05 (9)	0.05 (9)	0.504 (2)	0.432 (4)	0.432 (4)
Glass	0.41 (1)	0.382 (3)	0.376 (4)	0.21 (8)	0.326 (7)	$*N/A(11)$	0.1 (10)	0.34 (6)	0.36 (5)	0.39 (2)	0.168 (9)
Iris	0.87 (1)	0.786 (4)	0.664 (9)	0.47 (11)	0.738 (5)	0.69 (7)	0.67 (8)	0.7 (6)	0.804 (2)	0.796 (3)	0.534 (10)
Iris2D	0.9 (2)	0.84 (6)	0.84 (6)	0.666 (11)	0.738 (8)	0.69 (9)	0.68 (10)	0.87 (3)	0.904 (1)	0.866 (4)	0.866 (4)
Jain	1 (1)	0.37 (4)	0.178 (9)	0.258 (7)	0.156 (10)	$*N/A(11)$	0.25 (8)	0.71 (2)	0.374 (3)	0.312 (5)	0.312 (5)
Moons	1 (1)	0.23 (8)	0.23 (8)	0.208 (10)	0.072 (11)	1 (1)	1 (1)	0.27 (5)	0.284 (4)	0.234 (6)	0.234 (6)
Mouse	0.98 (1)	$*N/A(8)$	$*N/A(8)$	0.388 (6)	$*N/A(8)$	$*N/A(8)$	0.02 (7)	0.39 (5)	0.632 (2)	0.602 (3)	0.476 (4)
Pathbased	0.74 (1)	0.528 (3)	0.386 (9)	0.572 (2)	0.504 (6)	$*N/A(11)$	0.04 (10)	0.39 (8)	0.524 (4)	0.52 (5)	0.464 (7)
seeds	0.71 (4)	0.734 (1)	0.598 (7)	0.304 (9)	0.556 (8)	$*N/A(11)$	0.07 (10)	0.6 (6)	0.724 (2)	0.718 (3)	0.686 (5)
Smiley	1 (1)	0.206 (9)	0.234 (5)	1 (1)	0.216 (7)	0.45 (4)	1 (1)	0.17 (11)	0.218 (6)	0.21 (8)	0.184 (10)
Varied	0.94 (1)	0.804 (6)	0.646 (7)	0.432 (9)	0.936 (2)	$*N/A(11)$	0.03 (10)	0.55 (8)	0.84 (3)	0.836 (4)	0.834 (5)
VaryDensity	1 (1)	$*N/A(7)$	$*N/A(7)$	$*N/A(7)$	$*N/A(7)$	$*N/A(7)$	0.05 (6)	0.64 (5)	0.886 (2)	0.86 (3)	0.86 (3)
WDBC	0.7 (1)	0.028 (7)	0.028 (7)	0.016 (10)	0.028 (7)	$*N/A(11)$	0.03 (6)	0.14 (4)	0.2 (3)	0.096 (5)	0.498 (2)
Wine	0.45 (1)	0.42 (6)	0.408 (7)	0.356 (8)	0.428 (4)	$*N/A(11)$	0.07 (10)	0.43 (3)	0.432 (2)	0.424 (5)	0.168 (9)
Rank	**24**	**99**	**134**	**137**	**115**	**148**	**127**	**116**	**61**	**77**	**113**

[a]N/A denotes that the value is not reported because it failed to detect more than one cluster

Table 15 Total rank for each algorithm based on the average results

Algorithm	HS	CS	VM	AMI	ARI	Grand AVG
NPIR	**65**	**51**	**49**	**52**	**48**	**51**
k-means++	107	119	107	104	109	113
BIRCH	118	112	120	118	121	122
HDBSCAN	78	146	128	134	137	130
EM	116	116	115	110	118	116
MST	158	122	145	149	146	147
HC-SL	131	98	124	129	117	125
HC-CL	97	107	96	94	100	103
GA	115	115	113	107	103	112
PSO	84	97	80	76	78	80
DCMVO	99	79	92	87	82	95

considered hard for density based clustering algorithms such as HDBSCAN because its clusters have different densities and such algorithms cannot handle such difference. Data sets with circular patterns such as circles and smiley cannot be handled properly using centoid based algorithms because the centroid of a circular shape can be closer to other data points rather than the data points of the circular pattern. In addition, Moons, Jain, Aniso, and Flame data sets have data points which interleave in the X and Y axis and are also hard for centroid based algorithms because two data points in different classes can have the same distance to a specific centroid. In contrast, NPIR performs well for such data sets due to the random behavior of the election operation, the deterministic behavior of the selection operation for calculating the similarity of the nearest neighbors, and the fixed shapes avoidance over the course of iterations for the assignment operation.

Table 16 Total rank for each algorithm based on the best results

Algorithm	HS	CS	VM	AMI	ARI	AVG of bests
NPIR	**23**	**27**	**23**	**23**	**21**	**24**
k-means++	99	104	94	94	94	99
BIRCH	131	122	133	128	134	134
HDBSCAN	96	151	136	144	143	137
EM	113	114	112	108	114	115
MST	158	122	147	149	149	148
HC-SL	131	101	125	130	120	127
HC-CL	113	120	114	109	115	116
GA	61	70	61	59	53	61
PSO	79	87	76	74	77	77
DCMVO	119	102	114	107	100	113

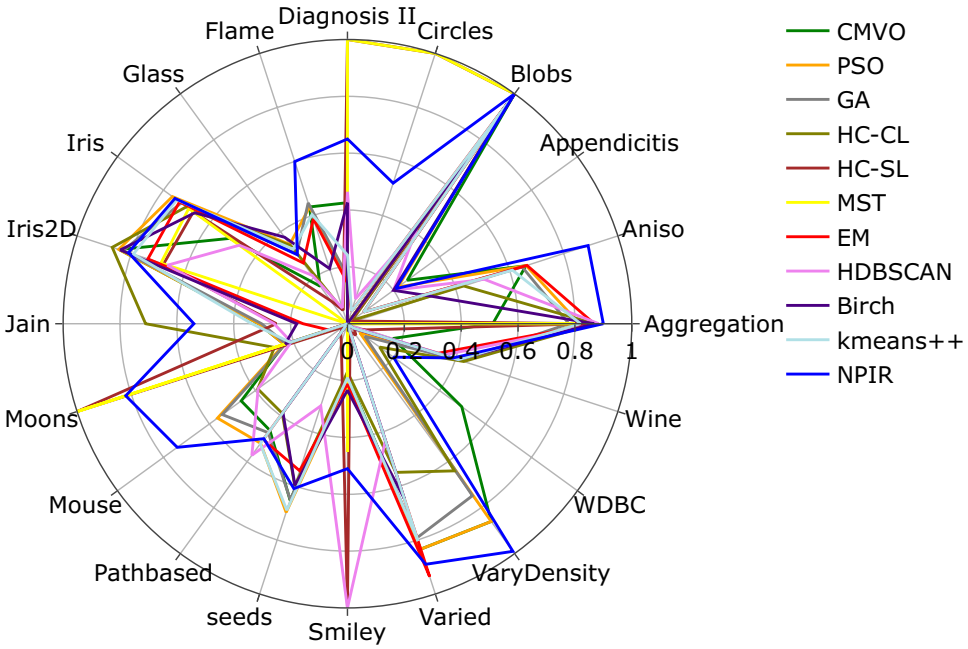


Fig. 10 Performance comparison for the average evaluation of grand average score of all the measures for different algorithms

In addition, the average results for the evaluation measures show that PSO has the closest total ranking to NPIR for VM, AMI, ARI and the grand average measures. In contrast, DCMVO has the closest total ranking to NPIR for CS and HDBSCAN has the closest total ranking to NPIR for HS. Since VM satisfies both homogeneity and completeness of the clusters which reflects both HS and CS, then we cannot conclude that HDBSCAN nor DCMVO are competing for NPIR since they only satisfy either the homogeneity or the completeness of the prediction but not the combination of the two which is reflected by VM. On the other hand, MST and HC-SL perform very well for some data sets such as Blobs, Circles, Diagnosis II, and Moons but fail to detect most of the other data sets and has very low ranking compared to the other algorithms. Thus, we can only conclude that MST and HC-SL are competing to NPIR for some of the data sets and that PSO is competing for NPIR for the average results since it has the closest performance results for most of the measures for the selected data sets.

We can also observe that the performance of all clustering algorithms decreases for data sets with higher dimensions such as Appendicitis, Glass, WDBC, and Wine data sets. In the case of NPIR, this can be explained due to the adoption of euclidean distance for measuring the distances between the points. It is well-known that high dimensions euclidean distance loses pretty much all meaning [79]. For future work, we plan to investigate the efficiency of adoption other distance measures such as Manhattan distance, cosine distance, Minkowski distance, and correlation distance to overcome this limitation.

On the other hand, the best results for each evaluation measure show that NPIR , MST, and HC-SL outperform the other algorithms having the value of 1 for Circles, Diagnosis II and Moons data sets. However, NPIR

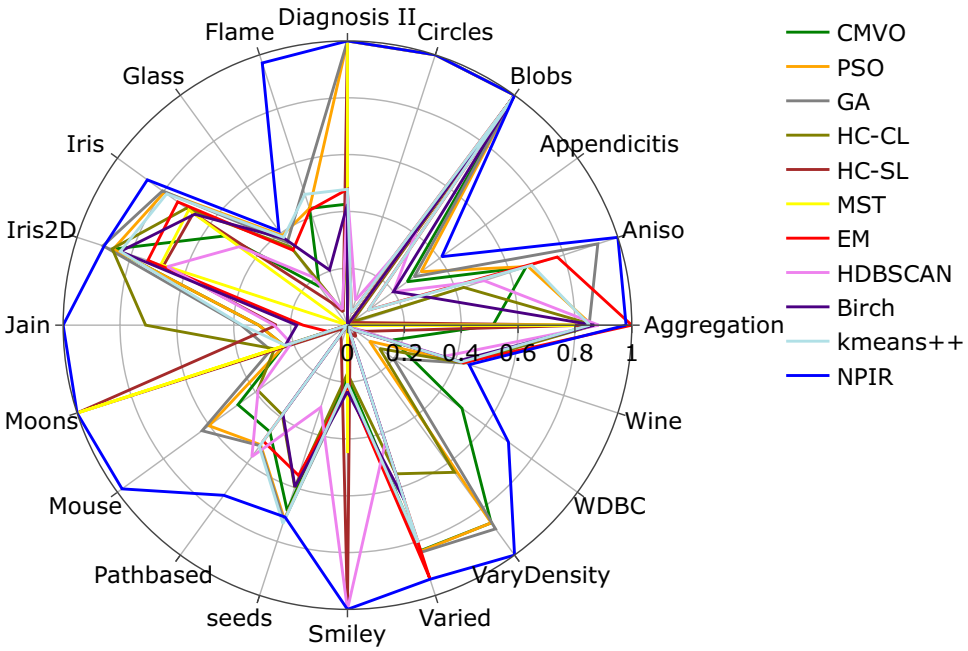


Fig. 11 Performance comparison for the best evaluation of average of bests score of all the measures for different algorithms

Table 17 Average computation time (seconds)

Data set	NPIR	kmeans++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	18.055	0.006	0.098	0.045	0.1	0.017	0.125	0.172	0.767	0.461	299.33
Aniso	388.909	0.003	0.063	0.051	0.024	0.026	0.063	0.109	0.779	0.284	257.97
Appendicitis	1.840	0.002	0.004	0.017	0.02	0.004	0.016	0.031	0.288	0.250	16.87
Blobs	49.973	0.003	0.051	0.04	0.006	0.025	0.031	0.016	0.313	0.341	281.57
Circles	49.255	0.002	0.038	0.057	0.01	0.024	0.016	0.016	1.078	0.278	78.75
Diagnosis II	0.943	0.002	0.008	0.006	0.006	0.010	0.000	0.047	0.412	0.185	16.20
Flame	11.970	0.003	0.008	0.014	0.025	0.014	0.016	0.031	0.225	0.188	24.05
Glass	1.125	0.004	0.011	0.013	0.023	0.015	0.240	0.016	0.400	0.234	127.12
Iris	0.900	0.002	0.005	0.005	0.016	0.013	0.007	0.031	0.328	0.219	37.63
Iris2D	1.410	0.003	0.004	0.008	0.005	0.006	0.004	0.031	0.504	0.172	36.28
Jain	15.433	0.002	0.024	0.012	0.015	0.009	0.016	0.031	0.289	0.194	28.14
Moons	534.936	0.002	0.042	0.043	0.008	0.021	1.085	0.031	0.247	0.274	78.77
Mouse	20.372	0.008	0.015	0.01	0.085	0.010	0.000	0.063	0.222	0.219	69.59
Pathbased	15.358	0.004	0.02	0.008	0.01	0.009	0.000	0.047	0.188	0.203	52.87
Seeds	3.517	0.003	0.009	0.009	0.008	0.007	0.016	0.016	0.203	0.188	66.78
Smiley	22.837	0.005	0.056	0.014	0.017	0.010	0.016	0.031	0.250	0.222	191.83
Varied	277.342	0.004	0.073	0.038	0.014	0.024	0.016	0.016	0.296	0.264	168.21
VaryDensity	1.879	0.013	0.005	0.004	0.072	0.009	0.016	0.062	0.203	0.210	34.16
WDBC	23.000	0.006	0.015	0.022	0.011	0.050	0.047	0.094	0.266	0.210	37.97
Wine	2.755	0.004	0.02	0.006	0.022	0.016	0.016	0.078	0.266	0.333	36.45

outperforms all other algorithms having the value of 1 for Aniso, Jain, and Vary density. This indicates correct clustering for all the data points for these data sets. In addition, NPIR has a recognizable minimum value for the total ranking of the best results for all the evaluation measures which indicates that NPIR is better than the other algorithms for the selected data sets and its ranking is very promising for other data sets. Furthermore, GA has the closest total ranking of the best results for all the measures due to its exploratory behavior. In contrast, HDBSCAN, BIRCH, and MST have the worst total ranking of the best results for these data sets.

A summary of the ranking is displayed by Tables 15 and 16 which show the total rank of the average and best

Table 18 Best computation time (seconds)

Data set	NPIR	kmeans++	BIRCH	HDBSCAN	EM	MST	HC-SL	HC-CL	GA	PSO	DCMVO
Aggregation	15.000	0.004	0.038	0.022	0.000	0.011	0.125	0.172	0.767	0.461	287.79
Aniso	299.287	0.002	0.044	0.031	0.014	0.020	0.063	0.109	0.779	0.284	168.37
Appendicitis	1.522	0.001	0.002	0.002	0.002	0.004	0.016	0.031	0.288	0.250	13.95
Blobs	47.270	0.002	0.031	0.032	0.000	0.020	0.031	0.016	0.313	0.341	170.48
Circles	44.487	0.001	0.033	0.040	0.000	0.020	0.016	0.016	1.078	0.278	76.49
Diagnosis II	0.750	0.000	0.000	0.000	0.000	0.004	0.000	0.047	0.412	0.185	14.06
Flame	9.248	0.000	0.000	0.000	0.004	0.007	0.016	0.031	0.225	0.188	20.43
Glass	0.954	0.000	0.000	0.000	0.009	0.006	0.240	0.016	0.400	0.234	108.13
Iris	0.728	0.001	0.003	0.003	0.006	0.007	0.007	0.031	0.328	0.219	33.40
Iris2D	1.088	0.000	0.003	0.003	0.00	0.004	0.004	0.031	0.504	0.172	33.31
Jain	12.614	0.001	0.017	00.00	0.011	0.006	0.016	0.031	0.289	0.194	25.67
Moons	342.906	0.001	0.028	0.031	0.005	0.020	1.085	0.031	0.247	0.274	76.96
Mouse	14.852	0.000	0.011	0.008	0.066	0.008	0.000	0.063	0.222	0.219	64.28
Pathbased	10.864	0.002	0.014	0.007	0.006	0.006	0.000	0.047	0.188	0.203	49.08
Seeds	2.876	0.001	00.00	0.006	0.006	0.006	0.016	0.016	0.203	0.188	40.14
Smiley	19.202	0.002	0.036	0.01	0.000	0.008	0.016	0.031	0.250	0.222	111.41
Varied	188.273	0.002	0.055	0.033	0.000	0.021	0.016	0.016	0.296	0.264	157.21
VaryDensity	1.578	0.01	0.003	0.003	0.058	0.004	0.016	0.062	0.203	0.210	31.71
WDBC	17.047	0.000	0.012	0.018	0.005	0.037	0.047	0.094	0.266	0.210	35.17
Wine	2.236	0.000	0.011	0.004	0.012	0.006	0.016	0.078	0.266	0.333	33.57

results for each algorithm, respectively. As shown from the tables, NPIR has the minimum total rank for all the evaluation measures compared to the other algorithms for both average and best results.

In addition, a radar chart for the average and best results for the grand average and the average of bests scores are displayed by Figs. 10 and 11, respectively. The two figures summarize the clustering performance of all the algorithms for each data set. Looking into the grand average results in Fig. 10, we can observe that NPIR has a better performance for the majority of the data sets having the radar lines of the other algorithms surrounded by the NPIR radar line for most parts of the radar. On the other hand, NPIR has a better performance for almost all the data sets for the average of bests results in Fig. 11 having the radar line of all the algorithms surrounded by the NPIR radar. In case of MST and HC-SL, only some data sets as discussed earlier have the same value as NPIR and thus have identical radar lines for these data sets. It is also observed that the radar line for NPIR is almost hitting the edges for most parts of the chart which indicates that the algorithm is generating the maximum value of 1 in the best results for many data sets resulting in correct clustering for all the data points.

Tables 17 and 18 show the average and best CPU time spent by NPIR compared to other algorithms. As per results, NPIR computation time is acceptable and fast enough to handle similar data sets. Compared to the other algorithms, NPIR is faster than DCMVO for most of the data sets but tend to be slower than the other algorithms. The main reason is due to the time taken for constructing the distance vectors at the preparatory stage and the frequent elections of points. However, NPIR is able to handle similar data sets in reasonable time having high quality results for the majority of the data sets as discussed earlier. It is also possible to use Graphics Processing Unit (GPU) to decrease the processing time.

4.4.2 Qualitative analysis

In this section, a two-dimensional plotting for the best results of selected data sets from the aforementioned ones is presented. Each predicted cluster is colored differently than the other predicted clusters. The selected data sets are challenging for researchers as they are of non-spherical shapes, contain clusters of different densities and sizes, and the data points of different clusters interleave in the X or Y axis. Figures 12, 13, 14, 15, 16, 17 and 18 show the visualization of clustering quality results for the best run for each of the selected data sets.

Aggregation data set has 7 clusters of different size having some connection between them. The challenges of this data set include the varying sizes of clusters and the connection between clusters. Figure 12 shows that some algorithms predict some of the clusters but fail to predict all the clusters correctly. However, NPIR almost assigns the data instances to the correct clusters. In addition, EM

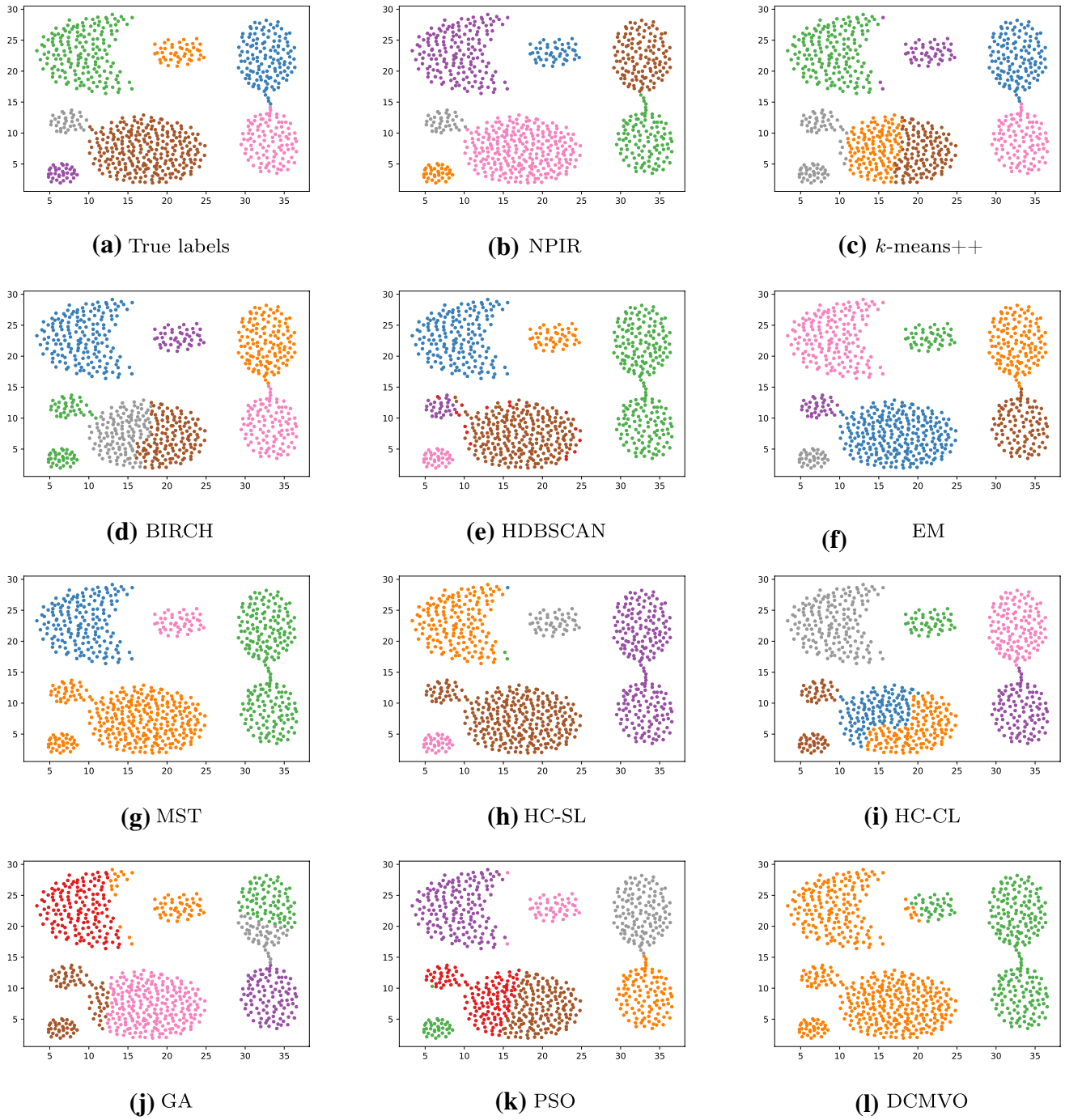


Fig. 12 Aggregation data set displayed in two-dimensional space for **a** True labels; **b** NPIR; **c** *k*-means++; **d** BIRCH; **e** HDBSCAN; **f** EM; **g** MST; **h** HC-SL; **i** HC-CL; **j** GA; **k** PSO; **l** DCMVO with a different color for each cluster (color figure online)

has similar performance to NPIR while DCMVO has the worst visual performance.

Circles data set has 2 clusters of different sizes forming 2 circles that are not connected to each other. This data set is challenging having circular clusters. In addition, one circle is contained within the other one which is considered very hard to predict for the clustering algorithms specially the centriod-based algorithms because both circles have the same centriod but are different in size. Figure 13 shows that NPIR, MST, and HC-SL assign the data instances to the correct clusters while all the other algorithms fail to recognize the target clusters. This data set shows how other algorithms have very poor performance and that NPIR has a recognizable performance over most of the other algorithms.

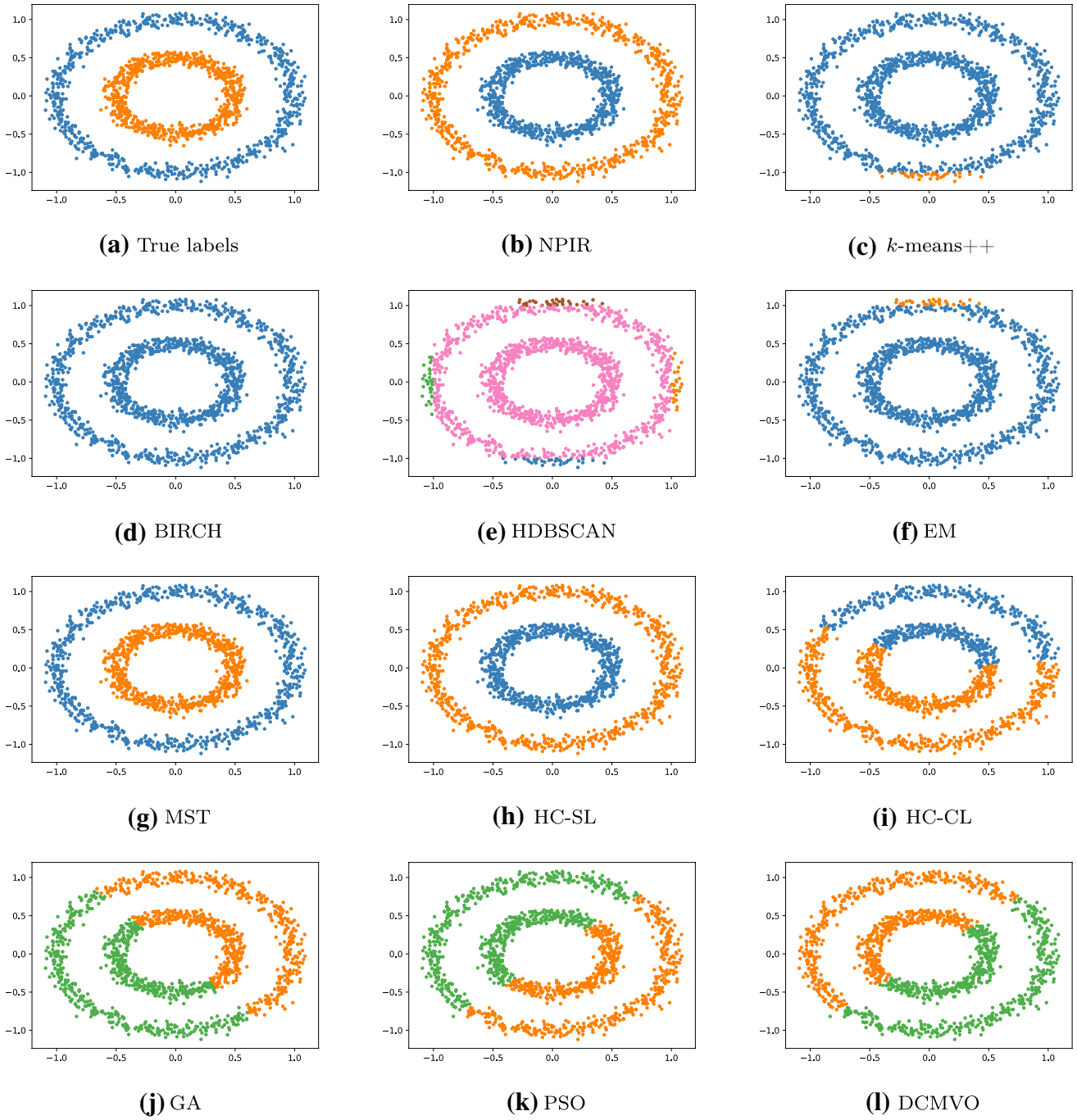


Fig. 13 Circles data set displayed in two-dimensional space for **a** True labels; **b** NPIR; **c** *k*-means++; **d** BIRCH; **e** HDBSCAN; **f** EM; **g** MST; **h** HC-SL; **i** HC-CL; **j** GA; **k** PSO; **l** DCMVO with a different color for each cluster (color figure online)

Jain data set has 2 clusters of different densities that are not connected with each other but interleave in the X and Y axis for some points which makes it challenging for the clustering algorithm. Figure 14 shows that NPIR assigns the data instances to the correct clusters while all the others fail to recognize the target clusters for the points that interleave in the X and Y axis. This data set also shows how other algorithms have very poor performance and that NPIR has a recognizable performance over all the other algorithms.

Moons data set has 2 clusters of similar densities that are not connected with each other but interleave in the X and Y axis for some points which makes it challenging for the clustering algorithm. Figure 15 shows that NPIR, MST, and HC-SL assign the data instances to the correct clusters

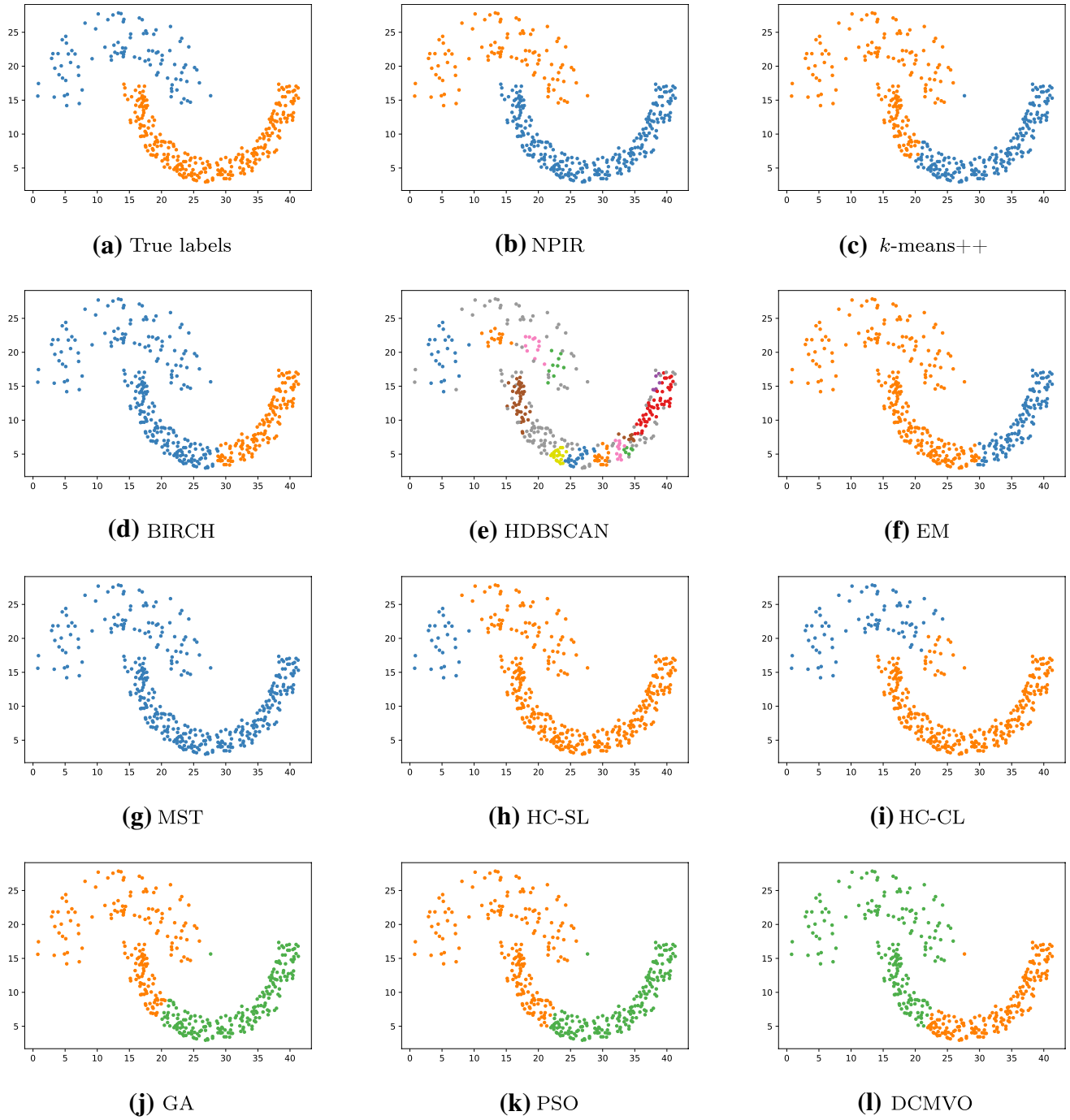


Fig. 14 Jain data set displayed in two-dimensional space for **a** True labels; **b** NPIR; **c** k-means++; **d** BIRCH; **e** HDBSCAN; **f** EM; **g** MST; **h** HC-SL; **i** HC-CL; **j** GA; **k** PSO; **l** DCMVO with a different color for each cluster (color figure online)

while all the others fail to recognize the target clusters for the points that interleave in the X and Y axis. This data set also shows how other algorithms have very poor performance and that NPIR has a recognizable performance over most of the other algorithms.

Smiley data set has 4 clusters that are not connected forming a shape of a face having clusters for the face boundary, the eyes, and the mouth. This data set is considered challenging as the face boundary interleaves in the X and Y axis with the eyes and mouth. In addition, the centroid of the circular face boundary is closer to the data points of the other clusters which forms the eyes and mouth rather than the data points of the circular face pattern. Figure 16 shows that all the algorithms except NPIR, HDBSCAN, and HC-SL fail to

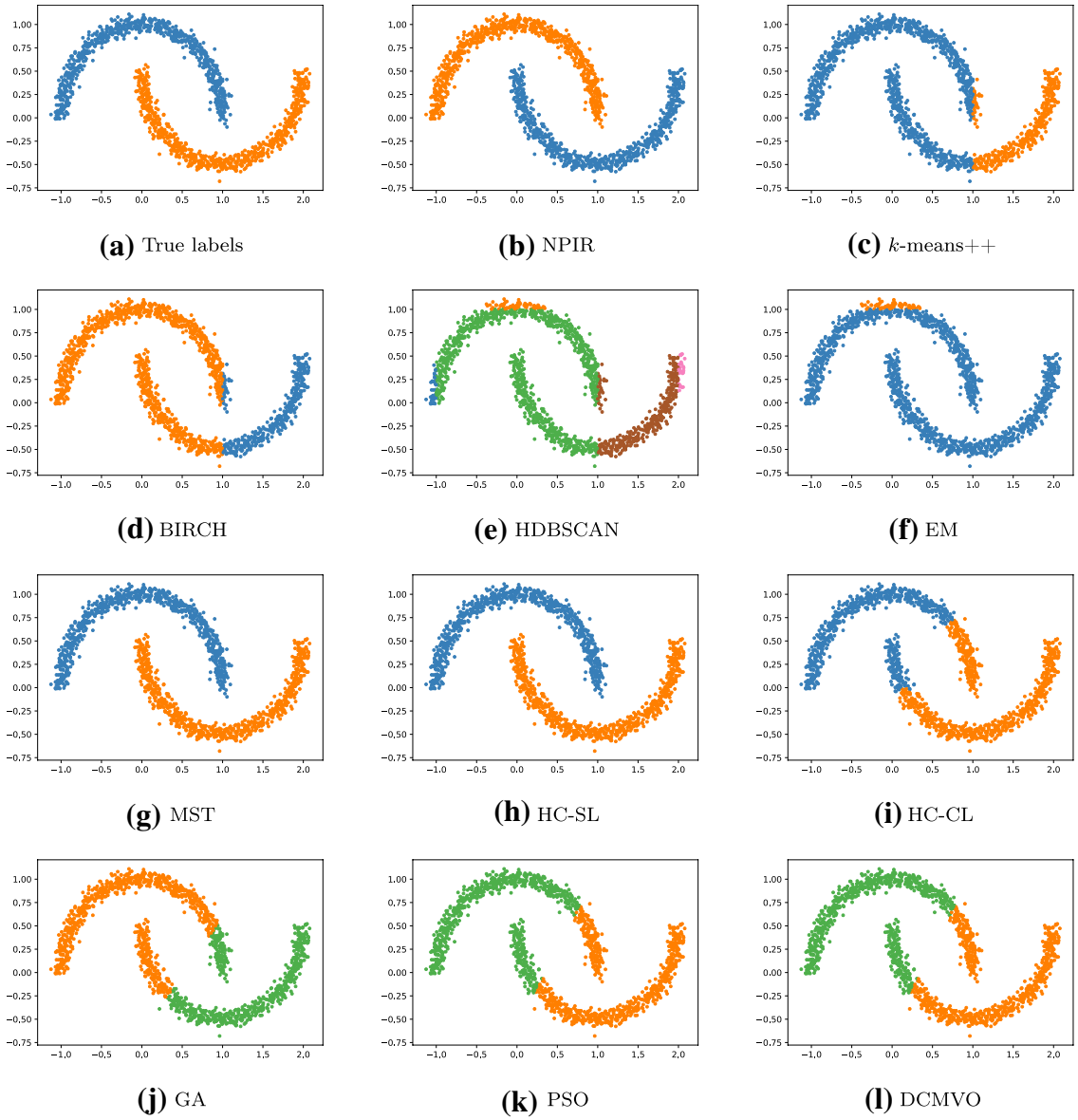


Fig. 15 Moons data set displayed in two-dimensional space for **a** True labels; **b** NPIR; **c** *k*-means++; **d** BIRCH; **e** HDBSCAN; **f** EM; **g** MST; **h** HC-SL; **i** HC-CL; **j** GA; **k** PSO; **l** DCMVO with a different color for each cluster (color figure online)

predict the face boundary. BIRCH, EM, and PSO have predicted the correct clusters for the eyes and mouth but not the face boundary. However, NPIR and HDBSCAN have predicted the face boundary, the eyes, and the mouth correctly.

Varied data set has 3 clusters with different densities that are connected with each other. The challenges of this data set include having different clusters densities and having connected clusters. This data set is also considered hard for density-based clustering algorithms such as HDBSCAN because the clusters have different densities and such algorithm cannot handle such differences having very bad results as observed. Figure 17 shows that most algorithms fail to predict the correct clusters. However, NPIR almost predicts the correct clusters for the data points. In addition, EM has the closest performance to NPIR for this data set compared to the other algorithms. The other algorithms fail to distinguish data points at the connections between clusters resulting in bad clustering.

Vary Density data set has 3 clusters with different densities and sizes that are connected with each other. The challenges of this data set include having different clusters densities and sizes and having connected clusters. This data set is also considered hard for density-based clustering algorithms such as HDBSCAN because the clusters have different densities and such algorithm cannot handle such differences having very bad results as observed.

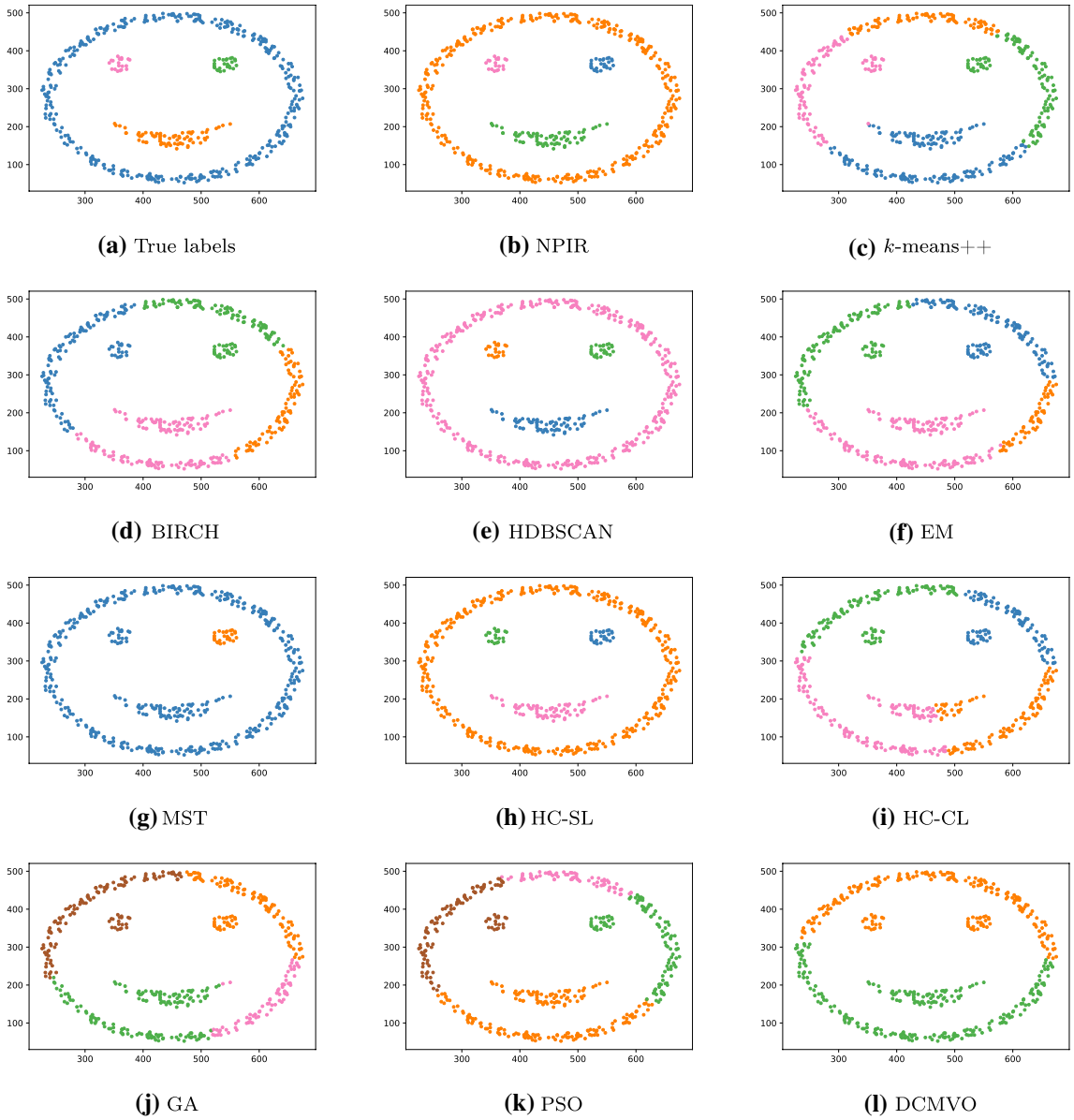


Fig. 16 Smiley data set displayed in two-dimensional space for **a** True labels; **b** NPIR; **c** *k*-means++; **d** BIRCH; **e** HDBSCAN; **f** EM; **g** MST; **h** HC-SL; **i** HC-CL; **j** GA; **k** PSO; **l** DCMVO with a different color for each cluster (color figure online)

Figure 18 shows that all algorithms fail to predict the correct clusters. However, NPIR predicts the correct clusters for the data points. In addition, PSO, GA, and DCMVO have the closest performance to NPIR for this data set compared to the other algorithms but still they do not predict data points at the connections between clusters. The other algorithms clustered all points at one cluster resulting in bad clustering.

As a conclusion to the qualitative analysis of the best results, NPIR has successfully predicted the clusters for almost all the data sets. It outperforms the other algorithms in the selected data sets and is promising for any other data sets. This can be used as an indication of the behavior of the algorithm as it predicts non-spherical shapes, shapes with clusters of different densities and sizes, and shapes where data points of different clusters interleave in the X or Y axis.

4.5 Mall customer segmentation case study

Customer behavior and purchasing data for large stores like malls can be used for gaining future insights about the behavior of these customers. Customer data such as age, gender, annual income, and spending score can be obtained using membership cards.

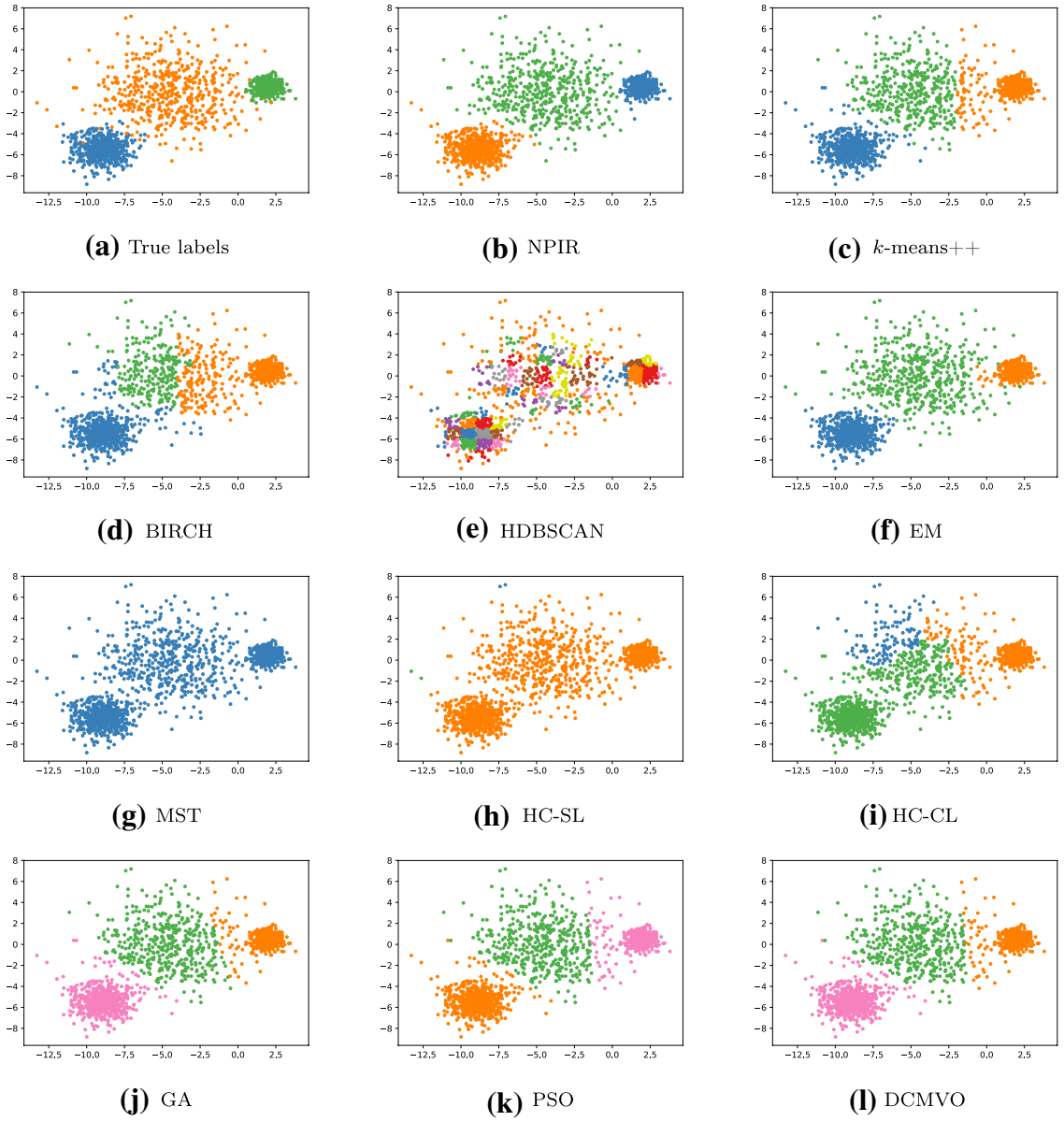


Fig. 17 Varied data set displayed in two-dimensional space for **a** True labels; **b** NPIR; **c** *k*-means++; **d** BIRCH; **e** HDBSCAN; **f** EM; **g** MST; **h** HC-SL; **i** HC-CL; **j** GA; **k** PSO; **l** DCMVO with a different color for each cluster (color figure online)

The goal of this case study is to segment customers according to their annual income and purchasing behavior. Strategic plans can be built according to those customers who can be easily converge to spending more on purchased items.

We use mall customer segmentation data set from kaggle[11] repository which has 100 instances and 5 features. We segment customers according to two features which are the annual income and the spending score to get insights about different type of customers. The distribution of the annual income and the spending score can be found in Figs. 19 and 20. We can observe from the two figures that normal distribution is present for these two features. This means that the mall customers have different values for their annual income and that most of them have an income that falls under the range of $50 and $80. Moreover, the spending behavior of the customers is almost normally distributed and most of the customers fall under the range of 40–60 score.

We apply NPIR on the mall customer data set with the values of 5, 0.1, and 100 for k, IR, and i. k is determined using the silhouette coefficient measure [80] which is a

[11] https://www.kaggle.com/.

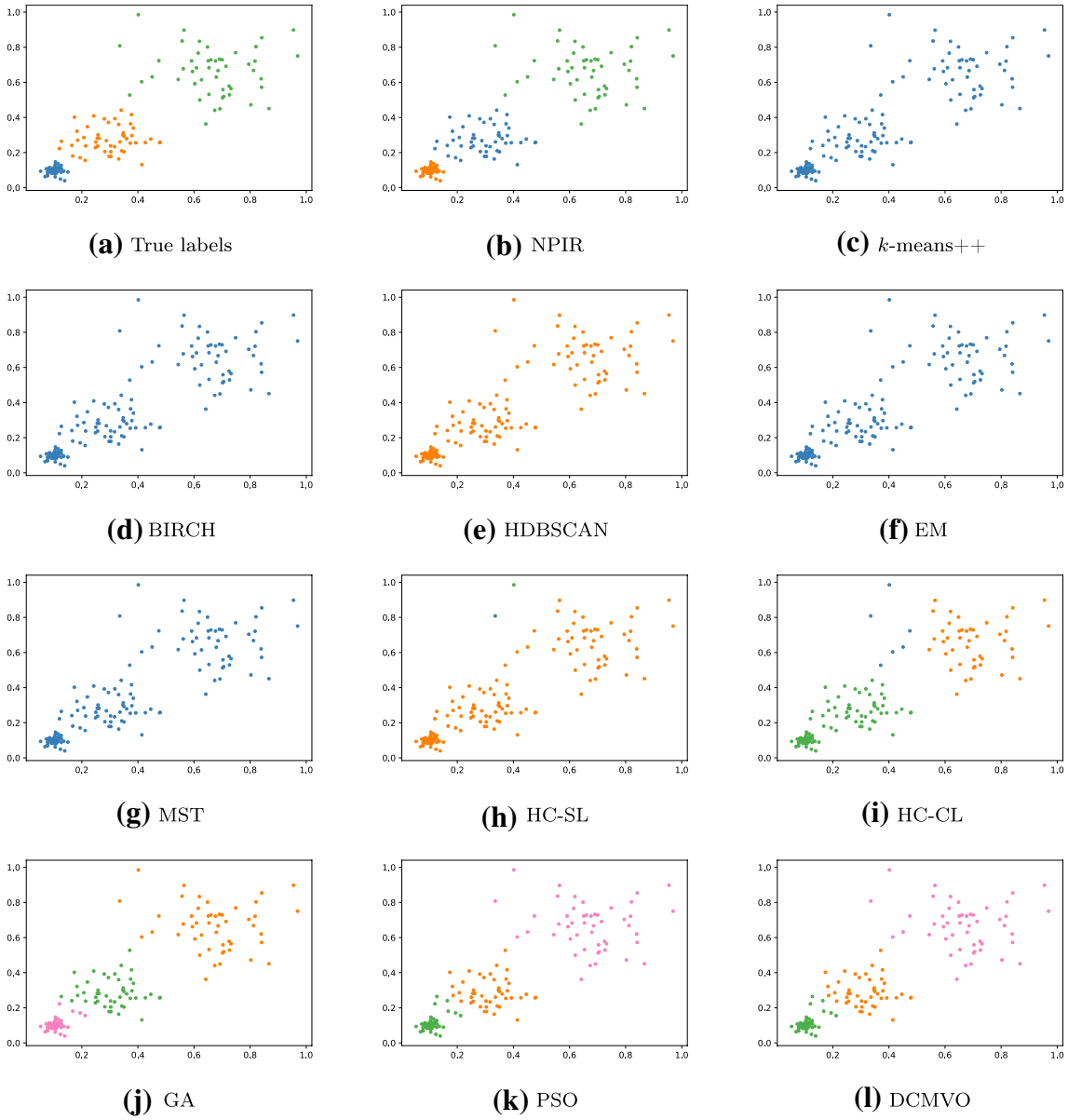


Fig. 18 Vary Density data set displayed in two-dimensional space for **a** True labels; **b** NPIR; **c** *k*-means++; **d** BIRCH; **e** HDBSCAN; **f** EM; **g** MST; **h** HC-SL; **i** HC-CL; **j** GA; **k** PSO; **l** DCMVO with a different color for each cluster (color figure online)

common approach to select the right number of clusters before performing the clustering task [81]. The *IR* and *i* values are selected as they are the dominant parameters selected in the experiments discussed previously.

The results of the segmentation for the mall customers using NPIR is shown in Table 19 and Fig. 21. It is observed from the figure and the table that most of the customers are concentrated in the middle of the figure which means that those customers have intermediate annual income and they normally spend their money purchasing items in the mall. We refer to these customers as standard customers. Customers with low income are referred to as sensible or careless customers according to their spending behavior as those who have low spending are sensible, while those who have high rate of spending are considered careless as they have low income. In contrast, customers with high income are referred to as careful or target customers according to their spending behavior as those who have low spending are considered careful customers because they are thrifty customers in spending their high income in the mall while those who have high rate of spending are considered the target customers that is the main goal of the study. Those target customers can be

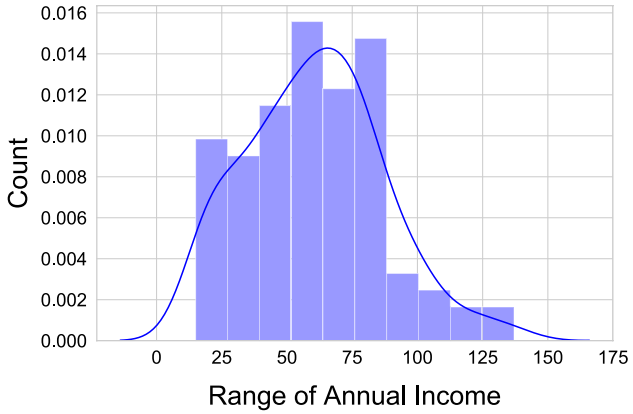


Fig. 19 Distribution of annual income

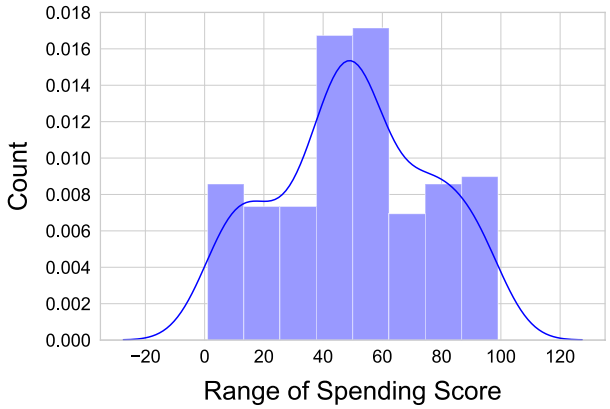


Fig. 20 Distribution of spending score

Table 19 Segmentation of customers

Cluster No.	#Instances	Label	Color
0	23	Sensible	Blue
1	79	Standard	Magenta
2	39	Target	Cyan
3	38	Careful	Red
4	21	Careless	Green

driven to higher spending rate through strategic planning of the mall owners.

We further analyzed the distribution of each segment of customers in Figs. 22 and 23. Figure 22 shows the box plot for the annual income of customers while Fig. 23 shows the box plot for the spending score of customers. Interquartile range, high values, low values are presented as the box, the upper whiskers. and the lower whiskers, respectively [82, 83]. It is observed from the figures that target customers have the highest income and spending while the sensible

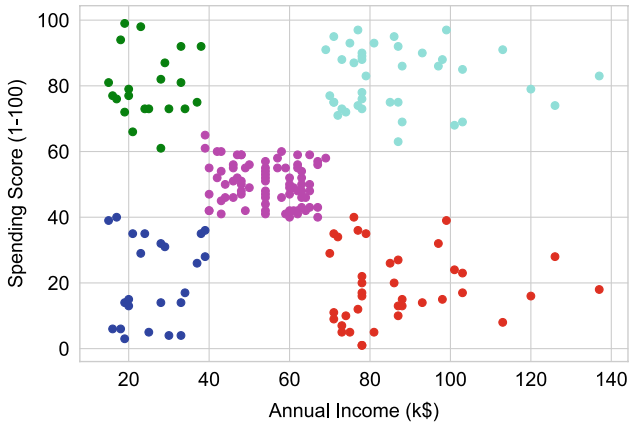


Fig. 21 Visualization of clusters

customers have the lowest income and spending. Standard customers have intermediate income and spending. In addition, careless customers have low income but high spending while careful customers have high income with low spending. It is also observed that target customers and careful customers with high income have some outliers with a very high income which is larger than $120,000. In addition, the values of spending and income for standard customers are close to the mean values with compact box which indicates stability in this cluster of customers.

5 Conclusion and future work

This paper proposes a new clustering algorithm named Nearest Point with Indexing Ratio (NPIR). NPIR tries to overcome the limitations of some clustering algorithms by avoiding the fixed shaped clustering and identifying arbitrary shapes and non-spherical distribution of points. The main idea of the proposed algorithm is to find the nearest point for an already assigned point using the distance between these points and then cluster the nearest point to the same cluster of the assigned point. It is based on the nearest neighbor search technique and a random and iterative behavior of three operations which are election, selection, and assignment in the aim of generating quality clustering results.

Based on the conducted experiments, the following concluding remarks can be noted:

– NPIR outperforms most of the algorithm which are *k*-means++, BIRCH, HDBSCAN, EM, GA, MST, HC-SL, HC-CL, PSO, and DCMVO in terms of Homogeneity Score, Completeness Score, V-measure, Adjusted

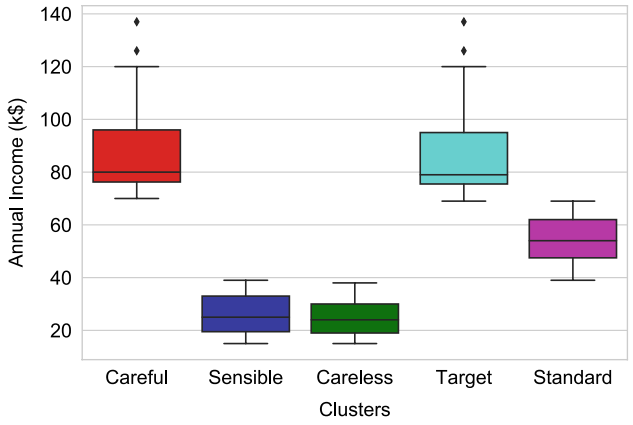


Fig. 22 Box plot of annual income per cluster

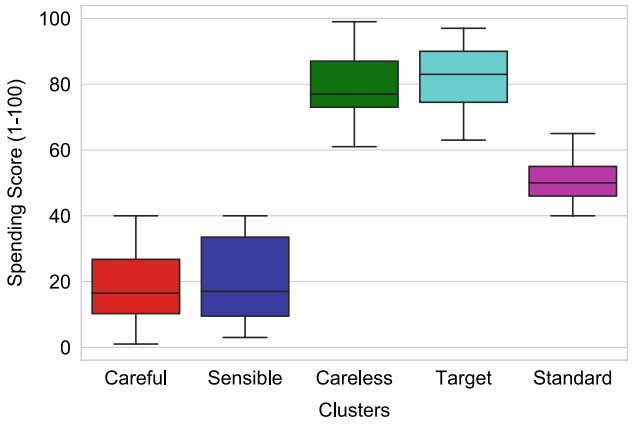


Fig. 23 Box plot of spending score per cluster

Rand Index, and Adjusted Mutual Information for 20 selected data sets.
- NPIR tries to solve the limitations of other algorithms for data sets of arbitrary shapes, non-spherical distribution of points, and different clusters sizes or densities.
- NPIR tries to avoid the termination of cluster shaping process at an early stage by integrating a flexible mechanism of moving any point away from its cluster and assigning it to another cluster which can correct wrong assignments of points and wrong selections of the initial points.
- PSO, MST, and HC-SL clustering algorithms are competing algorithms for NPIR.
- NPIR can be applied on real-life applications such as customer segmentation.

Much work can be done in the future to optimize the current implementation of the algorithm and to validate its efficiency for different applications which can be summarizes as follows:

- The algorithm can be evaluated on other real-life applications including bioinformatics, medical images, face recognition, image segmentations, geographic information systems, and much more.
- The algorithm can be evaluated on large scale data sets using the K-dimensional tree data structure.
- The implementation of the algorithm can be enhanced to dynamically detect the correct number of clusters (k).
- Optimization algorithms can be applied on NPIR to predict the best values of IR, i, and initial points.
- Fuzziness can be added to the algorithm to support overlapping clustering.
- The application of hierarchy clustering in NPIR can be investigated for a possibility to obtain better clustering quality.
- Investigations on the efficiency of adopting other distance measures other than euclidean distance can be performed to overcome the limitation of clustering low-dimensional data sets.

Compliance with ethical standards

Conflict of interest This is hereby certify that the paper is original, neither the paper nor a part of it is under consideration for publication anywhere else and has not been previously published anywhere. We have also no conflicts of interest to disclose.

References

1. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Amsterdam
2. Frank E, Hall M, Trigg L, Holmes G, Witten IH (2004) Data mining in bioinformatics using weka. Bioinformatics 20(15):2479
3. Kumar S, Pant M, Kumar M, Dutt A (2018) Colour image segmentation with histogram and homogeneity histogram difference using evolutionary algorithms. Int J Mach Learn Cybern 9(1):163
4. Santos BO, Valença J, Júlio E (2017) Detection of cracks on concrete surfaces by hyperspectral image processing. In: Automated visual inspection and machine vision ii, international society for optics and photonics, vol 10334
5. Khan Z, Ni J, Fan X, Shi P (2017) An improved k-means clustering algorithm based on an adaptive initial parameter estimation procedure for image segmentation. Int J Innovat Comput Inf Control 13(5):1509
6. Reddy S, Parker A, Hyman J, Burke J, Estrin D, Hansen M (2007) Image browsing, processing, and clustering for participatory sensing: lessons from a DietSense prototype. In: Proceedings of the 4th workshop on embedded networked sensors (ACM), pp 13–17
7. Zhang C, Wang P (2000) A new method of color image segmentation based on intensity and hue clustering. In: Proceedings

15th international conference on pattern recognition (IEEE), vol 3, pp 613–616
8. Liu A, Su Y, Nie W, Kankanhalli MS (2017) Hierarchical clustering multi-task learning for joint human action grouping and recognition. IEEE Trans Pattern Anal Mach Intell 39(1):102
9. Silva S, Suresh R, Tao F, Votion J, Cao Y (2017) A multi-layer k-means approach for multi-sensor data pattern recognition in multi-target localization. arXiv preprint arXiv:1705.10757
10. Nasrabadi NM (2007) Pattern recognition and machine learning. J Electron Imaging 16(4):049901
11. Mei JP, Wang Y, Chen L, Miao C (2017) Large scale document categorization with fuzzy clustering. IEEE Trans Fuzzy Syst 25(5):1239
12. Brodić D, Amelio A, Milivojević ZN (2017) Clustering documents in evolving languages by image texture analysis. Appl Intell 46(4):916
13. Kou G, Peng Y, Wang G (2014) Evaluation of clustering algorithms for financial risk analysis using MCDM methods. Inf Sci 275:1
14. Wang X, Garibaldi JM (2005) A comparison of fuzzy and non-fuzzy clustering techniques in cancer diagnosis. In: Proceedings of the 2nd international conference in computational intelligence in medicine and healthcare, BIOPATTERN conference, vol 28. Costa da Caparica, Lisbon, Portugal
15. Jang H, Hur Y, Lee H (2016) Identification of cancer-driver genes in focal genomic alterations from whole genome sequencing data. Sci Rep 6
16. Liu T, Rosenberg C, Rowley HA (2007) Clustering billions of images with large scale nearest neighbor search. In: 2007 IEEE workshop on applications of computer vision (WACV'07), pp 28–28
17. Oyelade O, Oladipupo O, Obagbuwa I (2010) Application of k means clustering algorithm for prediction of students academic performance. arXiv preprint arXiv:1002.2425
18. Alhalaweh A, Alzghoul A, Kaialy W (2014) Data mining of solubility parameters for computational prediction of drug-excipient miscibility. Drug Dev Ind Pharm 40(7):904
19. Estivill-Castro V (2002) Why so many clustering algorithms: a position paper. ACM SIGKDD Explor Newsl 4(1):65
20. Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recognit Lett 31(8):651
21. Tan PN et al (2006) Introduction to data mining. Pearson Education India, New Delhi
22. Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms (Society for Industrial and Applied Mathematics), pp 1027–1035
23. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: ACM sigmod record, vol 25 (ACM), pp 103–114
24. Campello RJ, Moulavi D, Zimek A, Sander J (2015) Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Trans Knowl Discov Data (TKDD) 10(1):5
25. Gower JC, Ross GJ (1969) Minimum spanning trees and single linkage cluster analysis. J R Stat Soc Ser C (Appl Stat) 18(1):54
26. Asano T, Bhattacharya B, Keil M, Yao F (1988) Clustering algorithms based on minimum and maximum spanning trees. In: Proceedings of the 4th annual symposium on computational geometry (ACM), pp 252–257
27. Murtagh F, Contreras P (2012) Algorithms for hierarchical clustering: an overview. Wiley interdisciplinary reviews. Data Min Knowl Discov 2(1):86
28. Legendre P, Legendre LF (2012) Numerical ecology, vol 24. Elsevier, Amsterdam
29. Everitt B, Landau S, Leese M (2001) Cluster analysis arnold. A member of the Hodder Headline Group, London, pp 429–438
30. Sheikh RH, Raghuwanshi MM, Jaiswal AN (2008) Genetic algorithm based clustering: a survey. In: 1st international conference on emerging trends in engineering and technology (IEEE), pp 314–319
31. Kennedy R, Eberhart J (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4, pp 1942–1948
32. Rana S, Jasola S, Kumar R (2011) A review on particle swarm optimization algorithms and their applications to data clustering. Artif Intell Rev 35(3):211
33. Alam S, Dobbie G, Koh YS, Riddle P, Rehman SU (2014) Research on particle swarm optimization based clustering: a systematic review of literature and techniques. Swarm Evol Comput 17:1
34. Shukri S, Faris H, Aljarah I, Mirjalili S, Abraham A (2018) Evolutionary static and dynamic clustering algorithms based on multi-verse optimizer. Eng Appl Artif Intell 72:54
35. Chen M, Li L, Wang B, Cheng J, Pan L, Chen X (2016) Effectively clustering by finding density backbone based-on knn. Pattern Recognit 60:486
36. Lu J, Zhu Q, Wu Q (2018) A novel data clustering algorithm using heuristic rules based on k-nearest neighbors chain. Eng Appl Artif Intell 72:213
37. Ester M, Kriegel HP, Sander J, Xu X et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol 96, pp 226–231
38. Ankerst M, Breunig MM, Kriegel HP, Sander J (1999) OPTICS: ordering points to identify the clustering structure. In: ACM sigmod record, vol 28, pp 49–60
39. Tzortzis G, Likas A (2014) The minmax k-means clustering algorithm. Pattern Recognit 47(7):2505
40. Frandsen PB, Calcott B, Mayer C, Lanfear R (2015) Automatic selection of partitioning schemes for phylogenetic analyses using iterative k-means clustering of site rates. BMC Evol Biol 15(1):13
41. Trivedi N, Kanungo S (2017) Performance enhancement of K-means clustering algorithm for gene expression data using entropy-based centroid selection. In: International conference on Computing, communication and automation (ICCCA), 2017 international conference on IEEE, pp 143–148
42. Kadir SN, Goodman DF, Harris KD (2014) High-dimensional cluster analysis with the masked EM algorithm. Neural Comput 26:2379
43. Al-Madi N, Aljarah I, Ludwig SA (2014) Parallel glowworm swarm optimization clustering algorithm based on MapReduce. In: 2014 IEEE symposium on swarm intelligence, IEEE, pp 1–8
44. Aljarah I, Ludwig SA (2012) Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In: 2012 4th world congress on nature and biologically inspired computing (NaBIC), IEEE, pp 104–111
45. Aljarah I, Ludwig SA (2013) Mapreduce intrusion detection system based on a particle swarm optimization clustering algorithm. In: 2013 IEEE congress on evolutionary computation, IEEE, pp 955–962
46. Cui X, Zhu P, Yang X, Li K, Ji C (2014) Optimized big data k-means clustering using MapReduce. J Supercomput 70(3):1249
47. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495
48. Aljarah I, Mafarja M, Heidari AA, Faris H, Mirjalili S (2020) Multi-verse optimizer: theory, literature review, and application in data clustering. In: Nature-inspired optimizers, Springer, pp 123–141
49. Aljarah I, Mafarja M, Heidari AA, Faris H, Mirjalili S (2019) Clustering analysis using a novel locality-informed grey

wolf-inspired clustering approach. In: Knowledge and information systems, Springer, pp 1–33
50. Faris H, Aljarah I, Al-Betar MA, Mirjalili S (2018) Grey wolf optimizer: a review of recent variants and applications. Neural Comput Appl 30(2):413–435
51. Martins JA, Mazayev A, Correia N, Schütz G, Barradas A (2017) Gacn: Self-clustering genetic algorithm for constrained networks. IEEE Commun Lett 21(3):628
52. Rahman MA, Islam MZ (2014) A hybrid clustering technique combining a novel genetic algorithm with k-means. Knowl Based Syst 71:345
53. Deng Y, Liu Y, Zhou D (2015) An improved genetic algorithm with initial population strategy for symmetric TSP. Math Problems Eng
54. Liang X, Li W, Zhang Y, Zhou M (2015) An adaptive particle swarm optimization method based on clustering. Soft Comput 19(2):431
55. Ni Q, Pan Q, Du H, Cao C, Zhai Y (2017) A novel cluster head selection algorithm based on fuzzy clustering and particle swarm optimization. IEEE/ACM Trans Comput Biol Bioinform (TCBB) 14(1):76
56. Daoud AS, Sallam A, Wheed ME (2017) Improving Arabic document clustering using K-means algorithm and Particle Swarm Optimization. In: Intelligent systems conference (IntelliSys), (IEEE, 2017), pp 879–885
57. Hoffmann BS (2010) Similarity search with set intersection as a distance measure
58. Anton H (2013) Elementary linear algebra, Binder ready version. Wiley, New York
59. Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315(5814):972
60. Maneewongvatana S, Mount DM (1999) It's okay to be skinny, if your friends are fat. In: Center for geometric computing 4th annual workshop on computational geometry, vol 2, pp 1–8
61. Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509
62. Pelleg D, Moore A (2000) Accelerating exact k-means algorithms with geometric reasoning. Carnegie-Mellon University, Pittsburgh **(Tech. rep.)**
63. Rosenberg A, Hirschberg J (2007) V-measure: a conditional entropy-based external cluster evaluation measure. In: EMNLP-CoNLL, vol 7, pp 410–420
64. Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837
65. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825
66. Rand WM (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66(336):846
67. Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193
68. Romano S, Vinh NX, Bailey J, Verspoor K (2016) Adjusting for chance clustering comparison measures. J Mach Learn Res 17(1):4635
69. Dheeru D, Karra Taniskidou E (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed June 2019
70. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor 11(1):10
71. Bandyopadhyay S, Saha S (2013) Some single-and multiobjective optimization techniques. In: Unsupervised classification, Springer, pp 17–58
72. Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization method in multiobjective problems. In: Proceedings of the 2002 ACM symposium on applied computing (ACM), pp 603–607
73. Gong C, Chen H, He W, Zhang Z (2017) Improved multi-objective clustering algorithm using particle swarm optimization. PLoS One 12(12):e0188815
74. Maulik U, Bandyopadhyay S (2000) Genetic algorithm-based clustering technique. Pattern Recognit 33(9):1455
75. Chang DX, Zhang XD, Zheng CW (2009) A genetic algorithm with gene rearrangement for k-means clustering. Pattern Recognit 42(7):1210
76. Beg A, Islam MZ (2015) Clustering by genetic algorithm-high quality chromosome selection for initial population. In: IEEE 10th conference on Industrial electronics and applications (ICIEA), (IEEE, 2015), pp 129–134
77. Liu Y, Wu X, Shen Y (2011) Automatic clustering using genetic algorithms. Appl Math Comput 218(4):1267
78. Siddiqi UF, Sait SM (2017) A new heuristic for the data clustering problem. IEEE Access 5:6801
79. Aggarwal CC, Hinneburg A, Keim DA (2001) On the surprising behavior of distance metrics in high dimensional space. In: International conference on database theory, Springer, pp 420–434
80. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53
81. Kodinariya TM, Makwana PR (2013) Review on determining number of cluster in k-means clustering. Int J 1(6):90
82. Aljarah I, Ala'M AZ, Faris H, Hassonah MA, Mirjalili S, Saadeh H (2018) Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cogn Comput 10:478–495
83. Faris H, Hassonah MA, Ala'M AZ, Mirjalili S, Aljarah I, (2018) A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture. Neural Comput Appl 30(8):2355–2369