Expert Systems With Applications 213 (2023) 119213

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Overlapping community detection with adaptive density peaks clustering and iterative partition strategy

Yunyun Niu [a], Detian Kong [b], Ligang Liu [a], Rong Wen [c], Jianhua Xiao [b,d,*]

[a] School of Information Engineering, China University of Geosciences in Beijing, Beijing 100083, China
[b] The Research Center of Logistics, Nankai University, Tianjin 300071, China
[c] Singapore Institute of Manufacturing Technology, Singapore 138634, Singapore
[d] The Laboratory for Economic Behaviors and Policy Simulation, Nankai University, Tianjin 300071, China

ARTICLE INFO

Keywords:
Community detection
Overlapping community
Density peaks clustering
Real social network
Synthetic network

ABSTRACT

The community structure is a collection of individuals with common characteristics that commonly exists in complex networks. The detection of community structures aids in mining information in the network and exploring the evolution mechanism of complex network systems. Compared with other traditional community-detection algorithms, the density peak clustering (DPC) algorithm, which has attracted extensive attention, can detect communities with arbitrary shapes through more efficient and accurate clustering. Although many scholars have proposed improvements to DPC, the proper determination of the cut-off distance d_c, which is essential for selecting cluster centers, has generally been ignored. Therefore, in this study, we propose an overlapping community-detection algorithm based on adaptive DPC with an iterative partition strategy known as ODPI, which adaptively selects d_c based on different network scales and features. Unlike the DPC algorithm, which manually selects cluster centers, the ODPI algorithm uses an iterative k-means clustering method to select community centers. Extensive experiments have been conducted on both real social networks and synthetic networks to demonstrate that ODPI has satisfactory performance on networks with both a complex structure distribution and complex weight distribution.

1. Introduction

Community detection is a critical means of studying complex networks. It can aid in mining the information in a network and exploring the evolution mechanisms of complex network systems. Many earlier studies, such as weighted community clustering (Prat-Pérez et al., 2014) and game theory-based methods (Bu et al., 2019), only focused on non-overlapping communities. However, the interaction among different communities, which refers to the behavior of the interaction through overlapping nodes, tends to be closer to reality. For example, in a network of biological molecules, communities have different biological functions. A single gene often carries various biological functions, and thus, cannot simply be categorized into a certain community.

Density peaks clustering (DPC) is an algorithm that is employed for non-overlapping community detection. Compared to traditional community-detection algorithms, the advantage of DPC lies in its ability to detect communities with arbitrary shapes efficiently and accurately. However, DPC suffers from several inherent limitations: (1) it can only detect non-overlapping communities, which differ from most social networks; (2) it introduces a cut-off distance d_c that needs to be set manually when initializing the local density of nodes; and (3) it selects clustering centers manually, which often leads to inaccurate results on large-scale datasets.

Therefore, many scholars have proposed different strategies to enhance DPC. Most of these methods have mainly focused on extending DPC to detect overlapping communities (Bai et al., 2017), creating novel distance functions (Du et al., 2018), or selecting cluster centers adaptively (Liang & Chen, 2016). Despite their effectiveness, these algorithms still need to select the cut-off distance d_c based on experience, as with the vanilla DPC. However, for different scales of social networks, using different d_c will produce different quality results. Consequently, determining such a parameter appropriately remains a challenging task.

To this end, we propose a novel overlapping community-detection algorithm based on adaptive DPC using an iterative partition strategy (ODPI). In particular, ODPI calculates the cut-off distance d_c adaptively based on different network scales and features, which means that our

* Corresponding author at: The Research Center of Logistics, Nankai University, Tianjin 300071, China.
E-mail addresses: yniu@cugb.edu.cn (Y. Niu), kdt@mail.nankai.edu.cn (D. Kong), liuligang@yeah.net (L. Liu), wenr@simtech.a-star.edu.sg (R. Wen), jhxiao@nankai.edu.cn (J. Xiao).

https://doi.org/10.1016/j.eswa.2022.119213
Received 20 May 2022; Received in revised form 16 October 2022; Accepted 1 November 2022
Available online 7 November 2022

d_c can adapt to different network topologies automatically. The main contributions of our approach are three-fold:

- A new distance function is proposed to initialize the distance information between nodes, which considers the weight of the direct links between nodes, influence of common neighbors, and network topology. Thus, the distance function is applicable to both weighted and unweighted social networks.
- A new strategy is designed to calculate the cut-off distance d_c adaptively based on different network scales and features, which means that the value of d_c can be set automatically instead of being manually determined across different datasets.
- A new iterative partition strategy is adopted to aid in the adaptive selection of cluster centers after initializing the local density and separation distance of all nodes. This new cluster center assignment method further assists in achieving high-quality results on large-scale datasets.

The remainder of this paper is organized as follows. Section 2 briefly describes the traditional overlapping community-detection algorithms and the principle of DPC. Section 3 presents the proposed ODPI algorithm. In Section 4, we introduce the experimental setup for examining the effectiveness of our proposed ODPI, including the compared baselines, evaluation metrics, and test datasets. Section 5 elaborates on the experimental results and provides an in-depth analysis. Finally, Section 6 concludes the study and highlights potential directions for future research.

2. Related work

This section provides an overview of traditional overlapping community-detection algorithms and DPC-based algorithms. Table 1 summarizes all classes of algorithms explained in this section with a short introduction, as well as their advantages and limitations.

2.1. Traditional overlapping community detection algorithm

Traditional overlapping community-detection algorithms can be classified into five main categories: clique percolation-based methods, link partition-based methods, local expansion and optimization-based methods, fuzzy detection-based methods, and label propagation-based methods.

1. Clique Percolation based Methods (CPMs). CPMs discover communities by searching for adjacent completely connected subgraphs (k-clique) (Derényi et al., 2005). These methods consider completely connected subgraphs as nodes to construct a new network, and then detect communities on the new network. The nodes in the new network represent completely connected subgraphs; therefore, the overlapping nodes among communities can be obtained by mapping them to the original network. The CPMd algorithm is an improvement of the CPM algorithm (Palla et al., 2005) and it can be used in directed graphs. This approach has led to a new definition of the k-clique.

In general, CPMs need to determine a parameter k, and different k values lead to distinct performance differences. Hence, CPMs need to search the k-clique to detect overlapping communities. However, it is difficult to identify complete k-cliques in sparse complex networks, and such algorithms typically achieve poor performance on these networks.

2. Link Partition based Methods (LiPMs). LiPMs consider networks as link graphs that can be handled by clustering algorithms, and then detect communities on the new link graph (Wang et al., 2015). The advantage of LiPMs is that, by converting the original networks into link graphs, many suitable existing algorithms can be employed to handle them and obtain the community structure. Finally, to restore the link graphs, a community structure with overlapping nodes is obtained. It is worth noting that LiPMs partition links rather than nodes; that is, if a node has a link that is allocated to multiple communities, it is considered an overlapping node.

Ahn et al. proposed an overlapped community-detection algorithm based on link partition (Ahn et al., 2010). Their algorithm maps the original network onto an edge graph and then merges the two edges with the highest similarity iteratively. It uses partition density assessment criteria to determine the optimal partition level, and finally, the edge graph is restored to communities with nodes. Overlapping nodes are detected according to all nodes and their communities.

In summary, LiPMs are intuitional and can easily detect overlapping nodes, but their reliability has not been demonstrated properly (Xie et al., 2013) because they are based on the fuzzy definition of the community.

3. Local Expansion and Optimization based Methods (LEOMs). LEOMs rely on a partial community structure construction process based on local expansion and optimization (Wang et al., 2015). The OSLOM method optimizes the local statistical significance of the communities (Lancichinetti et al., 2011). It first identifies sub-communities in the network and defines the objective functions of the communities. Subsequently, it expands the sub-communities by optimizing the objective functions. The community structure can be obtained from these expanded communities. Another popular local expansion method is the seed-set expansion. Whang et al. use the personalized PageRank for seed expansion (Whang et al., 2013).

4. Fuzzy Detection based Methods (FDMs). FDMs use the linking strength between node pairs and communities. In general, a relationship vector is calculated for each node (Gregory, 2011) and then combined with the modularity function to obtain the community partition by maximizing the modularity using the greedy strategy. A notable shortcoming of FDMs is that the dimension of the relationship vector, which can significantly affect the performance, needs to be determined empirically in advance. For example, Wang et al. combined separation and detection methods with local optimization methods (Wang et al., 2012), whereby a partition is obtained from any algorithm that detects disjoint communities, following which an evaluation score is used to determine whether to add or remove nodes.

Furthermore, spectral theory can be used to detect overlapping communities (Zhang et al., 2007). This method computes several vectors that are related to the Laplace matrix of the graph, and then applies the fuzzy C-means clustering method for clustering on this eigenvector; finally, the results are re-transcribed on the graph to obtain the coverage.

5. Label Propagation based Methods (LaPMs). LaPMs propagate mutually influencing labels between nodes and can be extended to support overlapping community detection by allowing a node to have various labels. Several LaPMs are available, such as SPAEM (Ren et al., 2009), BMLPA (Wu et al., 2012), and MLPA (Rees & Gallagher, 2013). Among these algorithms, the most popular are COPRA (Gregory, 2010) and SLPA (Xie et al., 2011).

CORPA enables multiple labels to be assigned to each node and has been extended to support overlapping community detection by allowing multiple labels on a node. The most common labels are maintained under the intervention of probability thresholds, and the results indicate that a node may belong to multiple communities. Wu et al. proposed connectivity to reflect the inclination of a node community toward its neighboring community; therefore, they designed a new COPRA-based connectivity degree known as COPRA-CD. In COPRA-CD, all nodes are initialized with a unique community identifier and an affiliation factor of 1 (Wu & Zhang, 2015). In particular, its community identifier is updated using a combination of neighborhood labels, and the corresponding attribution coefficient is obtained by normalizing the sum of all neighborhood attribution coefficients. The experimental results demonstrated that both the quality and stability of the community detection were enhanced.

SLPA is an extension of the non-overlapping community-detection algorithm LPA (Raghavan et al., 2007). The speaker (label propagation node) and listener (label receiving node) are introduced based on the

Table 1
Summary of the state-of-the-art approaches.

Approaches	Basic feature	Advantage	Limitation
CPMs (Derényi et al., 2005) (Palla et al., 2005)	CPMs discover communities through searching adjacent complete connected subgraphs (k-clique).	CPMs may get a better result with more fully connected subgraphs.	CPMs usually achieve poor performance on sparse complex networks.
LiPMs (Ahn et al., 2010)	LiPMs consider networks as link graphs which can be handled by clustering algorithms, and then detect communities on the new link graph.	LiPMs are intuitional and can detect overlapping nodes easily.	LiPMs' reliability is not demonstrated properly, because they are based on the fuzzy definition of community.
LEOMs (Lancichinetti et al., 2011) (Whang et al., 2013)	LEOMs rely on partial community structure constructing process based on local expansion and optimization.	LEMOs can detect clusters in networks accounting for edge directions, weights, and community dynamics.	LEOMs only consider the structural characteristics and may return slightly less accurate results than other methods.
FDMs (Zhang et al., 2007) (Gregory, 2011)	FDMs utilize the linking strength between node pairs and communities. Usually a relationship vector is calculated for each node and then obtain the community partition by maximizing modularity.	FDMs can reduce the difficulty of detection by exploiting the fuzziness of the overlap.	The dimension of the relationship vector, which can heavily affect the performance, needs to be empirically determined in advance.
LaPMs (Gregory, 2010) (Xie et al., 2011)	LaPMs propagate mutually influencing labels between nodes and extend to support overlapping community detection via allowing a node to have various labels.	LaPMs have an advantage over simplicity, efficiency and speed.	The method has a low time complexity and it is easy to operate, but the algorithm has great uncertainty.
AMs (Chen et al., 2010) (Zhou et al., 2018) (Moscato et al., 2019)	A number of agents (nodes) are used for investigation of the input network. These agents consider different network feature in their investigations and make community detection decisions independently.	AMs can be easily improved by designing different agent strategies.	The obtained solution may correspond to a local equilibrium which are the states such that no agent can improve its own utility.
DPC (Rodriguez & Laio, 2014) (Du et al., 2016) (Xu et al., 2019)	DPC can rapidly cluster nodes with the same characteristics into a community based on the concept of clustering.	DPC detects communities with arbitrary shapes efficiently and accurately.	The original DPC exhibits several shortcomings and needs to be extended to be more practical.

LPA. First, each node is initialized to specify a label randomly. Thereafter, speakers send the label with most of their labels, and listeners receive the label with most of all labels that are sent. The algorithm based on label propagation interacts in a manner of propagation. First, SLPA offers the advantages of simplicity, efficiency, and speed; furthermore, the community structure that is detected by SLPA tends to be highly uncertain (Wang et al., 2015), which indicates that the community structure that is discovered in each run may be different.

Jokar et al. recently proposed a new method known as fuzzy balanced link density label propagation (BLDLP) (Jokar et al., 2021) for overlapping community detection based on the synergy of the BLDLP algorithm and fuzzy theory. The proposed method does not require prior information regarding the number of network communities for community discovery, and can successfully detect overlapping nodes and communities in synthetic and real-world networks.

6. Agent based Methods (AMs). In agent-based algorithms, several agents (nodes) are used to investigate the input network. These agents consider different network features in their investigations and make community-detection decisions independently. Badie et al. proposed an efficient agent-based algorithm known as SATOCI that uses the closeness of nodes (Badie et al., 2013) and operates based on label propagation (Raghavan et al., 2007).

Game theory-based methods also belong to this category. In general, the community formation process can be modeled as a game, and the game process is terminated when not all agents can improve their own utility. Chen et al. presented the first randomized algorithm to model the dynamics of community formation using a game-theoretic approach (Chen et al., 2010). Zhou et al. proposed a new gain function with similarity to calculate the utility of each agent, whereby the overlapping community structure is naturally presented when all utilities of all agents converge (Zhou et al., 2018).

Moscato et al. extended the above-mentioned well-established game-theoretic approaches and proposed a novel strategy for discovering communities in social graphs (Moscato et al., 2019). Their efficiency results revealed the possibility of using the system in industrial applications.

2.2. DPC

The DPC algorithm can rapidly cluster nodes with the same characteristics into a community based on the concept of clustering (Rodriguez & Laio, 2014), and has high accuracy and efficiency. DPC also exhibits several shortcomings: (1) the adjacency matrix representing the complex network needs to be processed and transformed into a distance matrix containing the distance information between all nodes that are input by the DPC; (2) human participation is required for selecting cluster centers; and (3) overlapping community structures containing overlapping nodes cannot be identified. In view of these shortcomings, the improvements are mainly implemented according to four aspects: (1) defining a new distance function to transform the adjacency matrix representing the complex network; (2) changing the method of calculating the local density of the nodes; (3) adaptively selecting clustering centers without human participation; and (4) extending the DPC to support the detection of overlapping community structures containing overlapping nodes.

Complex networks are usually represented by adjacency matrices. However, the DPC algorithm needs to input a distance matrix containing all distance information between the nodes. Therefore, it is necessary to convert the adjacency matrix into a distance matrix. Several methods have been proposed to initialize the distance between nodes using network structure information. For example, Du et al. used the Floyd algorithm to calculate the shortest path length between two nodes in a complex network as the distance between them (Du

et al., 2018). As the minimum distance between any nodes needs to be calculated, this concept is only effective on small-scale networks. In the OCDDP algorithm proposed by Bai et al. (2017), α intermediate nodes of the nodes are considered when calculating the distance between nodes. However, a long time is required to calculate the node distance for a large α. Moreover, the OCDDP algorithm is quite sensitive to α; thus, it is non-trivial to set appropriate values of α for various complex networks.

When DPC initializes the local density of a node, it traverses the sum of contributions of other nodes to this node. This method ignores the contribution of neighboring nodes, which leads to high computational costs for large-scale complex networks. To resolve this issue, many scholars have proposed improvement methods, the most common being the concept of K-nearest neighbors (KNN) (Du et al., 2016; Xie et al., 2016). The rationale behind this is that a shorter distance to a node indicates a greater influence on this node.

DPC selects a clustering center artificially on a two-dimensional decision graph that is composed of the local density and separation distance. Several advanced approaches have been proposed to address the problem of selecting cluster centers with human participation. For example, in the 3DC algorithm (Liang & Chen, 2016), the divide-and-conquer and density-reachable ideas are considered as recursive ending conditions to determine the clustering center. In the DPOCD (Sun et al., 2021) and EADP (Xu et al., 2019) algorithms, cluster centers are selected adaptively using a linear fitting-based strategy.

2.3. Basic elements of DPC

DPC measures two metrics for each point in the network: the local density ρ and separation distance δ (Nicosia et al., 2009). Clustering centers are believed to have a relatively high local density and separation distance.

Various methods are available for calculating the local density, two of which are described here. Both require a manually set parameter cut-off distance d_c. The first method for calculating the local density ρ_i of node v_i is presented in Eq. (1):

$$\rho_i = \sum_j \epsilon(dist(i,j) - d_c) \tag{1}$$

$$\epsilon(x) = \begin{cases} 1 & x \le 0 \\ 0 & x > 0 \end{cases} \tag{2}$$

where $dist(i,j)$ is the distance between node v_i and node v_j. This is the count of the number of nodes that are sufficiently close to node v_i.

The second method for calculating the local density is to use the Gaussian kernel function, as indicated in Eq. (3). When node v_j is closer to node v_i, the its contribution to the local density of node v_i is greater.

$$\rho_i = \sum_j \exp\left(-\left(\frac{dist(i,j)}{d_c}\right)^2\right). \tag{3}$$

The separation distance of node v_i is the distance between v_i and v_j, where v_j is the closest node to v_i with a local density that is greater than v_i. If v_i is the node with the largest local density in the social network, its separation distance is assigned to the maximum distance between v_i and any other node. The separation distance of node v_i is defined as

$$\delta_i = \begin{cases} \min_{j:\rho_j > \rho_i} dist(i,j), & \exists \rho_j > \rho_i \\ \max_{j:j \ne i} dist(i,j), & \nexists \rho_j > \rho_i \end{cases} \tag{4}$$

Once the local density ρ and separation distance δ have been calculated for each node, DPC manually selects the nodes with relatively larger ρ and δ values as the cluster centers. Finally, the remaining nodes are assigned to the community that is represented by the closest cluster center based on distance.

3. ODPI: an overlapping community detection algorithm based on adaptive density peaks clustering with iterative partition strategy

As noted previously, DPC cannot detect overlapping communities in social networks; it requires a distance matrix as its input, which is usually unavailable in social networks. To resolve these issues, adaptive DPC for overlapping community detection in social networks is proposed, which is equipped with a custom distance function to measure the distance between any two nodes, an adaptive local density calculation method, a cluster center assignment method using iterative k-means clustering, and a strategy for allocating overlapping communities for nodes. In the following section, we explain each module in detail.

3.1. Distance function

To use DPC in social networks, it is necessary to calculate the distance between any two nodes in the network. However, a social network is usually described as an adjacent matrix or linking weight matrix. In particular, in terms of an unweighted social network, the adjacent matrix is a binary matrix, where 1 indicates that two nodes are linked and 0 means that there is no link between the two nodes. For a weighted social network, the elements in the matrix are non-negative real numbers representing the linking weights between two nodes. Hence, it is necessary to convert this matrix into a distance matrix.

The adjacent matrix can be regarded as a special linking weight matrix, where the linking weights of any two nodes can be either 0 or 1. It is obvious that two nodes with a larger linking weight implies a closer relationship such that the distance between them should be smaller. Hence, a straightforward method for transforming the linking weight matrix is to use the reciprocals of the elements in the linking weight matrix as elements in the distance matrix. However, in social networks, many "zero" elements generally exist in the linking weight matrix, which leads to many infinite values in the distance matrix following the conversion, thereby impeding subsequent calculations. Therefore, we propose a novel distance function to convert the linking weight matrix by considering common neighbors to avoid the overly infinite value issue.

In a social network $G = (V, E)$, V denotes the set of nodes and E denotes the set of edges. $A \in \mathbb{R}^{|V|*|V|}$ is the linking weight matrix and A_{ij} denotes the linking weight between nodes v_i and v_j. When the distance between two nodes is calculated, it is essential to consider not only the linking weight between them, but also their common neighbors (Bhat & Abulaish, 2014; Xu et al., 2019). In general, two nodes without a direct link in E possess a zero linking weight. However, this does not mean that they have no relationship and should be assigned an infinite distance. In fact, two nodes with no direct links but with common neighbors in the social network also indicates a potential relationship between them, as they may interact via their common friends in the future. Intuitively, when two nodes share more common neighbors, their potential relationships are closer; that is, the probability of their becoming friends in the future is higher. In this sense, the distance between them should be smaller instead of infinite.

Based on the above analysis, we calculate the distance between nodes v_i and v_j. First, we calculate the linking strength that is contributed by the common neighbors; that is, $lc(i,j)$, as indicated in Eq. (5).

$$lc(i,j) = \sum_{p \in V_{ij}} w_{ipj} * \exp\left(-\left(\frac{w_{ipj} - maxw}{r * t + \eta}\right)\right), \tag{5}$$

where V_{ij} denotes the set of common neighbors between v_i and v_j; $w_{ipj} = \min(A_{ip}, A_{pj})$ is smaller between A_{ip} and A_{pj}; $maxw$ is the largest weight A; $r = maxw - \min(A)$ is the range of weights in A, where $\min(A)$ is the smallest weight in A; $t \in [0,1]$ controls the influence of common neighbors on the linking strength, and η is a small positive number to

prevent the denominator from being zero. If the two linking weights of their common friends are large, the contribution of this friend to $lc(i,j)$ is significant. In this paper, t is set to 0.4 based on empiricism (Xu et al., 2019).

In addition to $lc(i,j)$, another metric is used to measure the linking strength that is contributed by the common neighbors between nodes v_i and v_j. In particular, B_{ij} is the proportion of common neighbors among all neighbors based on the Jaccard similarity coefficient. It is calculated using Eq. (6), where N_i and N_j are the sets of neighbors of nodes v_i and v_j, respectively. Note that if $|N_i \cup N_j|$ is 0, then B_{ij} is set to 0 directly.

$$B_{ij} = \begin{cases} \dfrac{|N_i \cap N_j| + 1}{|N_i \cup N_j|}, & if \quad |N_i \cup N_j| \neq 0, \\ 0, & else \end{cases} \tag{6}$$

Consequently, the final linking strength $ls(i,j)$ between nodes v_i and v_j can be calculated by considering both $lc(i,j)$ and $B(ij)$, as indicated in Eq. (7).

$$ls(i,j) = \frac{(lc(i,j) + A_{ij} + B_{ij}) * (|V_{ij}| + 1)}{\min(I_i, I_j)}, \tag{7}$$

where $|V_{ij}|$ is the number of common neighbors between v_i and v_j, and the multiplier $(|V_{ij}| + 1)$ in the numerator helps to ensure that $ls(i,j)$ is larger if v_i and v_j have more common neighbors, as they will have more opportunities to interact. In the denominator, I_i and I_j are the sums of the weights of the outward links of nodes v_i and v_j, respectively, when G is a directed graph. If G is an undirected graph, I_i and I_j are the sums of the weights of the links of nodes v_i and v_j. The term $\min(I_i, I_j)$ is primarily used to adjust the value of $ls(i,j)$ and further increases its reliability. This is mainly because if two nodes have many common friends, despite the fact that they both have a large circle of friends (they are "social butterflies"), it cannot be deduced that they will become close friends in the future. In this case, $ls(i,j)$ should be smaller given a larger $\min(I_i, I_j)$.

Based on Eq. (7), the linking weight matrix can be transformed into a linking strength matrix. Finally, the distance matrix can be obtained using Eq. (8).

$$dist(i,j) = \frac{1}{ls(i,j) + \epsilon} \tag{8}$$

where $dist(i,j)$ represents the distance between node v_i and node v_j, and ϵ is a small positive real number in case the denominator is zero. According to Eqs. (7) and (8), if two nodes have no direct link and no common neighbors between them, their linking strength will be zero, and thus, the distance between them will be $1/\epsilon$. If a node is isolated, its linking strength to any other node is zero and the distance to any other node should be assigned $1/\epsilon$ directly. Following the calculation using Eq. (8), the linking strength matrix is converted into the distance matrix. The algorithm flow of the distance function is presented in Algorithm 1.

Algorithm 1: Distance Function

Input: The adjacent matrix $A_{N\times N}$, parameter t, the number of nodes N
Output: The distance matrix $D_{N\times N}$

$D_{N\times N} \leftarrow zero(N)$
for $i \leftarrow 1$ to N **do**
  for $j \leftarrow 1$ to N **do**
    if v_i *or* v_j is an isolated node **then**
      $d_{ij} \leftarrow \frac{1}{\epsilon}$
      $d_{ji} \leftarrow d_{ij}$
      continue
    else
      $V_{ij} \leftarrow$ *common* nodes of v_i and v_j
      for $v_p \in V_{ij}$ **do**
        $w_{ipj} \leftarrow \min(A_{ip}, A_{jp})$
        $lc(i,j) = \sum_{p\in V_{ij}} w_{ipj} * \exp(-(\frac{w_{ipj} - maxw}{r*t+\eta}))$
      $B_{ij} = \begin{cases} \frac{|N_i \cap N_j| + 1}{|N_i \cup N_j|}, & if \quad |N_i \cup N_j| \neq 0, \\ 0, & else \end{cases}$
      $ls(i,j) = \frac{(lc(i,j) + A_{ij} + B_{ij}) * (|V_{ij}| + 1)}{min(I_i, I_j)}$
      $d_{ij} = \frac{1}{ls(i,j)+\epsilon}$
      $d_{ji} \leftarrow d_{ij}$
return D

3.2. Adaptive local density calculation

Given the converted distance matrix, every node in the social network is assigned a local density attribute ρ. The local density is calculated using the Gaussian kernel function and only considers the nearest K neighbors (Xu et al., 2019). A simple method for using the Gaussian kernel function to calculate the local density is presented in Eq. (3), where d_c is the cut-off distance. The cut-off distance d_c is usually a predetermined parameter provided by the user. However, different d_c values will produce results with highly varying quality. The performance of the DPC-based clustering algorithm is significantly affected by these parameters (Du et al., 2016). Therefore, before calculating the local density, it is necessary to calculate d_c for different networks adaptively (Yaohui et al., 2017), as indicated in Eq. (9):

$$d_c = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\theta_i^K - \mu^K)^2} \tag{9}$$

where $N = |V|$ denotes the total number of nodes in the social network, θ_i^K is the distance between node v_i and its Kth nearest neighbor, and μ^K is the mean value of all nodes, which is defined as follows:

$$\mu^K = \frac{1}{N}\sum_{i=1}^{N}\theta_i^K \tag{10}$$

$$K = \max(\lfloor \frac{1}{|V|}\sum_{v_i\in V} degree(v_i) \rfloor, 1) \tag{11}$$

K is set to the floor integer of the average degree of the entire social network unless the average degree of the entire social network is less than 1, see Eq. (11). If the average degree of the entire social network is less than 1, K is set to be 1. After the calculation of the cut-off distance, the local density of node v_i can be obtained using Eq. (12), where KNN_i is the set of the K-nearest neighbors of node v_i:

$$\rho_i = \sum_{j\in KNN_i} exp\left(-\left(\frac{dist(i,j)}{d_c}\right)^2\right). \tag{12}$$

It can be observed from Eq. (9) that d_c is the standard deviation of the distance of node v_i to its Kth nearest neighbor. Thus, the dynamic calculation of d_c for different networks makes the clustering algorithm more adaptive, and the robustness on complex datasets is further enhanced.

3.3. Separation distance calculation

ODPI must also calculate the separation distance δ_i for every node v_i in the social network. ODPI adopts Eq. (4) of the DPC for the calculation.

3.4. Adaptive selection of cluster centers with iterative partition strategy

ODPI adopts the recursion k-means clustering method to select cluster centers. Prior to the subsequent procedures, the local density ρ_i and separation distance δ_i of each node should be normalized to the same range, as expressed by the following equations, where $\rho = [\rho_1, \rho_2, \ldots, \rho_{|V|}]$, $\delta = [\delta_1, \delta_2, \ldots, \delta_{|V|}]$.

$$\rho_i^* = \frac{\rho_i - \min(\rho)}{\max(\rho) - \min(\rho)} \tag{13}$$

$$\delta_i^* = \frac{\delta - \min(\delta)}{\max(\delta) - \min(\delta)} \tag{14}$$

Note that nodes with both a small local density ρ and separation distance δ will have no opportunity to be selected as cluster centers (Xu et al., 2019). Therefore, we discard nodes whose ρ and δ are both below a predefined threshold (i.e., the average value of the top 80% of nodes). Outlier nodes in the community are also discarded in this step. Thus, the set of remaining nodes is denoted as V_c. The product of ρ^* and δ^* for the nodes in V_c is calculated using Eq. (15):

$$\gamma_i = \rho_i^* * \delta_i^*. \tag{15}$$

Subsequently, the k-means clustering algorithm can be applied to V_c according to the γ values of the nodes obtained from Eq. (15). In our study, we set $k = 2$, which indicates that the k-means clustering algorithm will separate the nodes into two groups, where one group, known as the H-group, includes nodes with relatively high γ values, whereas the other group, known as the L-group, contains nodes with relatively small γ values. Specifically, the first step of the k-means clustering process yields the H-group and L-group, and the nodes in the H-group are selected as cluster centers and added to the empty set Centers. However, cluster centers may remain in the L-group, and therefore, nodes in the L-group will be assigned to set V_c and the k-means clustering process will be repeated; that is, it will be applied again to the new V_c according to the γ values, and a new H-group and L-group will be generated. Thereafter, a criterion is utilized to determine whether the nodes in the new H-group are selected as cluster centers, which involves comparing the average γ value of V_c, and the average of the highest γ value in the L-group and lowest γ value in the H-group. This is formulated as follows:

$$\frac{1}{|V_c|}\sum_{v_i \in V_c} \gamma_i < \frac{1}{2}\left(\max_{v_j \in L-group} \gamma_j + \min_{v_k \in H-group} \gamma_k\right). \tag{16}$$

If the former is no less than the latter, no more nodes are selected as cluster centers and the recursion process ends; otherwise, the nodes in the H-group are also selected as cluster centers and added into the set $Centers$; the nodes in the L-group are assigned to set V_c, and the recursion process continues.

The recursion k-means clustering process is repeated until it meets the ending criterion, and the nodes in the set $Centers$ are considered as cluster centers of the social network. The flow of the algorithm for the adaptive selection of cluster centers is presented in Algorithm 2.

Algorithm 2: Cluster Center Selection

Input: Local density vector ρ={ρ_1,ρ_2,...,ρ_N}, separation distance vector δ={δ_1,δ_2,...,δ_N}
Output: The set of cluster centers C

```
C ← ∅
V_c ← ∅
for i ← 1 to N do
    ρ_i^* = (ρ_i − min(ρ)) / (max(ρ) − min(ρ))
    δ_i^* = (δ_i − min(δ)) / (max(δ) − min(δ))
Filtering out nodes with relatively small density or separation distance;
for v_i ∈ V_c do
    γ_i = ρ_i^* * δ_i^*
Dividing V_c into H-group and L-group by K-means clustering algorithm;
C ← C ∪ H-group
V_c ← L-group
while true do
    if (1/|V_c|) Σ_{v_i∈V_c} γ_i < (1/2)(max_{v_j∈L-group} γ_j + min_{v_k∈H-group} γ_k) then
        Dividing V_c into H-group and L-group by K-means clustering algorithm;
        C ← C ∪ H-group
        V_c ← L-group
    else
        break
return C
```

3.5. Overlapping communities allocation

In this process, all nodes in the network are allocated to their communities. ODPI performs a two-step allocation procedure. In the first step, all non-center nodes find their closest cluster centers, the local density of which is higher, and are allocated to the corresponding communities. In the second step, the candidate overlapping nodes are assigned to the communities to which they belong.

Definition 1 (*Candidate Overlapping Node*)**.** For a node v_i, if v_i and all of its neighbors belong to the same community, v_i is not a candidate overlapping node; otherwise, it is a candidate overlapping node.

In the social network $G = (V, E)$, $Centers$ denotes the set of cluster center nodes, as explained in Section 3.4. Accordingly, $|Centers|$ communities exist. For ease of presentation, we use the cluster center node ID to represent the corresponding community label. For example, given a community with center node v_p, its community label is p. Let $cl = (cl_1, cl_2, \ldots, cl_{|V|})$ be the sequence of community labels of all nodes to which they belong. All elements in the set are initialized to 0. For node v_p in $Centers$, set cl_p to p. Thereafter, the first-step allocation starts. Specifically, for a node $v_i \notin Centers$, calculate its distance to all cluster center nodes with a local density that is higher than v_i in $Centers$, assign it to the corresponding community of the closest cluster center node with a higher local density, and finally, mark v_i as labeled.

Note that if node v_i has an equal distance (including $1/\epsilon$) to two or more cluster center nodes with a larger local density, its belonging community will not be determined and node v_i will be marked as unlabeled. Until all nodes are marked as labeled or unlabeled, an iterated breadth-first label spread process is conducted to allocate the unlabeled nodes to a community. In particular, set U is initialized and appended to all unlabeled nodes. Subsequently, the iteration process starts: given $U \neq \Phi$, for each node $v_i \in U$, if all its neighbors are unlabeled, v_i remains unlabeled and the next node in U is handled; if one of its neighbors is labeled, v_i can be allocated to a community in this iteration. For all labeled neighbors of v_i (denoted as N_i), the community set of nodes in N_i is C_{N_i}. If there is only one community in C_{N_i}, v_i is allocated to this community; if there are multiple communities in C_{N_i}, the total linking strength between v_i and all neighbors belonging to the same community is calculated. The community with the maximum total linking strength will ultimately acquire v_i. Once all nodes in U have been processed, the nodes that have been allocated to a community are removed from U and marked as labeled. In this case, a new iteration will start.

Following the first-step allocation, all nodes in the network are allocated to one community. Accordingly, all candidate overlapping nodes can be determined according to Definition 1. In the second step, all candidate overlapping nodes are considered and allocated to the overlapping community. Specifically, for each candidate overlapping node v_i, there is a probability of affiliating any community to which its neighbors belong. The affiliation degree for community c of node v_i, which is denoted as AD_i^c, is calculated using Eq. (17), where KNN_i is the set of the KNNs of node v_i, and $c \in \bigcup_{v_j \in KNN_i} cl_j$,

$$AD_i^c = \sum_{j \in KNN_i, cl_j = c} ls(i,j) \frac{\sum_{p \in KNN_j, cl_p = c} ls(j,p)}{\sum_{p \in KNN_j} ls(j,p)}. \tag{17}$$

If the affiliation degree for community c of node v_i is higher than the threshold σ when multiplying the affiliation degree for the community cl_i of node v_i; that is, $AD_i^c > \sigma \times AD_i^{cl_i}$, it will also be assigned to

Algorithm 3: Overlapping Community's Allocation

Input: The selected cluster centers $Centers$, the adjacent matrix $A_{N\times N}$ and threshold σ
Output: The allocation for all data nodes

$cl \leftarrow Set$ of cluster labels of all nodes after first-step allocation
$R \leftarrow zeros(N, |Centers|)$ matrix of final allocation results
for $i \leftarrow 1$ to $|Centers|$ **do**
  $v_p \leftarrow Centers_i$
  $cl_p \leftarrow i$
  $r_{pi} \leftarrow 1$
/* First-step allocation */
for $i \leftarrow 1$ to N **do**
  if $v_i \notin Centers$ **then**
    $cl_i \leftarrow cl_p$, where $d_{ip} = min_{j:\rho_j>\rho_i} d_{ij}$
/* Second-step allocation */
for $i \leftarrow 1$ to N **do**
  if v_i is not a candidate overlapping node **then**
    $C \leftarrow U_{j\in knn_j} cl_j$
    for *all* $c \in C$ **do**
      $AD_i^c = \sum_{j\in knn_j} ls(i,j) \frac{\sum_{p\in knn_j, cl_p=c} ls(j,p)}{\sum_{p\in knn_j} ls(j,p)}$
      if $AD_i^c > \sigma AD_i^{cl_i}$ **then**
        $r_{ic} \leftarrow 1$
return R

community c. The threshold σ represents the tendency of the algorithm to assign candidate overlapping nodes to overlapping communities, and the smaller this value is, the more likely it is to assign nodes to overlapping communities.

In fact, Eq. (17) is derived based on the intuition that people in social networks tend to be affected by their friends (Li et al., 2016). Furthermore, people tend to join a community with which their friends are more familiar. Hence, the calculation of the affiliation degree of node v_i considers not only the relationship between v_i and v_j (i.e., $ls(i,j)$), but also the familiarity of its friend v_j with the community; that is, if v_j has a more frequent relationship with its community, more influence of the community will spread to its friends, which is the rationale underlying the fraction part in Eq. (17). The algorithm flow for the overlapping community allocation process is described in Algorithm 3.

3.6. Complexity analysis

Sections 3.1 to 3.5 presented the core modules of ODPI. The overall algorithm flow of ODPI is provided in this subsection, see Algorithm 4. Suppose there exists a network with N nodes and C cluster centers. The time complexity of ODPI is determined by the following three aspects: (a) converting the adjacent matrix into a distance matrix ($O(N^2)$), (b) sorting γ ($O(NlogN)$) and selecting cluster centers ($O(C)$) for the worst case, and (c) assigning non-center nodes for two steps, where step 1 requires ($O(N)$) and step 2 requires ($O(NC)$). Thus, the overall time complexity of ODPI is ($O(N^2 + NlogN + C + N + NC)$).

4. Experimental setup

This section introduces the experimental setup, including the baseline algorithms for comparison, the evaluation metrics used to assess the performance, and the test datasets.

4.1. Baseline algorithms

Seven baseline algorithms were considered and compared with our proposed ODPI.

Algorithm 4: ODPI

Input: The adjacent matrix $A_{N\times N}$, the number of nodes N, parameters t, σ
Output: The allocation for all data nodes

$d_c \leftarrow Calculate$ the adaptive cutoff distance
$D \leftarrow calDisMatrix(A_{N\times N}, t)$
$O \leftarrow findOverlappingCandidates(A_{N\times N}, N)$
$\rho \leftarrow zeros(1, N)$
$\delta \leftarrow zeros(1, N)$
for $i \leftarrow 1$ to N **do**
  $\rho_i = \sum_{j\in knn_i} \exp(-(\frac{dist(i,j)}{d_c})^2)$
  $\delta_i = min_{j:\rho_j>\rho_i} dist(i,j), \exists j, \rho_j > \rho_i$
$C = selectCenters(P, \delta)$
$R = allocateNodes(C, A_{N\times N}, O, \sigma)$
return R

- CPM (Li et al., 2014) is an algorithm for overlapping community detection based on maximal cliques.
- LFM (Whang et al., 2016) is an overlapping community-detection algorithm that uses a seed expansion approach. A seed expands and constructs a community until local optimality is achieved.
- SLPA (Xie et al., 2011) is a fast algorithm for overlapping community detection in large-scale networks. It spreads labels based on dynamic interaction rules.
- DEMON (Coscia et al., 2014) is a hierarchical and overlapping community algorithm that uses a local-priority approach. Each node will vote for its surrounding communities and these constitute local communities; subsequently, the local communities will be combined to construct larger communities.
- MOSES (McDaid & Hurley, 2010) is a scalable algorithm based on a statistical model of community structures that can detect highly overlapping community structures.
- EADP (Xu et al., 2019) is also an extended adaptive density peaks clustering for overlapping community detection in social networks.
- GTA (Zhou et al., 2018) is a game-theoretic algorithm for detecting overlapping community structures in networks.

Note that the results that are generated by SLPA are nondeterministic. To ensure robustness, we used the average result of 10 rounds as the final SLPA result.

4.2. Evaluation metrics

The metrics used in this study to evaluate the performance of the algorithms are described as follows. Specifically, the metrics evaluated the performance from two different perspectives: the overlapped normalized mutual information (ONMI), Ω index, and F-score measure the accuracy (Xu et al., 2019), whereas Q_{ov} examines the structure.

- The ONMI (Lancichinetti et al., 2009) is an extension of a measure of similarity that was borrowed from information theory for overlapping communities. The built-in modular structure of real networks can be compared with that delivered by the algorithm.
- The Ω Index (Collins & Dent, 1988) is used to measure the degree of matching between the real partition and that discovered by the algorithm, where 0 denotes totally different and 1 denotes completely identical.
- The F-Score (Yang & Leskovec, 2013) is the average of the $F1$-score of the best-matching ground-truth community to each detected community and the $F1$-score of the best-matching detected community to each ground-truth community.

Table 2
Statistics of eight real-world networks.

Dataset	Type	Number of nodes	Number of edges	Number of communities	Standard deviation of weight	Real community structure
Dolphin	Unweighted	62	159	2	–	Known
Football	Unweighted	115	616	12	–	Known
Karate	Weighted	34	78	2	1.21	Known
Political	Unweighted	105	414	3	–	Known
Power	Unweighted	4941	6594	–	–	Unknown
NetScience	Weighted	1589	2742	–	0.43	Unknown
Cond-mat	Weighted	16726	47594	–	0.81	Unknown
Ca-GrQc	Unweighted	5242	14496	–	–	Unknown

However, these metrics can only assess the performance of algorithms on datasets with ground-truth communities. Alternatively, the community labels of datasets should be known. For datasets that do not hold ground-truth communities, the metric that assesses the structure will come into play.

- Q_{ov} (Nicosia et al., 2009) is an overlapping version of modularity that evaluates the quality of overlapping communities based on the definition of the community structure.

For all the above metrics, a higher value indicates a higher quality of the overlapping community-detection results and a better algorithm performance.

4.3. Test datasets

Experiments were conducted on both real-world and synthetic networks using the ODPI and baseline algorithms.

Real-World Networks. A total of eight real-world networks were adopted: four labeled networks and four unlabeled networks were used.

- Dolphin Network (Lusseau et al., 2003) is a social network of bottlenose dolphin populations that includes 62 dolphins.
- College Football Network (Girvan & Newman, 2002) is a representation of the schedule of US college football games. The nodes in the network represent teams and the edges represent games between two teams.
- Karate (Zachary, 1977) is a social network of relationships in a karate club. The nodes in the network represent individuals and the edges represent a consistent interaction between two individuals.
- The US Political Books Network (Krebs, 2008) is a social network analysis of the purchase patterns of political books from the sales data of Amazon.
- Power (Zhang et al., 2017) is an unweighted network that consists of 4941 power grids of western American states.
- Netscience (Zhang et al., 2017) is a weighted collaboration network consisting of 1589 scientists working on network theory and experiments.
- Cond-mat (Zachary, 1977) is a weighted collaboration network consisting of 16726 scientists working on preprints of concentrated material.
- Ca-GrQc (Leskovec et al., 2007) is a co-authorship network regarding cosmology. If two authors have co-published a paper, an edge exists between them.

All four metrics described in Section 4.2 were calculated for the four labeled datasets with ground-truth communities. Only the modularity metric (Q_{ov}) was calculated for the other four unlabeled datasets without ground-truth communities. The statistics for the datasets are listed in Table 2.

Synthetic Networks. Owing to the limited datasets of known real community structures and the small number of nodes in these datasets, it is difficult to test the performance of algorithms comprehensively from multiple angles. Therefore, it was necessary to generate synthetic datasets that conform to the general characteristics of a complex network. We used synthetic networks to test the algorithms on different datasets. The LFR (Lancichinetti et al., 2008) benchmark was adopted to generate synthetic networks in this study. LFR provides various parameters to control the characteristics of synthetic networks, including the number of nodes N, average degree k, maximum degree k_{max}, mixing parameter for the topology μ_t, mixing parameter for the weights μ_w, minimum for the community sizes C_{min}, maximum for the community sizes C_{max}, number of overlapping nodes O_n, and number of memberships of the overlapping nodes O_m.

In this study, the parameters of the LFR were set as follows:: $N \in \{1000, 3000, 5000\}$, $k = 10$, $C_{min} = 4\% * N$, $C_{max} = 6\% * N$, $\mu_t \in \{0.0, 0.1, 0.2, 0.3\}$, $\mu_w \in \{0.0, 0.1, 0.2, 0.3\}$, $O_n = 10\% * N$, $O_m \in [2, 8]$. We generated 10 network instances using identical parameter settings and ran the algorithm on the 10 network instances. The average results of the 10 rounds were considered as the final results of the algorithm under the parameter settings.

5. Experimental results and analysis

This section presents the experimental results and an in-depth analysis thereof, with the goal of answering the following three research questions:

- **(RQ1)** How does the adaptive cut-off distance d_c affect the performance of ODPI?
- **(RQ2)** Does our proposed ODPI outperform the baseline algorithms on the real-world networks?
- **(RQ3)** Does our proposed ODPI achieve better performance than the baseline algorithms on the synthetic networks?

The impact of the adaptive cut-off distance d_c (RQ1) is discussed in Section 5.1. The experimental results of the ODPI and baseline algorithms on the real-world (RQ2) and synthetic networks (RQ3) are analyzed in Sections 5.2 and 5.3, respectively.

5.1. Impacts of the adaptive cut-off distance (RQ1)

As noted previously, DPC requires the calculation of the local density of each node using the Gaussian kernel function presented in Eq. (3), which involves the cut-off distance d_c. In general, d_c is considered as a constant and is established empirically (Yaohui et al., 2017). However, an improper setting of d_c may lead to a disappointing performance for different social networks.

Therefore, in this section, we investigate the impact of the adaptive cut-off distance on the performance of our proposed ODPI. To this end, we tested the ODPI on six different networks, namely the four labeled networks and the two LFR synthetic networks, as described in Section 4.3. In particular, in LFR-1, $N = 1000$, $\mu_t = 0.3$, $\mu_w = 0.3$, $O_m = 2$; in LFR-2, $N = 1000$, $\mu_t = 0.0$, $\mu_w = 0.0$, $O_m = 2$. To verify the impact of the adaptive cut-off distance derived from Eq. (9), we compared the performance of the ODPI by empirically setting d_c in the

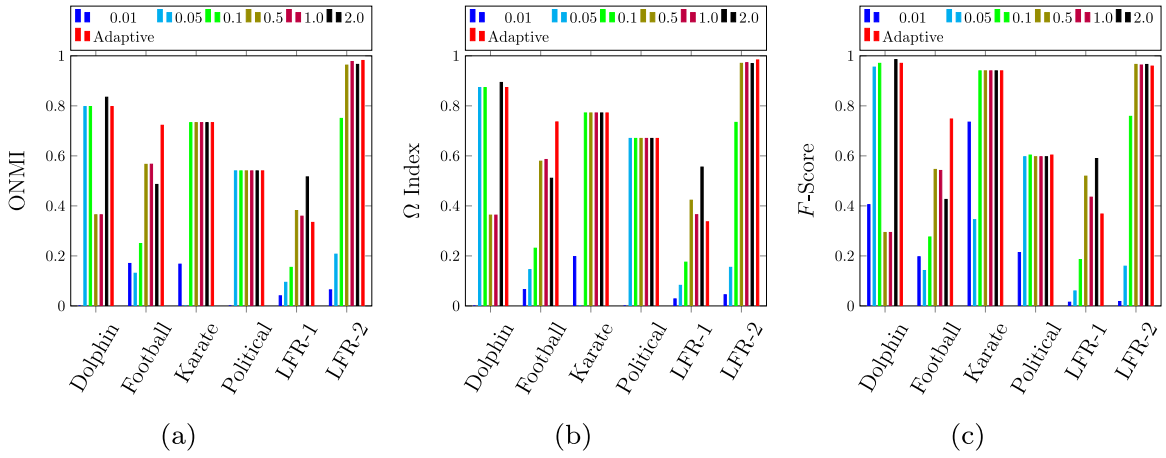

Fig. 1. Results of accuracy on different networks with different d_c.

range of 0.01, 0.05, 0.1, 0.5, 1, and 2. The accuracy results for these networks are depicted in Fig. 1.

It can be observed from Fig. 1 that different d_c settings led to different results in terms of the accuracy metrics overall, which suggests that ODPI should not treat d_c as a constant and set it empirically. Several interesting and detailed findings can be noted from the results: **(1)** In the Dolphin Network, although the best setting for d_c was 2, the results with the adaptive setting were quite close to the best results. **(2)** In the College Football Network, a large gap could be observed between the results from the adaptive setting and those from the empirical settings of d_c, which validates the effectiveness of the adaptive setting. **(3)** Both the Karate and US Political Books Networks were less sensitive to different settings of d_c, and the adaptive setting still achieved the best performance. **(4)** In LFR-1, there was no consistent setting of d_c to maximize the three accuracy metrics simultaneously. In contrast, the adaptive setting of d_c helped to ensure a nearly optimal result across all three metrics. **(5)** In LFR-2, the adaptive setting of d_c achieved the best results on both the ONMI and Ω index; regarding the F-score, although it was not the highest, the difference from the best result was quite minor.

In summary, the use of an adaptive setting of d_c remarkably increased the performance of ODPI. For certain networks, such as the College Football Network, the adaptive setting of d_c obviously led to the best performance. For other networks, leveraging the adaptive setting of d_c facilitated the best or near-best performance.

5.2. Results on real-world networks (RQ2)

The ODPI and baseline algorithms mentioned in Section 4.1 were evaluated on the real-world networks introduced in Section 4.3. Networks with ground-truth communities (i.e., those that contain nodes with known community labels) can be assessed using both the accuracy and structure metrics, whereas only the structural metric can be calculated for networks without ground-truth communities.

Table 3 lists the accuracy results of the algorithms for the four labeled networks. Several important observations can be made. First, ODPI achieved the best results on the Political Network in terms of the ONMI and Ω index, and the best results on the Karate Network in terms of the F-score. Second, MOSES obtained the best results on the College Football Network. This is mainly attributed to the fact that the cluster centers of the College Football Network do not have significant characteristics, whereas ODPI attempts to work on universal social networks and is not designed to detect such small variations. However, MOSES achieved unsatisfactory performance on the other networks, as it sacrifices its generality but is specialized for networks such as the College Football Network. In contrast, our ODPI, which is a generic algorithm, achieved acceptable results. For example, the ONMI of ODPI on the Football Network was only 4% lower than that of MOSES. Third, our ODPI achieved the second-best performance on the Dolphin Network, and it outperformed SLPA by 39.5% in terms of the ONMI. Finally, EADP achieved the best results on the Karate Network in terms of the ONMI and Ω index, and ODPI achieved the second-best scores on these two metrics. However, on the Political Network, which has a more complicated network structure, ODPI performed better than EADP regarding the ONMI and Ω index. In summary, ODPI can satisfactorily detect communities in real-world networks.

Table 4 presents the structural results of these algorithms for all real-world networks. It can be observed that ODPI achieved the best Q_{ov} value on Dolphin, Political, and Cond-mat; LFM performed the best on the Power and Ga-GrQc datasets; DEMON was superior on the Football dataset; SLPA achieved the best Q_{ov} value on the Karate dataset; and EADP achieved the best performance on the NetScience dataset. On the Football dataset, the Q_{ov} of ODPI was 22% lower than that of DEMON, which was the worst performance achieved by the ODPI. On the Power dataset, the Q_{ov} of the ODPI was 11% lower than that of LFM. On NetScience, ODPI was surpassed by SLPA by 13% with respect to Q_{ov}. For Ca-GrQc, the Q_{ov} of ODPI was 10% lower than that of LFM. The ODPI results were close to those of the first place. Alternatively, for datasets in which the ODPI did not achieve the best performance, the Q_{ov} values were still competitive in comparison with the baselines. In summary, according to the above analysis, the ODPI can operate effectively from the perspective of structural quality.

5.3. Results on synthetic networks (RQ3)

Experiments were conducted on the LFR synthetic networks. We first explore the sensitivity performance of the ODPI on the LFR synthetic networks with different construction parameters, and then present and discuss a performance comparison between ODPI and the baseline algorithms.

5.3.1. Sensitivity analysis

Performance of ODPI with different LFR settings of μ_w. Fig. 2 depicts the performance of ODPI with different μ_w settings, where μ_w was varied in $\{0.1, 0.2, 0.3\}$, O_m was varied in $[2, 8]$ in increments of 1, and $N = 1000$. Note that μ_w represents the mixing parameter for the weights of the LFR synthetic network and O_m is the number of memberships of the overlapping nodes. According to the figure, the ONMI, Ω index, and F-Score decreased with an increase in O_m. This is because O_m denotes the number of communities to which an overlapping node belongs. A larger O_m indicates a greater difficulty in discovering the correct communities for an overlapping node; therefore, the accuracy decreases as O_m increases. Similarly, as μ_w increased, the ONMI, Ω index, and F-Score decreased. More specifically, the performance of ODPI on the

Table 3
Results of accuracy on real-world labeled networks.

	CPM	LFM	SLPA	DEMON	MOSES	EADP	ODPI
ONMI							
Dolphin	0.3306	0.3074	0.6105	0.3895	0.2122	**1.0000**	0.8516
Football	0.2577	0.4148	0.5339	0.3102	**0.7897**	0.7299	0.7574
Karate	0.1744	0.1791	0.7096	0.2823	0.2273	**0.8372**	0.7329
Political	0.4288	0.2655	0.3987	0.2593	0.1960	0.5039	**0.5394**
Ω **Index**							
Dolphin	0.3451	0.3629	0.6995	0.2793	0.1464	**1.0000**	0.9143
Football	0.0568	0.4277	0.4136	0.0074	**0.9090**	0.7655	0.7357
Karate	0.0941	0.0727	0.7222	0.0819	0.1328	**0.8823**	0.7716
Political	0.5956	0.4773	0.6108	0.1819	0.3000	0.6671	**0.6698**
F-Score							
Dolphin	0.24	0.1216	0.4971	0.285	0.0873	**1.0000**	0.97
Football	0.2395	0.2260	0.4470	0.0514	**0.7562**	0.7122	0.7475
Karate	0.2977	0.1296	0.7533	0.1133	0.1837	0.9395	**0.94**
Political	0.39	0.0670	0.2848	0.205	0.0442	**0.8134**	0.6033

Table 4
Results of Q_{ov} on real-world networks.

	CPM	LFM	SLPA	DEMON	MOSES	EADP	ODPI	GTA
Dolphin	0.7395	0.6965	0.8792	0.7790	0.5604	0.7395	**0.9363**	0.6860
Football	0.8620	0.5884	0.7396	**0.8687**	0.6670	0.6954	0.6819	0.7000
Karate	0.8319	0.8479	**0.8863**	0.8690	0.3527	0.7530	0.8325	0.7470
Political	0.9025	0.7768	0.8854	0.8997	0.6317	0.8342	**0.9252**	0.8280
Power	0.1758	**0.9148**	0.8062	0.2008	0.1933	0.6536	0.8113	–
Netscience	0.8184	0.9393	0.9249	0.8552	0.8010	**0.9774**	0.8506	–
Cond-mat	0.7569	0.7705	0.7465	0.8195	0.6815	0.8202	**0.8841**	–
Ca-GrQc	0.7200	**0.8691**	0.8130	0.8145	0.6434	0.8173	0.7823	–

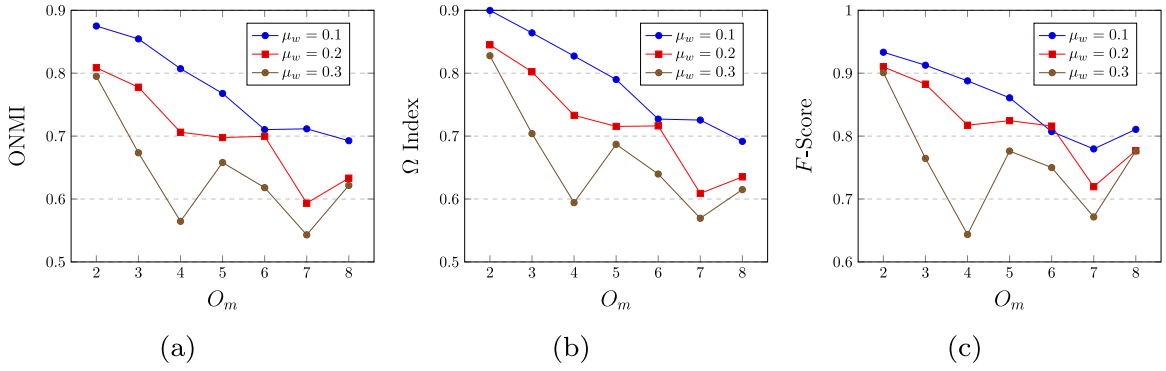

Fig. 2. Performance of ODPI with different μ_w.

network with $\mu_w = 0.2$ significantly exceeded that on the network with $\mu_w = 0.3$, whereas it was surpassed by that on the network with $\mu_w = 0.1$. This is mainly attributed to the fact that μ_w indicates the diversity of weights of the edges; a larger μ_w indicates that the network is more complicated and the community structure is more difficult to discover.

Performance of ODPI with different LFR settings of μ_t. The performance of ODPI with different settings of μ_t is illustrated in Fig. 3, where μ_t was varied in $\{0.1, 0.2, 0.3\}$, O_m was varied in $[2, 8]$ in increments of 1, and $N = 1000$. Note that μ_t represents the mixing parameter for the topology of the LFR synthetic networks. It can be observed from the figure that with the increase in O_m, the ONMI, Ω index, and F-score all decreased dramatically. The trends were similar but more obvious than the impact of μ_w, which suggests that the increase in the topological complexity makes the task more difficult to solve than the increase in the weight diversity.

Sensitivity to parameter σ. In Section 3.5, parameter σ was incorporated to determine whether a candidate overlapping node belongs to a community. If the affiliation degree for community c of node v_i, which is denoted as AD_i^c, is higher than σ, node v_i belongs to community c. In this section, the sensitivity of the ODPI to σ is analyzed.

The results are displayed in Table 5, where the LFR synthetic networks contain 1000 nodes, the number of overlapping nodes O_n varied in $\{50, 200, 400\}$; and the parameter σ varied in $\{0.01, 0.03, 0.05, 0.1, 0.5\}$. Note that O_f represents the number of overlapping nodes discovered by ODPI; O_f/O_n reflects the ability of ODPI to discover overlapping nodes; $O_f \& O_n$ is the number of common nodes in O_f and O_n, which yields the number of correct overlapping nodes discovered by ODPI; $\frac{O_f \& O_n}{O_n}$ is the accuracy of overlapping nodes detection, where a higher $\frac{O_f \& O_n}{O_n}$ means a better ability to detect correct overlapping nodes; and $\frac{O_f - (O_f \& O_n)}{O_n}$ is the proportion of incorrect detected overlapping nodes in real overlapping nodes, which reflects the error rate of ODPI in overlapping nodes detection.

It can be observed from Table 5 that with an increase in σ, the accuracy Of $\frac{O_f \& O_n}{O_n}$ of ODPI first increased, reached its peak, and finally, gradually decreased. In particular, with $\sigma = 0.01$, the accuracy $\frac{O_f \& O_n}{O_n}$ of ODPI was approximately 75%; the ODPI can achieve higher accuracy

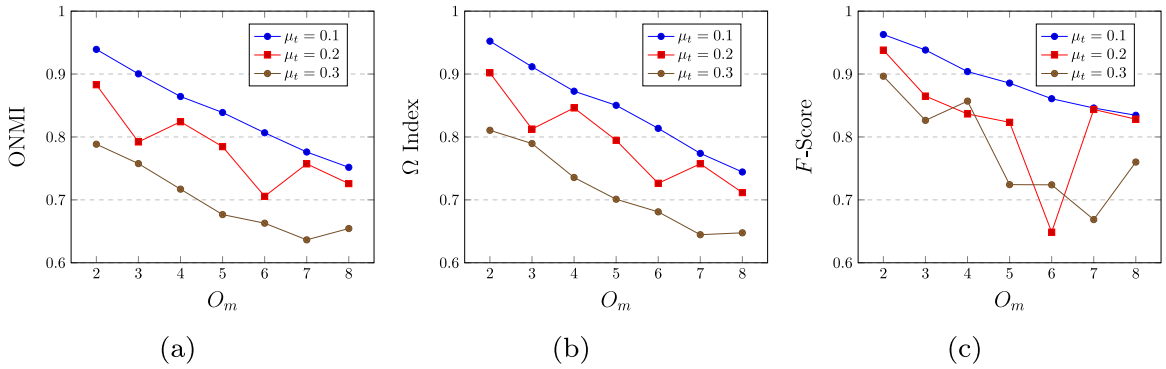

Fig. 3. Performance of ODPI with different μ_t.

Table 5
Influence of different σ on ODPI.

O_n	σ	O_f	O_f/O_n	$O_f \& O_n$	$\frac{O_f \& O_n}{O_n}$	$\frac{O_f - (O_f \& O_n)}{O_n}$
50	0.01	45	90%	39	78%	2%
200	0.01	160	80%	148	74%	6%
400	0.01	281	70.25%	281	70.25%	0%
50	0.03	46	92%	46	92%	0%
200	0.03	170	85%	170	85%	0%
400	0.03	318	79.50%	288	72%	7.50%
50	0.05	44	88%	42	84%	4%
200	0.05	151	75.50%	149	74.50%	1%
400	0.05	318	79.50%	290	72.50%	7%
50	0.1	34	68%	31	62%	6%
200	0.1	110	55%	100	50%	5%
400	0.1	220	55%	209	52.25%	2.75%
50	0.5	21	42%	21	42%	0%
200	0.5	94	47%	94	47%	0%
400	0.5	150	37.50%	150	37.50%	0%

with less actual overlapping nodes O_n. When $\sigma = 0.03$, the accuracy increased to approximately 85%, and when $\sigma = 0.05$, the accuracy was 75% again. When $\sigma = 0.1$, the accuracy was approximately 50%, and when $\sigma = 0.5$, it was reduced to approximately 45%. The experimental results demonstrate that $\sigma = 0.03$ was the best setting, and the performance of ODPI decreased with a further increase in σ. This is because a larger σ means a greater unwillingness to accept a node as a community member, which will decrease the number of overlapping nodes that are detected by ODPI. Moreover, the performance with $\sigma = 0.01$ was lower than that with $\sigma = 0.03$. The reason for this is that when σ is too small, more nodes that are not actual overlapping nodes will be treated as overlapping nodes by ODPI, which will increase the error rate $\frac{O_f - (O_f \& O_n)}{O_n}$.

5.3.2. Comparison of ODPI and baseline algorithms

Comparison about accuracy on different networks with varied μ_t. The experimental results in Section 5.3.1 reveal that an increase in μ_t will increase the complexity of the LFR synthetic networks. Therefore, four different LFR synthetic networks were constructed with parameter $\mu_t \in \{0.0, 0.1, 0.2, 0.3\}$, and the other parameters were set to default values. The ODPI and baseline algorithms were used on these networks, and the results of the accuracy are depicted in Fig. 4.

It can be observed that with the increase in μ_t, the LFR synthetic networks became increasingly complicated and the task of detecting the community became more difficult. However, compared with the baseline algorithms, ODPI could maintain its high performance in all situations. In contrast, several baseline algorithms exhibited an obvious performance decrease as μ_t increased, indicating that these algorithms cannot handle high-complexity networks. For example, CPM could reach first place in the ONMI when $\mu_t = 0.0$, but as μ_t increased to 0.3, its ONMI decreased to almost 0. This clearly indicates that CPM cannot handle high-complexity networks because it is based on the k-clique, and it is difficult to discover a suitable k-clique for complex networks. Moreover, the ONMI of SLPA slightly exceeded that of ODPI when $\mu_t < 0.1$, whereas its ONMI was significantly surpassed by that of ODPI with $\mu_t = 0.3$.

Comparison about accuracy on different networks with varied O_m. We further explored the performance of the ODPI and baseline algorithms on different LFR networks with different settings of μ_t and μ_w, and with a varied O_m. Specifically, we first generated two groups of LFR synthetic networks with $\mu_t = 0.2$ and $\mu_t = 0.3$; O_m varied in the range of $[2, 8]$ in increments of 1, and the other parameters were default values. Each group contained seven networks. The ODPI and baseline algorithms were employed on these networks. The results are plotted in Figs. 5 ($\mu_t = 0.2$) and Fig. 6 ($\mu_t = 0.3$).

Regardless of μ_t being 0.2 or 0.3, the ONMI, Ω index, and F-score of ODPI were almost all higher than those of the baseline algorithms. When $\mu_t = 0.2$, the performance of ODPI was better than that of EADP, and when μ_t increased to 0.3, the performances of ODPI and EADP were similar. Although the performances of ODPI and EADP became similar with the increasing complexity of the networks, ODPI generally performed better. Regarding CPM, which is based on k-clique theory, it is difficult to fulfill the k-clique structure for complex networks. Consequently, the performance of CPM was abnormally poor. In conclusion, the results demonstrate that the ODPI can handle more complex networks and achieve better performance than the baseline algorithms.

Moreover, LFR networks with different μ_w values were studied. Specifically, two groups of LFR synthetic networks with $\mu_w = 0.2$ and $\mu_w = 0.3$ were generated, while O_m was varied from 2 to 8 in increments of 1. Figs. 7 and 8 present the accuracy results of the ODPI and baseline algorithms. It can be observed that when $\mu_w = 0.2$ and $\mu_w = 0.3$,

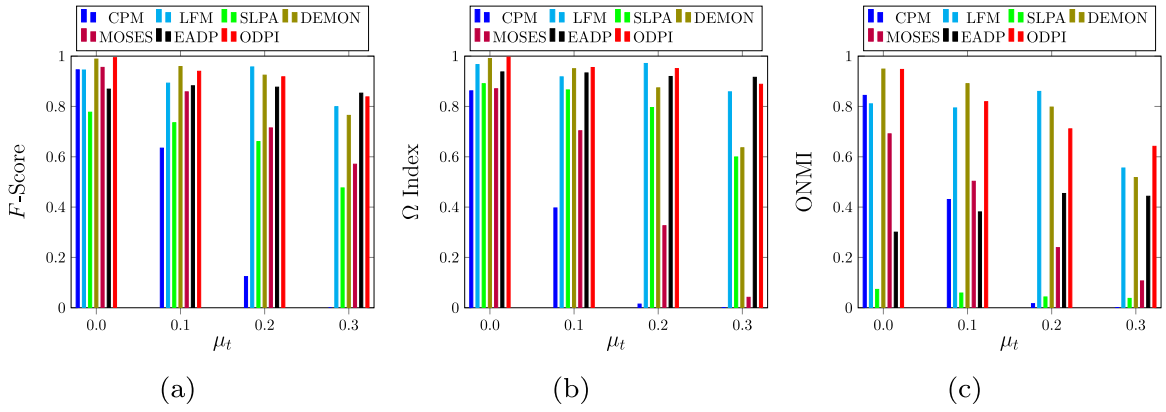

Fig. 4. Results of accuracy on LFR synthetic networks.

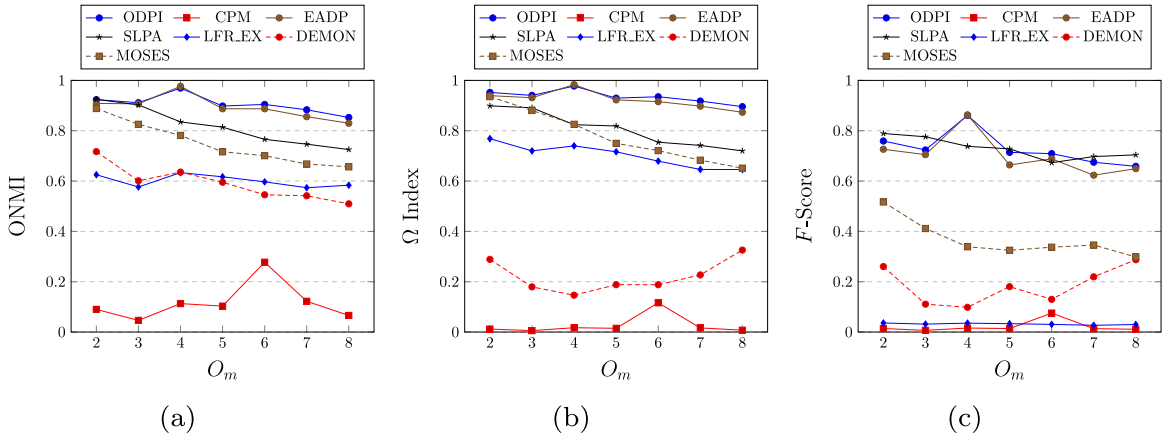

Fig. 5. Results of accuracy on LFR networks with $\mu_t = 0.2$.

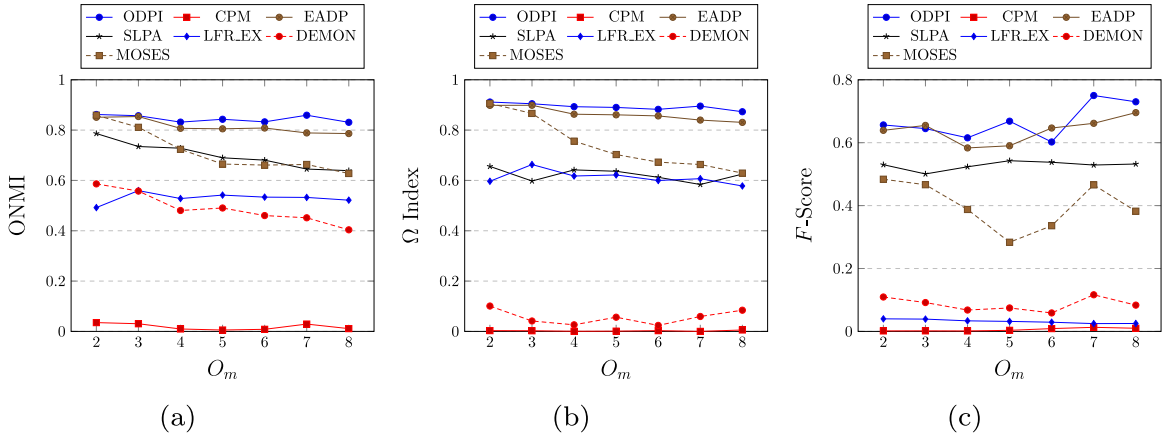

Fig. 6. Results of accuracy on LFR networks with $\mu_t = 0.3$.

SLPA obtained an F-score that was close to that of ODPI, but its ONMI and Ω index were substantially lower than those of ODPI. EADP and ODPI were similar; however, ODPI generally performed slightly better than EADP. It can be concluded that ODPI is an effective overlapping community-detection algorithm and offers the advantage of handling complex networks.

5.4. Efficiency analysis

To demonstrate the efficiency of the algorithms, the computational times for each algorithm on different networks are discussed in this section. The algorithms were run on a 64-bit Ubuntu machine with an Intel Xeon Gold 6248R CPU @ 3.00 GHz and 54 GB memory. To make the computational time results more reliable, the result for each algorithm was the average running time of 10 implementations.

First, we ran the ODPI on 10 different LFR synthetic networks with different N, which varied in $[1000, 10000]$ in increments of 1000. Fig. 9 presents the results. As discussed in Section 3.6, the time complexity of ODPI is almost $O(N^2)$, and the trend of the curve in Fig. 9 coincides with the results of the time complexity analysis. For the case of 10000 nodes, the computational time was 1699 s, which is also an acceptable result.

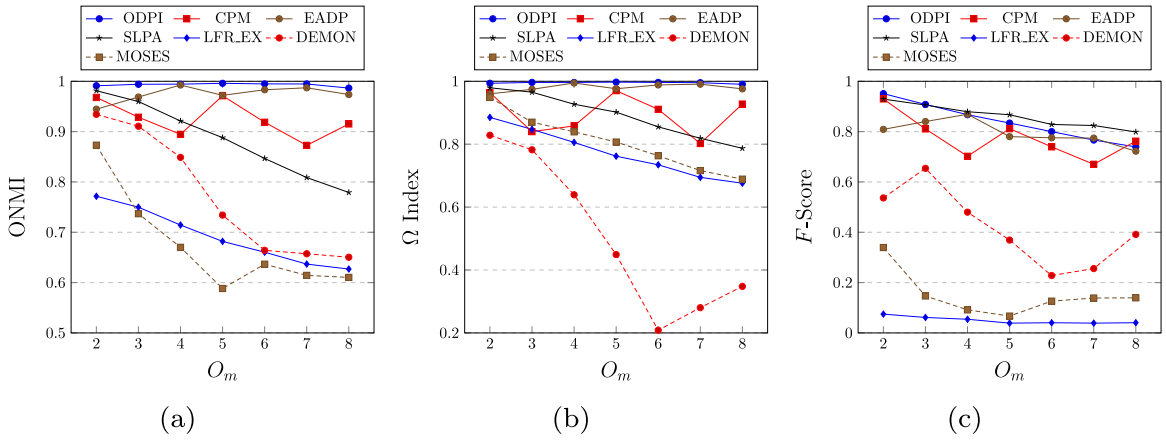

Fig. 7. Results of accuracy on LFR networks with $\mu_w = 0.2$.

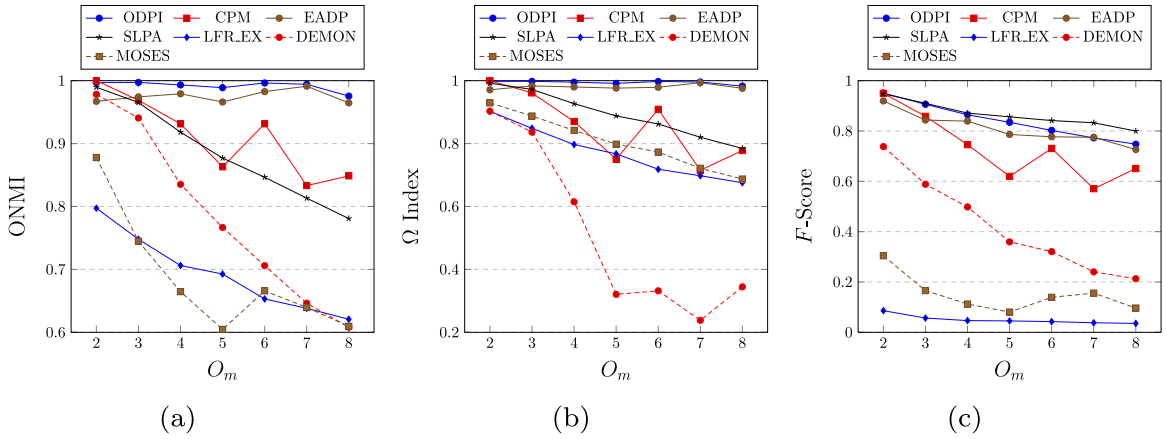

Fig. 8. Results of accuracy on LFR networks with $\mu_w = 0.3$.

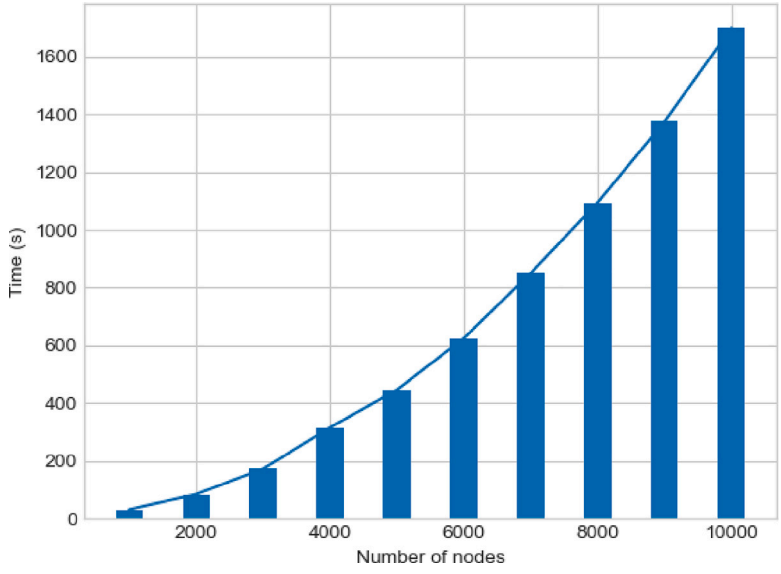

Fig. 9. Running time on synthetic networks of different sizes.

Table 6 lists the running times of different algorithms on real-world networks. Apart from GCE and MOSES, which are implemented in C++, all other algorithms, including ODPI, are implemented in Python. The first four real networks (Dolphin, Football, Karate, and Political) are relatively small, whereas the other four (Power, NetScience, Cond-mat, and Ca-GrQc) are larger, among which Cond-mat is the largest with 16726 nodes.

As both EADP and ODPI are DPC-based algorithms, it can be observed that their time consumption was similar. The performance of ODPI did not differ substantially from that of the other algorithms on

Table 6
Running time (seconds) on real-world networks.

Datasets	CPM	LFM	SLPA	GCE	DEMON	MOSES	EADP	ODPI
Dolphins	0.006	0.014	1.084	0.035	0.059	8.924	0.042	0.076
Football	0.022	0.034	3.995	0.049	0.233	9.521	0.185	0.236
Karate	0.006	0.012	0.555	0.037	0.033	8.001	0.025	0.712
Political	0.019	0.026	3.646	0.049	0.183	10.686	0.146	0.235
Power	0.063	10.055	52.833	0.324	1.655	13.937	161.898	163.951
NetScience	0.046	0.373	18.413	0.189	1.256	16.077	19.271	19.006
Cond-mat	12.426	13.196	328.430	2.973	42.451	89.657	2544.977	2491.475
Ca-GrQc	3.475	11.335	95.284	0.575	14.742	41.700	249.41	248.280

the small datasets, whereas the time consumption of ODPI increased significantly on the large datasets. It is worth noting that although the MOSES algorithm has excellent time performance on large-scale datasets, it consumes more time on small-scale datasets than other algorithms. Although the running speed of ODPI is not a significant advantage, the actual running time is within the acceptable range. It can be concluded that the ODPI achieves acceptable results in terms of performance, and it can be argued that this is an acceptable trade-off with speed.

6. Conclusions and future works

We have proposed a novel ODPI algorithm for detecting overlapping communities in social networks. In particular, we found that different cut-off distances d_c may lead to highly varying algorithm performance for different social networks. To this end, our proposed ODPI can calculate d_c adaptively based on different network scales and features, which indicates that our d_c can be adapted to different network typologies. We tested the performance of our ODPI algorithm on different complex networks by setting different values of d_c, and the final results demonstrate the effectiveness of our adaptive d_c. The experimental results on both real social networks and synthetic networks in comparison with six state-of-the-art baselines revealed that ODPI can achieve better performance on networks with complex weight distributions and complex structure distributions. This confirms the effectiveness of the ODPI in detecting overlapping communities.

Overall, our current work has focused on avoiding manual operations to enhance the accuracy of the ODPI further. Future works can be conducted based on the following aspects:

(1) Adaptive searching for σ. ODPI introduces σ into the calculation of candidate overlapping nodes and the allocation process of overlapping nodes (Section 3.5). In this study, this value was set based on experience, which often affects the algorithm performance. Therefore, the relationship between the parameter and network can be considered in future research, and adaptive parameters can be designed according to the characteristics of different networks.
(2) Optimization of ODPI for efficient operation. At present, ODPI must transform the adjacent matrix into a distance matrix. The time complexity of the procedure is $O(N^2)$, which is not practical for complex networks with large-scale nodes in real-world application scenarios. Therefore, parallel algorithms can be designed to accelerate the calculation process.
(3) Consideration of node attribute information. ODPI considers only the link relationships between nodes in the network. However, in an actual complex network, node information, which is probably critical for community detection, may not be reflected in the edge link. Therefore, the node attribute information should be considered to determine the community structure more accurately.

CRediT authorship contribution statement

Yunyun Niu: Conceptualization, Methodology. **Detian Kong:** Methodology. **Ligang Liu:** Software, Writing – original draft. **Rong Wen:** Writing – review & editing. **Jianhua Xiao:** Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [grant numbers 62172373, 61872325, 62072258, 61772290]; the Fundamental Research Funds for the Central Universities, China [grant number 2652019028]; the Special Foundation for Philosophy and Social Science Laboratories of Ministry of Education of China [grant number ZX20220103].

References

Ahn, Y.-Y., Bagrow, J. P., & Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *Nature*, *466*, 761–764.
Badie, R., Aleahmad, A., Asadpour, M., & Rahgozar, M. (2013). An efficient agent-based algorithm for overlapping community detection using nodes' closeness. *Physica A*, *392*, 5231–5247.
Bai, X., Yang, P., & Shi, X. (2017). An overlapping community detection algorithm based on density peaks. *Neurocomputing*, *226*, 7–15.
Bhat, S. Y., & Abulaish, M. (2014). Hoctracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, *27*, 1013–1019.
Bu, Z., Li, H.-J., Zhang, C., Cao, J., Li, A., & Shi, Y. (2019). Graph k-means based on leader identification, dynamic game, and opinion dynamics. *IEEE Transactions on Knowledge and Data Engineering*, *32*, 1348–1361.
Chen, W., Liu, Z., Sun, X., & Wang, Y. (2010). A game-theoretic framework to identify overlapping communities in social networks. *Data Mining and Knowledge Discovery*, *21*, 224–240.
Collins, L. M., & Dent, C. W. (1988). Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. *Multivariate Behavioral Research*, *23*, 231–242.
Coscia, M., Rossetti, G., Giannotti, F., & Pedreschi, D. (2014). Uncovering hierarchical and overlapping communities with a local-first approach. *ACM Transactions on Knowledge Discovery from Data*, *9*, 1–27.
Derényi, I., Palla, G., & Vicsek, T. (2005). Clique percolation in random networks. *Physical Review Letters*, *94*, Article 160202.
Du, M., Ding, S., & Jia, H. (2016). Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowledge-Based Systems*, *99*, 135–145.
Du, M., Ding, S., Xu, X., & Xue, Y. (2018). Density peaks clustering using geodesic distances. *International Journal of Machine Learning and Cybernetics*, *9*, 1335–1349.
Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*, 7821–7826.
Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics*, *12*, Article 103018.
Gregory, S. (2011). Fuzzy overlapping communities in networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2011*, Article P02017.
Jokar, E., Mosleh, M., & Kheyrandish, M. (2021). Overlapping community detection in complex networks using fuzzy theory, balanced link density, and label propagation. *Expert Systems*, Article e12921.
Krebs, V. (2008). http://www.orgnet.com/divided.html.
Lancichinetti, A., Fortunato, S., & Kertész, J. (2009). Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, *11*, Article 033015.
Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, *78*, Article 046110.

Lancichinetti, A., Radicchi, F., Ramasco, J. J., & Fortunato, S. (2011). Finding statistically significant communities in networks. *PLoS One*, *6*, Article e18961.

Leskovec, J., Kleinberg, J., & Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *1*(2–es).

Li, Y., Liu, C., Zhao, M., Li, R., Xiao, H., Wang, K., & Zhang, J. (2016). Multi-topic tracking model for dynamic social network. *Physica A*, *454*, 51–65.

Li, J., Wang, X., & Cui, Y. (2014). Uncovering the overlapping community structure of complex networks by maximal cliques. *Physica A*, *415*, 398–406.

Liang, Z., & Chen, P. (2016). Delta-density based clustering with a divide-and-conquer strategy: 3dc clustering. *Pattern Recognition Letters*, *73*, 52–59.

Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., & Dawson, S. M. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, *54*, 396–405.

McDaid, A., & Hurley, N. (2010). Detecting highly overlapping communities with model-based overlapping seed expansion. In *2010 international conference on advances in social networks analysis and mining* (pp. 112–119). IEEE.

Moscato, V., Picariello, A., & Sperli, G. (2019). Community detection based on game theory. *Engineering Applications of Artificial Intelligence*, *85*, 773–782.

Nicosia, V., Mangioni, G., Carchiolo, V., & Malgeri, M. (2009). Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics: Theory and Experiment*, *2009*, Article P03024.

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435*, 814–818.

Prat-Pérez, A., Dominguez-Sal, D., & Larriba-Pey, J.-L. (2014). High quality, scalable and parallel community detection for large real graphs. In *Proceedings of the 23rd international conference on world wide web* (pp. 225–236).

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, *76*, Article 036106.

Rees, B. S., & Gallagher, K. B. (2013). Detecting overlapping communities in complex networks using swarm intelligence for multi-threaded label propagation. In *Complex networks* (pp. 111–119). Springer.

Ren, W., Yan, G., Liao, X., & Xiao, L. (2009). Simple probabilistic algorithm for detecting community structure. *Physical Review E*, *79*, Article 036111.

Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, *344*, 1492–1496.

Sun, L., Ye, T., Sun, J., Duan, X., & Luo, Y. (2021). Density-peak-based overlapping community detection algorithm. *IEEE Transactions on Computational Social Systems*.

Wang, J., Ren, J., Li, M., & Wu, F.-X. (2012). Identification of hierarchical and overlapping functional modules in ppi networks. *IEEE Transactions on Nanobioscience*, *11*, 386–393.

Wang, C., Tang, W., Sun, B., Fang, J., & Wang, Y. (2015). Review on community detection algorithms in social networks. In *2015 IEEE international conference on progress in informatics and computing* (pp. 551–555). IEEE.

Whang, J. J., Gleich, D. F., & Dhillon, I. S. (2013). Overlapping community detection using seed set expansion. In *Proceedings of the 22nd ACM international conference on information & knowledge management* (pp. 2099–2108).

Whang, J. J., Gleich, D. F., & Dhillon, I. S. (2016). Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Transactions on Knowledge and Data Engineering*, *28*, 1272–1284.

Wu, Z.-H., Lin, Y.-F., Gregory, S., Wan, H.-Y., & Tian, S.-F. (2012). Balanced multi-label propagation for overlapping community detection in social networks. *Journal of Computer Science and Technology*, *27*, 468–479.

Wu, X., & Zhang, C. (2015). Multi-label propagation for overlapping community detection based on contribution degree. *Journal of the China Society for Scientific and Technical Information*.

Xie, J., Gao, H., Xie, W., Liu, X., & Grant, P. W. (2016). Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors. *Information Sciences*, *354*, 19–40.

Xie, J., Kelley, S., & Szymanski, B. K. (2013). Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm Computing Surveys (Csur)*, *45*, 1–35.

Xie, J., Szymanski, B. K., & Liu, X. (2011). Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *2011 IEEE 11th international conference on data mining workshops* (pp. 344–349). IEEE.

Xu, M., Li, Y., Li, R., Zou, F., & Gu, X. (2019). Eadp: An extended adaptive density peaks clustering for overlapping community detection in social networks. *Neurocomputing*, *337*, 287–302.

Yang, J., & Leskovec, J. (2013). Overlapping community detection at scale: a non-negative matrix factorization approach. In *Proceedings of the 6th ACM international conference on web search and data mining* (pp. 587–596).

Yaohui, L., Zhengming, M., & Fang, Y. (2017). Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy. *Knowledge-Based Systems*, *133*, 208–220.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*, 452–473.

Zhang, L., Pan, H., Su, Y., Zhang, X., & Niu, Y. (2017). A mixed representation-based multiobjective evolutionary algorithm for overlapping community detection. *IEEE Transactions on Cybernetics*, *47*, 2703–2716.

Zhang, S., Wang, R.-S., & Zhang, X.-S. (2007). Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A*, *374*, 483–490.

Zhou, X., Zhao, X., Liu, Y., & Sun, G. (2018). A game theoretic algorithm to detect overlapping community structure in networks. *Physics Letters. A*, *382*, 872–879.