

Article

ADPSCAN: Structural Graph Clustering with Adaptive Density Peak Selection and Noise Re-Clustering

Xinyu Du , Fangfang Li ^{*}, Xiaohua Li and Ge Yu 

School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China; xinyudu@stumail.neu.edu.cn (X.D.); lixiaohua@mail.neu.edu.cn (X.L.); yuge@mail.neu.edu.cn (G.Y.)

* Correspondence: lifangfang@cse.neu.edu.cn

Abstract: Structural graph clustering is a data analysis technique that groups nodes within a graph based on their connectivity and structural similarity. The Structural graph clustering SCAN algorithm, a density-based clustering method, effectively identifies core points and their neighbors within areas of high density to form well-defined clusters. However, the clustering quality of SCAN heavily depends on the input parameters, ϵ and μ , making the clustering results highly sensitive to parameter selection. Different parameter settings can lead to significant differences in clustering results, potentially compromising the accuracy of the clusters. To address this issue, a novel structural graph clustering algorithm based on the adaptive selection of density peaks is proposed in this paper. Unlike traditional methods, our algorithm does not rely on external parameters and eliminates the need for manual selection of density peaks or cluster centers by users. Density peaks are adaptively identified using the generalized extreme value distribution, with consideration of the structural similarities and interdependencies among nodes, and clusters are expanded by incorporating neighboring nodes, enhancing the robustness of the clustering process. Additionally, a distance-based structural similarity method is proposed to re-cluster noise nodes to the correct clusters. Extensive experiments on real and synthetic graph datasets validate the effectiveness of our algorithm. The experiment results show that the ADPSCAN has a superior performance compared with several state-of-the-art (SOTA) graph clustering methods.

Keywords: structural clustering; adaptive peak selection; generalized extreme value distribution; distance-based similarity



Citation: Du, X.; Li, F.; Li, X.; Yu, G. ADPSCAN: Structural Graph Clustering with Adaptive Density Peak Selection and Noise Re-Clustering. *Appl. Sci.* **2024**, *14*, 6660. <https://doi.org/10.3390/app14156660>

Academic Editor: Augusto Ferrante

Received: 4 July 2024

Revised: 24 July 2024

Accepted: 26 July 2024

Published: 30 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Graphs or networks represent a fundamental data structure utilized for depicting data without a fixed schema [1]. Composed of nodes and edges, nodes symbolize objects while edges denote the relationships between these objects. Clustering and classification tasks find applications across various domains. For instance, the latest methods in hyperspectral image (HSI) classification, such as M3FuNet [2], and the CC-MIDNN [3] method for estimating arrival times, illustrate their wide-ranging impact. This paper focuses primarily on graph clustering tasks. Graph clustering is a critical task in graph data analysis, aimed at dividing vertices into distinct groups where connections within groups are dense, and those between groups are sparse. This technique is widely applied in identifying functional modules in metabolic networks [4], communities in social networks [5], research teams in collaborative networks [6], and handwritten character recognition [7,8].

Numerous graph clustering methods have been proposed to date, encompassing a range of approaches such as graph partitioning [9], hierarchical methods [10], density-based methods [11,12], grid-based methods [13], spectral methods [14] and propagation-based methods [15]. Most existing methods successfully partition clusters. However, many methods still face challenges in practical. In particular, selecting the appropriate graph clustering technique requires careful consideration of specific application contexts and

data characteristics. For instance, the SCAN [16] is a structural graph clustering method. The core premise of the SCAN algorithm is that if two nodes are sufficiently similar in terms of their neighboring nodes, they are likely to belong to the same cluster. Unlike other clustering algorithms, SCAN can identify the unique roles of different nodes, such as identifying hub nodes that connect distinct clusters and outliers that are loosely connected to any cluster [16]. Although the SCAN algorithm has been successfully implemented in various fields, its clustering accuracy is heavily dependent on the selection of parameters ϵ and μ . This dependency makes the clustering results extremely sensitive to parameter adjustments, and varying these parameters can lead to significant differences in clustering results [17]. Additionally, according to the SCAN algorithm's definition, any boundary nodes that are neither core nodes nor connected to core nodes are considered noise. This categorization may erroneously label many boundary nodes that have specific connectivity patterns as noise, especially in sparsely connected graphs where the lack of sufficient connections among nodes tends to increase the prevalence of noise nodes. Therefore, the main issues affecting the accuracy of structural graph clustering are:

1. How to reduce or avoid the impact of parameter variations on clustering results?
2. How to correctly assign misclassified noise points to the appropriate clusters?

Considering the limitations of existing solutions, we propose a new structural graph clustering algorithm called ADPSCAN, which includes adaptive density peak selection and noise re-clustering parts. This algorithm executes structural graph clustering in three main steps. Firstly, we use our defined methods for local density and dependency similarity to construct a decision graph based on the input data. Secondly, to adaptively identify density peaks in the decision graph, we propose using the Generalized Extreme Value (GEV) distribution to fit the local density and dependency similarity. This fitting process yields thresholds for local density and dependency similarity, which are then used to identify density peak points. These points serve as potential cluster centers and form clusters with their first-order neighbors. If two density peak points are first-order neighbors, they are merged into the same cluster. Lastly, to accurately handle nodes that may be misclassified as noise but actually have specific connection patterns, we propose a distance-based structural similarity method. This method calculates the structural similarity between noise points and the existing clusters, reassigning the noise points to the cluster with the highest structural similarity.

Our main contributions can be summarized as follows:

1. ADPSCAN: We propose a structural graph clustering algorithm that utilizes adaptive density peak selection to significantly reduce the dependency on parameters ϵ and μ , characteristic of traditional SCAN algorithms. Our method utilizes a distance-based structural similarity metric to accurately reclassify noise nodes. Our method not only enhances the flexibility of the clustering process but also boosts the algorithm's adaptability to diverse datasets.
2. Adaptive Density Peaks Selection: We propose a method that utilizes the generalized extreme value distribution to automatically select density peaks from decision graphs, overcoming the limitations of existing density peak-based structural graph clustering algorithms that require manual selection of cluster centers.
3. Noise Re-clustering: To address the issue of misclassifying noise nodes, we propose a distance-based structural similarity method that effectively reassigns noise nodes to the correct clusters.
4. Experimental Validation: We conduct extensive experiments on various real and synthetic network datasets. The results demonstrate that our approach significantly enhances clustering accuracy.

2. Related Work

In this section, we review the Density Peaks Clustering (DPC) algorithm and structural graph clustering techniques.

Density Peaks Clustering. The DPC algorithm [18] is a density-based clustering technique primarily employed to identify cluster centers through local density and distance metrics. One of the primary advantages of this method is its inherent ability to identify distinct clusters naturally, without the necessity of predefining the number of clusters. The essence of the DPC method lies in pinpointing data points that have the highest local density and are significantly distant from other high-density points, designating them as cluster centers. While the DPC algorithm demonstrates robust clustering capabilities, its performance can be compromised when dealing with non-uniform datasets. Recently, several variants of the DPC have been proposed to enhance the original algorithm. CPF [19] improves clustering by partitioning data into areas of distinct densities before identifying density peaks within each area, thus grouping points within the same cluster. The BC-DPC algorithm [20] refines cluster center identification by creating balanced densities to mitigate differences among clusters. However, due to fundamental differences in problem definitions, these methods do not apply to structural graph clustering.

Structural Graph Clustering. The SCAN algorithm [16] is a well-known method within the field of graph clustering. Its core concept is based on a simple yet powerful principle: if two nodes are sufficiently similar within their neighborhood, then they are likely to belong to the same cluster. This method not only identifies clustering results but also recognizes the unique roles of different nodes within the graph. For instance, the algorithm can identify hub nodes that connect different clusters and outliers that do not have strong connections to any cluster. This capability allows for a deeper understanding of the intrinsic structure and relationships within the graph.

Despite its effectiveness, the SCAN algorithm [16] exhibits lower efficiency when handling large-scale graphs, leading to the development of various optimized versions. One line of research has focused on overcoming the efficiency challenges of the original SCAN algorithm in processing large graphs. For instance, the SCAN++ algorithm [1] reduces redundant similarity calculations significantly by accurately identifying vertex types and their two-hop neighbors. Additionally, the pSCAN algorithm [11], a state-of-the-art method based on pruning techniques, optimizes the computation sequence of vertices and implements timely pruning strategies to accelerate clustering. This approach avoids unnecessary similarity calculations after vertex types have been determined, thereby enhancing processing efficiency.

On another front, methods like GS*-Query [17], developed based on the GS*-Index [17] algorithm, offer an efficient response to clustering queries. SCAN-XP [21], ppSCAN [22], and GBBSIndexSCAN [23] leverage parallel computing technologies to expedite the generation of clustering results, making them particularly suitable for the rapid processing of large-scale network data.

Although all these methods have made advancements in the field of graph clustering, they often fail to adequately address issues related to parameter dependency or clustering accuracy. This paper proposes a structural graph clustering algorithm based on the adaptive selection of density peaks, which effectively eliminates reliance on input parameters. Additionally, we propose a distance-based structural similarity method for accurately managing noise nodes, thereby enhancing the quality of the clustering process, as well as its level of automation.

3. Problem Definition

In this paper, we focus on an unweighted undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, with $|V|$ represented by n and $|E|$ by m . The objective of this paper is to develop a new structural graph clustering algorithm that adaptively selects density peaks as potential cluster centers, thereby reducing dependency on parameters and improving clustering accuracy by reassigning noise points. The key parameters involved in the algorithm are summarized in Table 1.

Table 1. The Summary Of Key Notations.

Not.	Definition and Description
$G = (V, E)$	Unweighted undirected graph, with V as nodes and E as edges.
$N[u]$	Structural neighbors of node u , including u itself.
$\deg[u]$	Degree of node u , i.e., the size of $N[u]$.
$N(u)$	Open neighborhood of u , i.e., its direct neighbors.
$\sigma(u, v)$	Structural similarity between nodes u and v .
μ_{uv}	Relative similarity between u and its neighbor v .
ρ_u	Local density of u , summing $\sigma(u, v)$ values under a normal distribution.
δ_u	Dependency similarity of u , the max similarity to neighbors with higher density.
$\text{DirREACH}(u, v)$	Direct density reachability from u to v if u is a peak.
$\text{REACH}(u, v)$	Density reachability if there is a path of peaks between u and v .

Definition 1 (Structural Neighbors). For a graph $G(V, E)$, the structural neighbors of a vertex u include its neighboring vertices and itself, denoted as $N[u] = \{v \in V \mid (u, v) \in E\} \cup \{u\}$.

Note that, the degree of u , denoted by $\deg[u]$, is the cardinality of $N[u]$ (i.e., $\deg[u] = |N[u]|$). The open neighborhood of u , denoted by $N(u)$, is the set of neighbors of u (i.e., $N(u) = \{v \in V \mid (u, v) \in E\}$).

Definition 2 (Structural Similarity). The structural similarity between two vertices u and v in a graph is defined as the number of common neighbors between u and v divided by the square root of the product of the degrees of u and v . This metric is formulated as shown in Equation (1) and denoted by $\sigma(u, v)$.

$$\sigma(u, v) = \frac{|N[u] \cap N[v]|}{\sqrt{\deg[u] \cdot \deg[v]}} \quad (1)$$

In the algorithm design section, two metrics are defined for each vertex: local density and dependency similarity. Utilizing these definitions, density peak vertices can be identified, and the algorithm's extension steps can then be used to form the final set of clusters.

Definition 3 (Local Density). Given $\mu_{uv} = \frac{\sigma_t}{\sigma(u, v)}$, where vertex v is a structural neighbor of vertex u , the local density of vertex u is defined as the sum of the values of $\sigma(u, v)$ under a standard normal distribution for μ_{uv} , as shown in Equation (2) and denoted by ρ_u .

$$\rho_u = \sum_{v \in N[u]} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\mu_{uv}^2}{2}\right) \quad (2)$$

Definition 4 (Dependency Similarity). The maximum structural similarity between vertex u and any of its neighboring vertices that have a higher local density than u is referred to as dependency similarity. This measure is presented in Equation (3) and denoted by δ_u .

$$\delta_u = \max_{v: \rho_v > \rho_u, v \in N(u)} (\sigma(u, v)) \quad (3)$$

For example, in Figure 1a, the first-order neighbors of node u are v_1 , v_2 , and v_3 . Among these, v_1 and v_3 have higher local densities than node u . The dependency similarity δ_u is the maximum structural similarity between u and either v_1 or v_3 .

Based on local density and dependency similarity, a definition for density peak vertices can be established. A vertex u is designated as a density peak vertex if it possesses a relatively high local density ρ_u and a relatively low dependency similarity δ_u .

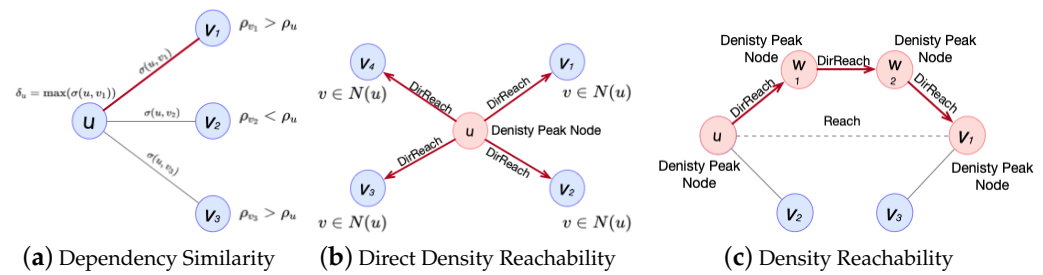


Figure 1. Visualization of Key Definitions in the Problem Statement.

Definition 5 (Direct Density Reachability). *If a vertex is a density peak, then it is directly density-reachable from all its structural neighbors. For instance, if vertex u is a density peak and vertex v is a structural neighbor of vertex u , then vertex u is directly density-reachable to vertex v , which is formally defined as follows:*

$$\text{DirREACH}(u, v) \Leftrightarrow \text{DensityPeak}(u) \wedge v \in N(u) \quad (4)$$

For example, in Figure 1b, node u is a density peak, and node u is directly density-reachable to all its first-order neighbors.

Definition 6 (Density Reachability). *If there exists a path between two vertices such that every vertex along the path is a density peak, then these two vertices are considered density-reachable. Vertices that are density-reachable are grouped into the same cluster. For instance, if vertex u is a density peak and there is a path from vertex u to vertex v where every vertex on the path is a density peak, then vertex u is density-reachable to vertex v , which is formally defined as follows:*

$$\text{REACH}(u, v) \Leftrightarrow \exists \text{ path } P(u \rightarrow v) : \forall w \in P, \text{DensityPeak}(w) \quad (5)$$

Here, w represents each vertex w along the path $P(u \rightarrow v)$, emphasizing that each of these vertices is a density peak. For example, in Figure 1c, there is a path from u to v_1 that passes through w_1 and w_2 , where u , w_1 , w_2 , and v_1 are all density peaks. Therefore, u is density-reachable to v_1 .

4. Method

In this section, we propose a structural graph clustering algorithm named ADPSCAN, which includes adaptive density peak selection and noise re-clustering parts. Our algorithm aims to address the parameter dependency issues inherent in traditional SCAN algorithms, significantly enhancing clustering accuracy. Traditional SCAN algorithms typically rely on manually setting parameters ϵ and μ . To overcome this challenge, we propose analyzing data trends in decision graphs and utilizing the generalized extreme value distribution to automatically select thresholds for local density and dependency similarity, thereby further enabling the selection of density peak nodes. This automated approach identifies density peaks as potential clustering centers, reducing the manual intervention required and making the clustering process more automated. Additionally, we propose distance-based structural similarity to effectively handle noise nodes, accurately reassigning these nodes to appropriate clusters based on their structural similarities with clusters. This further enhances the algorithm's accuracy. Framework of our algorithm is shown in Figure 2. We will now provide a detailed description of each component.

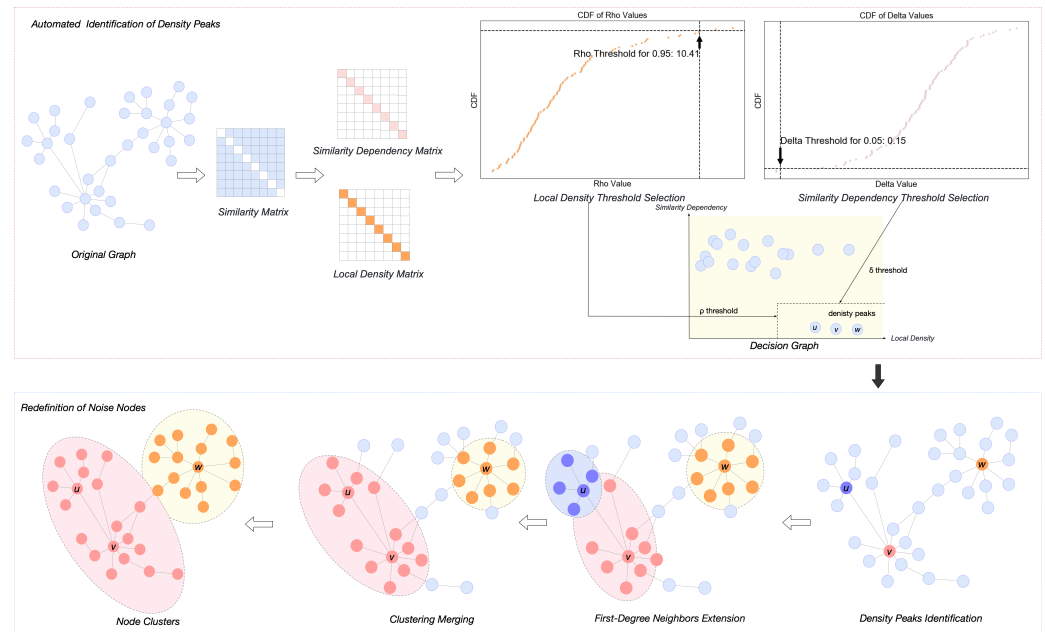


Figure 2. Overview of ADPSCAN.

4.1. Adaptive Density Peaks Selection

In this section, we present our method for the automated identification of density peaks in structural graph clustering. Our approach fundamentally relies on analyzing decision graphs and utilizing the generalized extreme value distribution to autonomously determine clustering centers. This greatly reduces the dependency on parameter selection that is common in traditional clustering algorithms. In Section 4.1.1, we define decision graphs and describe their role in our proposed methodology. In Section 4.1.2, we detail the observed characteristics of decision graphs and their correlation with the generalized extreme value distribution. Subsequently, in Section 4.1.3, we introduce the generalized extreme value distribution and validate its close fit to the observed characteristics of decision graphs through preliminary experiments. Finally, in Section 4.1.4, we introduced the process of adaptive density peak selection and the pseudo code.

4.1.1. Decision Graph

Decision graph is a commonly utilized tool in density peaks clustering algorithms, employed for visualizing and assisting in the selection of clustering centers. In this graphical representation, each point within the graph represents a data point; the horizontal axis typically indicates the local density of the point (i.e., the number or density of surrounding points), while the vertical axis represents the distance to the nearest point with a higher density. In this paper, in accordance with Definitions 3 and 4, we design decision graphs where ρ is plotted on the x-axis and δ on the y-axis, representing the local density and dependency similarity of each node in the graph, respectively.

4.1.2. Observation

Nodes on the decision graph exhibit several distinctive features:

1. **Cluster Centers:** Nodes characterized as potential cluster centers typically possess a higher local density and a lower dependency similarity, consistently appearing in the lower right quadrant of the decision graph.
2. **Non-cluster Nodes:** Conversely, nodes that are not cluster centers often display high values of both ρ and δ or low values of ρ and high values of δ , typically found in the upper right or upper left quadrants.

Observations indicate that cluster centers typically exhibit higher local density (ρ) and lower dependency similarity (δ), while non-cluster centers display the opposite char-

acteristics, with lower local density and higher dependency similarity. This distribution characteristic provides a critical insight: it is feasible to distinguish between cluster centers and non-cluster centers by setting appropriate thresholds. Specifically, determining suitable thresholds for ρ and δ can effectively identify potential cluster centers. The selection of these thresholds not only impacts the accuracy of clustering but also directly influences the efficiency of the clustering process. Next, we will explore in detail how to automatically determine these thresholds through the algorithm, optimizing the identification of cluster centers and reducing the dependency on manual intervention. This step is a key component of the adaptive density peak identification in our proposed structural graph clustering algorithm.

By observing the local density graph in Figure 3a and the dependency similarity graph in Figure 3b of nodes, it is apparent that certain nodes exhibit significantly higher local density and significantly lower dependency similarity, albeit in smaller numbers. The presentation of extreme values in local density and dependency similarity characteristics is crucial here. The generalized extreme value distribution is used to model extreme values in samples, particularly suitable for extreme value theory. Thus, employing statistical methods to determine thresholds for local density and dependency similarity is appropriate. Therefore, the decision graph is divided into two sections by these thresholds, with cluster centers located in the lower right quadrant and other nodes in the opposite section. Therefore, cluster centers can be automatically identified.

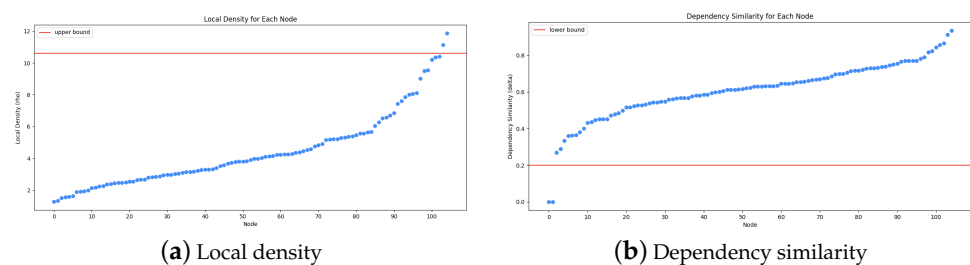


Figure 3. Local density and dependency similarity distribution.

4.1.3. Generalized Extreme Value Distribution

Local density and dependency similarity approximately follow the Generalized Extreme Value (GEV) distribution. In statistics, the GEV distribution is a class of continuous probability distributions used to model the maximum or minimum values of samples. It plays a pivotal role in extreme value theory, particularly in identifying and analyzing extreme behaviors in datasets, such as the identification of cluster centers in this paper. The GEV distribution encompasses three types of extreme value distributions: Gumbel, Fréchet, and Weibull, each corresponding to different tail behaviors. The GEV distribution can be expressed as follows:

$$G(z; \mu, \sigma, \xi) = \exp \left(- \left[1 + \xi \left(\frac{z - \mu}{\sigma} \right) \right]^{-\frac{1}{\xi}} \right) \quad (6)$$

In the formula, z represents a standardized variable, while μ , σ , and ξ are the location, scale, and shape parameters, respectively. The value of the shape parameter ξ determines the type of the GEV distribution. In this paper, we employ maximum likelihood estimation to automatically derive these parameters. As demonstrated in Figure 4a,b, the fits for local density and dependency similarity are performed using the generalized extreme value cumulative distribution, which aligns perfectly with the empirical distributions.

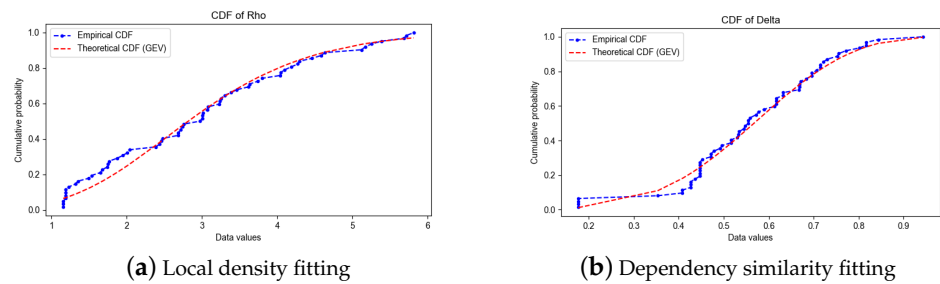


Figure 4. Fitting plots of GEV.

After obtaining the three parameters through maximum likelihood estimation, x_p can be obtained using the Equation (7), where p represents the probability of the quantile, and $\hat{\sigma}$, $\hat{\mu}$, and $\hat{\xi}$ denote the estimated parameters of the generalized extreme value distribution:

$$\hat{x}_p = \begin{cases} \hat{\mu} - \frac{\hat{\sigma}}{\hat{\xi}}(1 - y_p^{-\hat{\xi}}), & \text{if } \hat{\xi} \neq 0; \\ \hat{\mu} - \hat{\sigma} \log y_p, & \text{if } \hat{\xi} = 0, \end{cases} \quad (7)$$

$$y_p = -\log p \quad (8)$$

where $y_p = -\log p$ is the quantile function corresponding to the probability p . Specifically, $x_{p\rho}$ represents the threshold for local density, and $x_{p\delta}$ represents the threshold for dependence similarity. These thresholds are computed using the parameters $\hat{\sigma}$, $\hat{\mu}$, and $\hat{\xi}$. If the local density of node i exceeds $x_{p\rho}$ and the dependence similarity is less than $x_{p\delta}$, it is considered a density peak node. In the following section, we will introduce the specific design used to determine thresholds through the generalized extreme value distribution.

4.1.4. Adaptive Density Peaks Selection Algorithm Procedure

The method for adaptively selecting density peaks is outlined in Algorithm 1. First, the upper tail of local density is fitted to the GEV distribution to obtain the parameter set `gev_params_rho`. Subsequently, the lower tail of dependency similarity is also fitted to the GEV distribution, yielding its distribution parameters, `gev_params_delta` (lines 1–2). Based on these parameter sets, we calculate the thresholds for ρ and δ . The threshold for ρ is determined by setting a high probability point to isolate extreme high values, while the threshold for δ is identified through a low probability point to recognize extreme low values (lines 3–4). The algorithm begins by initializing an empty list, `density_peaks`, to store the indices of the identified density peaks (line 5). It then iterates over each node i in the decision graph (line 6), checking whether the local density $\rho[i]$ exceeds the threshold `threshold_rho` and the dependency similarity $\delta[i]$ falls below the threshold `threshold_delta` (line 7). If both conditions are satisfied, the node i is appended to the `density_peaks` list (line 8). Finally, the algorithm returns the list of identified density peaks (line 9). The functions `fit_gev` and `find_threshold` are defined to fit the GEV distribution and determine the thresholds, respectively. The `fit_gev` function takes the data and the tail type as inputs and fits the GEV distribution accordingly (lines 10–15). For the upper tail, it directly fits the data, whereas for the lower tail, it fits the negated data to focus on the lower extremes. The `find_threshold` function calculates the threshold based on the fitted parameters and the specified probability. For the upper tail, it uses the probability directly, and for the lower tail, it adjusts the probability to reflect the lower extreme values (lines 16–20). And The probability is typically set to 0.95, as it represents a confidence level that ensures the selected thresholds capture the significant extreme values while maintaining robustness against outliers.

Algorithm 1: Adaptive Density Peaks Selection

Input : Local density values, ρ ; Dependency similarity values, δ
Output: Density Peaks

```

1 // Fit GEV distribution
2 gev_params_rho ← fit_gev( $\rho$ , 'upper')
3 gev_params_delta ← fit_gev( $\delta$ , 'lower')
4 // Determine threshold
5 threshold_rho ← find_threshold(gev_params_rho, 'upper', rho_prob)
6 threshold_delta ← find_threshold(gev_params_delta, 'lower', delta_prob)
7 density_peaks ← empty list
8 for each node  $i$  in the decision graph do
9     if  $\rho[i] \geq \text{threshold\_rho}$  and  $\delta[i] \leq \text{threshold\_delta}$  then
10         // Identify density peaks
11         density_peaks.append( $i$ )
12 return density_peaks
13 Function fit_gev(data, tail):
14     if tail == 'upper' then
15         // Fit upper tail
16         params ← genextreme.fit(data)
17     else
18         // Fit lower tail
19         params ← genextreme.fit(-data)
20     return params
21 Function find_threshold(params, tail, probability):
22     if tail == 'upper' then
23         // Upper tail threshold
24         return genextreme.ppf(probability, *params)
25     else
26         // Lower tail threshold
27         return -genextreme.ppf(1 - probability, *params)

```

4.2. Redefining Noise Nodes

In this section, we propose a distance-based structural similarity measurement method used for effectively partitioning noise nodes in graph data. Our method relies on distance information between nodes in the graph to identify and reclassify nodes erroneously labeled as noise. Initially, we introduce the definition of structural similarity and its application in our approach. Finally, we demonstrate the specific design of this method in clustering algorithms.

4.2.1. Calculate Structural Similarity

In contrast to Definition 2, when addressing the problem of partitioning noise nodes, structural similarity is computed by comparing the sets of neighboring nodes reachable within a specific hop range in the graph. Specifically, for any two nodes u and v in the graph $G = (V, E)$, we define their similarity $\text{sim}(u, v)$ as follows:

$$\text{sim}(u, v) = \frac{|N_L(u) \cap N_L(v)|}{|N_L(u) \cup N_L(v)|} \quad (9)$$

where $N_L(u)$ and $N_L(v)$ respectively represent the sets of neighbors within L hops for nodes u and v . For each node, we recursively compute the set of neighbor nodes reachable within up to L hops. Specifically, for any node u , the set of neighbors reachable within L hops is obtained by merging the adjacency sets of neighbors reachable within $L - 1$ hops of u .

Through this approach, we address the situation where the structural similarity between any arbitrarily non-adjacent nodes is always zero. This method quantifies the structural similarity between any two nodes in the graph, providing a means for in-depth analysis based on data structure for subsequent noise nodes processing.

4.2.2. Re-Clustering of Noise Nodes

During the initial clustering process, some nodes may be erroneously labeled as noise due to various reasons. Through distance-based structural similarity, we can effectively reassess the cluster membership of these nodes. Our method first focuses on noise nodes and computes their distance-based structural similarity with each cluster's centroid nodes to evaluate their similarity. The noise nodes are then assigned to the cluster with the highest similarity to the centroid, and the clustering results are updated accordingly. The pseudo-code for our method is provided in Algorithm 2.

Algorithm 2: Calculate Distance-Based Structural Similarity

Input: Graph $G = (V, E)$, hop
Output: Dictionary D representing distance similarity

```

1 Initialize result as list of sets for each node for up to hop hops;
2 for  $h$  from 0 to  $hop - 1$  do
3   foreach node  $u$  in  $V$  do
4     if  $h = 0$  then
5       // Set initial adjacency
6       result[ $u$ ][ $h$ ]  $\leftarrow$  set( $adj$ );
7     else
8       // For each neighbor
9       foreach neighbor  $n$  in  $adj$  do
10        // Update result for hop
11        result[ $u$ ][ $h$ ]  $\leftarrow$  result[ $u$ ][ $h$ ]  $\cup$  result[ $n$ ][ $h - 1$ ];
12 Initialize distance_similarity as empty dictionary;
13 foreach node  $x$  in  $V$  do
14   Initialize distance_similarity[ $x$ ];
15   foreach node  $y$  in  $V$  do
16     if  $x = y$  then
17       distance_similarity[ $x$ ][ $y$ ]  $\leftarrow$  1;
18     else
19       // Compute intersection and union
20       Set Intersection and Union of result[ $x$ ][ $hop - 1$ ] and result[ $y$ ][ $hop - 1$ ];
21       if Union empty then
22         distance_similarity[ $x$ ][ $y$ ]  $\leftarrow$  0;
23       else
24         // Similarity is intersection over union
25         distance_similarity[ $x$ ][ $y$ ]  $\leftarrow$   $\frac{\text{Intersection}}{\text{Union}}$ ;
26 return distance_similarity;
```

Firstly, we establish a result list, to store collections of neighbor nodes for each node in the graph by the number of hops (line 1). For each hop from 0 to hop-1, the algorithm iterates over all nodes in the graph. At hop 0, the direct neighbor set of each node is directly stored in the corresponding entry in the result list. For hops greater than 0, the algorithm traverses through each node's neighbors, incorporating the neighbor's previous hop neighbor set into the current node's current hop neighbor collection (lines 2–8). Then, we establish a dictionary distance_similarity, to store the similarity degree between any two nodes (line 9). For every pair of nodes, x and y , within the graph, if the nodes are identical, their similarity is set to 1. If different, their similarity is calculated based on the intersection and

union of their neighbor sets at the maximum hop count. If the union is empty, the similarity is set to 0; otherwise, the similarity is calculated as the size of the intersection divided by the size of the union (lines 10–20).

For example, as shown in Figure 5, the distance-based structural similarity between nodes u and v is calculated using their L -hop neighborhood sets. Specifically, the method for calculating structural similarity is based on the ratio of the size of the intersection of these node sets to the size of their union. In the figure, the number of nodes n_1, n_2, \dots, n_7 represents the intersection of the neighborhood sets $N_L(u) \cap N_L(v)$, and the sum of the numbers of nodes s_1, s_2, \dots, s_5 and n_1, n_2, \dots, n_7 represents the union $N_L(u) \cup N_L(v)$. By dividing the size of the intersection by the size of the union, the distance-based structural similarity is obtained $\text{sim}(u, v)$.

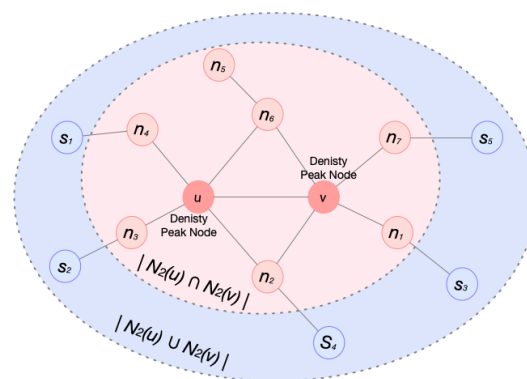


Figure 5. Distance-Based Structural Similarity.

Using the method above for computing structural similarity, we calculate the average structural similarity between noise nodes and each cluster. Specifically, this involves computing the average distance-based structural similarity between a node and all nodes in a cluster. If a cluster achieves the highest average similarity score, the noise node will be reassigned to this cluster. In the subsequent section, we will provide a detailed implementation of the method for reassigning noise nodes.

Proof. Let N be the set of all noise nodes. Let C_k denote the cluster k . Define the average structural similarity $\text{avg_sim}_{u,k}$ of a noise node u with the nodes in cluster k as:

$$\text{avg_sim}_{u,k} = \frac{1}{|C_k|} \sum_{v \in C_k} \text{sim}(u, v)$$

where $|C_k|$ represents the number of nodes in cluster k , and $\text{sim}(u, v)$ denotes the structural similarity between nodes u and v .

For a noise node u , compute its average structural similarity with each cluster C_k as follows:

$$\text{avg_sim}_{u,k} = \frac{1}{|C_k|} \sum_{v \in C_k} \text{sim}(u, v)$$

Let $\text{avg_sim}_{u,k}$ represent the average similarity of node u with the nodes in cluster k . If:

$$\text{avg_sim}_{u,k} \geq \text{avg_sim}_{u,j} \text{ for all } j \neq k,$$

then node u should be assigned to cluster k . This is because if $\text{avg_sim}_{u,k}$ is the largest among all clusters, then the overall similarity between node u and the nodes in cluster k is the highest. That is, the average similarity of node u with cluster k is greater than with any other cluster.

By assigning a noise node to the cluster with the maximum structural similarity, we ensure that the node is allocated to the most appropriate cluster. This method enhances the accuracy of clustering results and effectively manages noise nodes. \square

4.3. ADPSCAN

In this section, we summarize the work above, and the pseudocode of ADPSCAN is shown in Algorithm 3.

Algorithm 3: ADPSCAN

Input: Graph $G = (V, E)$
Output: Clusters

```

1 similarity  $\leftarrow$  calculate_similarity( $G$ );
2  $\rho \leftarrow$  calculate_rho(similarity,  $|V|$ );
3  $\delta \leftarrow$  calculate_delta(similarity,  $\rho$ ,  $|V|$ );
4 peak_list  $\leftarrow$  automatic_density_peaks_selection( $\rho, \delta$ )(Algo.1);
5 clusters  $\leftarrow$  get_clusters_reclassify_noise( $G$ , peak_list);
6 distance_similarity  $\leftarrow$  calculate_distance_structural_similarity( $G$ , peak_list, hop);
7 clusters  $\leftarrow$  reclassify_noise( $G$ , peak_list, distance_similarity);
8 return clusters;
```

Firstly, we calculate the structural similarity for all nodes in the graph (line 1). From this similarity, we calculate each node's local density (ρ) and dependency similarity (δ) (lines 2–3). We then identify density peaks using an automated density peaks selection method (line 4). We traverse each density peak node, initializing a cluster for unvisited peak nodes and adding their neighboring nodes to a pending queue. We continuously extract nodes from this queue and incorporate them into the current cluster. If these nodes are also density peak nodes, we add their unvisited neighbors to the queue to expand the cluster (line 5). Finally, for unprocessed nodes, identified as noise, we calculate the average distance-based structural similarity to the clusters and assign them to the cluster with the highest structural similarity (lines 6–7).

5. Experiments

In this section, we evaluate our proposed algorithm, ADPSCAN, on synthetic and real networks to demonstrate its advantages.

Selection of Comparative Methods. To assess the performance of ADPSCAN, we compare it against state-of-the-art structural graph clustering algorithms. Additionally, to thoroughly validate the effectiveness of our method, we also compare it with five other representative graph clustering algorithms.

- **DistanceSCAN** [24] is the state-of-the-art graph clustering algorithm based on distance, which utilizes structural similarities and distance information between nodes to identify core nodes and their community structures within a graph.
- **GS*-Query** [17] is a recently proposed graph clustering approach, which is an index-based approach.
- **DPSCAN** [25] is a density peak-based structural graph clustering method that identifies core nodes using density and similarity metrics.
- **FluidC** [15] is also a recently proposed graph clustering approach, which is based on the idea of fluids interacting in an environment.
- **Girvan-Newman** [26] (GN) is a method used in network analysis to detect community structure in networks. It identifies communities within networks by iteratively removing edges with high betweenness centrality, revealing clusters of densely interconnected nodes.
- **SCAN** [16] is a method that detects communities in networks by clustering nodes based on structural similarity and hop-reachability, distinguishing hubs and outliers effectively.

Evaluation Measures. To conduct a comprehensive comparison of the performance of various graph clustering algorithms, we employ two prevalent metrics: Normalized

Mutual Information (NMI) and Adjusted Rand Index (ARI), both of which are commonly used to gauge the effectiveness of clustering outcomes.

5.1. Synthetic Networks

In this section, we construct several synthetic networks to assess the performance of various graph clustering algorithms. We use the LFR benchmark network [9], which facilitates easy modifications of degree distributions and cluster sizes. Due to the high time complexity of GN and SCAN, our comparisons are limited to clustering algorithms such as DistanceSCAN, DPSCAN, GS*-Query, and FluidC.

Cluster Density: To assess the algorithms' responsiveness to network variations, the average degree (k) was adjusted, maintaining a fixed mixing parameter ($\lambda = 0.1$) while varying k from 5 to 25. Each network consisted of 5000 nodes. As k increased, the performance of five methods, evaluated by NMI, is displayed in Figure 6a. In Figure 6a, DistanceSCAN shows moderate performance at lower k values, but its performance improves as k increases. In contrast, FluidC's performance diminishes with increasing k . ADPSCAN maintains stable performance across the range, particularly excelling at $k = 15$, where it nearly achieves perfect clustering.

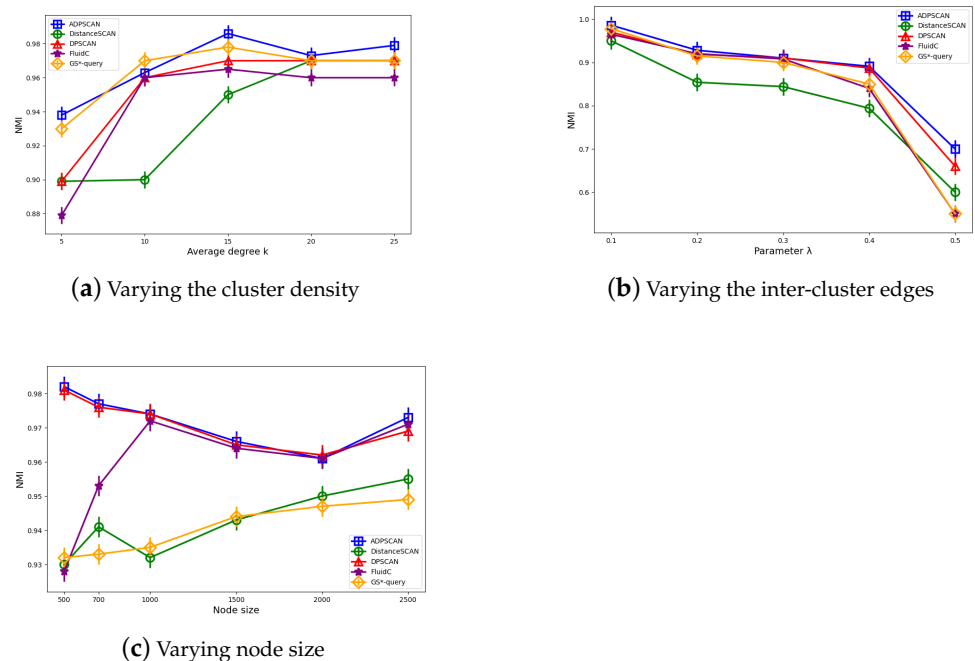


Figure 6. Performance of different algorithms on the LFR benchmark networks.

Inter-Cluster Edge: To evaluate the algorithms' sensitivity to changes in inter-cluster edges, the λ was varied from 0.1 to 0.5, with $k = 15$, generating networks with different inter-cluster connectivity. Each network comprised 5000 nodes. As λ increased, the performance of all five methods, assessed by NMI, is depicted in Figure 6b. Figure 6b shows a sharp decline in performance for all algorithms at $\lambda = 0.4$, with GS*-Query and FluidC experiencing the most rapid declines. Both ADPSCAN and GS*-Query approached perfect clustering at $\lambda = 0.1$.

Network Scale: To assess the adaptability of algorithms to networks of varying sizes, we fixed the mixing parameter $\lambda = 0.1$ and the average degree $k = 15$, and varied the number of nodes in the network from 500 to 2500. This setup allows us to observe the performance and stability of the algorithms as they handle networks of different sizes. As shown in Figure 6c, ADPSCAN consistently performs well, peaking at 500 nodes; DistanceSCAN steadily improves; DPSCAN peaks at 500 nodes but declines afterward; FluidC exhibits significant fluctuations on smaller datasets but gradually stabilizes as the node

size increases.; GS*-query starts lower but shows significant improvement, highlighting each algorithm's scalability and efficiency.

5.2. Real World Data

To evaluate various graph clustering algorithms, we employ three publicly available real-world networks. Complete descriptions and data for these networks are accessible through the UCI Network Data Repository (<https://networkdata.ics.uci.edu/index> (accessed on 25 July 2024)) and the Stanford Large Network Dataset Collection (<https://snap.stanford.edu/data/index.html> (accessed on 25 July 2024)).

Zachary's Karate Club Network: The Karate Club dataset is a well-known social network dataset. Collected by Wayne W. Zachary in the 1970s, it was derived from observing the social interactions among members of a karate club at an American university. The dataset includes 34 members and 78 edges representing social interactions among them. This network became particularly famous due to a case study where internal conflict led to the club's split. Zachary accurately predicted the members' alignment during the split, showcasing the potential of social network analysis in understanding dynamics of social structures. This dataset is frequently utilized to test the effectiveness of social network analysis and graph clustering algorithms.

Books About US Politics: The network dataset consists of books related to US politics, which were published around the 2004 presidential election. It represents books as nodes, sourced from sales on Amazon.com. The connections or edges between nodes indicate that books were often bought together by the same customers. This network includes 105 nodes and 441 edges. According to Mark Newman, these books are categorized into three groups: "liberal," "neutral," and "conservative."

Dolphin social network: This dataset offers a comprehensive analysis of the behavioral patterns of 62 bottlenose dolphins in New Zealand, resulting in the creation of a social network diagram that illustrates their interactions. In this network, each of the 62 nodes represents an individual dolphin, with edges between nodes indicating frequent interactions between dolphin pairs. The Dolphin network is frequently cited as a classic example for community detection and segmentation in complex network studies, illustrating the social connections among the dolphins, which are categorized into two distinct groups.

5.2.1. Clustering Quality Analysis of ADPSCAN

The experimental results on several real-world datasets demonstrate that the ADPSCAN algorithm excels in clustering accuracy. We evaluated its performance on three datasets—Karate, Polbooks, and Dolphins—and compared the results with other clustering algorithms, including DistanceSCAN, GS*-Query, DPSCAN, FluidC, GN, and SCAN. The results are summarized in Table 2.

Table 2. Performance of different graph clustering algorithms on real-world data sets.

Algorithm	Karate		Polbooks		Dolphins	
	ARI ^a	NMI ^b	ARI ^a	NMI ^b	ARI ^a	NMI ^b
ADPSCAN	1.000	1.000	0.668	0.601	0.935	0.889
DistanceSCAN	0.515	0.502	0.633	0.531	0.753	0.608
GS*-Query	0.725	0.684	0.673	0.581	0.759	0.784
DPSCAN	0.882	0.837	0.695	0.593	0.586	0.740
FluidC	0.882	0.837	0.635	0.547	0.408	0.459
GN	0.882	0.837	0.680	0.576	0.935	0.889
SCAN	0.725	0.628	0.601	0.537	0.601	0.672

^a ARI: Measures how similar two clustering results are, with values from −1 to 1. Higher values mean more similarity. ^b NMI: Measures the shared information between two clustering results, with values from 0 to 1. Higher values mean better agreement. The bold values indicate the best performance for each dataset.

On the Karate dataset, ADPSCAN outperformed the other five algorithms, achieving a perfect score of 1.000 in both ARI and NMI metrics. This indicates that ADPSCAN is highly effective in precisely delineating cluster boundaries within structurally simple and well-defined networks. Figure 7a illustrates the clustering results of ADPSCAN on the Karate dataset, successfully grouping the data into two clusters, which further validates the algorithm's effectiveness.

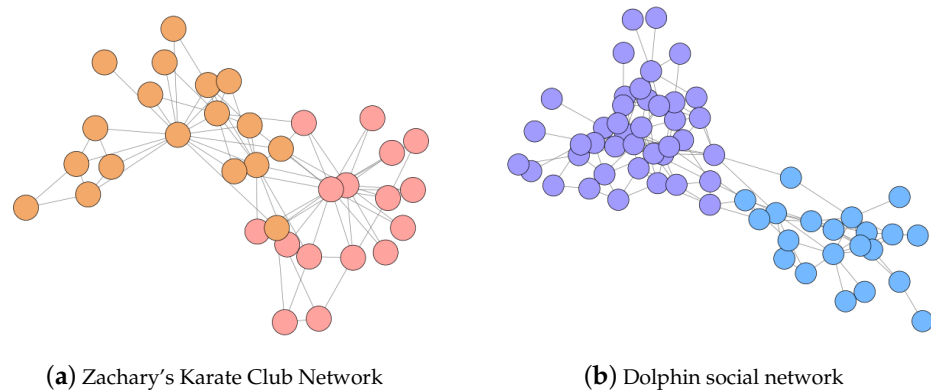


Figure 7. Performance of different algorithms on the LFR benchmark networks.

In the Polbooks dataset, ADPSCAN achieved an NMI score of 0.601, outperforming other compared algorithms. Although the ARI score is slightly lower than that of DPSCAN (0.668 vs. 0.695), this discrepancy can be attributed to the different focuses of the ARI and NMI metrics. ARI emphasizes accurately recovering the true group boundaries within the dataset, while NMI assesses the amount of shared information between the algorithmically identified groups and the true groups. The presence of nodes connecting multiple clusters in the Polbooks dataset contributes to the slight decrease in ARI score.

For the Dolphins dataset, ADPSCAN achieved the highest values for both ARI (0.935) and NMI (0.889), indicating exceptional performance in capturing the true clustering structure within biologically influenced complex networks. Figure 7b presents the clustering results of ADPSCAN on the Dolphins dataset, where the algorithm accurately identifies the majority of dolphins in their respective clusters.

Overall, the observation that ADPSCAN consistently achieves optimal NMI and ARI scores can be attributed to two key components of its design. The first component is the adaptive density peak selection module, this module accurately identifies potential cluster centers without requiring manual input, thereby ensuring the accuracy of clustering. Additionally, the second component is the noise re-clustering module, the high clustering quality of ADPSCAN is further enhanced by its ability to re-cluster noise points. By computing the distance-based structural similarity between noise points and clusters, ADPSCAN reassigns noise points to appropriate clusters, thereby improving overall clustering quality.

5.2.2. Performance of Noise Re-Clustering

To validate the effectiveness and practical feasibility of our method, we conducted ablation studies on multiple real-world datasets, including Karate, Polbooks, and Dolphins. These datasets are not only characterized by complex structures and diverse data characteristics but are also widely used to evaluate the performance of clustering algorithms. Our study focused particularly on assessing the impact of noise re-clustering. We designated the version of our algorithm without noise re-clustering as ADPSCAN-NR. The decision not to perform ablation studies on adaptive density peaks selection was deliberate. Adaptive density peaks selection forms the foundational step of our approach, responsible for identifying potential cluster centers without requiring manual input from users. The accuracy of this step directly influences the precision of subsequent clustering. Hence, we concentrated our efforts on evaluating how noise re-clustering enhances the final clustering results, ensuring the integrity of our method.

Across the Karate dataset, ADPSCAN achieved perfect scores with ARI and NMI both reaching 1.000 compared to ADPSCAN-NR. On the Polbooks dataset, ARI and NMI improved to 0.668 and 0.601, respectively, while on Dolphins, these metrics rose to 0.935 and 0.889. These significant performance improvements can be attributed to our noise re-clustering method, which calculates distance-based structural similarities between noise points and clusters, thereby enhancing clustering quality and stability (Table 3).

In conclusion, our ablation study underscores that by integrating adaptive threshold selection and noise re-clustering, our method substantially enhances clustering accuracy.

Table 3. Ablation Study Results for Noise Re-clustering.

Algorithm	Karate		Polbooks		Dolphins		Avg.ARI ^c	Avg.NMI ^d
	ARI ^a	NMI ^b	ARI ^a	NMI ^b	ARI ^a	NMI ^b		
ADPSCAN	1.000	1.000	0.668	0.601	0.935	0.889	0.868	0.845
ADPSCAN-NR	0.486	0.494	0.193	0.349	0.168	0.326	0.282	0.390

^a ARI: Measures how similar two clustering results are, with values from -1 to 1 . Higher values mean more similarity. ^b NMI: Measures the shared information between two clustering results, with values from 0 to 1 . Higher values mean better agreement. ^c Avg.ARI: Average Adjusted Rand Index across all datasets. ^d Avg.NMI: Average Normalized Mutual Information across all datasets. The bold values indicate the best performance for each dataset.

6. Conclusions

In this paper, a novel structural graph clustering algorithm, ADPSCAN, is proposed, based on the adaptive selection of density peaks. To tackle the high dependency of traditional methods on external parameters, the generalized extreme value distribution is utilized to adaptively pinpoint potential clustering centers. Preliminary experiments have confirmed the validity of this method. Additionally, considering the potential misclassification of nodes as noise in existing algorithms, a distance-based structural similarity strategy has been devised. This strategy precisely measures the average distance scores between noise points and clusters, effectively reassigning noise nodes to their appropriate clusters. Comprehensive experiments on both real-world and synthetic network datasets demonstrate the effectiveness of the ADPSCAN algorithm. The results highlight its broad application potential in fields such as social network analysis, collaborative network research, and metabolic network functional module identification. Despite its robust performance, ADPSCAN's complexity is relatively high, with a worst-case time complexity of $O(n^2)$, where n represents the number of nodes in the network. This high complexity makes it unsuitable for ultra-large networks, such as the UK-2002 dataset from the Web Algorithmics Laboratory. The primary computational cost stems from calculating distance-based structural similarity. Future research could address scalability and speed issues by incorporating indexing and caching mechanisms, or by leveraging parallel computing and distributed systems.

Author Contributions: Conceptualization, X.D.; methodology, F.L.; software, X.D.; validation, X.D.; formal analysis, X.L.; investigation, F.L.; data curation, F.L.; writing—original draft preparation, X.D.; writing—review and editing, F.L.; visualization, X.D.; supervision, G.Y.; project administration, X.L.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

- Shiokawa, H.; Fujiwara, Y.; Onizuka, M. Scan++ efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. *Proc. VLDB Endow.* **2015**, *8*, 1178–1189. [\[CrossRef\]](#)
- Chen, H.; Long, H.; Chen, T.; Song, Y.; Chen, H.; Zhou, X.; Deng, W. M³FuNet: An unsupervised multivariate feature fusion network for hyperspectral image classification. *IEEE Trans. Geosci. Remote. Sens.* **2024**, *62*, 5513015. [\[CrossRef\]](#)
- Deng, W.; Li, K.; Zhao, H. A flight arrival time prediction method based on cluster clustering-based modular with deep neural network. *IEEE Trans. Intell. Transp. Syst.* **2023**, *25*, 6238–6247. [\[CrossRef\]](#)
- Guimera, R.; Amaral, L.A.N. Functional cartography of complex metabolic networks. *Nature* **2005**, *433*, 895–900. [\[CrossRef\]](#) [\[PubMed\]](#)
- Santo, F. Community detection in graphs. *Phys. Rep.* **2010**, *486*, 75–174.
- Yang, Z.; Hong, C.; Jeffrey, X.Y. Graph Clustering Based on Structural/Attribute Similarities. *PVLDB* **2009**, *2*, 718–729.
- Hebbi, C.; Mamatha, H.R. Comprehensive dataset building and recognition of isolated handwritten kannada characters using machine learning models. *Artif. Intell. Appl.* **2023**, *1*, 179–190. [\[CrossRef\]](#)
- Bhosle, K.; Musande, V. Evaluation of deep learning CNN model for recognition of devanagari digit. *Artif. Intell. Appl.* **2023**, *1*, 114–118. [\[CrossRef\]](#)
- Ding, C.H.; He, X.; Zha, H.; Gu, M.; Simon, H.D. A min-max cut algorithm for graph partitioning and data clustering. Proceedings 2001 IEEE international conference on data mining. *IEEE*, **2001**, 107–114.
- Shiokawa, H.; Fujiwara, Y.; Onizuka, M. Fast algorithm for modularity-based graph clustering. *Proc. Aaai Conf. Artif. Intell.* **2013**, *27*, 1170–1176. [\[CrossRef\]](#)
- Chang, L.; Li, W.; Qin, L.; Zhang, W.; Yang, S. pSCAN: Fast and exact structural graph clustering. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 387–401. [\[CrossRef\]](#)
- Tariq, R.; Lavangnananda, K.; Bouvry, P.; Mongkolnam, P. An Edge-Based Approach to Partitioning and Overlapping Graph Clustering with User-Specified Density. *Appl. Sci.* **2024**, *14*, 380. [\[CrossRef\]](#)
- Chen X. Clustering based on a near neighbor graph and a grid cell graph. *J. Intell. Inf. Syst.* **2013**, *40*, 529–554. [\[CrossRef\]](#)
- Ni, L.; Manman, P.; Qiang, W. A Spectral Clustering Algorithm for Non-Linear Graph Embedding in Information Networks. *Appl. Sci.* **2024**, *14*, 4946. [\[CrossRef\]](#)
- Parés, F.; Gasulla, D.G.; Vilalta, A.; Moreno, J.; Ayguadé, E.; Labarta, J.; Suzumura, T. Fluid communities: A competitive, scalable and diverse community detection algorithm. In *Complexnetworks*; Cherifi, C., Cherifi, H., Karsai, M., Musolesi, M., Eds.; Springer: Cham, Switzerland, 2017; Volume 689, pp. 229–240.
- Xu, X.; Yuruk, N.; Feng, Z.; Schweiger, T.A. SCAN: A structural clustering algorithm for networks. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, USA, 12–15 August 2007; pp. 824–833.
- Wen, D.; Qin, L.; Zhang, Y.; Chang, L.; Lin, X. Efficient structural graph clustering: An index-based approach. *Proc. Vldb Endow.* **2017**, *11*, 243–255. [\[CrossRef\]](#)
- Rodriguez, A.; Laio, A. Clustering by fast search and find of density peaks. *Science* **2014**, *344*, 1492–1496. [\[CrossRef\]](#) [\[PubMed\]](#)
- Tobin, J.; Zhang, M. A Theoretical Analysis of Density Peaks Clustering and the Component-wise Peak-Finding Algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *46*, 1109–1120. [\[CrossRef\]](#) [\[PubMed\]](#)
- Zhang, Q.H.; Dai, Y.Y.; Wang, G.Y. Density peaks clustering based on balance density and connectivity. *Pattern Recogn.* **2023**, *2022*, 109052. [\[CrossRef\]](#)
- Tomokatsu, T.; Hiroaki, S.; Hiroyuki, K. SCAN-XP: Parallel Structural Graph Clustering Algorithm on Intel Xeon Phi Coprocessors; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1–7.
- Yulin, C.; Shixuan, S.; Qiong, L. Parallelizing Pruning-based Graph Structural Clustering. In Proceedings of the 47th International Conference on Parallel Processing, Eugene, OR, USA, 13–16 August 2018.
- Tom, T.; Laxman, D.; Julian, S. Parallel Index-Based Structural Graph Clustering and Its Approximation. In Proceedings of the 2021 International Conference on Management of Data, Auckland, New Zealand, 7–10 December 2021; pp. 1851–1864.
- Liu, K.; Wang, S.; Zhang, Y.; Xing, C. An Efficient Algorithm for Distance-Based Structural Graph Clustering. *Proc. ACM Manag. Data* **2023**, *1*, 45. [\[CrossRef\]](#)
- Wu, C.; Gu, Y.; Yu, G. Dpscan: Structural graph clustering based on density peaks. In *International Conference on Database Systems for Advanced Applications*; Springer International Publishing: Cham, Switzerland, 2019; pp. 626–641.
- Girvan, M.; Newman, M.E. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [\[CrossRef\]](#) [\[PubMed\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.