symmetry

MDPI

Article

Accelerating Density Peak Clustering Algorithm

Jun-Lin Lin [1,2]

[1] Department of Information Management, Yuan Ze University, Taoyuan 32003, Taiwan; jun@saturn.yzu.edu.tw; Tel.: +886-3-463-8800 (Ext. 2611)
[2] Innovation Center for Big Data and Digital Convergence, Yuan Ze University, Taoyuan 32003, Taiwan

Received: 28 May 2019; Accepted: 26 June 2019; Published: 2 July 2019

Abstract: The Density Peak Clustering (DPC) algorithm is a new density-based clustering method. It spends most of its execution time on calculating the local density and the separation distance for each data point in a dataset. The purpose of this study is to accelerate its computation. On average, the DPC algorithm scans half of the dataset to calculate the separation distance of each data point. We propose an approach to calculate the separation distance of a data point by scanning only the neighbors of the data point. Additionally, the purpose of the separation distance is to assist in choosing the density peaks, which are the data points with both high local density and high separation distance. We propose an approach to identify non-peak data points at an early stage to avoid calculating their separation distances. Our experimental results show that most of the data points in a dataset can benefit from the proposed approaches to accelerate the DPC algorithm.

Keywords: clustering; density-based clustering; density peak

1. Introduction

Clustering is the process of categorizing objects into groups (called clusters) of similar objects and is a widely-used data mining technique both in academic and applied research [1,2]. Many clustering methods appear in the literature, but they differ in the notion of similarity. For example, the *k*-means algorithm [3] represents each cluster by a centroid, and those objects near the same centroid are deemed similar; the DBSCAN algorithm [4] defines the notion of density and deems the objects in a continuous region with a density exceeding a specified threshold as similar; some studies measure the similarity using the concept of symmetry.

The *k*-means algorithm is an example of the partitioning-based clustering methods, and most of the partitioning-based clustering methods can find only spherical shaped clusters [5]. In contrast, the DBSCAN algorithm is an example of the density-based clustering methods, which can not only find clusters of arbitrary shapes but also detect outliers [5]. Although a density-based clustering method usually requires more execution time than a partitioning-based clustering method does, it can often discover meaningful clustering results that a partitioning-based clustering method cannot reveal. Several applications of clustering to real-world problems use both of these approaches to extract different clustering results of the same dataset, to highlight different aspects of the data.

The Density Peak Clustering (DPC) algorithm, proposed by Rodriguez and Laio [6], is a new density-based clustering method that has received much attention for the past few years [7–17]. It accelerates the clustering process by first searching for the density peaks in a dataset, and then constructing clusters from the density peaks. To search density peaks, DPC must calculate two quantities for each data point: local density and separation distance (see Section 2 for details) [9]. Then, data points with relatively high local density and separation distance are selected as the density peaks. Many works refer to the density peak of a cluster as the “center” of the cluster. Since density-based clustering methods yield clusters of arbitrary shapes, the notion of “center” is somewhat misleading. This work uses “density peak” instead of “center” to avoid confusion.

The contribution of this work is to propose two methods (called ADPC1 and ADPC2) that accelerate the DPC algorithm. The first method ADPC1 accelerates the calculation of separation distances and yields the same clustering results as that of the DPC algorithm. The second method ADPC2 accelerates the DPC algorithm by identifying a significant portion of the non-peak data points and avoiding calculating their separation distances. Since calculating the separation distances for all data points is a time-consuming step with $O(N^2)$ time complexity where N is the number of data points, our proposed methods can significantly speed up the DPC algorithm.

The rest of this work is organized as follows: Section 2 reviews related work, with a focus on the DPC algorithm. Sections 3 and 4 propose our methods. Section 5 presents the experimental results. Finally, Section 6 concludes this study.

2. Related Works

2.1. Clustering Methods

In the literature, clustering methods have been classified into several categories [18]: partitioning-based methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods. Partitioning-based methods (e.g., *k*-means and possibilistic *c*-means) focus on discovering compact and hyperellipsoidally shaped clusters. With *k*-means, the clustering results are sensitive to outliers. The possibilistic *c*-means (PCM) method is resilient to outliers, but it requires additional parameters γ, one for each cluster. Adaptive PCM algorithm [19] allows the parameters γ to change as the algorithm evolves.

Hierarchical methods work by iteratively (or recursively) dividing a large cluster into small clusters (or by combining small clusters as a large cluster). As a result, their clustering results can be represented by a dendrogram. Bianchi, et. [20] proposed a clustering method that forms clusters by iterative partitioning of an undirected graph.

Density-based methods discover clusters that are continuous regions with a high local density within the regions. Unlike the partitioning-based methods, density-based methods yield clusters of arbitrary shapes. Grid-based methods use a grid data structure to quantize the data space into a finite number of cells and perform the clustering operations directly on the cells. Model-based methods try to fit the data to some mathematical model.

Some clustering methods do not fit nicely into the above categorization. For example, subspace clustering [21] methods identify clusters based on their association with subspaces in high-dimensional spaces.

2.2. Density Peak Clustering Algorithm

As described in Section 1, the DPC algorithm [6] must calculate the local density and the separation distance for each data point. Given a dataset X, the local density $\rho(x_i)$ of a data point $x_i \in \mathrm{X}$ is the number of data points in the neighborhood of x_i. That is:

$$\rho(x_i) = |\mathrm{B}(x_i)| \tag{1}$$

where $\mathrm{B}(x_i)$ denotes the neighborhood of x_i and is defined as the set of data points in X whose distance to x_i is less than a user-specified parameter d_c. That is:

$$\mathrm{B}(x_i) = \{x_j \in \mathrm{X} | \mathrm{d}(x_i, x_j) < d_c\} \tag{2}$$

where $\mathrm{d}(x_i, x_j)$ represents the distance between x_i and x_j. Notably, Equations (2) and (1) use the parameter d_c as a hard threshold to derive the neighborhood and the local density of a data point, respectively.

The value of d_c can be chosen so that the average number of neighbors of a data point is around p% of the number of the data points in X, and the suggested value [6] for p is between 1 and 2. For small datasets, Rodriguez and Laio [6] suggested using an exponential kernel to calculate the local density, as shown in Equation (3):

$$\rho(x_i) = \sum_{x_j \in X} exp\left(-\frac{d(x_i, x_j)^2}{{d_c}^2}\right) \tag{3}$$

The separation distance $\delta(x_i)$ of x_i is the minimum distance from x_i to any other data point with a local density > $\rho(x_i)$, or the maximum distance from x_i to any other data point in X if there exists no data point with a local density > $\rho(x_i)$, as shown in Equation (4):

$$\delta(x_i) = \begin{cases} \min\limits_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j), \text{if } \rho(x_i) < \max\limits_{x_j \in X} \rho(x_j) \\ \max\limits_{x_j \in X} d(x_i, x_j), \text{otherwise.} \end{cases} \tag{4}$$

For ease of exposition, we use $\sigma(x_i)$ to denote the index *j* of the data point x_j that is the nearest to x_i and $\rho(x_j) > \rho(x_i)$, and if no such data point exists, $\sigma(x_i)$ is set to *i*, as shown in Equation (5):

$$\sigma(x_i) = \begin{cases} \operatorname*{argmin}\limits_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j), \text{if } \rho(x_i) < \max\limits_{x_j \in X} \rho(x_j) \\ i, \text{otherwise.} \end{cases} \tag{5}$$

Notably, there may be more than one data point that is the nearest to x_i and has a local density $> \rho(x_i)$. According to Laio's Matlab implementation of the DPC algorithm [22], if this situation happens, then $\sigma(x_i)$ is randomly chosen from the indexes of those data points with the highest local density among all the data points that are the nearest to x_i and have a local density $> \rho(x_i)$.

Once $\rho(x_i)$ and $\delta(x_i)$ of each data point have been determined, the DPC algorithm uses the following assumption to select density peaks: if a data point $x_i \in X$ is a density peak, then x_i must be surrounded by many data points (i.e., $\rho(x_i)$ is large) and must be at a relatively high distance from other data points with a local density greater than $\rho(x_i)$ (i.e., $\delta(x_i)$ is large). To assist choosing the density peaks, the DPC algorithm plots each data point in a decision graph, which is a two-dimensional graph with the local density and the separation distance as the horizontal and vertical axes, respectively. Data points with both high local density and high separation distance are manually selected as the density peaks. Alternatively, one can set a threshold on $\gamma(x_i) = \rho(x_i)\delta(x_i)$ and select data points with $\gamma(x_i)$ greater than the threshold as density peaks [6].

After all density peaks have been determined, each density peak acts as the starting point of a cluster, and thus the number of density peaks equals the number of clusters. Each non-peak data point is assigned to the same cluster as its nearest data point of higher density, i.e., data points x_i is assigned to the cluster that contains $x_{\sigma(x_i)}$. Let y_i denote the cluster label of data point x_i, then $y_i = y_{\sigma(x_i)}$.

Algorithm 1 shows the DPC algorithm. Notably, it is important to sort the data points by their local density descendingly in Step 2 so that calculating $\delta(x_i)$ and $\sigma(x_i)$ in Step 3 and the cluster assignment in Step 6 can be done efficiently. Without Step 2, for each data point x_i, Step 3 would require scanning all data points in X to find the data points with a local density > $\rho(x_i)$. With Step 2, Step 3 only needs to scan the data points located before x_i in X, and, thus, reduces the running time of Step 3 by half on average. Additionally, with Step 2, data points with higher local density are processed earlier in Step 6. Since $\rho(x_{\sigma(x_i)}) > \rho(x_i)$, $y_{\sigma(x_i)}$ will be determined before y_i in Step 6, and, thus, Step 6 can complete cluster assignment in $O(N)$ time.

Algorithm 1. DPC algorithm.

```
Input: the set of data points $X \in \mathbb{R}_{N\times M}$ and the parameters $d_c$ for defining
the neighborhood, and $d_r$ for selecting density peaks
Output: the label vector of cluster index $y \in \mathbb{R}_{N\times 1}$
Algorithm:
1.  Calculate $\rho(x_i)$ for each $x_i \in X$ using either (1) or (3).
2.  Sort all data points in X by their local densities descendingly.
```

```
3.  Calculate $\delta(x_i)$ and $\sigma(x_i)$ for each $x_i \in X$ using (4) and (5), respectively.
4.  Select data points with $\rho(x_i)\delta(x_i) > d_r$ as density peaks.
5.  For each density peak $x_i$, set $y_i = i$.   // starting point of each cluster
6.  For each non-peak data point $x_i$, set $y_i = y_{\sigma(x_i)}$.   // cluster assignment
7.  Return y.
```

Appendix A describes Laio's implementation details for Step 3 of the DPC algorithm. Specifically, we discuss how it handles two ambiguous situations when calculating the separation distance using Equation (4).

3. Accelerating APC by Scanning Neighbors Only

As described earlier, for each data point $x_i \in X$, Step 3 of the DPC algorithm in Algorithm 1 requires to scan half of X on average to find $x_{\sigma(x_i)}$, i.e., the data point nearest to x_i and with a local density > $\rho(x_i)$. Observation 1 shows that we can find $x_{\sigma(x_i)}$ by scanning only the neighbors of x_i, if the local density of x_i is less than the maximal local density of its neighbors. Most data points satisfy this condition, and the size of a data point's neighborhood is much smaller than the size of X, so the time complexity of Step 3 can be reduced from $O(N^2)$ to $O(Nb)$ where N denotes the number of data points in X, and b denotes the average neighborhood size.

Observation 1. *If $\rho(x_i) < \max_{x_j \in B(x_i)} \rho(x_j)$ for some data point $x_i \in X$, then the data point nearest to x_i and with a local density > $\rho(x_i)$ is in $B(x_i)$, i.e., $x_{\sigma(x_i)} \in B(x_i)$.*

Based on Observation 1, we rewrite Equations (4) and (5) to Equations (6) and (7) below. In Algorithm 2, we propose an accelerated version of DPC (called ADPC1), which produces the same clustering results as DPC does, but in less time:

$$\delta(x_i) = \begin{cases} \min\limits_{j:x_j \in B(x_i) \wedge \rho(x_j) > \rho(x_i)} d(x_i, x_j), & \text{if } B(x_i) \neq \phi \text{ and } \rho(x_i) < \max\limits_{x_j \in B(x_i)} \rho(x_j) \\ \min\limits_{j:x_j \in X \wedge \rho(x_j) > \rho(x_i)} d(x_i, x_j), & \text{if } B(x_i) = \phi \text{ or } \left(\rho(x_i) = \max\limits_{x_j \in B(x_i)} \rho(x_j) \text{ and } \rho(x_i) \neq \max\limits_{x_j \in X} \rho(x_j)\right) \\ \max\limits_{x_j \in X} d(x_i, x_j), & \text{if } \rho(x_i) = \max\limits_{x_j \in X} \rho(x_j). \end{cases} \quad (6)$$

$$\sigma(x_i) = \begin{cases} \operatorname*{argmin}\limits_{j:x_j \in B(x_i) \wedge \rho(x_j) > \rho(x_i)} d(x_i, x_j), & \text{if } B(x_i) \neq \phi \text{ and } \rho(x_i) < \max\limits_{x_j \in B(x_i)} \rho(x_j) \\ \operatorname*{argmin}\limits_{j:x_j \in X \wedge \rho(x_j) > \rho(x_i)} d(x_i, x_j), & \text{if } B(x_i) = \phi \text{ or } \left(\rho(x_i) = \max\limits_{x_j \in B(x_i)} \rho(x_j) \text{ and } \rho(x_i) \neq \max\limits_{x_j \in X} \rho(x_j)\right) \\ i, & \text{if } \rho(x_i) = \max\limits_{x_j \in X} \rho(x_j). \end{cases} \quad (7)$$

Algorithm 2. ADPC1 algorithm.

```
Input: the set of data points $X \in \mathbb{R}_{N \times M}$ and the parameters $d_c$ for defining the neighborhood, and $d_r$ for selecting density peaks
Output: the label vector of cluster index $y \in \mathbb{R}_{N \times 1}$
Algorithm:
1.  Calculate $\rho(x_i)$ and $B(x_i)$ for each $x_i \in X$ using either (1) and (2) or (3) and (2).
2.  Sort all data points in X by their local density descendingly.
3.  Calculate $\delta(x_i)$ and $\sigma(x_i)$ for each $x_i \in X$ using (6) and (7).
4.  Select data points with $\rho(x_i)\delta(x_i) > d_r$ as density peaks.
5.  For each density peak $x_i$, set $y_i = i$.   // starting point of each cluster
6.  For each non-peak data point $x_i$, set $y_i = y_{\sigma(x_i)}$.   // cluster assignment
7.  Return y.
```

*The parts different from the DPC in Algorithm 1 are highlighted in red.

Notably, in Step 1 of Algorithm 1, the DPC algorithm uses $\mathrm{B}(x_i)$ to calculate local density $\rho(x_i)$, but, afterwards, $\mathrm{B}(x_i)$ is no longer needed. However, in Algorithm 2, the ADPC1 algorithm needs to keep $\mathrm{B}(x_i)$ for calculating $\delta(x_i)$ and $\sigma(x_i)$ in Step 3. If $\rho(x_i) < \max_{x_j \in \mathrm{B}(x_i)} \rho(x_j)$, then we only need to scan $\mathrm{B}(x_i)$ to calculate $\sigma(x_i)$ and $\delta(x_i)$. If $\rho(x_i) < \max_{x_j \in \mathrm{B}(x_i)} \rho(x_j)$ does not hold, then $\sigma(x_i)$ and $\delta(x_i)$ are calculated the same way as in the DPC algorithm, i.e., scanning half of the dataset X on average. Since it is often that the local density of a data point is less than the maximal local density of its neighbors, ADPC1 can greatly reduce the execution time. Appendix B describes the implementation details for Step 3 of the ADPC1 algorithm.

4. Accelerating APC by Skipping Non-Peaks

Both DPC and ADPC1 need to calculate the separation distance $\delta(x_i)$ for each data point x_i. Recall that the purpose of calculating $\delta(x_i)$ is to assist determining whether x_i is a density peak. Therefore, if we can determine x_i as a non-peak data point at an early stage, then there is no need to calculate $\delta(x_i)$. Observation 2 shows the necessary condition of a density peak, which can be applied to detect most non-peak data points in a dataset.

Observation 2. *If $\rho(x_i) < \max_{x_j \in B(x_i)} \rho(x_j)$ for some data point $x_i \in \mathrm{X}$, then x_i cannot be a density peak.*

If x_i is not a density peak, then we can omit to calculate $\delta(x_i)$ by simply assigning $\delta(x_i)$ to a small value, say 0. However, without calculating $\delta(x_i)$, we do not know $\sigma(x_i)$, i.e., the index of the data point nearest to x_i and with a local density $> \rho(x_i)$. Notably, $\sigma(x_i)$ is needed for cluster assignment in Step 6 of the DPC and ADPC1 algorithms. To resolve this problem, we use the index of the data point with the highest local density in the neighborhood of x_i as a surrogate for $\sigma(x_i)$ and redefine Equations (6) and (7) as Equations (8) and (9) below:

$$\delta(x_i) = \begin{cases} 0, & \text{if } \mathrm{B}(x_i) \neq \phi \text{ and } \rho(x_i) < \max_{x_j \in \mathrm{B}(x_i)} \rho(x_j) \\ \min_{j: x_j \in \mathrm{X} \wedge \rho(x_j) > \rho(x_i)} d(x_i, x_j), & \text{if } \mathrm{B}(x_i) = \phi \text{ or } \left(\rho(x_i) \geqq \max_{x_j \in \mathrm{B}(x_i)} \rho(x_j) \text{ and } \rho(x_i) \neq \max_{x_j \in \mathrm{X}} \rho(x_j)\right) \\ \max_{x_j \in \mathrm{X}} d(x_i, x_j), & \text{if } \rho(x_i) = \max_{x_j \in \mathrm{X}} \rho(x_j). \end{cases} \tag{8}$$

$$\sigma(x_i) = \begin{cases} \operatorname{argmax}_{j: x_j \in \mathrm{B}(x_i)} \rho(x_j), & \text{if } \mathrm{B}(x_i) \neq \phi \text{ and } \rho(x_i) < \max_{x_j \in \mathrm{B}(x_i)} \rho(x_j) \\ \operatorname{argmin}_{j: x_j \in \mathrm{X} \wedge \rho(x_j) > \rho(x_i)} d(x_i, x_j), & \text{if } \mathrm{B}(x_i) = \phi \text{ or } \left(\rho(x_i) \geqq \max_{x_j \in \mathrm{B}(x_i)} \rho(x_j) \text{ and } \rho(x_i) \neq \max_{x_j \in \mathrm{X}} \rho(x_j)\right) \\ i, & \text{if } \rho(x_i) = \max_{x_j \in \mathrm{X}} \rho(x_j). \end{cases} \tag{9}$$

Notably, Equations (8) and (9) only modify the first case of Equations (6) and (7), i.e., when the local density of x_i is less than the maximal local density of its neighbors. Based on Equations (8) and (9), we propose another accelerated version of DPC (called ADPC2), which is the same as ADPC1 in Algorithm 2 except that Step 3 of ADPC2 uses Equations (8) and (9) instead of Equations (6) and (7) to calculate $\delta(x_i)$ and $\sigma(x_i)$, as shown in Algorithm 3. Notably, because ADPC1 and ADPC2 calculate $\sigma(x_i)$ differently, their clustering results can be slightly different from each other. Appendix C describes the implementation details for Step 3 of the ADPC2 algorithm.

Algorithm 3. ADPC2 algorithm.

Input: the set of data points $\mathrm{X} \in \mathbb{R}_{N\times M}$ and the parameters d_c for defining the neighborhood, and d_r for selecting density peaks

Output: the label vector of cluster index $\mathrm{y} \in \mathbb{R}_{N\times 1}$

Algorithm:

1. Calculate $\rho(x_i)$ and $\mathrm{B}(x_i)$ for each $x_i \in \mathrm{X}$ using either (1) and (2) or (3) and (2).

```
2. Sort all data points in X by their local density descendingly.
3. Calculate $\delta(x_i)$ and $\sigma(x_i)$ for each $x_i \in X$ using (8) and (9).
4. Select data points with $\rho(x_i)\delta(x_i) > d_r$ as density peaks.
5. For each density peak $x_i$, set $y_i = i$.    // starting point of each cluster
6. For each non-peak data point $x_i$, set $y_i = y_{\sigma(x_i)}$.    // cluster assignment
7. Return y.
```

*The parts different from the DPC in Algorithm 1 are highlighted in red.

5. Performance Study

5.1. Test Datasets

In this study, we use 12 well-known two-dimensional synthetic datasets to demonstrate the performance of the proposed algorithms. Dataset Spiral [23] consists of three spiral-shaped clusters. Dataset Flame [24] consists of two non-Gaussian clusters of points, where both clusters are of different sizes and shapes. Dataset Aggregation [25] consists of seven perceptually distinct (non-Gaussian) clusters of points. Dataset R15 [26] consists of 15 similar Gaussian clusters that are positioned on concentric circles. Dataset D31 [26] consists of 31 similar Gaussian clusters that are positioned along random curves. Datasets A1, A2, and A3 [27] contain 20, 35, and 50 circular clusters, respectively, where each cluster has 150 points. Datasets S1, S2, S3, and S4 [28] each contain 15 Gaussian clusters, where the degree of cluster overlapping is S1 < S2 < S3 < S4. Appendix D gives a detailed characterization of these datasets.

5.2. Experiment Setup

The experiment was divided into two tests. Test 1 used a hard threshold to calculate the local density, as defined in Equations (1) and (2); Test 2 used an exponential kernel to calculate the local density, as defined in Equation (3). In both tests, the value of d_c for defining the neighborhood is determined by the parameter p, as suggested in [6] and described in Section 2. We varied the value of p from 0.5 to 4 with a step size of 0.5. A large p implied a large d_c and consequently a large neighborhood.

In this experimental study, we compared the performance of the proposed ADPC1 and ADPC2 against DPC. Recall that both ADPC1 and ADPC2 accelerated the way to derive the separation distances of those data points with a local density less than the maximal local density of their neighbors. Thus, we calculated the proportion (denoted by $\breve{R}$) of such data points in a dataset for various p values, i.e., $\breve{R} = \frac{\breve{N}}{N}$ where $\breve{N}$ is the number of such data points in the dataset, and N is the total number of data points in the dataset. Usually, both $\breve{N}$ and $\breve{R}$ grow with a large neighborhood (i.e., a large d_c or p). Thus, the proposed ADPC1 and ADPC2 should perform better with a larger p.

Since these three algorithms only differ on how to calculate the separation distance, we collected and compared their execution time for calculating the local density and the separation distance, i.e., from Step 1 to Step 3 of these algorithms in Algorithms 1 and 2. Then, for ease of comparison, we calculated the percentage of execution time improvement of ADPC1 (or ADPC2) over DPC by the difference of the execution times of DPC and ADPC1 (or ADPC2), divided by the execution time of DPC.

5.3. Experiment Results

5.3.1. Test 1: Use a Fixed Threshold for Local Density

In Test 1, a fixed threshold is used to determine the neighborhood for calculating the local density of each data point. Table 1 shows the value of $\breve{R}$, i.e., the proportion of data points with a local density less than the maximal local density of their neighbors. According to Table 1, except for some small datasets (e.g., Spiral, Flame, Aggregation, and R15 datasets) and small p combinations, the value of $\breve{R}$ is usually greater than 80% in most cases, indicating that a large proportion of the data points in a dataset can be benefited from ADPC1 and ADPC2 to accelerate the calculation of their separation distances. Please refer to Table A9 in Appendix E for the value of $\breve{N}$, i.e., the number of data points with a local density less than the maximal local density of their neighbors.

A larger p implies a larger d_c, and thus a larger neighborhood range and probably more neighbors in the neighborhood. Intuitively, for a data point with a larger number of neighbors, it becomes less likely that the local density of the data point is greater than the maximal local density of its neighbors. Therefore, as the value of p increases, the value of $\breve{R}$ tend to increase (with some exceptions).

Table 1. The value of $\breve{R}$ for various dataset and p combinations. (Using a fixed threshold)

Dataset	p = 0.5	p = 1	p = 1.5	p = 2	p = 2.5	p = 3	p = 3.5	p = 4
Spiral	19.87%	28.85%	35.26%	44.55%	51.60%	59.94%	66.67%	74.36%
Flame	31.25%	59.58%	73.75%	86.67%	88.75%	92.08%	94.17%	96.25%
Aggregation	77.28%	89.34%	94.42%	96.45%	96.83%	96.57%	96.57%	97.46%
R15	58.00%	81.67%	89.50%	93.00%	93.83%	95.00%	94.00%	94.50%
D31	95.23%	97.68%	97.87%	97.97%	97.19%	87.94%	65.61%	80.06%
A1	95.13%	98.50%	98.90%	98.93%	99.03%	99.10%	98.80%	98.73%
A2	97.90%	99.07%	98.90%	98.76%	99.09%	99.31%	99.73%	99.75%
A3	98.64%	98.91%	98.84%	99.41%	99.69%	99.77%	99.81%	99.87%
S1	86.10%	96.12%	98.24%	98.72%	98.90%	99.12%	99.48%	99.46%
S2	87.02%	96.68%	98.68%	99.06%	99.00%	99.08%	99.36%	99.22%
S3	89.48%	97.64%	98.84%	99.18%	99.54%	99.40%	99.54%	99.64%
S4	87.10%	96.36%	98.60%	99.10%	99.28%	99.36%	99.44%	99.52%

Table 2 shows the percentage of execution time improvement of ADPC1 and ADPC2 over DPC. Except for the two small datasets Spiral and Flame at p = 0.5, both ADPC1 and ADPC2 substantially reduced the execution time of DPC. ADPC2 took less time than ADPC1 did for most dataset and p value combinations. For the execution time of the three algorithms, please see Table A11 in Appendix E.

For most cases in Table 1, the values of $\breve{R}$ were large and did not change much as the value of p increased. As a result, the impact of p's value on the execution time improvement was not obvious in Table 2. To show that the impact of $\breve{R}$ on the percentage of execution time improvement, consider the case of dataset D31 at p = 3 and 3.5. In Table 1, the value of $\breve{R}$ dropped from 87.94% at p = 3 to 65.61% at p = 3.5. The corresponding case in Table 2 showed that at p = 3, ADPC1 (or ADPC2) incurred the execution time improvement over DPC by 77.86% (or 80.44%). However, at p = 3.5, ADPC1 (or ADPC2) incurred the execution time improvement over DPC by only 47.06% (or 48.53%). This example shows that a large $\breve{R}$ helps ADPC1 and ADPC2 to reduce the percentage of execution time improvement. However, if a small p is applied on a small dataset, then the resulting $\breve{R}$ value is too small, causing ADPC2 to perform slower than DPC does (e.g., datasets Flame and Spiral at p = 0.5).

Table 2. Percentage of execution time improvements over DPC (using a fixed threshold).

Dataset	Algorithm	p = 0.5	p = 1	p = 1.5	p = 2	p = 2.5	p = 3	p = 3.5	p = 4
Spiral	ADPC1	7.89%	12.31%	54.43%	38.46%	0.00%	33.33%	33.33%	0.00%
	ADPC2	−18.42%	29.42%	43.04%	10.09%	33.34%	0.07%	0.04%	66.62%
Flame	ADPC1	50.00%	0.01%	50.03%	100.00%	50.09%	49.99%	100.00%	50.00%
	ADPC2	-0.01%	50.01%	50.10%	50.00%	50.00%	100.00%	50.08%	50.07%
Aggregation	ADPC1	67.86%	70.59%	77.78%	78.95%	78.95%	73.69%	80.00%	72.23%
	ADPC2	69.09%	76.47%	83.33%	84.21%	84.21%	78.95%	75.00%	77.78%
R15	ADPC1	20.00%	54.55%	54.55%	72.73%	72.73%	75.00%	72.73%	72.73%
	ADPC2	40.00%	54.55%	72.73%	72.73%	72.73%	83.33%	81.82%	81.82%
D31	ADPC1	79.36%	82.37%	83.02%	82.14%	81.43%	77.86%	47.06%	53.88%
	ADPC2	79.56%	83.56%	84.29%	83.93%	83.21%	80.44%	48.53%	56.47%
A1	ADPC1	79.77%	83.14%	83.33%	82.26%	81.15%	80.84%	80.44%	78.73%

	ADPC2	80.16%	83.91%	84.47%	83.77%	83.41%	83.47%	83.03%	81.79%
A2	ADPC1	83.35%	84.11%	83.38%	82.51%	81.51%	80.42%	79.92%	79.12%
	ADPC2	83.98%	84.61%	82.13%	84.12%	83.37%	82.76%	82.50%	82.05%
A3	ADPC1	84.56%	83.59%	82.57%	82.33%	81.67%	80.11%	79.22%	79.09%
	ADPC2	84.91%	85.11%	84.61%	83.90%	83.54%	83.07%	82.73%	81.99%
S1	ADPC1	65.66%	79.50%	81.64%	81.86%	81.78%	81.06%	80.13%	79.39%
	ADPC2	66.22%	80.29%	82.98%	83.46%	83.64%	83.44%	82.75%	82.03%
S2	ADPC1	67.13%	79.81%	82.37%	82.14%	81.55%	80.94%	79.89%	78.85%
	ADPC2	67.55%	80.76%	83.45%	83.63%	83.56%	82.95%	82.44%	81.96%
S3	ADPC1	71.51%	81.36%	82.43%	82.43%	81.86%	80.73%	79.89%	78.85%
	ADPC2	71.78%	82.45%	83.65%	83.89%	83.60%	82.99%	82.57%	81.93%
S4	ADPC1	68.75%	79.65%	82.05%	81.99%	80.95%	80.54%	79.87%	79.17%
	ADPC2	69.03%	80.46%	83.27%	83.60%	82.97%	82.95%	82.40%	82.06%

5.3.2. Test 2: Use an Exponential Kernel for Local Density

In Test 2, an exponential kernel (see Equation (3)) is used to calculate the local density of each data point. Table 3 shows the value of $\breve{R}$ for various dataset and p combinations. Please refer to Table A10 in Appendix E for the value of $\breve{N}$. Similar to Table 1 in Test 1, a large proportion of the data points can be benefited from ADPC1 and ADPC2. Furthermore, each value of $\breve{R}$ in Table 3 is greater than its corresponding value in Table 1. That is, for the same dataset and the same p value, an even larger proportion of data points can be benefited from ADPC1 and ADPC2 using the exponential kernel than using a fixed threshold to calculate the local density. In Test 2, a larger p value always incurs a larger $\breve{R}$ values in Table 3. The results are consistent with that of Test 1.

Table 3. The value of $\breve{R}$ for various dataset and p combinations. (Using an exponential kernel)

Dataset	$p = 0.5$	$p = 1$	$p = 1.5$	$p = 2$	$p = 2.5$	$p = 3$	$p = 3.5$	$p = 4$
Spiral	47.44%	80.45%	91.67%	96.47%	98.72%	99.04%	99.04%	99.04%
Flame	49.17%	78.33%	91.25%	96.25%	96.67%	97.50%	97.50%	97.92%
Aggregation	92.01%	97.59%	98.35%	98.73%	98.86%	98.98%	99.11%	99.11%
R15	73.17%	87.83%	92.67%	95.33%	96.00%	96.83%	97.33%	97.33%
D31	96.23%	98.19%	98.74%	98.94%	99.00%	99.03%	99.19%	99.52%
A1	97.13%	98.90%	99.23%	99.27%	99.33%	99.33%	99.37%	99.43%
A2	98.57%	99.28%	99.31%	99.35%	99.56%	99.73%	99.79%	99.87%
A3	98.99%	99.29%	99.37%	99.67%	99.77%	99.88%	99.91%	99.91%
S1	91.84%	97.60%	98.88%	99.26%	99.56%	99.66%	99.66%	99.68%
S2	92.10%	98.20%	99.36%	99.60%	99.64%	99.68%	99.70%	99.70%
S3	94.54%	98.64%	99.32%	99.58%	99.66%	99.70%	99.74%	99.76%
S4	93.04%	98.20%	99.06%	99.42%	99.48%	99.60%	99.70%	99.74%

Table 4 shows the percentage of execution time improvement of ADPC1 and ADPC2 over DPC. ADPC1 always took less time than DPC did, except at $p = 0.5$ for Spiral dataset; ADPC2 always took less time than DPC did, except at $p = 0.5$ for Flame dataset. In general, both ADPC1 and ADPC2 required substantially less execution time than DPC did. ADPC2 usually achieved higher improvement than ADPC1 did; however, the difference is small. For the execution time of the three algorithms, please see Table A12 in Appendix E.

Table 4. Percentage of execution time improvements over DPC (using an exponential kernel).

Dataset	Algorithm	$p = 0.5$	$p = 1$	$p = 1.5$	$p = 2$	$p = 2.5$	$p = 3$	$p = 3.5$	$p = 4$
Spiral	ADPC1	−11.97%	7.48%	17.07%	23.06%	21.44%	6.68%	21.43%	14.30%
	ADPC2	7.40%	21.15%	26.25%	6.52%	21.43%	20.00%	21.44%	21.44%
Flame	ADPC1	12.50%	12.50%	12.50%	11.11%	25.00%	22.21%	12.50%	22.22%
	ADPC2	0.00%	12.50%	0.001%	22.22%	12.51%	22.22%	12.50%	22.22%
Aggregation	ADPC1	16.98%	15.00%	15.00%	17.07%	17.07%	15.00%	13.75%	13.75%
	ADPC2	14.52%	15.00%	16.25%	18.29%	18.29%	16.25%	16.25%	16.25%
R15	ADPC1	9.80%	10.42%	12.49%	14.58%	12.50%	16.33%	14.89%	14.89%

	ADPC2	9.80%	10.42%	12.49%	16.67%	14.58%	16.33%	14.89%	17.02%
D31	ADPC1	15.82%	15.49%	16.55%	16.08%	15.74%	16.01%	15.81%	15.51%
	ADPC2	15.90%	16.01%	17.80%	16.80%	17.15%	16.21%	15.43%	16.63%
A1	ADPC1	16.15%	15.56%	15.40%	15.73%	15.38%	15.32%	15.54%	14.99%
	ADPC2	16.99%	16.16%	16.19%	16.51%	16.61%	16.19%	16.88%	16.78%
A2	ADPC1	15.40%	16.05%	16.21%	16.81%	14.43%	16.21%	16.73%	15.75%
	ADPC2	15.90%	16.94%	16.49%	16.70%	16.52%	17.31%	17.38%	17.04%
A3	ADPC1	17.25%	16.59%	15.09%	16.76%	15.00%	16.18%	14.56%	14.76%
	ADPC2	16.84%	15.38%	16.08%	18.30%	15.68%	17.06%	15.70%	16.15%
S1	ADPC1	14.59%	15.16%	16.13%	15.32%	16.08%	17.20%	15.87%	17.60%
	ADPC2	13.83%	15.53%	15.58%	17.18%	16.07%	17.57%	16.69%	18.72%
S2	ADPC1	13.74%	16.19%	17.47%	16.00%	16.17%	11.36%	15.92%	15.73%
	ADPC2	14.07%	16.65%	17.76%	16.28%	16.88%	16.63%	17.40%	17.43%
S3	ADPC1	16.26%	16.52%	16.77%	16.25%	16.77%	17.31%	15.77%	16.29%
	ADPC2	14.75%	16.55%	15.87%	16.29%	15.52%	17.74%	16.46%	16.27%
S4	ADPC1	14.48%	16.03%	17.01%	15.91%	16.51%	16.08%	15.46%	15.83%
	ADPC2	13.56%	16.53%	16.55%	16.87%	16.26%	16.77%	16.28%	16.70%

Comparing Tables 2 and 4 show that the execution time improvement is greater in Test 1 than in Test 2. In Test 1, calculating the local density of a data point requires simply counting the number of data points in its neighborhood (see Equations (1) and (2)). However, in Test 2, calculating the local density of a data point is much time consuming because it requires calculating an exponential function N-1 times, where N is the number of data points in the dataset (see Equation (3)). The execution time collected in this study is the execution time for calculating the local density and the separation distance. All three algorithms use the same method to calculate the local density, and they are only differed on how to calculate the separation distance. That is, the execution time improvement of ADPC1 and ADPC2 over DPC is due to the improvement on how to calculate the separation distance. Since much more time was spent on calculating the local density in Test 2 than in Test 1, the percentage of execution time improvement is smaller in Test 2 than in Test 1.

6. Conclusions

As discussed in Section 3, if the local density of a data point x_i is less than the largest local density of its neighbors, then ADPC1 and ADPC2 can reduce the time complexity for calculating the separation distance of x_i from $O(N)$ to $O(|B(x_i)|)$ where N denotes the number of data points in the dataset, and $|B(x_i)|$ denotes the number of neighbors of x_i. Thus, the effectiveness of both ADPC1 and ADPC2 depends on the proportion of the data points satisfying this condition. The experimental results in Tables 1 and 3 show that most data points in a dataset satisfy this condition, except for some small datasets using a small neighborhood setting. Consequently, both ADPC1 and ADPC2 improve the execution time of DPC, as shown in Tables 2 and 4. Furthermore, in most cases, ADPC2 requires less execution time than ADPC1 does.

Consider the case that all data points in a continuous region have the same local density. Then, there exists no data point in the region with a local density less than the largest local density of its neighbors, and consequently, both ADPC1 and ADPC2 cannot accelerate the computation of the separation distance for the data points in this region. If the entire dataset contains many such regions, then the advantage of ADPC1 and ADPC2 diminishes. However, according to Tables 1 and 3, except for small datasets with a small neighborhood range (i.e., small d_c), both ADPC1 and ADPC2 are advantageous.

The proposed methods focus on accelerating the calculation of the separation distance. However, it is also possible to improve the DPC algorithm by accelerating the calculation of the local density [9]. Besides, the DPC algorithm has several shortcomings that have received much attention in the literature. First, choosing proper values for DPC's parameters is not straightforward, but it can highly affect the quality of the clustering results. To resolve this problem, [7] applied the concept of heat diffusion and [8] employed the potential entropy of the data field to determine the value of d_c. Additionally, [12] proposed a comparative technique to choose the density peaks. Thus, how to make the DPC algorithm more adaptive to the datasets with less human intervention is worthy of further investigation.

The local density of a data point x_i can be defined from two different perspectives. One is to specify a fixed distance and count the number of data points within the fixed distance from x_i. The DPC algorithm adopted this perspective. Another perspective is to specify a fixed number of neighbors and measure the distances of these neighbors to x_i. [13,14] adopted this perspective and defined new methods to calculate the local density based on the *k*-nearest neighbors of x_i. Since the definition of the local density significantly affects the clustering results, how to choose a proper method to define the local density is an important issue worthy of further investigation for density-based clustering algorithms.

Our future work intends to extend the DPC algorithm as a hierarchical clustering algorithm. Conceptually, the DPC algorithm builds a directed acyclic graph of all data points with an out-degree ≤ 1. Then, it selects several data points from the graph as the density peaks. Finally, it removes the outgoing links of the density peaks and breaks the graph into several subgraphs, each of which represents a cluster. By adding an ordering on the density peaks and incrementally removing the outgoing links of the density peaks according to this ordering, it is possible to yield the clustering results as a dendrogram. Furthermore, integrating the notion of central symmetry [29] or point symmetry [30] with the DPC algorithm for the detection of symmetry objects is also worthy of further investigation.

Funding: This research is supported by the Ministry of Science and Technology, Taiwan, under grant MOST 106-2221-E-155-038.

Acknowledgments: The author acknowledges the Innovation Center for Big Data and Digital Convergence at Yuan Ze University for supporting this study.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. Implementation Details for Calculating Separation Distance in DPC

Consider the case of more than one data points with a local density = the maximal local density in X. According to Equation (4), the separation distance of any data point x_i with the maximal local density will be set to the maximal distance from x_i to any point in X, i.e., $\max_{x_j \in \mathrm{X}} \mathrm{d}(x_i, x_j)$. Consequently, all data points with the maximal local density have high separation distances and, thus, will be chosen as the density peaks to form individual clusters, regardless that some of these data points may be near to each other. Notably, many data points with an equal local density are less likely to occur when Equation (3) is used for calculating local density because the Gaussian kernel in Equation (3) yields a floating-point value. However, the local density calculated using Equation (1) is an integer, and data points with an equal local density become common.

Laio's Matlab implementation of the DPC algorithm [22] resolved the above problem as follows. Recall that in Step 2 of the DPC algorithm in Algorithm 1, all data points in X are sorted by their local densities descendingly, i.e., $\rho(x_i) \geq \rho(x_j)$ for $i < j$. After Step 2, Laio used the ordering of the data points' positions in X instead of the ordering on local density for calculating separation distances. Specifically, Laio used Equations (A1) and (A2) instead of Equations (4) and (5) to calculate separation distances. Notably, in this work, we use x_i to denote the *i*th data point in X, and whenever the ordering of the data points in X is rearranged, the data point referred as x_i also changes. Notably, it is possible that more than one data point has the same local density, but each position in X can only be taken by one data point:

$$\delta(x_i) = \begin{cases} \max_{x_j \in \mathrm{X}} \mathrm{d}(x_i, x_j), & \text{if } i = 1 \\ \min_{j<i} \mathrm{d}(x_i, x_j), & \text{if } 1 < i \end{cases} \quad \text{(A1)}$$

$$\sigma(x_i) = \begin{cases} 1 & \text{if } i = 1 \\ \underset{j<i}{\operatorname{argmin}}\, d(x_i, x_j), & \text{if } 1 < i \end{cases} \tag{A2}$$

According to Equation (A1), only the separation distance of the first data point x_1 in X is set to the maximal distance, and for each data point $x_{i\neq 1}$, we only scan those data points located before x_i in X. Notably, with Equations (A1) and (A2), it is possible that $\sigma(x_{i\neq 1}) = j$ but $\rho(x_j) = \rho(x_i)$ because the ordering on local density is non-monotonically decreasing after Step 2 of the DPC algorithm in Algorithm 1. However, with Equations (4) and (5), if $\sigma(x_i) = j$ and $i \neq j$, then $\rho(x_j) > \rho(x_i)$ must hold. Thus, the definition of separation distance according to Equation (4) has been slightly modified in Equation (A1), and the difference is illustrated in Figure A1.

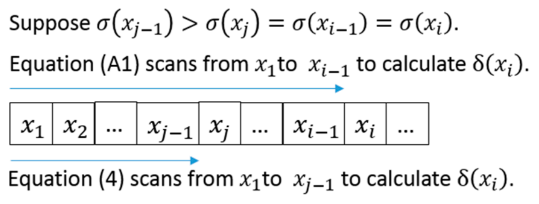

Suppose $\sigma(x_{j-1}) > \sigma(x_j) = \sigma(x_{i-1}) = \sigma(x_i)$.
Equation (A1) scans from x_1 to x_{i-1} to calculate $\delta(x_i)$.
x_1
x_2
...
x_{j-1}
x_j
...
x_{i-1}
x_i
...
Equation (4) scans from x_1 to x_{j-1} to calculate $\delta(x_i)$.

Figure A1. Difference between using Equation (4) and using Equation (A1) to calculate separation distance.

Figure A2 shows Laio's implementation for Step 3 of the DPC algorithm based on Equations (A1) and (A2). It is obvious that only the first data point is handled differently from the rest of the data points. Figure A3 shows the implementation of Step 3 of the DPC algorithm based on Equations (4) and (5). Data points with the maximal local density are handled in the same manner in Figures A2 and A3. However, for data points with a local density less than the maximal local density, Figure A3 faithfully implements Equations (4) and (5) to ensure that no data point with the same local density as x_i is scanned when calculating $\delta(x_i)$, as illustrated in Figure A1.

```
3.  $\delta(x_1) = \max_{x_j \in X} d(x_1, x_j)$;    $\sigma(x_1) = 1$;
    For $i = 2$ to $|X|$ do
        $\sigma(x_i) = \operatorname{argmin}_{j<i} d(x_i, x_j)$;
        $\delta(x_i) = \min_{j<i} d(x_i, x_j)$;
```

Figure A2. Implementation details of Step 3 of DPC algorithm (in Algorithm 1) based on Laio's implementation [22].

```
3.  $\delta(x_1) = \max_{x_j \in X} d(x_1, x_j)$;    $\sigma(x_1) = 1$;
    **For** $k = 2$ to $|X|$ **do**    //  for points with maximal local density in X
        **If** $\rho(x_k) = \max_{x_j \in X} \rho(x_j)$ **then**
            $\sigma(x_i) = \operatorname{argmin}_{j<i} d(x_i, x_j)$;   $\delta(x_i) = \min_{j<i} d(x_i, x_j)$;
        **else break**; // exit the for loop, and $x_k$ is the first point with $\rho(x_k) \neq \max_{x_j \in X} \rho(x_j)$
        **end if**
    **For** $i = k$ to $|X|$ **do**    //  for points with local density < maximal local density
        $\sigma(x_i) = \operatorname{argmin}_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j)$;    // only scan $x_j$ for $j < i$ until $\rho(x_j) \leq \rho(x_i)$.
        $\delta(x_i) = \min_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j)$;
```

Figure A3. Implementation details of Step 3 of DPC algorithm (in Algorithm 1) based on Equations (4) and (5).

Appendix B. Implementation Details for Calculating Separation Distance in ADPC1

Figure A4 gives a detailed description of Step 3 of the ADPC1 algorithm in Algorithm 2. Notably, before this step, all data points in X have been sorted by their local densities descendingly. To resolve the problem of multiple data points with the maximal local density, we adopt the same approach described in Appendix A, as shown in the first for loop of Figure A4. That is, only the separation distance of the first data point with the maximal local density in X is set to the maximal distance. For each data point $x_{i\neq 1}$ with the maximal local density, the separation distance $\delta(x_i)$ is set to the minimal distance from x_i to other data points located before x_i in X. Notably, the data points with the maximal local density are handled in the same manner in Figures A2, A3, and A4. The second for loop in Figure A4 applies Equations (6) and (7) to process the data points with local density < the maximal local density.

```
3.  $\delta(x_1) = \max_{x_j \in X} d(x_1, x_j)$;   $\sigma(x_1) = 1$;
    For $k$ = 2 to $|X|$ do   //  for points with maximal local density in $X$
        If $\rho(x_k) = \max_{x_j \in X} \rho(x_j)$ then
            $\sigma(x_i) = \operatorname{argmin}_{j<i} d(x_i, x_j)$;   $\delta(x_i) = \min_{j<i} d(x_i, x_j)$;
        else break; // exit the for loop, and $x_k$ is the first point with $\rho(x_k) \neq \max_{x_j \in X} \rho(x_j)$
        end if
    For $i = k$ to $|X|$ do   //  for points with local density < maximal local density
        If $B(x_i) = \phi$ or $\rho(x_i) = \max_{x_j \in B(x_i)} \rho(x_j)$ then   // the greatest density in $B(\boldsymbol{x}_i)$
            $\sigma(x_i) = \operatorname{argmin}_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j)$;   // only scan $x_j$ for $j < i$ until $\rho(x_j) \leq \rho(x_i)$.
            $\delta(x_i) = \min_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j)$;
        else   // not the greatest density in $B(x_i)$
            $\sigma(x_i) = \operatorname{argmin}_{j:x_j \in B(x_i) \wedge \rho(x_j)>\rho(x_i)} d(x_i, x_j)$;  // only scan $B(x_i)$
            $\delta(x_i) = \min_{j:x_j \in B(x_i) \wedge \rho(x_j)>\rho(x_i)} d(x_i, x_j)$;
        endif
```

Figure A4. Implementation details of Step 3 of ADPC1 algorithm (in Algorithm 2) based on Equations (6) and (7).

Appendix C. Implementation Details for Calculating Separation Distance in ADPC2

Figure A5 gives a detailed description of Step 3 of the ADPC2 algorithm in Algorithm 3. To resolve the problem of multiple data points with the maximal local density, we adopt the same approach described in Appendix A, as shown in the first for loop of Figure A5. The second for loop in Figure A5 bases on Equations (8) and (9) to process the data points with local density < the maximal local density.

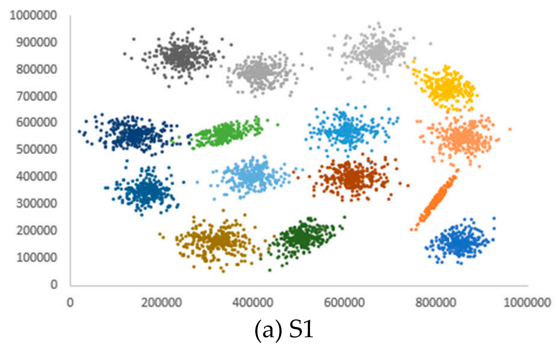
```
3.  $\delta(x_1) = \max_{x_j \in X} d(x_1, x_j)$;   $\sigma(x_1) = 1$;
    For $k$ = 2 to $|X|$ do   //  for points with maximal local density in $X$
```

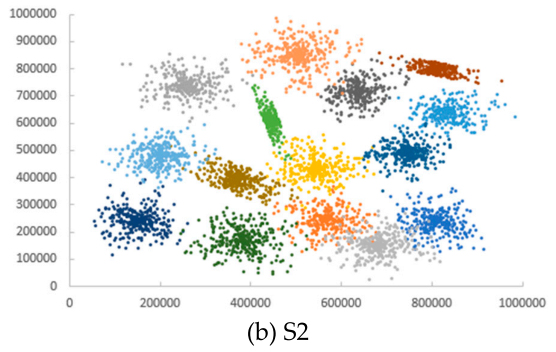
```
        If $\rho(x_k) = \max_{x_j \in X} \rho(x_j)$ then
            $\sigma(x_i) = \operatorname{argmin}_{j<i} d(x_i, x_j)$;   $\delta(x_i) = \min_{j<i} d(x_i, x_j)$;
        else break; // exit the for loop, and $x_k$ is the first point with $\rho(x_k) \neq \max_{x_j \in X} \rho(x_j)$
        end if
    For $i = k$ to $|X|$ do    //  for points with local density < maximal local density
        If $B(x_i) = \phi$ or $\rho(x_i) = \max_{x_j \in B(x_i)} \rho(x_j)$ then   // the greatest density in $B(x_i)$
            $\sigma(x_i) = \operatorname{argmin}_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j)$;   // only scan $x_j$ for $j < i$ until $\rho(x_j) \leq \rho(x_i)$.
            $\delta(x_i) = \min_{j:\rho(x_j)>\rho(x_i)} d(x_i, x_j)$;
        else    // not the greatest density in $B(x_i)$
            $\sigma(x_i) = \operatorname{argmax}_{j:x_j \in B(x_i)} \rho(x_j)$;   // use a surrogate
            $\delta(x_i) = 0$;
        endif
```

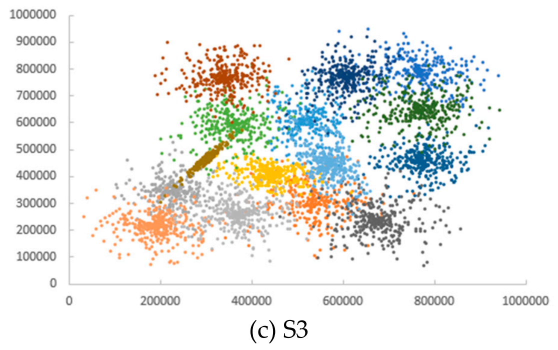
Figure A5. Implementation details of Step 3 of the ADPC2 algorithm (in Algorithm 3) based on Equations (8) and (9).

Appendix D. Datasets

Figures A6 and A7 show the data distribution of the 12 two-dimensional synthetic datasets used in Section 5. Table A8 describes the number of clusters and the number of points in these datasets.







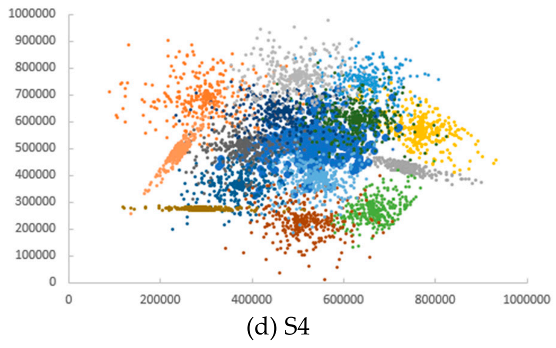
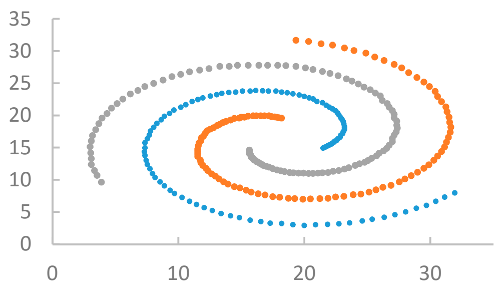
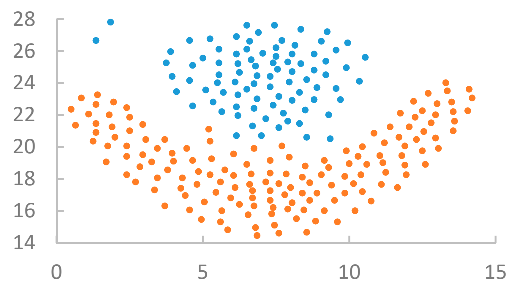


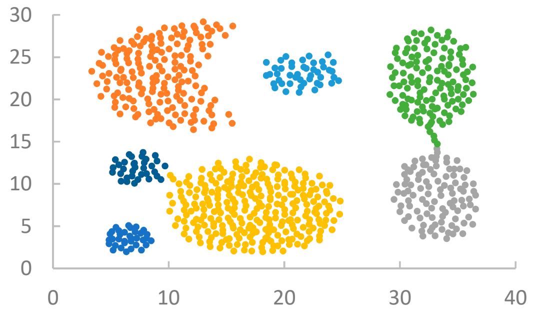
Figure A6. Data distribution of the 12 datasets (part 1).



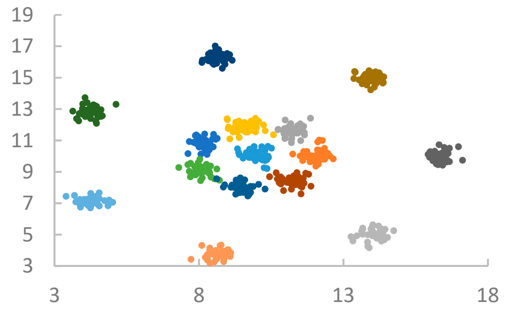
(a) Spiral



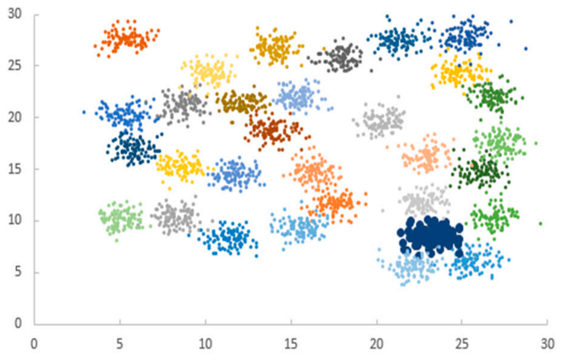
(b) Flame



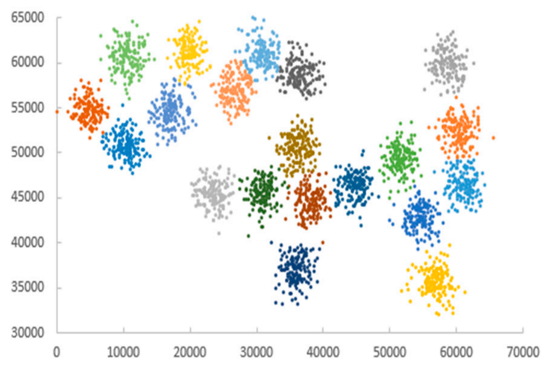
(c) Aggregation



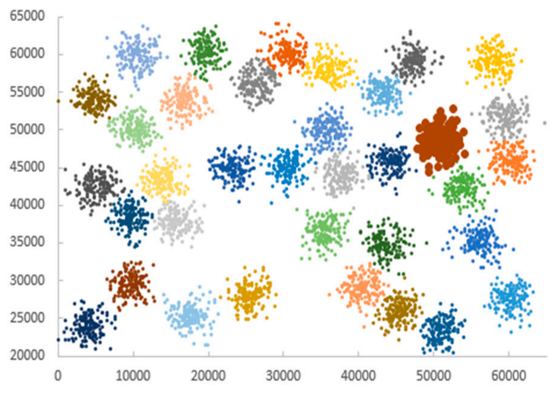
(d) R15



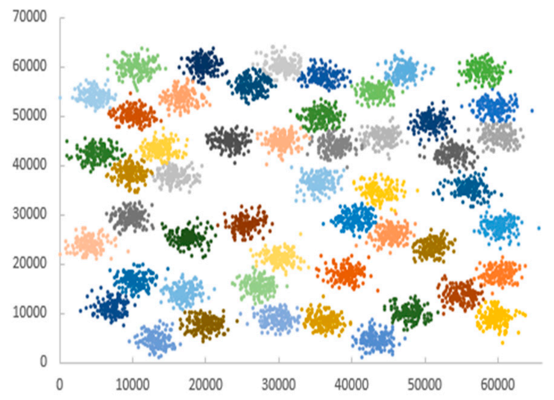
(e) D31



(f) A1



(g) A2



(h) A3

Figure A7. Data distribution of the 12 datasets (part 2).

Table A8. Number of points and number of clusters in the 12 datasets.

Dataset	Number of clusters	Number of points
Spiral	3	312
Flame	2	240
Aggregation	7	788
R15	15	600
D31	31	3100
A1	20	3000
A2	35	5250
A3	50	7500
S1	15	5000
S2	15	5000
S3	15	5000
S4	15	5000

Appendix E. More Experimental Results

Tables A9 and A10 show the number $\breve{N}$ of the data points with a local density < the maximal local density of their neighbors in Tests 1 and 2, respectively. Tables A11 and A12 show the execution time of the three algorithms (DPC, ADPC1 and ADPC2) in Tests 1 and 2, respectively.

Table A9. The value of $\breve{N}$ for various dataset and p combination (Test 1 uses a fixed threshold)

Dataset (N)	$p = 0.5$	$p = 1$	$p = 1.5$	$p = 2$	$p = 2.5$	$p = 3$	$p = 3.5$	$p = 4$
Spiral ($N = 312$)	62	90	110	139	161	187	208	232
Flame ($N = 240$)	75	143	177	208	213	221	226	231
Aggregation ($N = 788$)	609	704	744	760	763	761	761	768
R15 ($N = 600$)	348	490	537	558	563	570	564	567
D31 ($N = 3100$)	2952	3028	3034	3037	3013	2726	2034	2482
A1 ($N = 3000$)	2854	2955	2967	2968	2971	2973	2964	2962
A2 ($N = 5250$)	5140	5201	5192	5185	5202	5214	5236	5237
A3 ($N = 7500$)	7398	7418	7413	7456	7477	7483	7486	7490
S1 ($N = 5000$)	4305	4806	4912	4936	4945	4956	4974	4973
S2 ($N = 5000$)	4351	4834	4934	4953	4950	4954	4968	4961
S3 ($N = 5000$)	4474	4882	4942	4959	4977	4970	4977	4982
S4 ($N = 5000$)	4355	4818	4930	4955	4964	4968	4972	4976

Table A10. The value of $\breve{N}$ for various dataset and p value combination (Test 2 uses an exponential kernel)

Dataset (N)	$p = 0.5$	$p = 1$	$p = 1.5$	$p = 2$	$p = 2.5$	$p = 3$	$p = 3.5$	$p = 4$
Spiral ($N = 312$)	148	251	286	301	308	309	309	309
Flame ($N = 240$)	118	188	219	231	232	234	234	235
Aggregation ($N = 788$)	725	769	775	778	779	780	781	781
R15 ($N = 600$)	439	527	556	572	576	581	584	584
D31 ($N = 3100$)	2983	3044	3061	3067	3069	3070	3075	3085
A1 ($N = 3000$)	2914	2967	2977	2978	2980	2980	2981	2983
A2 ($N = 5250$)	5175	5212	5214	5216	5227	5236	5239	5243
A3 ($N = 7500$)	7424	7447	7453	7475	7483	7491	7493	7493
S1 ($N = 5000$)	4592	4880	4944	4963	4978	4983	4983	4984
S2 ($N = 5000$)	4605	4910	4968	4980	4982	4984	4985	4985
S3 ($N = 5000$)	4727	4932	4966	4979	4983	4985	4987	4988
S4 ($N = 5000$)	4652	4910	4953	4971	4974	4980	4985	4987

Table A11. The execution time in seconds (Test 1 uses a fixed threshold).

Dataset	Algorithm	*p* = 0.5	*p* = 1	*p* = 1.5	*p* = 2	*p* = 2.5	*p* = 3	*p* = 3.5	*p* = 4
Spiral	DPC	0.038101	0.046879	0.079211	0.052139	0.04688	0.046879	0.046879	0.046879
	ADPC1	0.035094	0.041108	0.036096	0.032086	0.046879	0.031252	0.031252	0.046879
	ADPC2	0.04512	0.033088	0.04512	0.046878	0.031251	0.046845	0.04686	0.015646
Flame	DPC	0.031253	0.031255	0.031273	0.031252	0.031253	0.031252	0.031252	0.031253
	ADPC1	0.015626	0.031253	0.015626	0	0.015599	0.015628	0	0.015628
	ADPC2	0.031255	0.015625	0.015605	0.015627	0.015627	0	0.0156	0.015606
Aggregation	DPC	0.252673	0.265685	0.281293	0.296938	0.296938	0.296938	0.312564	0.281312
	ADPC1	0.081215	0.078132	0.062506	0.062507	0.062506	0.078133	0.062506	0.078133
	ADPC2	0.0781	0.062507	0.04688	0.04688	0.046901	0.062507	0.078133	0.062505
R15	DPC	0.156265	0.171892	0.171892	0.171892	0.171892	0.187519	0.171891	0.171891
	ADPC1	0.125012	0.078131	0.078131	0.04688	0.046879	0.04688	0.046873	0.046878
	ADPC2	0.093759	0.07813	0.046879	0.046879	0.046878	0.031253	0.031253	0.031253
D31	DPC	4.281704	4.344213	4.375495	4.375465	4.375464	4.234823	3.187837	3.625386
	ADPC1	0.883543	0.7657	0.742746	0.781333	0.812588	0.937631	1.687681	1.672092
	ADPC2	0.875124	0.71418	0.687573	0.7032	0.734489	0.828213	1.640798	1.578292
A1	DPC	4.016051	4.078557	4.125437	4.141065	4.145534	4.160485	4.234813	4.187945
	ADPC1	0.81259	0.687572	0.687606	0.734455	0.781332	0.79696	0.828212	0.89072
	ADPC2	0.796961	0.656319	0.640726	0.67195	0.687573	0.687573	0.718826	0.762816
A2	DPC	12.48567	12.4857	12.59509	12.59508	12.59509	12.68885	12.76698	12.7982
	ADPC1	2.078376	1.984586	2.093839	2.203357	2.328372	2.484632	2.564224	2.672159
	ADPC2	2.000502	1.922079	2.250205	2.000213	2.093969	2.187732	2.234609	2.297118
A3	DPC	25.42454	25.53396	25.69023	25.72854	26.01839	26.11215	26.40587	26.33089
	ADPC1	3.926412	4.189912	4.47909	4.547357	4.769292	5.194029	5.486006	5.506579
	ADPC2	3.83774	3.802774	3.953544	4.1411	4.281703	4.420601	4.561412	4.742328
S1	DPC	11.42309	11.81372	11.65747	11.71463	11.75125	11.79816	11.79813	11.82938
	ADPC1	3.922292	2.422134	2.140852	2.125225	2.140851	2.234612	2.344002	2.437762
	ADPC2	3.8584	2.328372	1.984588	1.93774	1.922111	1.953329	2.035082	2.125227
S2	DPC	11.21994	11.45434	11.52083	11.5481	11.68874	11.64186	11.65748	11.67311
	ADPC1	3.687891	2.312742	2.031496	2.062718	2.15648	2.218986	2.344	2.469012
	ADPC2	3.641011	2.203357	1.906452	1.890826	1.922079	1.984584	2.047092	2.105681
S3	DPC	11.29808	11.48559	11.56373	11.56373	11.62623	11.59498	11.65748	11.67311
	ADPC1	3.21909	2.14085	2.031468	2.031467	2.109567	2.234612	2.343999	2.469044
	ADPC2	3.187838	2.015836	1.890857	1.86247	1.906452	1.97208	2.031466	2.109599
S4	DPC	11.25118	11.51685	11.57936	11.62623	11.56371	11.64183	11.71999	11.70437
	ADPC1	3.515999	2.344002	2.078346	2.093971	2.203359	2.265865	2.359625	2.437758
	ADPC2	3.484713	2.250242	1.937706	1.906451	1.968992	1.984586	2.062751	2.099317

Table A12. The execution time in seconds (Test 2 uses an exponential kernel).

Dataset	Algorithm	*p* = 0.5	*p* = 1	*p* = 1.5	*p* = 2	*p* = 2.5	*p* = 3	*p* = 3.5	*p* = 4
Spiral	DPC	0.217607	0.227606	0.24064	0.230642	0.218788	0.234399	0.218773	0.218806
	ADPC1	0.243648	0.210588	0.199559	0.177449	0.17189	0.218742	0.171892	0.187521
	ADPC2	0.201514	0.179477	0.177471	0.215608	0.171891	0.18752	0.171869	0.171893
Flame	DPC	0.12501	0.125012	0.125014	0.14064	0.125013	0.14064	0.125014	0.14064
	ADPC1	0.109386	0.109386	0.109386	0.125014	0.093759	0.109406	0.109387	0.109387
	ADPC2	0.12501	0.109385	0.125013	0.109386	0.10938	0.109386	0.109384	0.109385
Aggregation	DPC	1.299488	1.250166	1.250131	1.281362	1.281407	1.250169	1.250164	1.250162
	ADPC1	1.078868	1.062644	1.062612	1.062614	1.062644	1.062645	1.078269	1.078274
	ADPC2	1.11083	1.062613	1.046983	1.047022	1.047018	1.047019	1.047018	1.047018
R15	DPC	0.779632	0.750111	0.750057	0.750113	0.750075	0.765738	0.734483	0.73445
	ADPC1	0.703232	0.671957	0.656351	0.640727	0.656351	0.640728	0.625097	0.625066
	ADPC2	0.703232	0.671978	0.656351	0.625098	0.640724	0.640691	0.625097	0.60947
D31	DPC	20.34588	19.8146	19.90442	19.53329	19.65834	19.5177	19.54898	19.64271
	ADPC1	17.12682	16.74525	16.61117	16.39237	16.56423	16.39233	16.4581	16.59551
	ADPC2	17.11119	16.64242	16.36111	16.25173	16.28676	16.35301	16.53297	16.37674
A1	DPC	18.58013	18.08004	17.95503	17.98625	17.8769	17.8988	17.95503	17.8769
	ADPC1	15.57978	15.26725	15.18911	15.15783	15.12657	15.15679	15.16565	15.19659
	ADPC2	15.42351	15.15789	15.04847	15.01722	14.90703	15.00159	14.92343	14.87658
A2	DPC	56.31846	55.53715	55.05268	55.67778	55.25958	55.44339	55.5684	55.64653
	ADPC1	47.64571	46.62474	46.12989	46.31741	47.2863	46.45805	46.27054	46.87997
	ADPC2	47.3644	46.12753	45.97363	46.37992	46.12989	45.84858	45.91012	46.16624
A3	DPC	115.6921	112.1025	112.9651	115.0125	112.3618	112.9651	112.1213	112.8714
	ADPC1	95.73069	93.50614	95.91672	95.73115	95.51016	94.68273	95.79348	96.21016
	ADPC2	96.21336	94.86226	94.79854	93.96594	94.73864	93.69754	94.52285	94.63775
S1	DPC	53.11501	51.63048	50.92332	50.58346	50.44285	50.78661	50.19283	51.5836
	ADPC1	45.36419	43.80152	42.70859	42.8327	42.33265	42.0513	42.2279	42.50451

	ADPC2	45.77048	43.614	42.9889	41.89504	42.33735	41.86379	41.8169	41.92629
S2	DPC	51.88047	50.97416	50.89606	50.00531	50.36472	50.08341	50.73976	50.56783
	ADPC1	44.75324	42.71937	42.00446	42.00442	42.22323	44.39533	42.66078	42.61389
	ADPC2	44.58285	42.48891	41.85598	41.86382	41.86418	41.7544	41.9107	41.75443
S3	DPC	51.89613	50.14595	50.17026	49.89592	50.03655	50.63037	50.13029	50.03656
	ADPC1	43.45777	41.86382	41.75443	41.78568	41.64504	41.86382	42.22323	41.88364
	ADPC2	44.23907	41.84819	42.20761	41.77006	42.27011	41.65029	41.87944	41.89504
S4	DPC	51.45296	50.1772	50.42723	50.5679	50.36469	50.05215	50.00527	50.23967
	ADPC1	44.00467	42.13493	41.84819	42.52013	42.05134	42.00442	42.274	42.28573
	ADPC2	44.47347	41.88291	42.08256	42.03568	42.17635	41.66067	41.86382	41.84976

References

1. Aggarwal, C.C.; Reddy, C.K. *Data clustering: Algorithms and applications*. Chapman and Hall/CRC: Boca Raton, FL, USA, 2014.
2. Pham, G.; Lee, S.-H.; Kwon, O.-H.; Kwon, K.-R. A watermarking method for 3d printing based on menger curvature and k-mean clustering. *Symmetry* **2018**, *10*, 97.
3. MacQueen, J. Some methods for classification and analysis of multivariate observations, In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, Berkeley, Calif., 1967; University of California Press: Berkeley, Calif., pp 281–297.
4. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, **1996**, *96*, 226–231.
5. Han, J.; Kamber, M.; Pei, J. *Data mining: Concepts and techniques*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2011; p. 696.
6. Rodriguez, A.; Laio, A. Clustering by fast search and find of density peaks. *Science* **2014**, *344*, 1492.
7. Mehmood, R.; Zhang, G.; Bie, R.; Dawood, H.; Ahmad, H. Clustering by fast search and find of density peaks via heat diffusion. *Neurocomput.* **2016**, *208*, 210–217.
8. Wang, S.; Wang, D.; Li, C.; Li, Y.; Ding, G. Clustering by fast search and find of density peaks with data field. *Chinese J. Electron.* **2016**, *25*, 397–402.
9. Bai, L.; Cheng, X.; Liang, J.; Shen, H.; Guo, Y. Fast density clustering strategies based on the k-means algorithm. *Pattern Recognit.* **2017**, *71*, 375–386.
10. Mehmood, R.; El-Ashram, S.; Bie, R.; Dawood, H.; Kos, A. Clustering by fast search and merge of local density peaks for gene expression microarray data. *Sci. Reports* **2017**, *7*, 45602.
11. Liu, S.; Zhou, B.; Huang, D.; Shen, L. Clustering mixed data by fast search and find of density peaks. *Math. Problems Eng.* **2017**, *2017*, 7.
12. Li, Z.; Tang, Y. Comparative density peaks clustering. *Expert Syst. Appl.* **2018**, *95*, 236–247.
13. Du, M.; Ding, S.; Jia, H. Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowledge-Based Syst.* **2016**, *99*, 135–145.
14. Yaohui, L.; Zhengming, M.; Fang, Y. Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy. *Knowledge-Based Syst.* **2017**, *133*, 208–220.
15. Ding, S.; Du, M.; Sun, T.; Xu, X.; Xue, Y. An entropy-based density peaks clustering algorithm for mixed type data employing fuzzy neighborhood. *Knowledge-Based Syst.* **2017**, *133*, 294–313.
16. Yang, X.-H.; Zhu, Q.-P.; Huang, Y.-J.; Xiao, J.; Wang, L.; Tong, F.-C. Parameter-free laplacian centrality peaks clustering. *Pattern Recognit. Letters* **2017**, *100*, 167–173.
17. Cheng, S.; Duan, Y.; Fan, X.; Zhang, D.; Cheng, H. Review of Fast Density-Peaks Clustering and Its Application to Pediatric White Matter Tracts. *Annual Conference on Medical Image Understanding and Analysis*. Springer International Publishing: Cham, Switzerland, 2017; pp 436–447.
18. Han, J.; Kamber, M.; Pei, J. 10-cluster analysis: Basic concepts and methods. In *Data mining,* 3rd ed.; Han, J.; Kamber, M.; Pei, J., Eds. Morgan Kaufmann: Boston, MA, USA, 2012; pp 443–495.
19. Xenaki, S.D.; Koutroumbas, K.D.; Rontogiannis, A.A. A novel adaptive possibilistic clustering algorithm. *IEEE Trans. Fuzzy Syst.* **2016**, *24*, 791–810.
20. Bianchi, G.; Bruni, R.; Reale, A.; Sforzi, F. A min-cut approach to functional regionalization, with a case study of the italian local labour market areas. *Optim. Letters* **2016**, *10*, 955–973.
21. Deng, Z.; Choi, K.-S.; Jiang, Y.; Wang, J.; Wang, S. A survey on soft subspace clustering. *Inf. Sci.* **2016**, *348*, 84–106.

22. Laio, A. Available online: http://people.sissa.it/~laio/Research/Clustering_source_code/cluster_dp.tgz (accessed on 27 May 2019).
23. Chang, H.; Yeung, D.-Y. Robust path-based spectral clustering. *Pattern Recognit.* **2008**, *41*, 191–203.
24. Fu, L.; Medico, E. Flame, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioinf.* **2007**, *8*, 3.
25. Gionis, A.; Mannila, H.; Tsaparas, P. Clustering aggregation. *ACM Trans. Knowl. Discov. Data* **2007**, *1*, 4.
26. Veenman, C.J.; Reinders, M.J.T.; Backer, E. A maximum variance cluster algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 1273–1280.
27. Kärkkäinen, I.; Fränti, P. *Dynamic local search algorithm for the clustering problem*. University of Joensuu: Joensuu: Kuopio, Finland, 2002.
28. Fränti, P.; Virmajoki, O. Iterative shrinking method for clustering problems. *Pattern Recognit.* **2006**, *39*, 761–775.
29. Lin, J.; Peng, H.; Xie, J.; Zheng, Q. Novel clustering algorithm based on central symmetry. In Proceedings of the Internation Conference on Machine Learning and Cybernetics, Shanghai, China, 26–29 August 2004; pp. 1329–1334.
30. Bandyopadhyay, S.; Saha, S. A Point Symmetry-Based Clustering Technique for Automatic Evolution of Clusters. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 1441–1457.