Neurocomputing 337 (2019) 287–302

Contents lists available at ScienceDirect

Neurocomputing

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

EADP: An extended adaptive density peaks clustering for overlapping community detection in social networks

Mingli Xu, Yuhua Li*, Ruixuan Li, Fuhao Zou, Xiwu Gu

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

ARTICLE INFO

Article history:
Received 13 May 2018
Revised 12 December 2018
Accepted 25 January 2019
Available online 5 February 2019

Communicated by Dr Jiajun Bu

Keywords:
Overlapping community
Adaptive density peaks
Real social networks

ABSTRACT

Overlapping community detection plays an important role in studying social networks. The existing overlapping community detection methods seldom perform well on networks with complex weight distribution. Density peaks clustering (DPC) is capable of finding communities with arbitrary shape efficiently and accurately. However, DPC fails to be applied to overlapping community detection directly. In this paper, we propose an extended adaptive density peaks clustering for overlapping community detection, called EADP. To handle both weighted and unweighted social networks, EADP takes weights into consideration and incorporates a novel distance function based on common nodes to measure the distance between nodes. Moreover, unlike DPC choosing cluster centers by hand, EADP adopts a linear fitting based strategy to choose cluster centers adaptively. Experiments on real-world social networks and synthetic networks show that EADP is an effective overlapping community detection algorithm. Compared with the state-of-the-art methods, EADP performs better on those networks with complex weight distribution.

1. Introduction

It is very common that people in social networks join various communities. Community detection, which aims at finding out the community structure hidden in a network, becomes a very popular research direction in social network mining. There have been numerous algorithms proposed to find disjoint community structure, such as DBSCAN [1], spectral clustering [2] and so on. However, people in social networks are often characterized by multiple community memberships. For instance, a boy could be a member of music club and basketball club simultaneously as he may be interested in both music and basketball. For this reason, researches about overlapping community detection have drawn more and more attention. The clique percolation method (CPM) [3] is one of the representative algorithms. To our best knowledge, many of the existing overlapping community detection methods do not consider the linking weights, and methods that take weights into account seldom perform well on networks with dramatic weights variation.

Density peak clustering (DPC) [4], published in Science Journal, introduces a novel approach to find community structures. DPC is based on the assumption that cluster centers are surrounded by lower density points and the distance between cluster centers is relatively large. DPC is capable of detecting communities with arbitrary shape efficiently and accurately. However, communities in social networks are often overlapped, DPC only detects non-overlapping communities. Moreover, DPC requires a distance matrix as its input. However, as point out in [5], for most of the social networks, the input would be the adjacent matrix. Therefore, we need to construct distance matrix ourselves by making use of the linking information in adjacent matrix. What's more, DPC requires choosing cluster centers by hand, which will be not practical in the case of large networks.

Since DPC being proposed, many scholars have made various improvements on it. However, just the same as DPC, most of the improvements require the distance between nodes as input. Thus, we claim that those methods could not be used in social networks directly.

In this paper, we propose an extended adaptive density peaks clustering for overlapping community detection, called EADP. EADP incorporates a novel distance function based on common nodes to handle social networks directly. Furthermore, the distance function is designed to take linking weights into consideration such that EADP could be used in weighted networks. Our main contributions can be summarized as follows:

- Extend DPC to detect overlapping communities. To do this, we adopt a two-step allocation strategy. For the first step, we perform nodes assignation with the same strategy as DPC. Then, we calculate community memberships for each

* Corresponding author
E-mail address: idcliyuhua@hust.edu.cn (Y. Li).

https://doi.org/10.1016/j.neucom.2019.01.074
0925-2312/

non-center node, and add it to clusters where it has higher community memberships than it has in its initial community.
- Come up with a common nodes based distance function with linking weights considered. Through this distance function, EADP could be used directly in social networks. Moreover, when calculating distance between nodes, linking weights are considered. Therefore, EADP is suitable for both weighted and unweighted social networks.
- Utilize a linear fitting based strategy to select cluster centers adaptively. We choose cluster centers by performing linear fitting within the difference vector of γ, which is the product of local density ρ and separation distance δ.

The rest of this paper will be organized as follows. In Section 2, we discuss the related work. In Section 3, we introduce the preliminary of DPC. Section 4 presents the proposed algorithm EADP. Section 5 gives an introduction of experimental setup. Section 6 shows and analyses the experimental results on both real and synthetic datasets. Finally, we conclude this paper and give the vision of our future work in Section 7.

2. Related work

Our work focuses on overlapping community detection based on density peaks, so we will discuss traditional overlapping community detection methods and density peaks based methods, respectively.

2.1. Traditional overlapping community detection

Traditional overlapping community detection algorithms can be roughly divided into following categories: label propagation based methods, local expansion and optimization based methods, graph partitioning based methods, and link partitioning methods.

Label propagation based methods. This kind of methods define the propagation rules of nodes labels in the network. COPRA [6] and SLPA [7] are two typical algorithms. For COPRA, each node updates its belonging coefficients by averaging the coefficients of its neighbors at each time step in a synchronous manner. SLPA is a speaker-listener based information propagation algorithm, which spreads labels between nodes according to pairwise interaction rules. For label propagation methods tend to generate monster communities, Sattaria et al. [8] come up with a spreading activation based label propagation algorithm to solve this problem. Fan at al. [9] propose an algorithm taking the method of label propagation based on local max degree and neighborhood overlap for initial network partitioning. Label-based propagation methods have the advantages of simplicity, efficiency and rapidity, but the community structure found by them are of great uncertainty [10].

Local expansion and optimization based methods. Most of them rely on a local benefit function that characterizes the quality of a densely connected group of nodes. Lancichinetti et al. [11] propose LFM algorithm, which randomly selects a node as an initial seed, then starts from this seed node and expands outward to construct a community. The process is terminated until the fitness function reaches the local optimum. Statistical model based algorithm MOSES [12] and greedy group expansion algorithm GCE [13] are capable of detecting highly overlapping community structure. Liu et al. [14] combine the original network with the corresponding annealing network [15] into an integrated network, based on which UEOC algorithm is developed. Recently, Liu et al. [16] propose a hierarchical agglomerative clustering algorithm based on the local optimal expansion cohesion idea. Yu et al. [17] propose SEOCD, a seeds extension overlapping community detection algorithm using random walk strategy to find the seed communities with tight structures. The drawback of these methods is the clustering result is unstable for the seeds are randomly selected.

Graph partitioning methods. One category of graph partitioning methods is using non-negative matrix factorization. NMF model is first used to overlapping community detection by Zhang et al. [18], but it requires to know the number of communities in advance. Later, many scholars have proposed some improved methods, such as SNMF [19]. Besides, NMFOSC [20] handled this problem by feature matrix preprocessing and ranking optimization, so that it can discover network structure with unknown community number. Recently, an adaptive algorithm is proposed incorporating bayesian explanation in factorizing process to obtain the most appropriate count of communities [21]. Li et al. [22] propose an algorithm for overlapping community discovery based on semi-supervised matrix factorization and random walk. Another category is spectral clustering, which makes use of the singular vectors of the Laplacian matrix associated with a graph. Traditional spectral clustering methods focus on global quantities and may not be sensitive to very local information. Unlike traditional spectral techniques, Li et al. [23] and He et al. [24] prioritize and learn about local region information. The drawback is that the optimization is done in the entire eigenspace. Li et al. [25] propose Lemon (Local Expansion via Minimum One Norm), which does optimization in a partial invariant subspace constructed by the Krylov subspace. Other approaches find overlapping communities by optimizing a defined objective, such as in Ref. [26] using a measure related to connectivity.

Link partitioning methods. Different from those algorithms clustering nodes, link partitioning methods cluster links. If links that connect to a same node are clustered into different communities, then the node are assigned to multiple clusters. Ahn et al. [27] propose an edge-based overlapped community discovery method, which maps the original network into edge graph and repeatedly merges two edges with the highest similarity. Evans et al. [28–30] map the network as a weighted edge graph, and then cluster edges using non-overlapping community detection methods to find overlapping communities. Shi et al. [31] apply genetic operation to cluster on links and design an effective encoding schema. Sun et al. [32] propose a link-based label propagation algorithm LinkLPA, which performs label propagation with links. Link partitioning methods rely on a vague community definition, so it does not guarantee higher cluster quality than those node-based clustering methods [33].

Besides those kinds of methods above, Bhat et al. [34] propose a density-based approach called HOCTracker, which defines a dual-layered distance function with common nodes considered. However, when calculating the distance between two nodes, HOCTracker treats the direct and un-direct linking weights contributed by their common nodes equally. It is not reasonable because direct links should be more important and be given greater attention. Sheikholeslami et al. [35] put forward an ehonet-tensor based algorithm called DC-EgoTen, which adopts a divide-and-conquer strategy to detect community structure. Wang et al. [36] propose a model called local gravitation among data points, which adopts the idea that there exist distinct differences between the local resultant force (including magnitudes and directions) of the data points close to the cluster centers and at the boundary of the clusters. Another way is to use ensemble methods, which summarizes clustering solutions obtained by different clustering algorithms to unified solution, such as [37,38]. The advantage is that ensemble clustering methods give more reliable results, and the disadvantage is that time would be longer.

2.2. Density peaks based community detection

Another new community detection method is based on density peaks, called DPC [4], which is come up with by Rodriguez and Laio. Due to its high accuracy and efficiency, it has drawn widely

attention since proposed. Based on DPC, scholars have made various improvements which could be categorized into four classes: change the way of calculating local density, choose cluster centers adaptively, extend DPC to discover overlapping communities, and define a new distance metric.

The most common way to improve the calculation of local densities is introducing the idea of k nearest neighbors (KNN). When calculating the local density of a node, only its k nearest neighbors are considered instead of all the nodes. The advantage is that the local structure of nodes is considered. As far as we know, most improvements have adopted this approach, such as [5,39,40], etc. However, there are still some exceptions. Wang and Xu [41] estimates local densities with the idea of nonparametric multivariate kernel density estimation. Inspired by FJP algorithm [42,43] uses fuzzy neighborhood relation to compute the local density.

As for the selection of cluster centers, there are several different improvements. The 3DC algorithm [44], motivated by the divide-and-conquer strategy and the density-reachable concept in the DBSCAN framework, could automatically find the correct number of clusters in a recursive way. In [45], all points whose delta values are larger than the cutoff distance are chosen as initial cluster centers. At the last step, all density reachable clusters will be aggregated. Because of the aggregation step, cluster centers do not need to be found out very accurately. Li and Tang [46] proposes a comparative density peaks algorithm which adopts a new quantity as the alternative of δ to help select cluster centers. DenPEHC [47] adopts an linear fitting method to select cluster centers automatically. Unlike DenPEHC fitting all points on γ curve from back to front, we only fit several typical points within the difference vector of γ. Another difference is that we remove data points with small ρ or δ before starting the fitting process. Therefore, our algorithm EADP will be much faster than DenPEHC.

In terms of the improvements for overlapping community detection, the most common approach is to use a fuzzy method, such as [5] and [40]. First, a membership vector will be calculated. Then each node will be assigned to clusters where its memberships exceed to a pre-specified threshold. The main disadvantage is that the performance will be affected by the threshold and it is not easy to set it appropriately for most of the time.

For the improvements to distance calculation, Du et al. [48] introduce the idea of the geodesic distances to make DPC group data effectively with multi-manifold structures. Bai et al. [5] propose an overlapping community detection algorithm called OCDDP, which calculates the distance between a pair of nodes through their intermediate nodes. However, the distance function used by OCDDP are of high time complexity. Meanwhile, the parameter α representing the number of intermediate nodes is not easy to set properly. The biggest difference between OCDDP and our work is that OCDDP requires human intervention to select cluster centers. For most of the time, people could select cluster centers accurately when having prior knowledge, but it is not practical when handling large networks.

Besides the improvements above, Jiang et al. [49] propose DFC algorithm. Inspired by DPC and DBSCAN, DFC improves its capability in the case of datasets with varying sizes, varying densities and irregular shapes by merging density fragments according to the network structural similarity.

As far as we know, many of the existing work do not extend the DPC to social networks for overlapping community detection and automatic selection of cluster centers at the same time. Therefore, this is the research point of our work.

3. Preliminary

Our algorithm EADP is based on density peak proposed in [50], thus we give a brief introduction to its main idea here.

Density peaks clustering (DPC) is based on the assumption that cluster centers are characterized by higher density than their neighbors and by a relatively large distance from those nodes with higher density. It has two quantities: the local density ρ and the separation distance δ.

There are ways to calculate the local density ρ. One way is using cutoff distance shown in Eq. (1),

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

$$\chi(x) = \begin{cases} 1 & x \le 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where ρ_i is the local density of node v_i, $dist(i, j)$ is the distance between v_i and v_j, and d_c is the cutoff distance. The density of node v_i calculated with this method is equal to the number of nodes whose distance to v_i is less than d_c.

Another way utilizing Gaussian kernel is shown in Eq. (3). The closer node v_j is to v_i, the greater the contribution v_j will do to the density of v_i.

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

The separation distance δ_i is calculated according to Eq. (4). δ_i is the minimum distance between node v_i and any other nodes with higher density.

$$\delta_i = \min_{j:\rho_j > \rho_i} dist(i,j) \tag{4}$$

After calculating ρ and δ for each data node, the decision graph will be drawn with ρ as horizontal axis and δ as vertical axis. By observing this graph, data nodes with relatively large ρ and large δ will be chosen as cluster centers.

At last, each remaining data node will be assigned to the same cluster as the node with higher density and closest distance to it.

4. EADP: an extended adaptive density peaks clustering for overlapping community detection in social networks

To extend DPC to be capable of finding overlapping communities in both weighted and unweighted social networks, we mainly do following work: (1) Define a distance function to measure the distance between nodes in social networks. (2) Introduce the idea of linear fitting to choose cluster centers adaptively. (3) Change the allocation strategy adopted by DPC such that the communities could be overlapped.

Next, we will discuss our work one by one, which are the core parts of our method EADP. Finally, the overall picture of EADP will be presented.

4.1. Distance function

For social networks, the input is often the adjacent matrix, from which the linking information including linking relationships and linking weights could be obtained. For an unweighted network, the elements of the adjacent matrix are 1 or 0. 1 means there is a link and 0 means not. For a weighted network, the elements are nonnegative real values representing linking weights. To introduce density peaks methods into social networks, the adjacent matrix should be transferred to distance matrix. One simple idea is to set the reciprocals of linking weights as distances. However, for social networks are often sparse, the adjacent matrix will contain too many zero values. Therefore, there will be too many infinite values in distance matrix. In this paper, we define a novel distance function generalized for both weighted and unweighted networks.

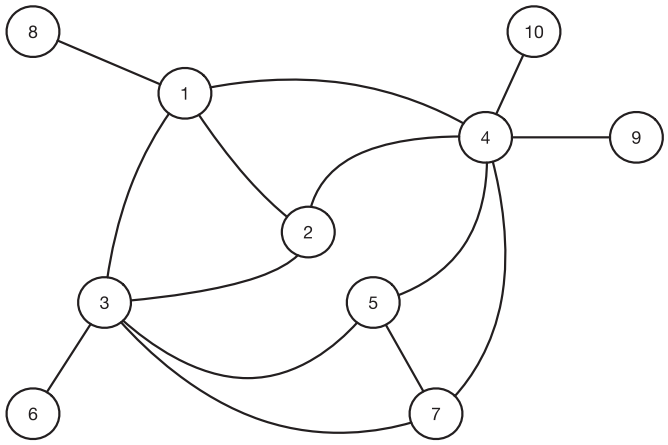

Fig. 1. A sample network.

Considering a social network as a graph $G = (V, E_w)$, where V is the set of nodes representing users and E_w is the set of edges among users. For a graph with n nodes, an $n*n$ adjacent matrix A is constructed, where A_{ij} is the linking weight between node v_i and node v_j.

Inspired by HOCTracker [34], we take common nodes into consideration when calculating the distance between a pair of nodes, instead of these two nodes only. There are two reasons. One is that two people may become friends someday through their mutual friends, even though they do not have any explicit interactions now. The other reason is that if two people share significant number of mutual friends, then they will have more chances to interact with each other. In other words, they will be closer because of their mutual friends. Here, we take the network in Fig. 1 as an example, node 3 and node 4 have no explicit interactions, but their distance should not be set to infinite because their mutual friends node 5 and node 7 will be the bridge to link them. That is, a link between them will exist someday. When calculating the distance between node 1 and node 2, we take their mutual friends node 3 and node 4 into consideration because node 3 and node 4 will increase the probability of their interactions.

To measure the distance between v_i and v_j, we first calculate $cc(i, j)$, the contribution that the common nodes do to their link strength. $cc(i, j)$ is defined as Eq. (5), where V_{ij} is set of the common nodes between v_i and v_j, w_{ipj} is the minimum of A_{ip} and A_{jp}, $maxw$ is the maximum weight in the whole network, r is the range of weights, $t \in [0, 1]$ is a parameter controlling how significantly a common node v_p influences the link strength between node v_i and v_j, η is a small positive value to avoid zero denominator. The greater the w_{ipj}, the greater the link strength contribution. It means that the more frequent their interactions with common node v_p, the greater the contribution v_p will do to their link strength.

$$cc(i, j) = \sum_{p \in V_{ij}} w_{ipj} * \exp\left(-\left(\frac{w_{ipj} - maxw}{r * t + \eta}\right)^2)\right) \tag{5}$$

Then, we calculate link strength between v_i and v_j with Eq. (6). The link strength is denoted as $ls(i, j)$. V_{ij} is the set of common nodes to which both v_i and v_j have links in the network, and $|V_{ij}|$ is the number of their common nodes. I_i and I_j are the total weight of outgoing links of v_i and v_j respectively. Link strength between v_i and v_j is not only dependent on their common nodes' contribution $cc(i, j)$, but also affected by how many common nodes they have. As explained above, more common nodes mean more chances for interactions and a closer distance. Note that large direct link weight does not always mean a close distance. For example, there are two people with frequent interactions, but they also have frequent or even more frequent interactions with their other friends, then we can't simply judge that these two people are close enough because of their large direct linking weight. Instead, we need to consider how much their direct interaction accounts for their total interactions. The greater the fraction is, the greater the link strength.

$$ls(i, j) = \frac{(cc(i, j) + A_{ij}) * (|V_{ij}| + 1)}{\min(I_i, I_j)} \tag{6}$$

Finally, the distance between node v_i and node v_j is calculated by Eq. (7), where ϵ is a non-negative real number to make sure the denominator is non-zero.

$$dist(i, j) = \frac{1}{ls(i, j) + \epsilon} \tag{7}$$

It should be noted that the distances of isolated nodes to any other nodes will be directly set to the largest distance $\frac{1}{\epsilon}$ without calculation. With the distance function, the distance matrix D could be obtained through adjacent matrix A. The process is presented as Algorithm 1.

Algorithm 1 CalDisMx.

Input: The adjacent matrix A
Output: The distance matrix D

```
for i = 1 to n − 1 do
    for j = i + 1 to n do
        if v_i or v_j is an isolated node then
            D_ij ← 1/ε;
            D_ji ← D_ij;
        else
            V_ij ← common nodes of v_i and v_j;
            for all v_p ∈ V_ij do
                w_ipj ← min(A_ip, A_jp);
                Calculate cc(i, j) via Equation (5);
            end for
            Calculate ls(i, j) via Equation (6);
            Calculate dist(i, j) via Equation (7);
            D_ij ← dist(i, j);
            D_ji ← D_ij;
        end if
    end for
end for
return D;
```

4.2. Local density calculation with KNN

Our algorithm EADP utilizes Gaussian kernel to calculate local density ρ. In contrast to DPC taking all nodes into account, EADP only considers k nearest neighbors. With the idea of KNN, the unique local structure of nodes will be considered. The local density of node v_i is defined as Eq. (8), where knn_i is a set containing the k nearest neighbors of node v_i. In this paper, we set k to the average degree of the network.

$$\rho_i = \sum_{j \in knn_i} \exp\left(-\left(\frac{dist(i, j)}{dc}\right)^2\right) \tag{8}$$

4.3. Adaptively selection of cluster centers

EADP adopts linear fitting method to select cluster centers automatically based on the difference vector of $\boldsymbol{\gamma}$.

Before calculating $\boldsymbol{\gamma}$, we need to do min-max normalization to ensure that $\boldsymbol{\rho}$ and $\boldsymbol{\delta}$ are in the same range, shown as Eqs. (9)

and (10).

$$\rho_i^* = \frac{\rho_i - \min(\boldsymbol{\rho})}{\max(\boldsymbol{\rho}) - \min(\boldsymbol{\rho})} \tag{9}$$

$$\delta_i^* = \frac{\delta_i - \min(\boldsymbol{\delta})}{\max(\boldsymbol{\delta}) - \min(\boldsymbol{\delta})} \tag{10}$$

It should be noted that we only keep those nodes whose ρ and δ is relatively large, the rest nodes will not be considered when choosing cluster centers. It's obvious that nodes with small ρ and δ are not competitive enough to be cluster centers. We assume that the nodes whose ρ and δ are both below the average corresponding value of the 80% nodes with smaller ρ and δ almost have no chance to be selected as cluster centers.

Next, for each node v_i, we calculate γ_i using Eq. (11).

$$\gamma_i = \rho_i^* \delta_i^* \tag{11}$$

Then, sort the elements in $\boldsymbol{\gamma}$ in ascending order. Let $\boldsymbol{\gamma^s}$ be the set containing the sorted $\boldsymbol{\gamma}$ elements and let $\boldsymbol{Ind}$ be the set containing the corresponding indices. Base on $\boldsymbol{\gamma^s}$, we calculate the difference vector $\boldsymbol{d\gamma}$ as Eq. (12) shows.

$$\boldsymbol{d\gamma} = \{d\gamma_i | d\gamma_i = \gamma_{i+1}^s - \gamma_i^s\} \tag{12}$$

Obviously, cluster centers are those nodes with anomalously large γ. Because large ρ and large δ will lead to large γ. Thus, we adopt the following strategy to select cluster centers.

(1) Initially, get the index of the maximum element in $\boldsymbol{d\gamma}$, denoted as idx. The corresponding maximal element is $d\gamma_{idx}$.
(2) Next, we do linear fitting to predict the value of $d\gamma_{idx}$ with the elements ahead, shown as Eq. (13).

$$\hat{d\gamma}_{idx} = a * idx + b, \tag{13}$$

a and b are two variables to make the fitting process reach a minimum mean square error (MSE), and $\hat{d\gamma}_{idx}$ is an estimation of $d\gamma_{idx}$. We denote the difference between the real value $d\gamma_{idx}$ and the estimated value $\hat{d\gamma}_{idx}$ as $\Delta\gamma$. $\Delta\gamma$ is calculated with Eq. (14).

$$\Delta\gamma = \hat{d\gamma}_{idx} - d\gamma_{idx}. \tag{14}$$

(3) We judge if the condition shown in Eq. (15) is met or not. The threshold is set to $2\Delta\gamma$ because we find that cluster centers could be well separated from non-center nodes when taking this value.

$$d\gamma_{idxn} > 2\Delta\gamma \tag{15}$$

The process will be terminated if the condition is not satisfied for the reason that there will not be nodes with significant γ any more. If the condition is satisfied, then nodes in set S will be added to the set of cluster centers denoted by $\boldsymbol{Clusters}$, where $S = \{v_i, i \in [Ind_{idx+1}, Ind_{|\boldsymbol{Ind}|}]\}$.
Eq. (15) indicates that only those nodes which have significant increase in $\boldsymbol{d\gamma}$ curve could be chosen as cluster centers. And the nodes behind them in $\boldsymbol{d\gamma}$ will be also chosen as cluster centers because of larger γ.
(4) repeat (1) until $idx < 3$ for we cannot do linear fitting with only one node.

When the linear fitting process terminates, the nodes in set $\boldsymbol{Clusters}$ are the selected cluster centers. To make it better understood, we give an example in Example 1.

Example 1. Selecting cluster centers with EADP using Karate dataset [51].

Karate is a weighted interaction network between 34 members of a Karate club, and there are two clusters in total. Fig. 2 shows the network topology, where red and green nodes form different communities. The thicker the edge, the bigger the weight. After filtering out those nodes whose ρ and δ is small, there are 10 nodes left that have chances to be cluster centers. For the 10 remaining nodes, we calculate their γ values, sort in ascending order, and then get the difference vector $\boldsymbol{d\gamma}$. $\boldsymbol{\gamma}$, $\boldsymbol{\gamma^s}$ and $\boldsymbol{d\gamma}$ are shown in Table 1.

In Fig. 3(a), the elements of $\boldsymbol{\gamma^s}$ are plotted. By observing Fig. 3(a), it is reasonable to choose the points within the red rectangle as cluster centers. As we can see, γ_9^s and γ_{10}^s are abnormally large compared to other points. Because $\gamma_9^s = \gamma_1$, and $\gamma_{10}^s = \gamma_8$, so the nodes corresponding to the points inside the red rectangle are node 1 and node 24.

Then, we will show how to choose cluster centers. Fig. 3(b) shows the elements of $\boldsymbol{d\gamma}$. First, we select the maximum of $\boldsymbol{d\gamma}$ in the entire dataset, that is $d\gamma_8$. Then we do linear fitting to predict its value using points with subscript in [1,7]. Because the real value is greatly larger than the predicted value, then nodes whose index in $\{Ind_i | i \geq 8\}$ will be chosen as centers. That is, node 1 and node 24 are selected. To make it clear, see

$$d\gamma_8 = \gamma_9^s - \gamma_8^s = \gamma_1 - \gamma_4,$$

node 1 should be chosen as a cluster center for its large γ. Any nodes with larger γ value than node 1 should be chosen as cluster center too. Because

$$\gamma_{10}^s > \gamma_9^s,$$

and

$$\gamma_{10}^s = \gamma_8,$$

the node is 24 should be chosen too.

Repeat this process, until we found the one $d\gamma_7$. Because there is no big difference between the real value and the predicted one, the process is terminated. As a result, node 1 and node 24 are selected as cluster centers, which is consistent with our observation in Fig. 3(a). As shown in Fig. 2, node 1 and 24 are surrounded by several nodes, and there are distant to each other. So it is also reasonable to consider them as centers.

The process is formally presented as Algorithm 2.

4.4. Overlapping allocation of remaining nodes

EADP follows a two-step procedure to assign the non-center nodes to multiple communities. In the first step, we assign each remaining node v_i to the same cluster as the node with higher density and minimum distance to it. Let $cl = (cl_1, cl_2, \ldots, cl_N)$ be the set containing the cluster labels of all nodes after the first-step assignation, where cl_i denotes the cluster label of v_i.

In the second step, we will reassign non-center nodes so that they could belong to more than one community. First of all, we give the definition of enveloped nodes.

Definition 1 (enveloped node)**.** Given a node v_i, v_i is an enveloped node if v_i and all its neighbors are in the same cluster.

For each non-center node, we first determine if it is an enveloped node. If not, we allocate it to its potential clusters with the following strategy.

(1) For each non-enveloped node v_i, we calculate its membership p_i^c for cluster c using Eq. (16), where c is one of the clusters to which at least one of its k-nearest neighbors belong (that is, $c \in \cup_{j \in knn_i} cl_i$).

$$p_i^c = \sum_{j \in knn_i, cl_j = c} ls(i,j) \frac{\sum_{p \in knn_j, cl_p = c} ls(j,p)}{\sum_{p \in knn_j} ls(j,p)} \tag{16}$$

As validated in literature, people in social networks are often influenced by their friends [52]. Accordingly, people tend

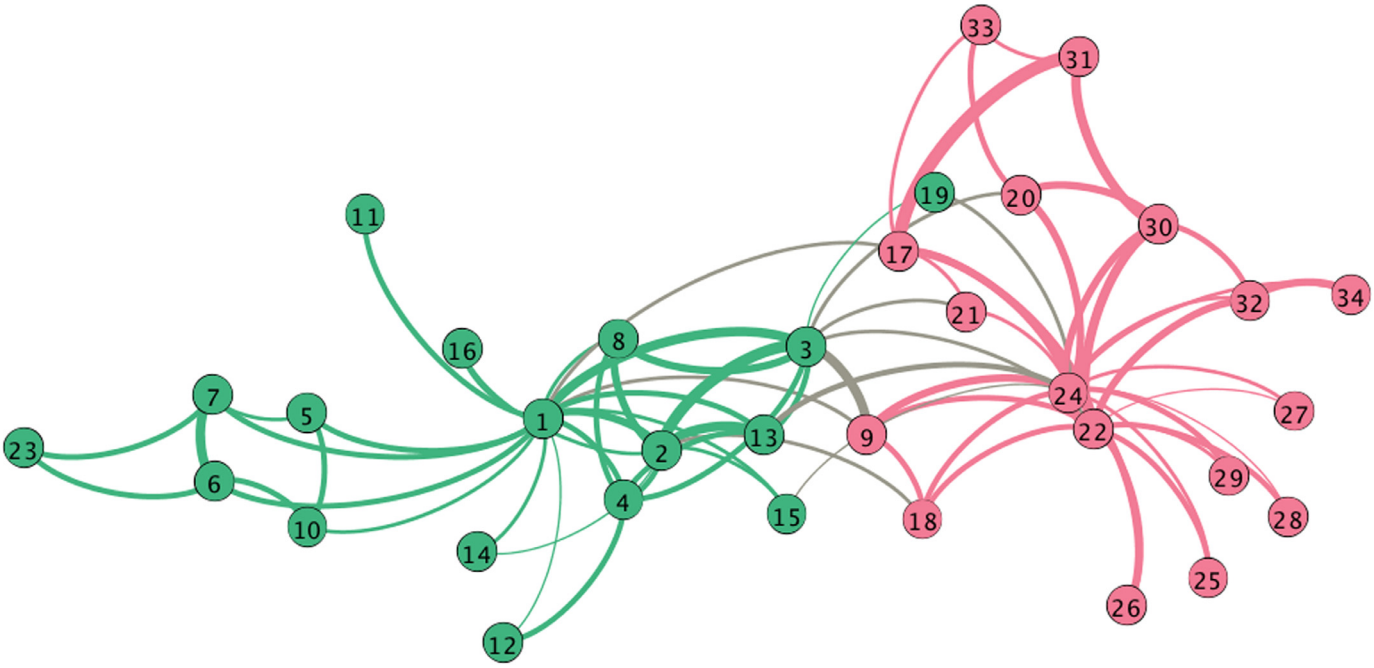

Fig. 2. Network topology of Karate dataset.

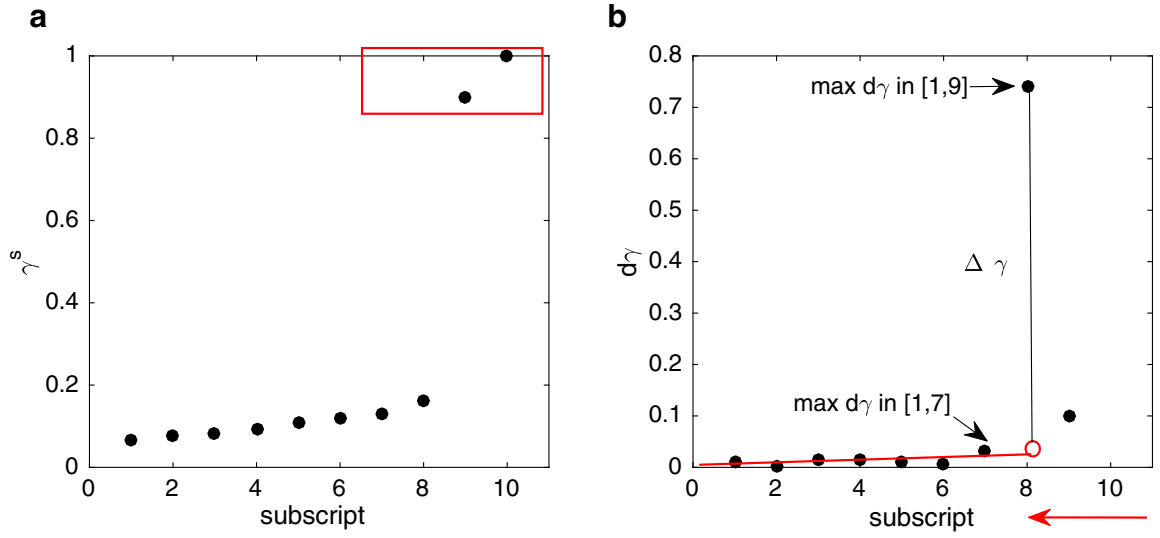

Fig. 3. Diagram to illustrate how to select cluster centers with linear fitting using the γ values in Table 1. (a) The diagram of γ values in Table 1. The points inside the rectangle should be chosen as clusters centers by our observation. (b) The process of linear fitting. Solid circles are elements of $\boldsymbol{d\gamma}$ and lines are the linearly fitted lines. First, we select the maximum of $\boldsymbol{d\gamma}$ in the entire dataset, and then do linear fitting to predict its value with the points in front. If there is a significant difference between the real value and the predicted value, then points behind will all be chosen as clusters. Repeat this process until there is no big difference occurred or there are only two points left.

Table 1
$\boldsymbol{\gamma}$, $\boldsymbol{\gamma_s}$, $\boldsymbol{d\gamma}$ used by example 1.

Node	1	3	4	6	9	17	18	24	30	32
$\boldsymbol{\gamma}$	**0.8999**	0.1098	0.1284	0.1606	0.0780	0.1207	0.0672	**1.0000**	0.0806	0.0936
subscript	**1**	2	3	4	5	6	7	**8**	9	10
$\boldsymbol{\gamma^s}$	0.0672	0.0780	0.0806	0.0936	0.1098	0.1207	0.1284	0.1606	**0.8999**	**1.0000**
Ind	7	5	9	10	2	6	3	4	**1**	**8**
subscript	1	2	3	4	5	6	7	8	**9**	**10**
$\boldsymbol{d\gamma}$	–	0.0108	0.0026	0.0130	0.0162	0.0109	0.0077	0.0323	**0.7393**	**0.1001**
subscript	–	1	2	3	4	5	6	**7**	**8**	**9**

The notations in this table such as $\boldsymbol{\gamma}$, $\boldsymbol{\gamma_s}$, $\boldsymbol{d\gamma}$ and ***Ind*** are consistent with the notations used above. Notation *subscript* is the element subscript of corresponding vector, start from 1.

Algorithm 2 SelectCenters.

Input: The set of local densities $\boldsymbol{\rho}$ and the set of separation distances $\boldsymbol{\delta}$
Output: The set of cluster centers ***Clusters***

/* Normalize ρ and δ to $[0, 1]$ */
for $i = 1$ to n **do**
  Normalize ρ_i via Equation (9) with ρ_i^* produced;
  Normalize δ_i via Equation (10) with δ_i^* produced;
end for
Filtering out nodes with relatively small density or separation distance;
/* Calculate γ for remaining nodes */
for all node v_i remained after filtering **do**
  Calculate γ_i via Equation (11);
end for
/* Sort gamma in ascending order */
$(\boldsymbol{\gamma^s}, \boldsymbol{Ind}) = sortAscending(\boldsymbol{\gamma})$;
Calculate the difference vector $\boldsymbol{d\gamma}$ via Equation (12);
$\boldsymbol{d\gamma} \leftarrow \{d\gamma_i | i \in Ind\}$;
$S \leftarrow \varnothing$;
/* Selecting cluster centers */
repeat
  $idx \leftarrow \max(\boldsymbol{d\gamma})$;
  Do linear fitting to predict $d\gamma_{idx}$, denoted as $\hat{d\gamma}_{idx}$;
  if Equation (15) are satisfied **then**
    $S \leftarrow \{v_i | i \in [Ind_{idx+1}, Ind_{|Ind|}]\}$;
    $\boldsymbol{d\gamma} \leftarrow \{d\gamma_i | i \in [1, idx-1]\}$;
  else
    break;
  end if
until $idx < 3$
add each node v_i in S to ***Clusters***;
return ***Clusters***;

to be in the same clusters as their friends. When calculating node v_i's membership with respect to cluster c, we not only consider the link strength between v_i and its k-nearest neighbors, but also the local structure of its k-nearest neighbors. For one of v_i's k-nearest neighbors v_j, how significantly v_j influences v_i' cluster label is not only related to its link strength to v_i but also how frequently v_j's cluster label occurred in its own neighbors. The more frequently, the greater v_j influences v_i. In Eq. (16), the fraction in right side represents how significantly v_j's k-nearest neighbors influences v_i' cluster label.

(2) Assign node v_i to cluster c if $\frac{p_i^c}{p_i^{cl_i}} \geq \sigma$. That is to say, if v_i's membership in cluster c is greater than that of its original cluster to some degree, v_i should also belong to cluster c.

The allocation process is presented as Algorithm 3.

4.5. EADP Method

From Sections 4.1 to 4.4, we have discussed the core parts of EADP, respectively. In this section, we give an overall picture of EADP as Algorithm 4 shows. Then, we will analyze the time complexity of EADP.

Suppose there is a network with n nodes and the number of clusters is m. The time complexity of computing the distance matrix is $O(vn^2)$, where v is the average number of common nodes. Note that v will be greatly less than n because social networks are often sparsely connected. Thus, the time complexity of computing the distance matrix nodes will be a little bigger than $O(n^2)$ but greatly less than $O(n^3)$. As for selecting cluster centers, sorting γ requires $O(n\log n)$ and selecting centers requires $O(m)$ for the worst case. The time complexity of allocating non-center nodes consists of two parts: $O(n)$ for the first step and $O(nm)$ for the second step. Thus, the overall time complexity of EADP is $O(vn^2 + nlogn + m + nm)$.

Algorithm 3 AllocateNodes.

Input: The selected cluster centers ***Clusters***, the adjacent matrix A and threshold σ
Output: The allocation for all data nodes;

$cl \leftarrow$ Set of cluster labels of all nodes after first-step allocation
$R \leftarrow n * |\boldsymbol{Clusters}|$ matrix of final allocation results;
/* Traverse ***Clusters*** to initialize cl and$R$ */
for $i = 1$ to $|\boldsymbol{Clusters}|$ **do**
  $v_p \leftarrow Clusters_i$;
  $cl_p \leftarrow i$
  $R_{pi} \leftarrow 1$;
end for
/* First-step allocation */
for all i such that v_i is not a cluster center **do**
  $cl_i \leftarrow cl_p$, where $\min_{j:\rho_j > \rho_i} dist(i, j) = dist(i, p)$;
  $R_{icl_i} \leftarrow 1$;
end for
/* Second-step allocation */
for $i = 1$ to n **do**
  if v_i is a non-enveloped node **then**
    $C \leftarrow \bigcup_{j \in knn_i} cl_j$;
    for all $c \in C$ **do**
      Calculate p_i^c via Equation (16);
      if $p_i^c \geq \sigma * p_i^{cl_i}$ **then**
        $R_{ic} \leftarrow 1$; /* Allocate node v_i to cluster c */;
      end if
    end for
  end if
end for
return R;

Algorithm 4 EADP.

Input: The adjacent matrix A
Output: The allocation for all data nodes

$D \leftarrow$ CalDisMx(A);
for $i = 1$ to n **do**
  Calculate ρ_i via Equation (8);
  Calculate δ_i via Equation (4);
end for
Clusters $\leftarrow$ SelectCenters($\boldsymbol{\rho}, \boldsymbol{\delta}$);
$R \leftarrow$AllocateNodes(***Clusters***, A);
return R;

5. Experiment setup

In this section, we will give a brief introduction of the baseline algorithms, community validation metrics and the datasets.

5.1. Baseline algorithms

The baseline algorithms include:

- *MOSES* [12], a statistical model based algorithm, which is capable of detecting highly overlapping community structure.
- *SLPA* [7], a speaker-listener based information propagation algorithm. It has a threshold parameter r, and we set it as 0.05, for it gives best results.

- *HOCTracker* [34], a density-based approach for detecting overlapping community structures.
- *OCDDP* [5], an extension of DPC for overlapping community detection, requires human intervention to choose cluster centers.
- SMFRW [22], an overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk.

It should be noted that, since SLPA and SMFRW are non-deterministic, their final results are the average performances of 10 repetitions.

5.2. Community validation metrics

We evaluate the quality of overlapping communities from both accuracy and structure perspectives. Metrics from the perspective of accuracy include:

- *Overlapped normalized mutual information (ONMI)*. We use the one proposed in [11], which is generalized to overlapping communities. This metric quantifies the level of correspondence between the ground-truth communities and those obtained from an algorithm.
- Ω *Index* [53]. It is based on pairs of nodes in agreement in two covers. A pair of nodes is considered to be in agreement if they are clustered in exactly the same number of communities (possibly none).
- *F-Score* [54]. It is the average of the *F*1-score of the best-matching ground-truth community to each detected community, and the *F*1-score of the best-matching detected community to each ground-truth community.

For some datasets with unknown ground-truth communities, the above validation metrics cannot be calculated. Instead, we evaluate the quality of detected communities with following metric:

- *Overlapping version of modularity* [50], denoted as Q_{ov}. This metric evaluates the quality of overlapping communities from structure perspective.

The higher the value of these metrics, the better the quality of detected communities.

5.3. Test suite of datasets

To compare our algorithm EADP with baseline algorithms more fairly, we conduct experiments on both real-world networks and synthetic networks.

5.3.1. Real-world networks

We have used twelve real-world datasets, including weighted and un-weighted datasets, labeled and unlabeled datasets. For unlabeled datasets, their ground-truth community structures are unknown, thus only Q_{ov} could be calculated.

Table 2 shows the details of these datasets. The labeled datasets include:

- *Dolphin network* [55], a social network of frequent associations between 62 dolphins.
- *NCAA college football network* [56], a social network consisting of 115 college football teams.
- *The US political books network* [1], a network of 105 books about US politics sold online by Amazon.
- *Karate* [51], a weighted Zachary's interaction network between 34 members of a Karate club.
- *School1*, a weighted school networks used by HOCTracker.
- *School2*, a weighted school networks used by HOCTracker.

The unlabeled networks include:

- *Power* [2], an unweighted network representing the topology of 4941 western states power grid in the United States.
- *Netscience* [2], a weighted co-authorships network of 1461 scientists working on network theory and experiment.
- *Hep-th* [2], a weighted collaboration network of 8361 scientists posting preprints on the high-energy theory archive at http://www.arxiv.org. 1995–1999
- *Astro-ph* [2], a weighted collaboration network of 16,706 scientists posting preprints on the astrophysics archive at http://www.arxiv.org, 1995–1999.
- *Cond-mat* [2], a weighted collaboration network of 16726 scientists posting preprints on the condensed matter archive at http://www.arxiv.org, 1995–1999
- *Cond-mat-2003* [2], an updated version of Cond-mat and contains 31163 scientists.

Fig. 4 shows the weight distribution of eight weighted networks. By observing this graph, we find that weights generally obey power-law distribution, which is a typical characteristic of social networks. In addition, karate is smaller and has simpler complex weight distributions compared to other weighted networks. Hep-th, Astro-ph, Cond-mat and Cond-mat-2003 are larger networks, thus it would be more difficult to detect community structures for them.

5.3.2. Synthetic networks

For we do not have enough instances of real networks to test our algorithm, we utilize LFR [57] benchmark to generate synthetic networks. LFR provides various parameters to control the network topology, including network size n, the average degree $\bar{k}$, minimal community sizes *minc*, maximal community sizes *maxc*, the mixing parameter *mut*, the mixing parameter for the weights *muw*, the percentage of overlapping nodes O_n, the number of communities O_m to which each overlapping node belongs.

We generate networks with following configuration: $n \in \{1000, 5000, 10, 000\}$, $\bar{k} = 10$, $minc = 10$, $maxc = 50$, $mut = 0.2$, $muw \in \{0.2, 0.3\}$, $O_n = 10\%$, $O_m \in \{2, 8\}$. The rest of the parameters of LFR generator are set to their default values. For each parameter set, we generate 10 instantiations and the results in all the experiments are reported by the average.

6. Experiments and analysis

In this section, we first show and analyse our experimental results of effectiveness evaluation on real-world networks and synthetic networks. Then, we will make a comparison about running time.

6.1. Experiments on real social networks

In this section, we compare EADP with baseline algorithms in real-world networks from three aspects: accuracy, structural quality and performance on weighted networks. Table 3 gives the experimental results with respect to ONMI, Ω Index, *F*-Score. And Table 4 gives the experimental results of Q_{ov}. Note that only Q_{ov} could be calculated for datasets with unknown ground-truth communities.

Comparison about accuracy. In terms of accuracy, EADP achieves three best ONMI results, five best Ω Index results and four best *F*-score results in six datasets with known ground-truth community structures. While MOSES, SLPA and HOCTracker get best ONMI

[1] http://www.orgnet.com/.

[2] http://www-personal.umich.edu/~mejn/netdata/.

Table 2
Details of real-world datasets.

Datasets	Type	Nodes	Edges	Communities	Standard deviation of weights	Ground truth
Dolphin	Unweighted	62	159	2	–	Known
Football	Unweighted	115	616	12	–	Known
Polbooks	Unweighted	105	441	3	–	Known
Karate	Weighted	34	78	2	1.21	Known
School1	Weighted	236	5899	11	10.83	Known
School2	Weighted	242	8316	11	10.14	Known
Power	Unweighted	4941	6593	–	–	Unknown
Netscience	Weighted	1461	2742	–	0.43	Unknown
Hep-th	Weighted	8361	15,751	–	1.18	Unknown
Astro-ph	Weighted	16,706	121,251	–	5.15	Unknown
Cond-mat	Weighted	16,726	47,594	–	0.81	Unknown
Cond-mat-2003	Weighted	31,163	120,029	–	0.88	Unknown

The notation "-" represents the unknown information.

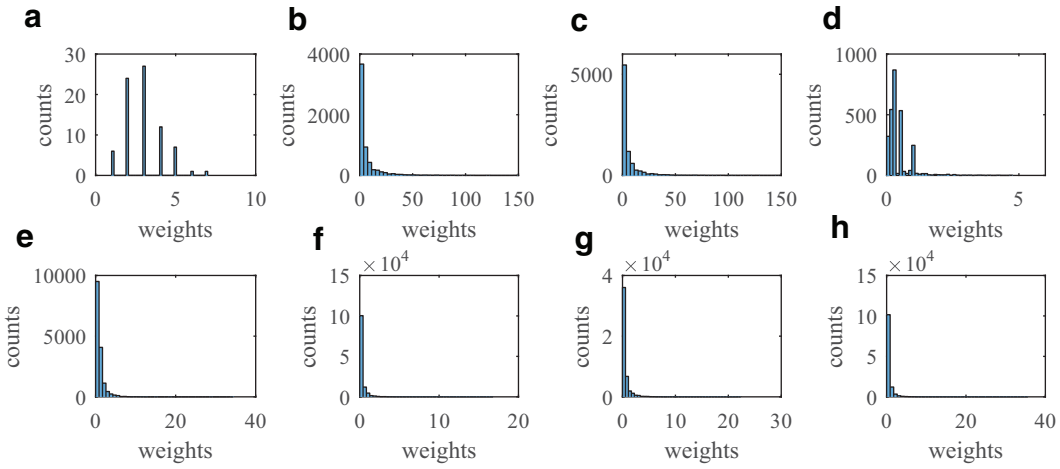

Fig. 4. Histogram showing the weight distribution of eight weighted networks, where counts represents the weight occurrences in each weight range. Subfigure (a), (b), (c), (d), (e), (f), (g) and (h) show the weight distribution of Karate, School1, School2, Netscience, Hep-th, Astro-ph, Cond-mat and Cond-mat-2003 respectively.

Table 3
Results of accuracy on real-world datasets with ground-truth communities.

	MOSES	SLPA	HOCTracker	OCDDP	SMFRW	EADP
ONMI						
Dolphin	0.190379	0.525838	0.853569	0.515732	0.198745	**1.000000**
Football	**0.789716**	0.624700	0.788826	0.682400	0.640130	0.729898
Polbooks	0.161567	**0.676260**	0.328542	0.337734	0.234262	0.503931
Karate	0.154515	0.325136	0.584211	0.395334	0.263420	**0.837171**
School1	0.548504	0.552176	0.707284	0.608474	0.491403	**0.752064**
School2	0.519014	0.667338	**0.727678**	0.584522	0.331552	0.723308
Ω Index						
Dolphin	0.132237	0.626055	0.913617	0.483398	0.234221	**1.000000**
Football	**0.910922**	0.566865	0.818665	0.747707	0.719227	0.765524
Polbooks	0.272700	0.618835	0.661239	0.665876	0.539437	**0.667100**
Karate	0.095037	0.756354	0.739145	0.826643	0.232725	**0.882258**
School1	0.282241	0.656602	0.734936	0.732153	0.564733	**0.785725**
School2	0.138556	0.690432	0.714809	0.618752	0.426704	**0.736380**
***F*-Score**						
Dolphin	0.262762	0.960737	0.886365	0.672680	0.409509	**1.000000**
Football	**0.919961**	0.833174	0.625872	0.778689	0.746988	0.712166
Polbooks	0.486769	0.801535	0.773323	0.803353	0.690496	**0.813413**
Karate	0.184818	0.862524	0.882331	0.686636	0.445714	**0.939450**
School1	0.568180	0.760600	0.697150	0.463061	0.624625	**0.799490**
School2	0.515577	**0.799018**	0.713717	0.200398	0.497046	0.761301

scores in Football, Polbooks and School2, respectively, EADP outperforms each of them greatly in other datasets. In detail, for ONMI scores, while MOSES is only better than EADP in Football by 7%, EADP is better than MOSES in Dolphin, Polbooks, Karate, School1 and School2 by 425%, 212%, 442%, 37% and 39%, respectively. SLPA only performs better than EADP in Karate by 25%, but EADP outperforms it in Dolphin, Football, Karate, School1 and School2 by 90%, 17%, 157%, 36% and 8%, respectively. As for HOCTracker, it gets better performance in Football and School2, but only slightly better. However, EADP is much better than HOCTracker in Dolphin, Polbooks, Karate and School1. Furthermore, when it comes to Ω Index and *F*-Score, EDAP gets best performance on most of the datasets. That is, EADP could detect communities with satisfactory accuracy.

Table 4
Results of Q_{ov} on real-world datasets with unknown ground-truth.

	MOSES	SLPA	HOCTracker	OCDDP	SMFRW	EADP
Dolphin	0.477751	0.685559	0.745779	**0.773529**	0.612292	0.739517
Football	0.657647	**0.724349**	0.641837	0.667688	0.675848	0.695351
Polbooks	0.299139	0.712718	0.829655	**0.839468**	0.791269	0.834244
Karate	0.240197	**0.822438**	0.502041	0.703645	0.200885	0.752966
School1	0.158076	**0.425005**	0.412473	0.363453	0.314169	0.411370
School2	0.014461	**0.379027**	0.368290	0.354710	0.216062	0.371781
Power	0.000000	0.629854	0.000000	**0.763156**	0.489503	0.653581
Netscience	0.000000	0.816739	0.000000	0.934115	0.495284	**0.977398**
Hep-th	0.414581	0.618767	0.443545	0.732873	0.177239	**0.744857**
Astro-ph	0.278388	0.474452	0.272157	0.708523	0.335920	**0.733640**
Cond-mat	0.419392	0.617963	0.358064	0.807591	0.316721	**0.820219**
Cond-mat-2003	0.303936	0.525449	0.280623	0.723583	0.320530	**0.725460**

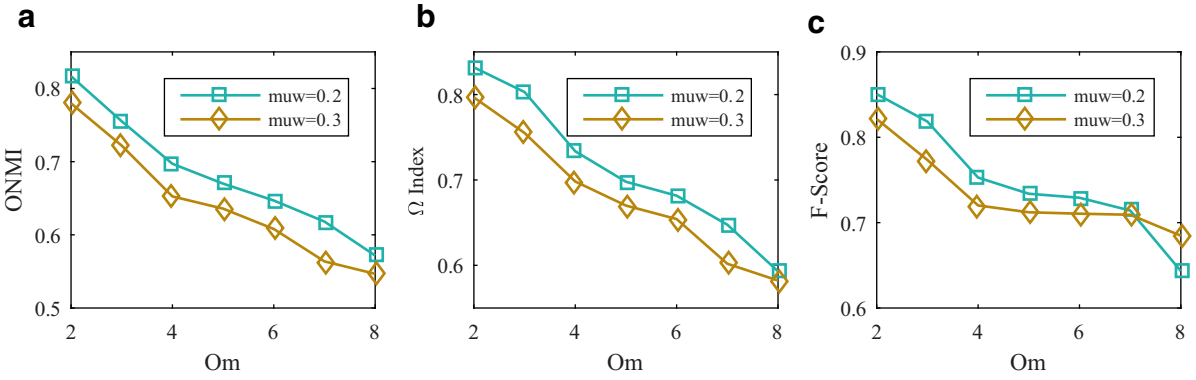

Fig. 5. EADP's performance on networks with $muw = \{0.2, 0.3\}$.

Comparison about structural quality. In Table 4, we could easily find that there are four zero Q_{ov} scores. That is because MOSES and HOCTracker fail to find any community in Power and Netscience. For small datasets like Karate, School1 and School2, it is SLPA instead of EADP that gets most of the best Q_{ov} scores. However, the results obtained by EADP are quite comparable. For some relatively larger datasets, such as NetScience, Hep-th, Astro-ph, Cond-mat and Cond-mat-2003, EADP gets much higher Q_{ov} scores than baseline algorithms including SLPA. As we know, it is often harder to detect correct communities within larger networks. Therefore, compared to baseline algorithms, EADP is more capable of finding correct community structures.

It could be seen that OCCDP also get excellent Q_{ov} results, which is because it relies on human to select cluster centers and human's priori knowledge will help. It should be noted that the good performance of OCDDP depends on the effort of human. In addition, it would be inefficient.

Comparison on weighted networks. Observing the performance on three weighted datasets with known community structures, we could find that EADP achieves excellent performance on ONMI, Ω Index and *F*-Score. For five weighted datasets with unknown community structures, EADP performs much better in term of Q_{ov} scores. To compare HOCTracker with EADP, we could find that EADP gets much better results. The reason is that different from HOCTracker, EADP treats direct and un-direct linking weights unequally, thus EADP could handle weighted networks better.

6.2. Experiments on synthetic networks

In this section, we first talk about the performance of EADP and how it is affected by parameter t and σ. Then we will make a comparison between EADP and baseline algorithms.

6.2.1. Investigation about EADP

Performance of EADP on networks with different muw. First, we test the performance of EADP in handling networks of different weight distributions along with varying the overlapping degree O_m. We use networks of 1000 nodes and the experimental results is shown in Fig. 5.

It is obvious that the ONMI, Ω Index and *F*-Score all decrease along with the growth of O_m. The reason might be that a bigger O_m means a node could belong to more communities and therefore it is a harder task, which leads to the performance decreasing. Parameter *muw* shows the similar phenomenon: the results of $muw = 0.3$ are consistently worse than those of $muw = 0.2$. As we know, the bigger the *muw*, the more complex the weight distribution. However, the results show that the performance of EADP is only a little bit worse when $muw = 0.3$ than the performance when $muw = 0.2$. This means that EADP is capable of handing those networks with relatively complex weight distribution.

Performance of EADP on networks with different n. We test how EADP performs on networks with different sizes along with varying the overlapping degree O_m. We use networks of 1000, 5000 and 10,000 nodes, and the results are shown in Fig. 6. Apparently, EADP performs better on networks with 1000 nodes than those with 5000 nodes, and the performance on 5000 nodes is better than that of 10,000 nodes. This results are consistent with our intuition that more nodes increase the difficulty of the clustering task, thus it is harder to find community structure accurately.

Sensitivity to parameter t and σ. For EADP introduces two parameters t and σ, here we test how they influence the performance of EADP. To do this, we do experiments on four kinds of synthetic networks with different O_m and *muw*, shown in Fig. 7.

Roughly, the performance of EADP first increases and then decreases as t increases. It could be found that EADP performs well when $t < 0.4$. When $O_m = 2$, greater σ gives better results. But

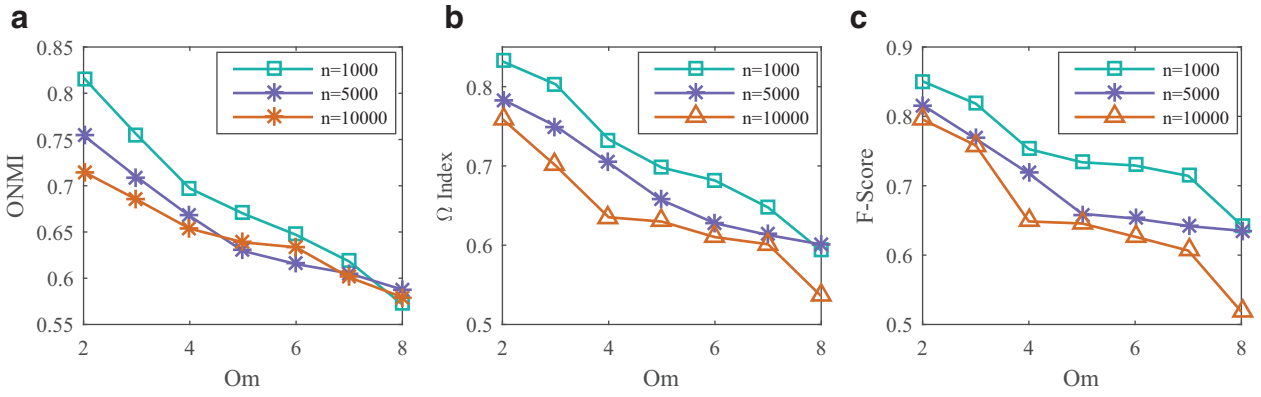

Fig. 6. EADP's performance on networks with $n = \{1000, 5000, 10,000\}$.

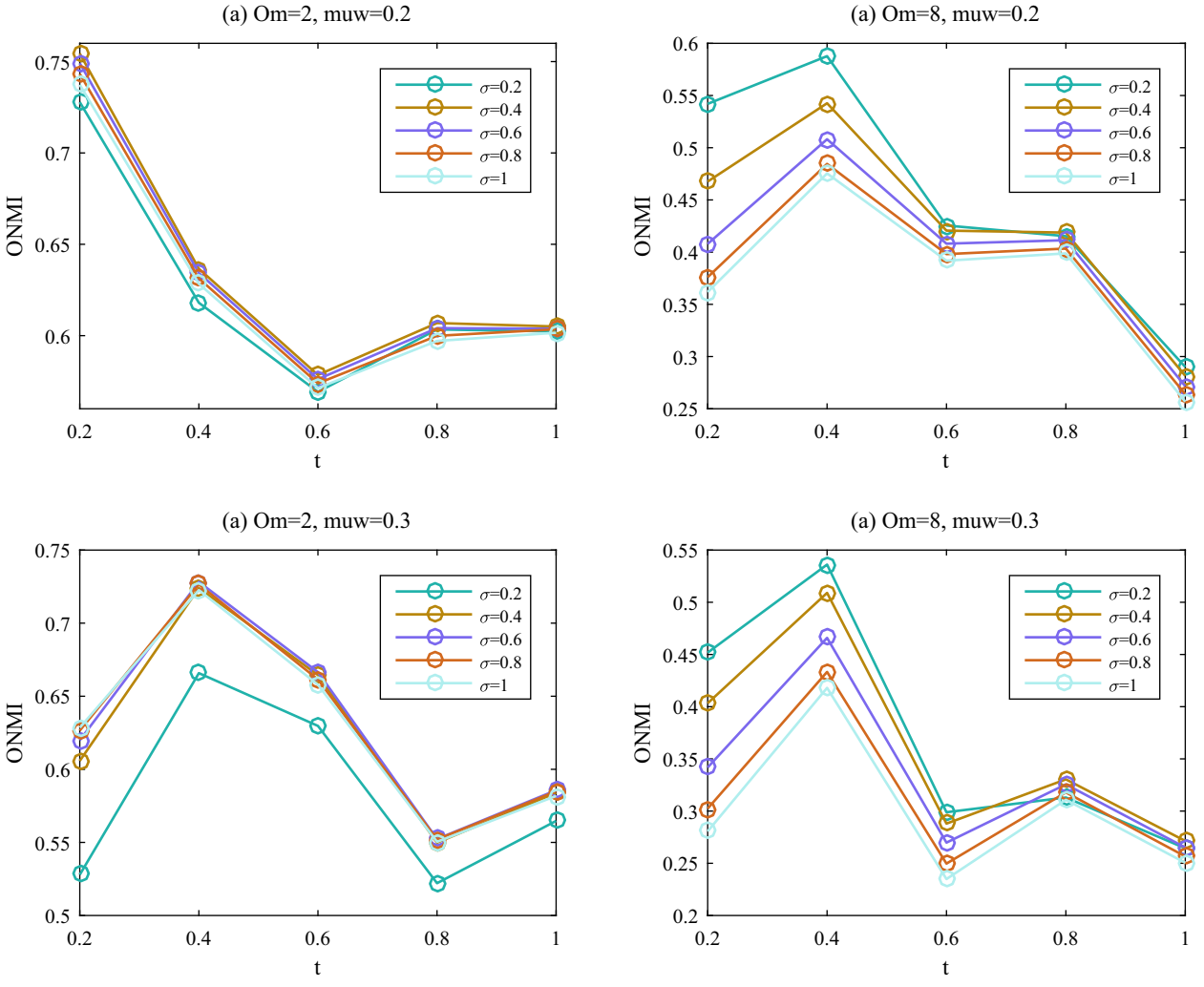

Fig. 7. Figure to explore how parameter t and σ influences the performance of EADP.

when $O_m = 8$, smaller σ gives better results. The reason is that greater O_m means higher degree of overlap, so a smaller σ allows a node belong to more communities, and a larger σ makes EADP more careful to assign node to a new community, which yields higher consistence to ground truth community structures.

6.2.2. Comparison with baseline algorithms

In this section, we focus on comparing the performance of EADP with baseline algorithms in case of detecting overlapping communities in synthetic networks with different weight distributions, degrees of overlap and sizes.

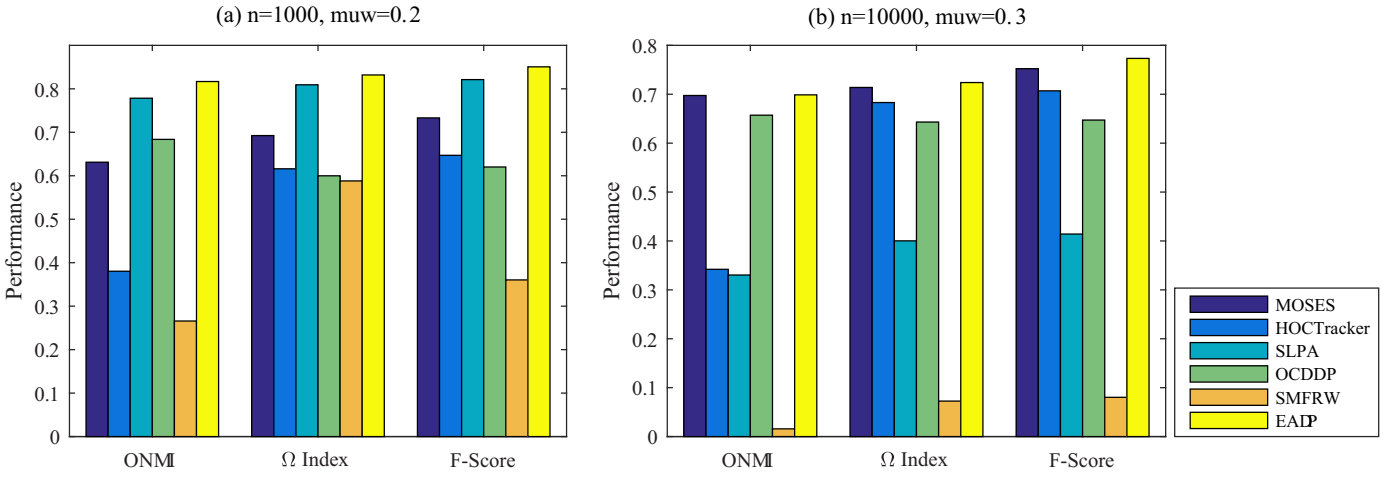

Fig. 8. Comparison results about accuracy.

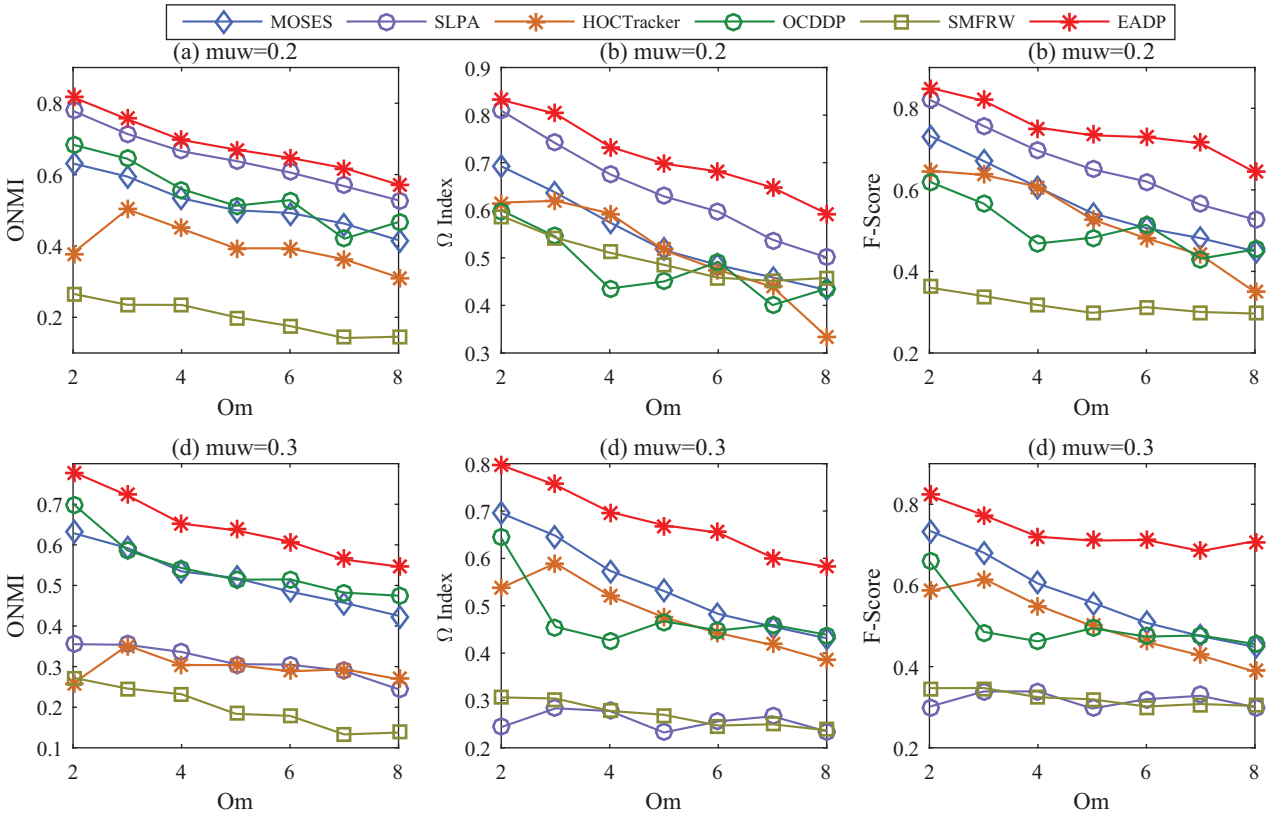

Fig. 9. Comparison results about *muw* with O_m varying.

Comparison about accuracy. We compare EADP with baseline algorithms on relatively simple network ($n = 1000$, $muw = 0.2$) and complex network ($n = 10,000$, $muw = 0.3$). The result is shown in Fig. 8. Apparently, EADP gets all the best results, whether on ONMI, or Ω Index or *F*-Score, which shows the effectiveness of EADP in detecting accurate overlapping community structures.

Comparison on networks with different muw. To test how EADP and baseline algorithms perform on networks with different weight distributions, we generate networks with $muw = \{0, 2, 0.3\}$. From Fig. 9, we could find that EADP performs better than all the baseline algorithms whenever *muw* is 0.2 or 0.3. When *muw* is set to 0.2, SLPA gets comparable ONMI, Ω Index and *F*-Score results compared to EADP. But when *muw* increases to 0.3, EADP performs much better than SLPA. Meanwhile, it is easy to find that the performance of SLPA deceases a lot when *muw* increases from 0.2 to 0.3. In contrast, EADP gives much more stable performance. The results indicate that SLPA is not competent to find communities in networks with relatively complex weight distributions.

Comparison with overlap degree varying. By observing Fig. 9, we could also see that the performances of all the algorithms including EADP show a downward trend as O_m increases. The reason is that a bigger O_m means higher degree of overlap and a more challenging clustering task, which leads to the performance decreasing. It is worth noticing that OCDDP does not strictly decreases along with the increase of O_m and there are some fluctuations, which is because of the uncertainty and potential mistakes caused by manual selection of the cluster centers.

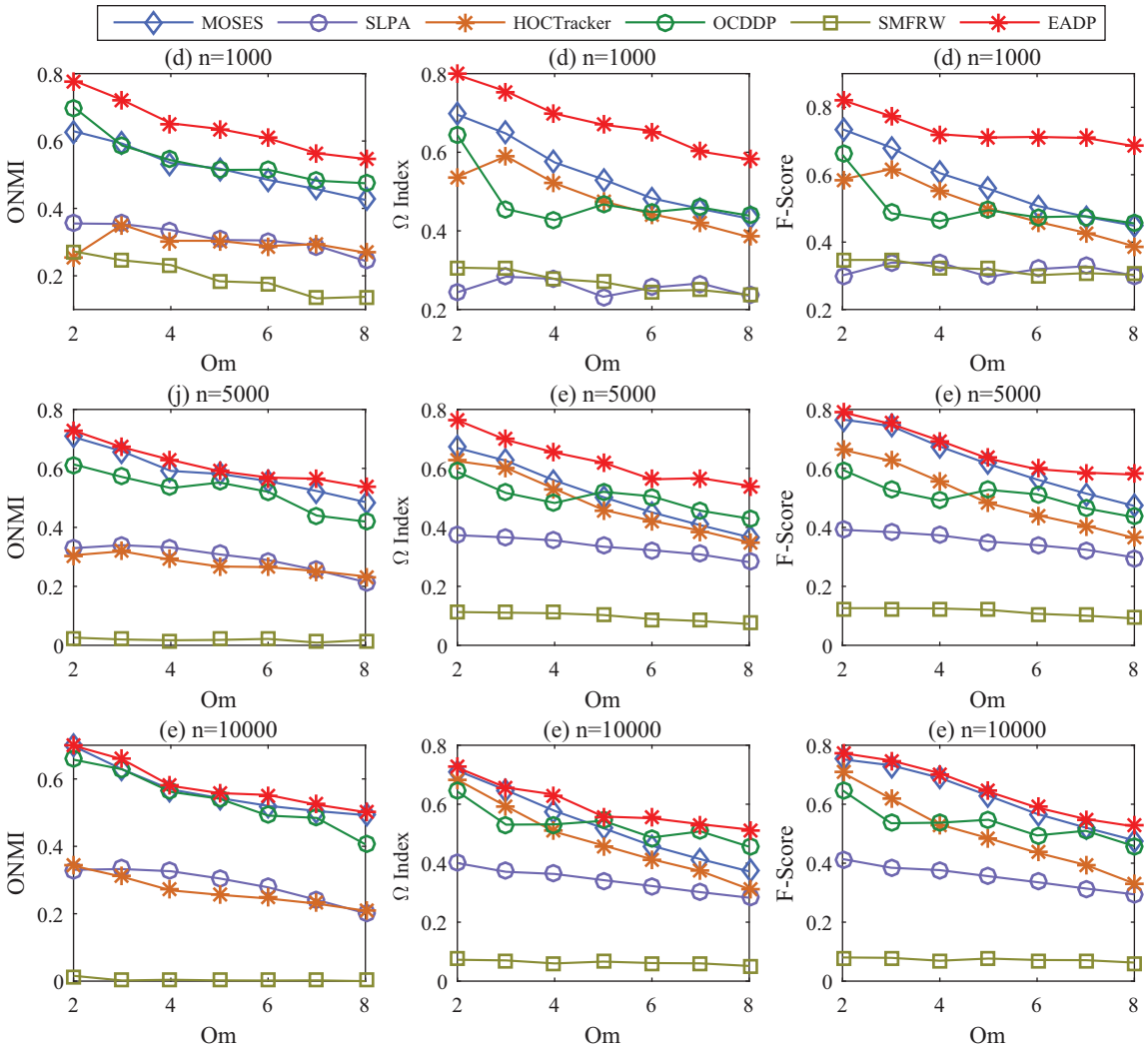

Fig. 10. Comparison results about n with O_m varying

Comparison on network with different size. We do experiments on network with $n = \{1000, 5000, 10, 000\}$, $muw = 0.3$ to test how algorithms will perform with different size of networks. From Fig. 10, we could see that EADP gets almost all the best results. And with the increase of nodes, MOSES achieves as excellent performance as EADP. As we know, larger networks will contain more overlapping nodes, thus it is hard to allocate overlapping nodes to correct communities. MOSES scales to large networks and is capable of detecting highly overlapped community structure, especially when there is variance in the number of communities each node belongs to. Therefore, MOSES could perform well on larger networks.

It could be concluded that EADP outperforms baseline algorithms on almost all the cases. While SLPA achieves comparable performance with EADP on networks with simple weight distribution, EADP achieves much better performance than SLPA when it comes to networks with more complex weight distributions. Compared to baseline algorithms, EADP could find overlapped community structures more accurately. Therefore, we could claim that EADP is an effective overlapping community detection algorithm suitable for both unweighted and weighted networks. For networks with more complex weight distribution, EADP has advantages over baseline algorithms.

Table 5
Comparison of running time with baseline algorithms.

Datasets	MOSES C++	SLPA Java	HOCTracker Java	SMFRW Python	EADP Matlab
Netscience	4 s	4 s	< 1 s	< 1 s	< 1 s
Power	3 s	2 s	< 1 s	468 s	5 s
Hep-th	34 s	44 s	7 s	15,169 s	15 s
Astro-ph	79 s	219 s	66 s	26,638 s	65 s
Cond-mat	30 s	172 s	18 s	25,713 s	68 s
Cond-mat-2003	56 s	384 s	65 s	45,584 s	262 s

6.3. Comparison of running time

In this section, we compare EADP with the baseline algorithms except OCDDP in terms of running time. Since OCDDP requires human to select cluster centers, it would be unfair to compare with it.

We use real-world datasets with more than 1000 nodes in Table 2, and the experimental results are shown in Table 5. To make the results more reliable, we run each algorithm in each dataset for ten times with a 64bit Linux machine of E5-2620 2.10 GHz CPU and 128GB memory, and then take the average spent time as our final result.

As we can see in Table 5, the time spent by EADP increases as the amount of data increases. However, the baseline algorithms do not show the same phenomenon and their results are unstable.

In dataset Astro-ph, Cond-mat and Cond-mat-2003, MOSES spends the longest time on Astro-ph, and shortest time in Cond-mat. MOSES uses a combination of heuristics, edge-expansion, to detect community structures, and the number of edges will affect the running time. Astro-ph has nearly the same number of nodes as Cond-mat, and three times as many edges as Cond-mat. Therefore, MOSES takes longer time in Astro-ph than Cond-mat. Apart from this, we can also see that MOSES takes shorter time in Power than NetScience, and shorter time in Cond-mat-2003 than Astro-ph, even Power and Cond-mat-2003 is bigger than Netscience and Astro-ph respectively. That is because MOSES selects edges randomly and a community is built around each selected edge. The initial edge selection process could lead to different expansion process and the time spent by MOSE varies.

As for SLPA, its time complexity is $O(Tn)$, where T is the user-defined maximum iterations. However, our experimental results do not completely follow this theoretical time complexity. For instance, SLPA spends less time in Power and Cond-mat than in Netscience and Astro-ph respectively. SLPA is based on label propagation algorithm (LPA), and its community detection process is dynamic and uncertain. Therefore, there is a probability that SLPA spends less time in larger datasets.

HOCtracker is a density-based algorithm with no need to set parameters manually. It uses a heuristic method to tune parameter η until the next community structure does not result gain or no-change in modularity score. In addition, HOCtracker has to calculate distance for each pair of node, and the complexity of this process is $O(n^2)$. Therefore, its time complexity would be $O(Tn^2)$, where T is the times for searching η. In the worst case, T is only 2 or 3. In Table 5, we could find that HOCtracker takes much less time in Cond-mat than in Astro-ph, although Cond-mat has almost the same number of nodes as Astro-ph. The reason is that Astro-ph requires fewer rounds to search η, thus HOCtracker could run much faster.

SMFRW is the slowest algorithm in all comparison algorithms, and it contains many time-consuming operations. SMFRW first calculates the transition probability for each pair of nodes, and then constructs feature matrix. At the same time, it combines a priori information to build a must-link matrix and a cannot-link matrix, which are merged into the feature matrix to form a new feature matrix. The time spent by SMFRW is involved with the number of nodes and edges. Astro-ph contains three times as many edges as Cond-mat, thus SMFRW spends much longer time in Astro-ph. Generally, SMFRW consumes more time along with the increase of the amount of data.

By comparing EADP with baseline algorithms, we could find that EADP is faster than SMFRW. In small datasets, EADP gets comparable time efficiency with MOSES, SLPA and HOCTracker. In larger datasets, EADP takes shorter time than SLPA and SMFRW, and takes longer time than MOSES and HOCTracker. Since EADP is extended to be able to handle social networks directly, it has to calculate the distance for each pair of node, which could be time-consuming. In addition, EADP is based on density peaks, and it contains several sort operations, which would spend more time when dealing with larger social networks. It would be our future work to speed up distance matrix calculation process and do some optimizations to scale EADP to very large social networks.

7. Conclusion and future work

In this paper, we extend DPC to be applied to overlapping community detection in social networks and propose our algorithm EADP. EADP adopts a two-step strategy to allocate non-center nodes so that communities could be overlapped. To handle social networks directly without extra work to construct the distance matrix, EADP incorporates a common nodes based distance function. Moreover, unlike DPC choosing cluster centers by hand, EADP utilizes a linear fitting based strategy to select cluster centers adaptively. Experimental results on real social networks and synthetic networks show the effectiveness of EADP about detecting overlapping communities. Compared to baseline algorithms, EADP gets much better performance on networks with more complex weight distribution. On those unweighted networks or networks with simple weight distribution, EADP is still capable of getting competitive results. In terms of running time, EADP gets comparable time efficiency in small networks. And in larger networks, the time efficiency of EADP could be positioned in the middle, not the best or the worst.

Our current work focuses on extending the method of density peaks clustering to social networks and improving the accuracy of the detected communities. As for our future work, we will focus on speeding up EADP to make it work fast for large scale social networks. In addition, we will study an adaptive method to choose parameters t and σ. Thus, the parameters will be learned from the given network and there is no need to set them manually.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under grants 2016QY01W0202 and 2016YFB0800402, National Natural Science Foundation of China under grants 61572221, U1836204, 61672254, 61433006, 61772219, U1401258 and 61502185, Major Projects of the National Social Science Foundation under grant 16ZDA092 and Guangxi High level innovation Team in Higher Education Institutions Innovation Team of ASEAN Digital Cloud Big Data Security and Mining Technology. We would like to express our thanks to the anonymous reviewers for their invaluable comments and suggestions and to the author of the baseline algorithms for sharing their code to us for comparison.

References

[1] M. Ester, H. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: Proceedings of the Kdd, 96, 1996, pp. 226–231.
[2] H. Liu, T. Liu, J. Wu, D. Tao, Y. Fu, Spectral ensemble clustering, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 715–724.
[3] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814.
[4] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[5] X. Bai, P. Yang, X. Shi, An overlapping community detection algorithm based on density peaks, Neurocomputing 226 (2017) 7–15.
[6] S. Gregory, Finding overlapping communities in networks by label propagation, New J. Phys. 12 (10) (2010) 103018.
[7] J. Xie, B.K. Szymanski, Towards linear time overlapping community detection in social networks, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2012, pp. 25–36.
[8] M. Sattari, K. Zamanifar, A spreading activation-based label propagation algorithm for overlapping community detection in dynamic social networks, Data Knowl. Eng. 113 (2018) 155–170.
[9] H. Fan, Y. Zhong, G. Zeng, Overlapping community detection based on discrete biogeography optimization, Appl. Intell. 48 (2018) 1–13.
[10] C. Wang, W. Tang, B. Sun, J. Fang, Y. Wang, Review on community detection algorithms in social networks, in: Proceedings of the 2015 IEEE International Conference on Progress in Informatics and Computing (PIC), IEEE, 2015, pp. 551–555.
[11] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New J. Phys. 11 (3) (2009) 033015.
[12] A. McDaid, N. Hurley, Detecting highly overlapping communities with model-based overlapping seed expansion, in: Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, 2010, pp. 112–119.

[13] C. Lee, F. Reid, A. McDaid, N. Hurley, Detecting highly overlapping community structure by greedy clique expansion, in: Proceedings of the 4th Workshop on Social Network Mining and Analysis, ACM, 2010, pp. 32–44.
[14] D. Jin, B. Yang, C. Baquero, D. Liu, D. He, J. Liu, A Markov random walk under constraint for discovering overlapping communities in complex networks, J. Stat. Mech.: Theory Exp. 2011 (05) (2011) P05031.
[15] M.E. Newman, S.H. Strogatz, D.J. Watts, Random graphs with arbitrary degree distributions and their applications, Phys. Rev. E 64 (2) (2001) 026118.
[16] H. Liu, L. Fen, J. Jian, L. Chen, Overlapping community discovery algorithm based on hierarchical agglomerative clustering, Int. J. Pattern Recognit. Artif. Intell. 32 (03) (2018) 1850008.
[17] Z. Yu, J. Chen, K. Quo, Y. Chen, Q. Xu, Overlapping community detection based on random walk and seeds extension, in: Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing, ACM, 2017, pp. 18–24.
[18] S. Zhang, R.-S. Wang, X.-S. Zhang, Uncovering fuzzy community structure in complex networks, Phys. Rev. E 76 (4) (2007) 046103.
[19] F. Wang, T. Li, X. Wang, S. Zhu, C. Ding, Community discovery using nonnegative matrix factorization, Data Mining Knowl. Discov. 22 (3) (2011) 493–521.
[20] N. Chen, Y. Liu, H.-C. Chao, Overlapping community detection using non-negative matrix factorization with orthogonal and sparseness constraints, IEEE Access 6 (2017) 21266–21274.
[21] X. Shi, H. Lu, G. Jia, Adaptive overlapping community detection with Bayesian nonnegative matrix factorization, in: Proceedings of the International Conference on Database Systems for Advanced Applications, Springer, 2017, pp. 339–353.
[22] W. Li, J. Xie, M. Xin, J. Mo, An overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk, Expert Syst. Appl. 91 (2018) 277–285.
[23] Y. Li, K. He, D. Bindel, J.E. Hopcroft, Uncovering the small community structure in large networks: a local spectral approach, in: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2015, pp. 658–668.
[24] K. He, Y. Sun, D. Bindel, J. Hopcroft, Y. Li, Detecting overlapping communities from local spectral subspaces, in: Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM), IEEE, 2015, pp. 769–774.
[25] Y. Li, K. He, K. Kloster, D. Bindel, J. Hopcroft, Local spectral clustering for overlapping community detection, ACM Trans. Knowl. Disc. Data 12 (2) (2018) 17.
[26] G. Bello-Orgaz, S. Salcedo-Sanz, D. Camacho, A multi-objective genetic algorithm for overlapping community detection based on edge encoding, Inf. Sci. 462 (2018) 290–314.
[27] Y. Ahn, S. Lehmann, J.P. Bagrow, Communities and hierarchical organization of links in complex networks, Technical Report, 2009.
[28] T.S. Evans, Clique graphs and overlapping communities, J. Stat. Mech.: Theory Exp. 2010 (12) (2010) P12037.
[29] T.S. Evans, R. Lambiotte, Line graphs of weighted networks for overlapping communities, Eur. Phys. J. B 77 (2) (2010) 265–272.
[30] T. Evans, R. Lambiotte, Line graphs, link partitions, and overlapping communities, Phys. Rev. E 80 (1) (2009) 016105.
[31] C. Shi, Y. Cai, D. Fu, Y. Dong, B. Wu, A link clustering based overlapping community detection algorithm, Data Knowl. Eng. 87 (2013) 394–404.
[32] H. Sun, J. Liu, J. Huang, G. Wang, X. Jia, Q. Song, Linklpa: a link-based label propagation algorithm for overlapping community detection in networks, Comput. Intell. 33 (2) (2017) 308–331.
[33] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3-5) (2010) 75–174.
[34] S.Y. Bhat, M. Abulaish, Hoctracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks, IEEE Trans. Knowl. Data Eng. 27 (4) (2015) 1013–1019.
[35] F. Sheikholeslami, G.B. Giannakis, Overlapping community detection via constrained parafac: a divide and conquer approach, in: Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 127–136.
[36] Z. Wang, Z. Yu, C.P. Chen, J. You, T. Gu, H.-S. Wong, J. Zhang, Clustering by local gravitation, IEEE Trans. Cybern. 48 (5) (2018) 1383–1396.
[37] Z. Yu, L. Li, J. Liu, J. Zhang, G. Han, Adaptive noise immune cluster ensemble using affinity propagation, IEEE Trans. Knowl. Data Eng. 27 (12) (2016) 3176–3189.
[38] Z. Yu, X. Zhu, H.-S. Wong, J. You, J. Zhang, G. Han, Distribution-based cluster structure selection, IEEE Trans. Cybern. 47 (11) (2017) 3554–3567.
[39] M. Du, S. Ding, H. Jia, Study on density peaks clustering based on k-nearest neighbors and principal component analysis, Knowl.-Based Syst. 99 (2016) 135–145.
[40] J. Xie, H. Gao, W. Xie, X. Liu, P.W. Grant, Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors, Inf. Sci. 354 (2016) 19–40.
[41] X.-F. Wang, Y. Xu, Fast clustering using adaptive density peak detection, Stat. Methods Med. Res. 26 (6) (2017) 2800–2811.
[42] E.N. Nasibov, G. Ulutagay, A new unsupervised approach for fuzzy clustering, Fuzzy Sets Syst. 158 (19) (2007) 2118–2133.
[43] M. Du, S. Ding, Y. Xue, A robust density peaks clustering algorithm using fuzzy neighborhood, Int. J. Mach. Learn. Cybern. (2017) 1–10.
[44] Z. Liang, P. Chen, Delta-density based clustering with a divide-and-conquer strategy: 3dc clustering, Pattern Recognit. Lett. 73 (2016) 52–59.
[45] L. Yaohui, M. Zhengming, Y. Fang, Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy, Knowl.-Based Syst. 133 (2017) 208–220.
[46] Z. Li, Y. Tang, Comparative density peaks clustering, Expert Syst. Appl. 95 (2018) 236–247.
[47] J. Xu, G. Wang, W. Deng, Denpehc: density peak based efficient hierarchical clustering, Inf. Sci. 373 (2016) 200–218.
[48] M. Du, S. Ding, X. Xu, Y. Xue, Density peaks clustering using geodesic distances, Int. J. Mach. Learn. Cybern. (2017) 1–15.
[49] J. Jiang, X. Tao, K. Li, Dfc: density fragment clustering without peaks, J. Intell. Fuzzy Syst. 34 (1) (2018) 525–536.
[50] V. Nicosia, G. Mangioni, V. Carchiolo, M. Malgeri, Extending the definition of modularity to directed graphs with overlapping communities, J. Stat. Mech.: Theory Exp. 2009 (03) (2009) P03024.
[51] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[52] Y. Li, C. Liu, M. Zhao, R. Li, H. Xiao, K. Wang, J. Zhang, Multi-topic tracking model for dynamic social network, Phys. A: Stat. Mech. Appl. 454 (2016) 51–65.
[53] L.M. Collins, C.W. Dent, Omega: a general formulation of the rand index of cluster recovery suitable for non-disjoint solutions, Multivar. Behav. Res. 23 (2) (1988) 231–242.
[54] J. Yang, J. Leskovec, Overlapping community detection at scale: a nonnegative matrix factorization approach, in: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, ACM, 2013, pp. 587–596.
[55] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations, Behav. Ecology Sociobiol. 54 (4) (2003) 396–405.
[56] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[57] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110.

Mingli Xu received the B.S. degree from Software Engineering, Chongqing University. She is currently pursuing the M.S. degree at Huazhong University of Science and Technology, Wuhan, China. Her current research interests include social network and machine learning.

Yuhua Li received the Ph.D. degree in computer application technology from Huazhong University of Science and Technology, Wuhan, China, in 2006. She is currently an associate professor in the College of Computer Science and Technology, Huazhong University of Science and Technology, China. She has published more than 40 journal and conference papers. She is a senior member of China Computer Federation (CCF). Her research interests include data mining, social network, machine learning, big data.

Ruixuan Li received the Ph.D. degree in computer application technology from Huazhong University of Science and Technology, Wuhan, China, in 2004. He is currently a professor in the College of Computer Science and Technology, Huazhong University of Science and Technology, China. He is a senior member of China Computer Federation (CCF), a member of IEEE and ACM. He has published more than 70 journal and conference papers. His research interests include social network, big data management, distributed computing, big data security.

Fuhao Zou received B.E. degree in computer science from Huazhong Normal University, Wuhan, Hubei, China, in 1998. And received M.S. and Ph.D. in computer science and technology from Huazhong University of Science and Technology (HUST), Wuhan, Hubei, China, in 2003 and 2006. Currently, he is an associate professor with the school of computer science and technology, HUST. His research interests include deep learning, multimedia understanding and analysis, big data analysis. He is senior member of China Computer Federation (CCF) and member of IEEE, ACM.

Xiwu Gu received the Ph.D. degree in computer application technology from Huazhong University of Science and Technology, Wuhan, China, in 2007. He is currently an associate associate research fellow in the College of Computer Science and Technology, Huazhong University of Science and Technology, China. He has published more than 30 journal and conference papers. His research interests include distributed computing, data mining, social computing, big data.