FastDEC: Clustering By Fast Dominance Estimation

Geping Yang*[1], Hongzhang Lv*[1], Yiyang Yang**[1], Zhiguo Gong**[2], Xiang Chen[3], and Zhifeng Hao[4]

[1] Guangdong University of Technology, Faculty of Computer, China
[2] University of Macau, State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science, Macau, China
[3] Sun Yat-Sen University, School of Electronics and Information Technology, China
[4] Shantou University, College of Engineering, China

Abstract. k-Nearest Neighbors (k-NN) graph is essential for the various graph mining tasks. In this work, we study the density-based clustering on the k-NN graph and propose FastDEC, a clustering framework by fast dominance estimation. The nearest density higher (NDH) relation and dominance-component (DC), more specifically their integration with the k-NN graph, are formally defined and theoretically analyzed. FastDEC includes two extensions to satisfy different clustering scenarios: FastDEC$_D$ for partitioning data into clusters with arbitrary shapes, and FastDEC$_K$ for K-Way partition. Firstly, a set of DCs is detected as the results of FastDEC$_D$ by segmenting the given k-NN graph. Then, the K-Way partition is generated by selecting the top-K DCs in terms of the inter-dominance (ID) as the seeds, and assigning the remaining DCs to their nearest dominators.
FastDEC can be viewed as a much faster, more robust, and k-NN based variant of the classical density-based clustering algorithm: Density Peak Clustering (DPC). DPC estimates the significance of data points from the density and geometric distance factors, while FastDEC innovatively uses the global rank of the dominator as an additional factor in the significance estimation. FastDEC naturally holds several critical characteristics: (1) excellent clustering performance; (2) easy to interpret and implement; (3) efficiency and robustness. Experiments on both the artificial and real datasets demonstrate that FastDEC outperforms the state-of-the-art density methods including DPC.

Keywords: Density Estimation · Clustering · k Nearest Neighbors.

1 Introduction

Density-based clustering is widely used in various fields such as computer vision and outlier detection. As a fast pre-processing technology, it is used to partition

* Equal Contribution
** Corresponding Author: yyygou@gmail.com,fstzgg@um.edu.mo

Table 1. Characteristics of existing density-based methods. CC: Connected-Component detection. OS: over-segmentation. K-Way: supports K-Way Partition. Para: primary parameters. (DR: density reachable. DH: density-higher. NDH: nearest density-higher.)

Method	Connectivity	CC	Avoid OS	K-Way	Accuracy	Efficiency	Para	Robustness
DBSCAN	DR	×	×	×	Low	High	τ, min_{pt}	Low
Meanshift	DH	×	×	×	Media	Low	τ	Media
Quickshift	NDH	×	×	×	Media	Media	τ	Media
DPC	NDH	×	×	✓	High	Low	τ	Media
Quickshift++	mutual k-NN	✓	✓	×	High	Low	k, β	Media
FINCH	1^{st}-NN	✓	×	✓	Media	High	$k = 1$	High
QuickDSC	mutual k-NN	✓	✓	✓	High	High	k, β	Media
FastDEC (Ours)	k-NN dominance	✓	✓	✓	High	High	k	High

the data into clusters with arbitrary shapes. Several classic and effective methods have been designed for such a task. This work aims to design an efficient, robust, and effective density-based clustering method. In Table 1, we list several perspectives that are significant to the density-based clustering, based on which, analyze the relevant methods including the classical and state-of-the-art.

Density-based clustering algorithms conduct the connectivity between data points. Meanshift [8], a mode-seeking procedure, iteratively shifts data points to the location with local maximum density, while the density is estimated by Gaussian Kernel. Such a mechanism makes use of a density-higher (DH) connectivity as the cluster structure. DBSCAN [15] adopts a Flat Kernel [19] to identify the density-reachable (DR) connectivity. It significantly reduces computational cost but ignores the mode information. Rather than shifting to a location, Quickshift [29] only moves to the NDH data point if one exists within the τ-radius ball. Both methods hill-climb to the local modes of density and cluster upon these modes. Such processes asymptotically detect all modes individually, leading to **over-segmentation (OS)** [20]. That is the nearby modes are clustered as distinctive clusters, although it is better to group them as a single cluster.

Quickshift++ [20] uses the mutual k-Nearest Neighbors (mutual k-NN) graph to avoid the OS. Two points are merged into a connected-component (CC) if a mutual k-NN edge is detected. Based on the obtained CCs, QuickDSC [36] introduces a density-higher (DH) based way to generate a K-Way partition. However, both methods still require an additional parameter β to balance the segmentation. On the other hand, Density Peak Clustering (DPC) [26] prevents the OS by a new factor: geometric distance, that is the distance from a point to its nearest density-higher (NDH) point. The higher this value, the less probability the point to be a mode. Besides, DPC is extremely slow since it requires finding every possible NDH connection. Cluster by First Integer Neighbor Hierarchy (FINCH) [27] applies the first nearest neighbor (1^{st}-NN) as the connectivity to form the CCs, based on which conduct the K-Way partition. Although FINCH is a parameter-free algorithm, it disregards the description of local modes, therefore may suffer from the OS as well.

To sum up (as Table 1), these perspectives are critical to our target. Connectivity is used to model the relationships between data points. Similar to the methods such as DBSCAN, and Meanshift, CCs can be returned directly as the final result, which leads to clusters with arbitrary shapes, but meanwhile makes the algorithm difficult to control, e.g., if K-Way partition is explicitly required. On the other hand, CC is relevant to the algorithm efficiency because the detected CCs as the densely connected points can be used to further form the clusters. From this point of view, DPC completely ignores such a density structure. Avoiding OS is key to the density-based methods as well. Finally, besides the efficiency and effectiveness, the algorithm robustness is discussed as well for some parameters are difficult to determine in practice [20, 34]. For example, Quickshift++ and QuickDSC are outstanding for their performances, however, they need to tune two parameters, k and β, for various datasets, which diminishes the robustness of algorithms.

In this work, FastDEC, clustering by fast dominance estimation, is proposed to fulfill the requirements above. The advantages of FastDEC are highlighted as follows:

1. Efficient, effective, and robust. It can be viewed as a k-NN graph based variant of DPC. According to our analysis, all DPC needs is a k-NN graph. Thus, FastDEC requires only one parameter, that is the number of nearest neighbors k.
2. Easy to interpret. To further support K-Way partition, dominance-component (DC) and inter-dominance estimation are formally defined and theoretically analyzed. In the inter-dominance estimation, the global rank of the NDH dominator is innovatively involved, which further enhances the performance of the algorithm;
3. Experiments on distinctive datasets demonstrate that FastDEC[1] outperforms the state-of-the-art including Quickshift++ [20], FINCH [27], and QuickDSC [36].

2 Related Work

Clustering is a critical analysis tool in data mining and machine learning fields. Density-based clustering methods such as DBSCAN [15], MeanShift [8], QuickShift [29], and so on [19], effectively identify the clusters with arbitrary shapes. These methods use a τ-radius ball to estimate the density, which is difficult to choose in practice. The k-NN DE based methods [2, 20, 36] are proposed recently to compute the density efficiently.

However, the mode-seeking clustering suffers from over-segmentation [20]. The mutual k-NN graph [6, 7, 21] is proposed to overcome this problem. The methods above cannot provide the K-Way partitions. BIRCH [35], Agglomerative clustering [11], and FINCH [27] are proposed to satisfy this requirement.

[1] FastDEC is released on `https://github.com/gepingyang/FastDEC`.

However, they do not utilize the density structure. The K-Mode [4] and the LK-Mode [31] are able to provide a K-Way partition, but neither of them considers the connected-component (CC).

DPC [26], an effective clustering algorithm, is proposed to address this limitation. The idea is simple: the importance of each point is evaluated from two aspects: (1) density; (2) geometric distance: distance to the NDH point. Based on two aspects jointly, the most significant K points as selected as the seeds and based on which clusters the remaining points. DPC suffers from two issues: (1) parameter τ is hard to determine; (2) computational expensive. Based on DPC, several works [3, 13, 23, 32] are designed to improve its effectiveness. Meanwhile, several methods are proposed to accelerate the DPC by space technology [25] or parallelization approach [1].

3 Preliminaries

Let $X = \{x_1, x_2, ..., x_n\}$ be the set of data points, where $x_i \in \mathbb{R}^f$ is an f-dimensional feature vector, and n is the number of points. The Euclidean distance is considered exclusively for clustering. A k Nearest Neighbors (k-NN) graph $G = (V, E)$ is conducted on X, where V denotes the vertex set, and E denotes the edge set. Distinguished from the K (capital letter), which corresponds to top-K significant points or K-Way partition, k-NN (lowercase letter) denotes the k Nearest Neighbors. For each $v \in V$, its k-NN is provided as $\mathcal{N}^k(v)$, where $\mathcal{N}^k(v)_{[j]}$ denotes the j^{th}-NN of v.

Let $d(v)$ denote the density of vertex v, it is referred to as the degree in spectral methods [28]. Given a k-NN graph G, $d(v)$ can be computed through a density estimator. Gaussian Kernel Density Estimator (GKDE) [9] is the most robust but requires a predefined sphere (by τ). DBSCAN uses a Flat Kernel Density Estimator (FKDE), and Quickshift++ adopts a k^{th}-NN Density Estimator (k^{th}-NN DE).

In FastDEC, given a vertex v, its density is computed by a typical k-NN Gaussian kernel density :

$$d(v) = \sum_{u \in \mathcal{N}^k(v)} \exp(-\frac{||v-u||^2}{2\sigma^2}) \tag{1}$$

where $||v-u||$ denotes the Euclidean distance between vertices v and u, and σ is the global bandwidth parameter. We straightly set σ as the mean value of all k-NN distances. The definitions of k-NN Flat kernel density estimator and k^{th} kernel density estimator can be found in Supplement Material.

Based on a density estimator, the density-higher relation is defined as follows:

Definition 1. *(density-higher) A vertex u is **density-higher (DH)** than v if $d(u) > d(v)$.*

Definition 2. *(nearest density-higher) A vertex u is the **nearest density-higher (NDH)** vertex of v if (1) u is DH than v and (2) there is no vertex w such that w is DH than v, and $||v-w|| < ||v-u||$.*

Both DH and NDH relations are asymmetric.

A vertex with the highest density in a local region is referred to as the mode [8]. However, depending on density only is insufficient to identify the modes correctly because of the over-segmentation (OS), that is the nearby vertices with local maximum densities are detected as distinctive modes [20]. From the connectivity perspective, they should be grouped as one component for their dense mutual relations. Quickshift++ [20] resolves this problem by introducing an additional tolerance parameter β to adjust the "closeness" between modes. Density Peak Clustering (DPC) addresses the OS by a new factor. For a vertex $v \in V$, $g(v)$, the geometric distance from v to its NDH vertex is defined as:

$$g(v) = \begin{cases} \min\limits_{u \in V : d(u) > d(v)} ||v-u||, & \text{if } u \text{ exists} \\ \max\limits_{u \in V} g(u), & \text{otherwise} \end{cases} \tag{2}$$

v is a local mode if its NDH vertex u exists; otherwise, v is a global mode.

The weight of a vertex is estimated by two factors: $w(v) = d(v) \cdot g(v)$. The idea is intuitive: the density value (e.g., $d(v)$) indicates the intensity of density, while geometric distance $g(v)$ reveals the scope of density. Thus, $w(v)$ jointly reflects the weight of v from two factors. However, DPC searches for every possible NDH relation with extremely high time-complexity $O(n^2)$. QuickDSC [36] uses a mutual k-NN graph [20] to reduce the search cost, but requires tuning a new parameter β. Our experiments demonstrate that the good settings of β are diverse in different datasets.

4 Proposed Framework

In this work, we propose FastDEC, a clustering framework by fast dominance estimation. Its general procedure is described in Figure 1.

1. k-NN density estimation: build the space index on X for the k-NN graph retrieval. Based on the k-NN graph G, estimate the densities of vertices by Equation 1;
2. Direct k-NN dominator (DkD) detection: identify the direct k-NN dominator (DkD) of all vertices from the G; Based on the DkD, a set of dominance-components can be explicitly identified, where the dominance-component (DC) is a special form of Connected-Component;
3. DC significance estimation: the significance of DCs is estimated from three factors. Based on which, the top-K DCs are selected as the seeds;
4. DC-based clustering: the remaining DCs are assigned to their Nearest Density Higher (NDH) DCs.

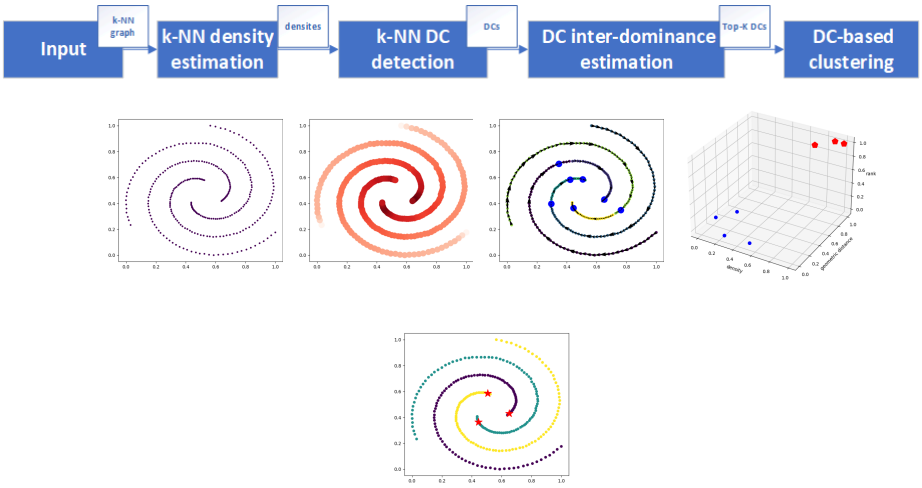

Fig. 1. FastDEC consists four primary stages: (1) k-NN density estimation; (2) Direct k-NN dominator (DkD) detection; (3) DC significance estimation; (4) DC-based clustering. Visualization is generated on Spiral. ($n = 312$, $k = 7$ and $K = 3$)

The first stage computes the vertex densities by accessing the k-NN graph G, which is essential for all density-based methods. **Note:** FastDEC can adopt an arbitrary density estimator. k-NN Gaussian Kernel DE (as Equation 1) is used as the default for several critical characteristics: (1) it is as robust as the τ-ball based DE (as Table 1); (2) k is easy to control and the only parameter (so as the whole framework FastDEC); (3) based on the sophisticated index technologies [10, 12], FastDEC can retrieve a large-scale k-NN graph efficiently. In the second stage, according to the given k, the DkDs of all vertices are captured as well. It is easy to show the equivalence between DkD and DC (to be introduced later). In the third stage, a significance estimation process is performed on the obtained DCs from three factors including density, geometric distance, and the global rank of the NDH vertex. The last stage follows the typical process of DPC [26] and QuickDSC [36]. The DkD detection and the DC significance estimation are the key components of FastDEC with k as the only control parameter.

4.1 Direct k-NN Dominator (DkD) Detection

Density alone is insufficient to correctly identify the modes/clusters because of the OS. To overcome it, Quickshift++ [20] is proposed to merge the nearby vertices as the connected-component (CC) by mutual k-NN edges. While Sarfraz et al. [27] utilize the symmetric 1^{st} Nearest Neighbor (1^{st}-NN) relation to segment the entire k-NN graph into CCs. The obtained CCs are used as the basic units for further clustering, therefore preventing the OS since the nearby vertices with high densities are grouped. Meanwhile, both methods enormously reduce the

graph size from $|V| = n$ to c, where c is the number of CCs, therefore accelerate the clustering process.

In FastDEC, a special CC, the dominance-component (DC) is defined by combining the NDH relation and k-NN graph. For vertex v, FastDEC attempts to find its NDH vertex from its k-NN only (i.e., $\mathcal{N}^k(v)$). Such a mechanism results in the direct k-NN dominance, which is defined as follows.

Definition 3. *(direct k-NN dominance) A vertex v is **directly k-NN dominated** by vertex u wrt. k if (1) $u \in \mathcal{N}^k(v)$, and (2) u is the NDH vertex of v.*

For better representation, we term v as the dominated vertex and u as the **direct k-NN dominator (DkD)** of v. Furthermore, the k-NN dominance is defined as follows:

Definition 4. *(k-NN dominance) A vertex v is **k-NN dominated** by u wrt. k if there exists a chain of vertices v_1,...,v_l, $v_1 = v$, $v_l = u$ such that v_i is directly k-NN dominated by v_{i+1}.*

k-NN dominance is a canonical extension of direct k-NN dominance. It is transitive as the DH, meanwhile, it is not symmetric. The notion of k-NN dominance-connected is introduced to guarantee symmetry.

Definition 5. *(k-NN dominance-connected) A vertex v is **k-NN dominance-connected** to a vertex u wrt. k if there exists a vertex w such that both, u and v are k-NN dominated by w.*

k-NN dominance-connectivity is a symmetric relation. Through direct k-NN dominance, a vertex iteratively reaches a vertex **without DkD**. The latter is more significant and representative, because of the higher density value. We define such kind of vertices as the k-NN mode:

Definition 6. *(k-NN mode) A vertex without k-NN dominator wrt. k, is a k-NN mode.*

k-NN mode is a vertex with the highest density in the local region (i.e., k-NN). The region is further expanded by the direct k-NN dominance. Intuitively, the vertices dominated by a k-NN mode have strong affiliations. Now, we define the k-NN dominance-component (DC).

Definition 7. *(k-NN dominance-component) Let V be the vertex set, and $m \in V$ be a k-NN mode. A k-NN dominance-component DC of m wrt. k is a non-empty subset of V satisfying the following conditions:*
(1) $\forall u, v$: if $u \in DC$ and v is k-NN dominance-connected to u wrt. k, then $v \in DC$. (Maximality & Connectivity)
(2) $\forall v$ are k-NN dominated by m wrt. k, then $v \in DC$. (Dominance)

Note that the k-NN DC has several critical characteristics:

Algorithm 1 Direct k-NN Dominator (DkD) Detection

Input: X, DB, k
Output: DkD
1: $G \leftarrow$ Conduct k-NN graph of X by querying DB; ▷ k-NN graph construction
2: $d \leftarrow$ Estimate k-NN densities via Equation 1; ▷ k-NN density estimation
3: Initialize DkD(v) as *None* for $v \in V$; ▷ DkD Initialization
4: **for** $v \in V$ **do**
5: **for** $j := 1, ..., k$ **do** ▷ search from near to far by varying j in $[1, k]$
6: **if** $d(v) < d(\mathcal{N}^k(v)_{[j]})$ **then** ▷ $\mathcal{N}^k(v)_{[j]}$ denotes the j^{th} nearest neighbor of v
7: DkD(v) := $\mathcal{N}^k(v)_{[j]}$; ▷ v is dominated by its j^{th}-NN
8: break;
9: **end if**
10: **end for**
11: **end for**
12: Return DkD

1. a DC has only a k-NN mode, that is the vertex with the highest density in the DC;
2. the density values along the path, that starts from a vertex to the k-NN mode, are guaranteed to be monotonically increasing;
3. the k-NN mode is an excellent representative of DC;
4. DCs are mutually disjoint;

The corresponding proofs are omitted because of the space limitation. **Note:** noise filtering [15] is compatible with FastDEC, although it is insignificant to the results of this work. It is disabled in all applicable methods [8, 19, 34] by setting $min_{pt} = 0$.

Our concern is to identify the DkD from X. The procedure is described as Algorithm 1, with the dataset X, the space index DB, and k as input. First, conduct the k-NN graph (Line 1), and then for each vertex $v \in V$, compute its density by an arbitrary DE (Line 2), and initialize its DkD as None (Line 3). Second, for each vertex v, we attempt to find its DkD (Lines 4-11), where $\mathcal{N}^k(v)_{[j]}$ denotes the j^{th}-NN of v. Finally, the identified DkD information is returned. In this stage, the vertex without DkD is referred to as a k-NN mode. Based on the DkD, we can extract the dominance-components (DCs) explicitly by performing a Hill-Climbing search on all vertices (as Algorithm 2).

4.2 DC Dominance Estimation

Fundamentally dominator-component (DC) is a set of vertices that are compactly linked through the nearest density-higher (NDH) relations. In Fig. 2 (a),

Algorithm 2 Hill-Climbing: convert the DkD to DCs $\mathcal{DC}^k$ and the corresponding modes M^k explicitly

Input: DkD
Output: $\mathcal{DC}^k, M^k$

$M^k \leftarrow \emptyset$;
$c := 0$; ▷ Count the number of DCs
for $v \in V$ **do**
  if DkD(v) is *None* **then**
    $M^k \leftarrow M^k \cup v$;
    $c := c + 1$;
    $\mathcal{DC}^k_c \leftarrow \{v\}$; ▷ Initialize $\mathcal{DC}^k$
  end if
end for
for $v \in V$ **do**
  for $m := 1, ..., c$ **do**
    if v is k-NN dominated by $M^k(m)$ **then**;
      $\mathcal{DC}^k_m \leftarrow \mathcal{DC}^k_m \cup v$;
    end if
  end for
end for
Return $\mathcal{DC}^k$, M^k

Algorithm 3 FastDEC$_D$: Typical Density-based Method

Input: $X \in \mathbb{R}^{n \times m}$, DB, k
Output: clusters C

DkD $\leftarrow$ Algorithm 1(X, DB, k); ▷ DkD Detection
$\mathcal{DC}^k, M^k \leftarrow$ Algorithm 2(DkD); ▷ Extract DCs from DkD explicitly
Return $C \leftarrow \mathcal{DC}^k$. ▷ Return DCs as the Clusters

the change rates of NDH relations of three k-NN relevant density estimators[2] are shown, where k varies from 5 to 15. For instance, assume we obtain n NDH relations if set $k = 10$. If we set $k = 11$, n' NDH relations are changed, then the change rate is reported as n'/n for $k = 11$. Results on change rate demonstrate that the k-NN Gaussian Kernel based NDH relation is more stable, so it is set as the default DE.

Besides, NDH is adopted by Quickshift [29], DPC [26], and QuickDSC [36]. The former two methods explicitly search all possible NDH relations and lead to a huge amount of search cost; the latter uses an additional parameter β together with a mutual k-NN graph to "avoid" the unnecessary search. Our proposed k-NN dominance-component is conducted on NDH relation as well but with the restriction of k-NN. The idea is intuitive: obtain the majority of

[2] Density-Reachable (DR) in DBSCAN [15] is equivalent to τ based Flat Kernel. For the sake of comparison, we use a k-NN based one.

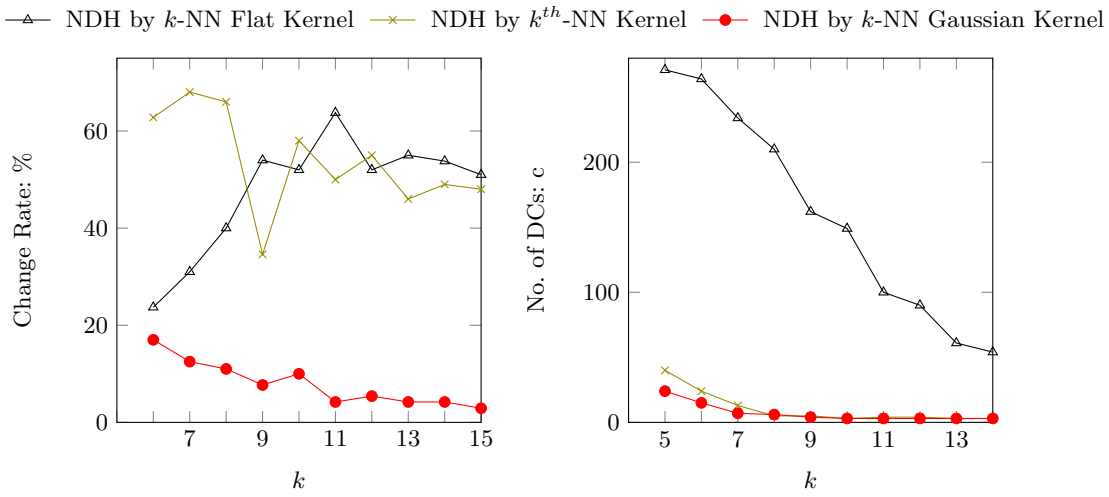

Fig. 2. The change rate of NDH relation and the No. of DCs (c) on Spiral by varying k.

such relations from the given k-NN graph, and perform the extra NDH search for the k-NN modes only. In Fig. 2 (b), the number of obtained DCs c is shown, with k varying from 5 to 15, where c is obtained by counting the number of k-NN modes. Again, NDH by k-NN Gaussian Kernel is the stablest. More interestingly and significantly, k can be used to control the number of DCs.

Based on the obtained components (e.g., CC/DC), the computational complexity is heavily reduced, and the algorithm is further enhanced with the capability of K-Way partition: select top-K significant DCs as the initial seeds/centers from the obtained c DCs. For CC-enabled methods, several strategies are proposed to find an ideal value of c. QuickDSC uses the parameter tuning in a brute-force manner, while FINCH applies the BIRTH [35] to identify the hierarchy structure of CCs, and according to which, generates K clusters. Regarding FastDEC, as k increases, the number of DCs $c \rightarrow c_{global}$, where c_{global} is the number of global modes. If k approaches 0, $c \rightarrow n$.

Algorithm 1 returns the DkD of all vertices. By revoking Algorithm 2, a set of DCs $\mathcal{DC}^k = \{DC_1, DC_2, ..., DC_c\}$ and the corresponding k-NN modes $M^k = \{m_1, m_2, ..., m_c\}$ are obtained for the given k. If $c \geq K$, the obtained DCs are sufficient for the subsequent process (e.g., DC dominance estimate). Otherwise, it is necessary to reduce the value of k to obtain at least K DCs. Fortunately, this step is extremely efficient by directly accessing the retained space index DB. Its detail is omitted for the space limitation.

Straightforwardly, similar to the typical density-based methods [8,15,19,20], the obtained DCs with arbitrary shapes can be reported as the clustering outcome. We term this method the FastDEC$_D$, its procedure is described as Algorithm 3. Except for the K-Way partition, FastDEC$_D$ satisfies most require-

Algorithm 4 FastDEC$_K$: K-Way Partition Method

Input: $X \in \mathbb{R}^{n \times m}$, k, K
Output: clusters C

DB $\leftarrow$ Build the space index on X;
▷ Stage 1:DkD Detection for X
DkD $\leftarrow$ Algorithm 1(X, DB, k);
▷ Stage 2: DC dominance estimation
$\mathcal{DC}^k, M^k \leftarrow$ Algorithm 2(DkD); ▷ Detect DCs and the corresponding modes explicitly
$DkD_M \leftarrow$ Algorithm 1(M^k, DB, n); ▷ Mode-based DkD Detection for M^k by setting $k = n$
▷ Estimate the k-NN significance of modes only.
for $m_i \in M^k$ **do**
    $s(m_i) \leftarrow$ estimate m_i by $DkD_M(m_i)$ via Equation 3;
end for
▷ Stage 3: DC-based clustering.
$C \leftarrow$ top-K DCs;
for $DC_i \in \mathcal{DC}$ and DC_i is not top-K DCs **do**
    Assign DC_i to the cluster $\hat{C} \in C$ that includes $DkD_M(m_i)$;
end for
Return C.

ments listed in Table 1. To further support the K-Way partition, FastDEC$_K$ is designed.

Similar to DPC and QuickDSC, the density and geometric distance are involved in estimation (as Equation 3). Additionally, an interesting and novel factor: the rank of the NDH neighbor is considered as well. For a vertex $v \in V$, its significance is defined as:

$$s(v) = d(v) \cdot g(v) \cdot r(v) \tag{3}$$

where $r(v)$ denotes the rank of its NDH neighbor in $\mathcal{N}^k(v)$. Clearly, $r(v)$ ranges from 1 to n. The larger $r(v)$, the more significant v is. This factor can be viewed as **global rank** information for weighting the dominance of v.

The mode in existing density-based methods [8,26] is τ-based. In QuickDSC [36], the geometric distance (e.g., $g(v)$) works as a special form of τ that is highly adaptive to the data point (e.g., v). For the significance estimation, the global rank information in terms of the $r(v)$ is involved. This factor indicates the relative position of v in the whole dataset X. As a result, our decision map evolves from 2-dimensional to 3-dimensional (as Figure 1).

On the other hand, it seems that the involvement of $r(v)$ heavily aggravates the search cost of estimation. Fortunately, the significance ($s(v)$) is consistent to the vertex weight ($w(v)$) adopted in DPC and QuickDSC.

Theorem 1. *Given a k-NN dominance component DC with its k-NN mode m, its significance value $s(m) > s(v)$ for all vertices $v \in DC$ if $w(m) > w(v)$.*

Proof. Rewrite $s(v) = w(v) \cdot r(v)$. It is easy to infer that $r(m) > r(v)$ definitely for all $v \in DC$, because $r(v) \leq k$ and $r(m) > k$. The theorem holds since $r(m) > r(v)$ and $w(m) > w(v)$ for all $v \in DC$.

Regarding the vertex weight $w(v)$, it is easy to infer that $d(m) > d(v)$ for all $v \in DC$ since m is the k-NN mode. For the geometric distance, $g(m) > g(v)$ in the majority of the cases [36]. Therefore, once the DCs and the corresponding modes are obtained, **we only need to estimate the significance of the modes**.

The general procedure is described as Algorithm 4. First, a space index DB is built on X (Line 1). Based on this, the DkD detection is executed for X (Line 2). Second, the DCs and the corresponding modes are explicitly found (Line 3). For each mode m_i, since its density is known, its geometric distance and the global rank are computed by querying the DB (Line 4). Meanwhile, the DkD information of the modes, denoted as DkD_M, is detected as well. Therefore, it is possible to **estimate the "n-NN significance" of modes** (Lines 5-7). The last stage of FastDEC follows the general process of DPC and QuickDSC: top-K significant modes are selected as the seeds (Line 8). The remaining DCs are assigned to the clusters according to the DkD_M (Lines 9-11).

4.3 Complexity Analysis and Implementation

FastDEC_K consists of an initial stage, and three major stages. FastDEC_D includes the initial stage, DkD detection, and a hill-climbing process. The initial stage requires building a space index (e.g., DB), and the DkD detection stage utilizes it to construct a k-NN graph. We use KD-Tree as the default space index, and Random Project Tree [10] + Nearest Neighbor Descend [12] are used to obtain the approximate k-NN by considering the scalability. Both ways require a complexity of $O(nlog(n)k)$. The DkD detection requires a complexity of $O(nk)$. DC significance estimation costs a complexity of $O(clog(n)k + cn)$ where c is the number of DCs. The last stage needs a complexity of cK. The most consuming step is the k-NN graph retrieval, the whole framework is still near-linear to n and k. FastDEC is implemented by Python, NumPy[3] is used for acceleration. We might further speed up our framework with parallel technologies.

5 Evaluation

To evaluate the performance of FastDEC, experiments are conducted on several artificial and real-world datasets of different sizes.

Datasets. The artificial and real-world datasets are selected from various sources. Details of the employed datasets are described in Table 2, and Principal Component Analysis (PCA), as a typical preprocessing of step dimension reduction, is applied on MNIST to reduce the number of feature dimensions from 784 to 20.

[3] https://numpy.org/

Table 2. Datasets In Evaluation.

Name	# instances	# features	# classes	type
banana-ball [24]	2,000	2	3	Artificial
Flame [17]	240	2	2	Artificial
R15 [30]	600	2	15	Artificial
Spiral [5]	312	2	3	Artificial
S2 [16]	5,000	2	15	Artificial
seeds [14]	210	7	3	Real-World
Banknote [14]	1,372	4	2	Real-World
Segmentation [14]	2, 310	19	7	Real-World
Phonemes [18]	4,509	258	5	Real-World
MFCCs [14]	7291	22	10	Real-World
MNIST [22]	70,000	20	10	Real-World

Table 3. ARI Comparison (%) on Artificial and Real-world datasets. The best and second best results are highlighted by **bold** and underline respectively.

	Typical Density-based Methods					K-Way Partition Methods				
Datasets	DBSCAN	Meanshift	Quickshift	Quickshift++	FastDEC$_D$	FINCH	DPC	QuickDSC	SNN-DPC	FastDEC
banana-ball	47.70	62.70	90.57	99.67	**99.83**	74.96	69.66	99.50	**99.83**	**99.83**
Flame	19.93	7.41	98.81	**100.00**	**100.00**	0.00	72.78	**100.00**	95.02	**100.00**
R15	26.37	98.21	92.78	**99.28**	**99.28**	**99.28**	73.94	**99.28**	**99.28**	**99.28**
Spiral	1.00	10.22	**100.00**	58.77	**100.00**	0.01	98.56	58.76	**100.00**	**100.00**
S2	0.00	**93.83**	92.64	92.80	93.67	88.01	48.15	93.47	92.89	**93.67**
seeds	38.39	**72.37**	61.66	70.76	70.76	39.10	68.23	77.27	**78.36**	76.64
Banknote	82.60	17.81	28.33	95.39	**95.67**	10.86	89.25	95.39	83.53	**96.53**
Segmentation	40.38	51.27	47.17	49.23	**59.66**	42.99	57.80	60.59	60.26	**60.94**
Phonemes	42.36	40.82	71.10	74.95	**76.51**	65.59	79.59	74.60	73.37	**76.51**
MFCCs	13.65	35.74	17.17	43.69	**47.02**	24.35	15.77	23.99	27.04	**31.74**
MNIST	6.74	21.98	N/A	46.41	**46.84**	N/A	N/A	35.96	N/A	**45.05**

Baselines. FastDEC$_D$ is compared with typical density-based methods: DBSCAN [15], Meanshift [8], Quickshift [29] and Quickshift++ [20]. FastDEC$_K$ is compared with the methods that support K-Way partition: FINCH [27], DPC [26], SNN-DPC [23], and QuickDSC [33]. More specifically, some recent DPC-based methods [1, 3, 13, 25, 32] are not involved, for the unavailability of their implementations.

Evaluation Metric. We use the Adjusted Rand Index (ARI), Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI) to evaluate the clustering results. For all metrics, the higher value indicates better performance. AMI-based results can be found in Supplement Material.

Experiment Setup. All experiments are executed on a Win-10 64-bits machine with Intel(R) Core $i5-9400F$ CPU($2.90GHz$), and 16 GB of main memory. For the k-NN based methods: Quickshift++, QuickDSC, FastDEC$_D$, SNN-DPC and FastDEC$_K$, k is varied from 5 to 200 with step size 5 in all datasets. Regarding τ-ball based methods: DBSCAN, Meanshift, Quickshift, and DPC, τ

Table 4. NMI (%) on Artificial and Real-world datasets. The best and second best results are highlighted by **bold** and underline respectively.

	Typical Density-based Methods					K-Way Partition Methods				
Datasets	DBSCAN	Meanshift	Qucikshift	Quickshift++	FastDEC$_D$	FINCH	DPC	QuickDSC	SNN-DPC	FastDE
banana-ball	67.98	61.72	88.53	99.30	**99.65**	80.17	68.73	99.10	**99.65**	**99.65**
Flame	37.45	32.88	97.06	**100.00**	**100.00**	15.25	79.76	**100.00**	89.94	**100.0**
R15	74.25	98.65	97.97	**99.42**	**99.42**	**99.42**	92.02	**99.42**	**99.42**	**99.42**
Spiral	**100.00**	45.44	**100.00**	59.57	**100.00**	40.10	98.05	59.57	**100.00**	**100.0**
S2	0.00	**94.62**	93.85	94.08	94.52	93.20	77.18	94.38	94.09	**94.52**
seeds	46.17	**69.00**	64.33	67.97	67.97	50.79	65.03	73.23	**74.10**	73.29
Banknote	73.58	26.63	44.63	91.94	**92.35**	13.19	83.00	91.94	78.49	**93.59**
Segmentation	60.16	62.37	65.44	71.13	**73.03**	59.71	70.78	72.45	68.34	**73.82**
Phonemes	60.28	60.75	73.12	84.00	**85.41**	72.10	**88.65**	83.53	82.74	85.41
MFCCs	52.30	68.00	66.06	**74.83**	74.22	61.51	49.35	58.27	61.27	**63.10**
MNIST	23.52	39.40	N/A	64.80	**66.56**	N/A	N/A	54.74	N/A	**66.25**

Table 5. Overall runtime (secs) on Artificial and Real-world datasets. The best and second best results are highlighted by **bold** and underline respectively.

	Typical Density-based Methods					K-Way Partition Methods				
Datasets	DBSCAN	Meanshift	Quickshift	Quickshift++	FastDEC$_D$	FINCH	DPC	QuickDSC	SNN-DPC	FastDEC
banana-ball	**0.02**	9.17	0.40	0.05	0.04	0.15	2.63	0.15	25.55	**0.04**
Flame	**0.00**	0.54	0.01	**0.00**	**0.00**	0.03	0.01	**0.00**	0.34	**0.00**
R15	**0.00**	1.01	0.03	0.01	**0.00**	0.03	0.07	0.01	1.94	**0.00**
Spiral	**0.00**	0.38	0.04	**0.00**	**0.00**	0.11	0.02	**0.00**	0.49	**0.00**
S2	0.05	12.44	1.66	0.05	**0.04**	0.57	4.56	0.07	207.52	**0.04**
seeds	**0.00**	0.16	0.01	**0.00**	**0.00**	0.04	0.02	**0.00**	0.43	**0.00**
Banknote	**0.01**	6.49	0.15	0.08	0.03	0.13	0.39	0.05	19.95	**0.02**
Segmentation	**0.09**	8.68	0.30	0.21	0.14	**0.14**	1.08	0.17	41.38	**0.14**
Phonemes	**0.27**	180.57	1.08	7.71	7.12	**0.59**	4.17	7.46	123.40	7.02
MFCCs	**0.74**	30.36	2.96	1.02	0.96	1.11	10.74	1.28	413.45	**1.08**
MNIST	**31.00**	8306.19	N/A	237.38	200.43	N/A	N/A	210.05	N/A	**178.44**

varies from 0.05 to 2 with step size 0.05 in all datasets. DBSCAN requires an additional parameter min_{pt} that corresponds to the noise filtering. Regarding PCA, we also use the implementation provided by Scikit-Learn. Besides, Quickshift++ and QuickDSC need a parameter β, we vary it from 0.1 to 0.3 with step size 0.1.

5.1 Comparison on Artificial and Real-World Datasets

The evaluation of the clustering results on real-world datasets is shown in Table 3, Table 4, and Table 5. The bold represents the best result, and the second-best result is highlighted by underline, "N/A" denotes we cannot obtain the clustering result on the experimental host (e.g., Out-of-Memory). By varying the control parameters, for each method, the best clustering result and the cor-

responding execution time is reported. The runtime 0.00 indicates that the runtime is less than 0.005 seconds. Note: rather than the Nearest Density Higher (NDH) relation, FastDEC adopts the Nearest Density Higher **or Equal** (NDHE) relation in the Segmentation.

Based on the tables above, several findings are drawn:

1. FastDEC (two versions) always achieves the best or the second-best results among all datasets. The performance of the τ-ball based methods (DBSCAN, DPC, Meanshift, Quickshift) is relatively poor. The performance of Quickshift++ and QuickDSC are competitive. The former is a typical density-based method, while the latter is a K-Way partition method;
2. Both Quickshift++ and QuickDSC need an additional parameter β which requires extra effort in tuning. FINCH is parameter-free by ignoring the mode information, thus its performance is relatively poor. For our method, only FastDEC$_D$ needs to tune the k, but still, the overall performance by varying k is outstanding and robust;
3. Regarding the algorithm runtime, there is no doubt that DBSCAN is the fastest method. FINCH is quite fast as well by utilizing the 1^{st}-NN graph. On the contrary, SNN-DPC is rather slow for extracting shared-nearest-neighbor relations. However, the methods above ignore the DH information and lead to a substantial reduction in clustering quality. FastDEC obtains the best or the second-best result in terms of the runtime on almost all datasets, which demonstrates its efficiency.

To sum up, FastDEC provides a balanced yet excellent solution that considers effectiveness, efficiency, and parameter-tuning.

5.2 Robustness Testing

Now, we test the robustness of the algorithm parameters. We only demonstrate the experiments on two datasets MFCCs and MNIST, more results are attached as Supplement Material. Again, comparisons are conducted for typical density-based methods and K-Way partition methods separately. For the former, Meanshift, Quickshift, DPC and DBSCAN use the primary parameter τ, but the remaining use the k. We convert them into the same granularity: for example, setting $k = 1$ is identical to setting $\tau = 0.01$. For parameter β, we use the suggested setting 0.3 [20]. Specifically, Figure 3 (a) and Figure 3 (b) show the NMI results of the typical density-based methods in the MFCCs and MNIST respectively. Figure 3 (c) and Figure 3 (d) are that of the K-Way partition methods respectively. According to Figure 3 (a) and Figure 3 (b), τ-based algorithms are parameter sensitive, so it is difficult to capture the best setting. On the other hand, the performance of k-based algorithms, especially the FastDEC$_D$, is quite stable. It again demonstrates our method is robust. Regarding the K-Way partition, the performance of FastDEC$_K$ is compromised as well. It completely outperforms the other algorithms including the parameter-free algorithm FINCH.

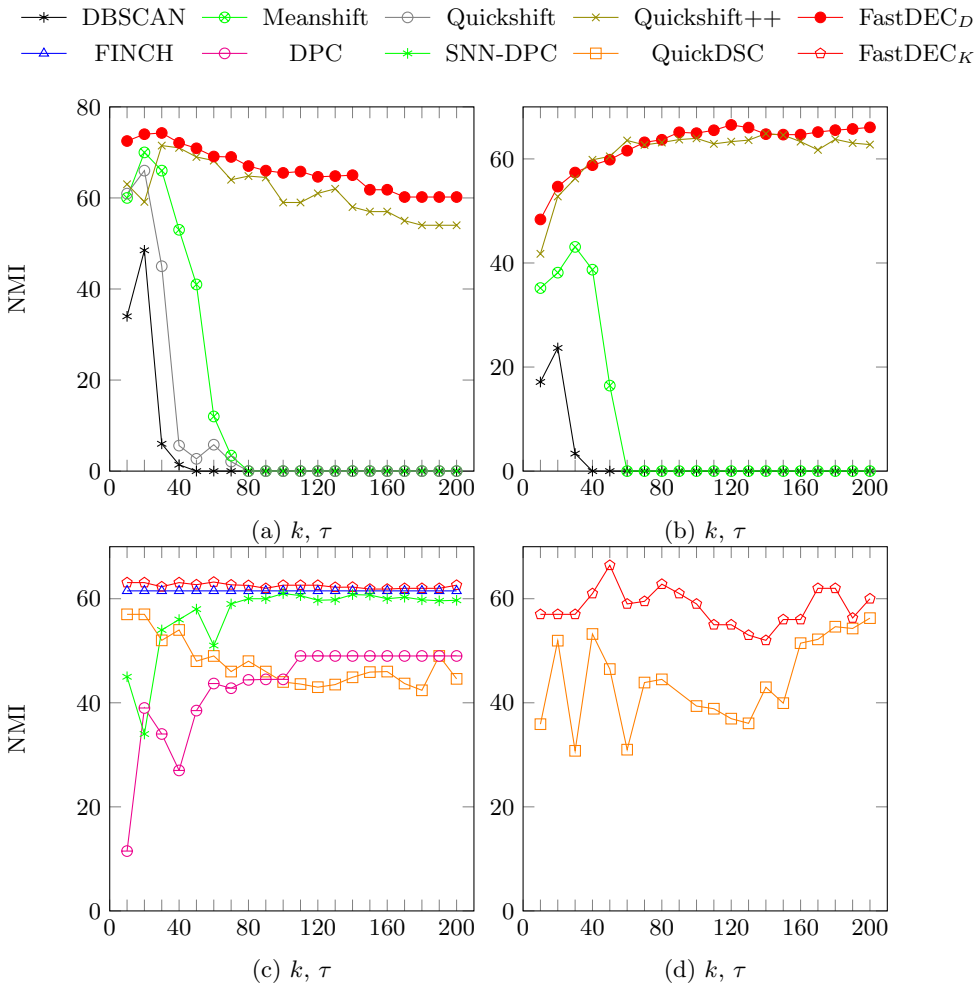

Fig. 3. Robustness Testing of parameters k and τ on MFCCs, MNIST.

6 Conclusion

In this paper, we propose FastDEC, a novel density-based clustering framework. It provides two versions to satisfy different requirements: FastDEC$_D$ for identifying the clusters with arbitrary shapes, and FastDEC$_K$ for K-Way partition. FastDEC detects the nearest density higher (NDH) relation from a k-NN graph and forms the dominance-component (DC), based on which, conduct the clusters. The experiments on real-world datasets demonstrate that our method has excellent performance, meanwhile, it is efficient and easy to tune. In the future, we consider a distributed version of FastDEC to handle the extremely large datasets.

Acknowledgment We thank the anonymous reviewers for their constructive comments and thoughtful suggestions. This work was supported in part by: National Key D&R Program of China (019YFB1600704, 2021ZD0111501), NSFC (61603101, 61876043, 61976052), NSF of Guangdong Province (2021A1515011941), State's Key Project of Research and Development Plan (2019YFE0196400), NSF for Excellent Young Scholars (62122022), Guangzhou STIC (EF005/FST-GZG/2019/GSTIC), NSFC-Guangdong Joint Fund (U1501254), the Science and Technology Development Fund, Macau SAR (0068/2020/AGJ, 0045/2019/A1, SKL-IOTSC(UM)-2021-2023, GDST (2020B1212030003).

References

1. Amagata, D., Hara, T.: Fast density-peaks clustering: Multicore-based parallelization approach. In: SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021. pp. 49–61. ACM (2021)
2. Angelino, C.V., Debreuve, E., Barlaud, M.: Image restoration using a knn-variant of the mean-shift. In: ICIP. pp. 573–576. IEEE (2008)
3. Cai, J., Wei, H., Yang, H., Zhao, X.: A novel clustering algorithm based on DPC and PSO. IEEE Access **8**, 88200–88214 (2020)
4. Carreira-Perpiñán, M.Á., Wang, W.: The k-modes algorithm for clustering. CoRR **abs/1304.6478** (2013)
5. Chang, H., Yeung, D.: Robust path-based spectral clustering. Pattern Recognit. **41**(1), 191–203 (2008)
6. Chaudhuri, K., Dasgupta, S.: Rates of convergence for the cluster tree. In: NIPS. pp. 343–351. Curran Associates, Inc. (2010)
7. Chaudhuri, K., Dasgupta, S., Kpotufe, S., von Luxburg, U.: Consistent procedures for cluster tree estimation and pruning. IEEE Trans. Inf. Theory **60**(12), 7900–7912 (2014)
8. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 790–799 (1995)
9. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
10. Dasgupta, S., Freund, Y.: Random projection trees and low dimensional manifolds. In: STOC. p. 537–546 (2008)
11. Davidson, I., Ravi, S.S.: Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In: PKDD. vol. 3721, pp. 59–70. Springer (2005)
12. Dong, W., Charikar, M., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: WWW. pp. 577–586. ACM (2011)
13. Du, M., Ding, S., Jia, H.: Study on density peaks clustering based on k-nearest neighbors and principal component analysis. Knowl. Based Syst. **99**, 135–145 (2016)
14. Dua, D., Graff, C.: UCI machine learning repository (2017), `http://archive.ics.uci.edu/ml`
15. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD. pp. 226–231 (1996)
16. Fränti, P., Virmajoki, O.: Iterative shrinking method for clustering problems. Pattern Recognit. **39**(5), 761–775 (2006)

17. Fu, L., Medico, E.: Flame, a novel fuzzy clustering method for the analysis of DNA microarray data. BMC Bioinform. **8** (2007)
18. Hastie, T., Friedman, J.H., Tibshirani, R.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics, Springer (2001)
19. Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: KDD. pp. 58–65 (1998)
20. Jiang, H., Jang, J., Kpotufe, S.: Quickshift++: Provably good initializations for sample-based mean shift. In: ICML. vol. 80, pp. 2299–2308. PMLR (2018)
21. Jiang, H., Kpotufe, S.: Modal-set estimation with an application to clustering. In: AISTATS. vol. 54, pp. 1197–1206. PMLR (2017)
22. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
23. Liu, R., Wang, H., Yu, X.: Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Inf. Sci. **450**, 200–226 (2018)
24. Myhre, J.N., Mikalsen, K.Ø., Løkse, S., Jenssen, R.: Robust clustering using a knn mode seeking ensemble. Pattern Recognit. **76**, 491–505 (2018)
25. Rasool, Z., Zhou, R., Chen, L., Liu, C., Xu, J.: Index-based solutions for efficient density peak clustering (extended abstract). In: 37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021. pp. 2342–2343. IEEE (2021)
26. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**(6191), 1492–1496 (2014)
27. Sarfraz, M.S., Sharma, V., Stiefelhagen, R.: Efficient parameter-free clustering using first neighbor relations. In: CVPR. pp. 8934–8943 (2019)
28. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
29. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: ECCV. vol. 5305, pp. 705–718. Springer (2008)
30. Veenman, C.J., Reinders, M.J.T., Backer, E.: A maximum variance cluster algorithm. IEEE Trans. Pattern Anal. Mach. Intell. **24**(9), 1273–1280 (2002)
31. Wang, W., Carreira-Perpiñán, M.Á.: The laplacian k-modes algorithm for clustering. CoRR **abs/1406.3895** (2014)
32. Xie, J., Gao, H., Xie, W., Liu, X., Grant, P.W.: Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors. Inf. Sci. **354**, 19–40 (2016)
33. Yang, Y., Deng, S., Lu, J., Li, Y., Gong, Z., U, L.H., Hao, Z.: Graphlshc: Towards large scale spectral hypergraph clustering. Inf. Sci. **544**, 117–134 (2021)
34. Yang, Y., Gong, Z., Li, Q., U, L.H., Cai, R., Hao, Z.: A robust noise resistant algorithm for POI identification from flickr data. In: IJCAI. pp. 3294–3300. ijcai.org (2017)
35. Zhang, T., Ramakrishnan, R., Livny, M.: SIGMOD. pp. 103–114. ACM Press (1996)
36. Zheng, X., Ren, C., Yang, Y., Gong, Z., Chen, X., Hao, Z.: Quickdsc: Clustering by quick density subgraph estimation. Inf. Sci. **581**, 403–427 (2021)