Information Sciences 596 (2022) 93–118

Contents lists available at ScienceDirect

Information Sciences

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

CEO: Identifying Overlapping Communities via Construction, Expansion and Optimization

Xiaoyu Ding [a,*], Hailu Yang [b], Jianpei Zhang [c], Jing Yang [c], Xiaohong Xiang [a]

[a] College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[b] School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China
[c] College of Computer Science and Technology, Harbin Engineering University, Harbin, Heilongjiang 150001, China

ARTICLE INFO

Article history:
Received 11 June 2020
Received in revised form 23 January 2022
Accepted 3 March 2022
Available online 5 March 2022

Keywords:
Overlapping community detection
Local expansion
Poor tolerance problem
Discarding low-quality seeds and communities
Node memberships

ABSTRACT

Overlapping communities are ubiquitous in real-world systems. For overlapping community detection, local expansion methods excel in scalability and efficiency yet have poor tolerance to low-quality seeds and communities. Based on our previous work, we introduce a more robust local-expansion-based overlapping community detection algorithm, named CEO, performing Construction, Expansion and Optimization sub-processes. To solve the poor fault tolerance problem, CEO discards low-quality seeds and communities in each sub-process based on optimizing node memberships. CEO was compared to thirteen noted algorithms by examining the performance on five groups of artificial networks and sixteen real-world networks with ground-truth communities. Experimental results showed CEO performs the best in identifying overlapping communities, which verifies the effectiveness of discarding low-quality seeds and communities in solving the poor fault tolerance problem.

1. Introduction

Many real-world systems can be modeled as networks where nodes represent objects and links represent interactions between objects. Community structure as a topological property of networks is essential for understanding the organization of real-world systems [1]. A community is intuitively described as a subgraph where the vertices are closely connected with each other but well separated from the rest of the network [2]. Community detection has gained increasing attention for decades. Most previously proposed methods focus on identifying disjoint communities where each node belongs to one community [3]. However, increasing evidence shows that many real-world systems are characterized by statistics of overlapping communities where a node may belong to multiple communities [4].

Mainstream overlapping community detection algorithms can be classified into local expansion methods [5], clique percolation methods [6], link partitioning methods [7], agent-based dynamical methods [8], fuzzy detection methods [9], multi-objective evolutionary methods [10], learning-based methods [11], etc. In addition, by performing on multiple groups of query nodes, community search methods can generate communities with overlaps [12]. Despite other methods, we focus on solving the poor fault tolerance problem of local expansion methods.

* Corresponding author.
E-mail addresses: dingxy@cqupt.edu.cn (X. Ding), yanghailu@hrbust.edu.cn (H. Yang), zhangjianpei@hrbeu.edu.cn (J. Zhang), yangjing@hrbeu.edu.cn (J. Yang), xiangxh@cqupt.edu.cn (X. Xiang).

https://doi.org/10.1016/j.ins.2022.03.012
0020-0255/

Local expansion methods iteratively perform seeding and expansion sub-processes in sequence. At each iteration, the seeding process first selects a node that has not been assigned to any community as a seed, and then the expansion process expands the community around the selected seed until the joining of its neighbors can no longer improve its quality. Each iteration generates a fully expanded community achieving the optimal configuration under specific conditions. The algorithm stops when each node in the network has been assigned to at least one community. **Obviously, local expansion methods have poor fault tolerance to low-quality seeds and communities. This is because once the algorithm generates a low-quality seed or community, the low-quality seed or community may lead to poor results of its subsequent seeding and expansion operations.**

Various technologies have been developed to get high-quality seeds and communities; however, the poor fault tolerance problem is still a blind spot in the study of local expansion methods. Some state-of-the-art studies inspire this work. Liu et al. [13] introduced a two-step node allocation strategy, which performs well in reducing the probability of nodes being wrongly assigned. Biswas et al. [9] suggested providing opportunities for all nodes to develop communities while removing faulty or redundant anchors. Ding et al. [14] showed that the reallocation of community boundaries helps to further optimize the quality of fully expanded communities. To solve the poor fault tolerance problem, we attempt to enable local expansion methods to discard low-quality seeds and communities in successive iterations. The major contributions of this paper are as follows:

- Based on our previous work [14], we introduce a more robust local-expansion-based overlapping community detection algorithm, named CEO, performing Construction, Expansion and Optimization sub-processes.
- To solve the poor fault tolerance problem, we enable the construction, expansion and optimization sub-processes to discard low-quality seeds and communities and reallocate community boundaries.
- We compared CEO to thirteen noted algorithms by examining the performance on five groups of artificial networks and sixteen real-world networks with ground-truth communities. Experimental results showed CEO performs the best in identifying overlapping communities, which verifies the robustness of CEO in solving the poor fault tolerance problem.

The rest of this work is organized as follows. Related work on is outlined in Section 2. Section 3 provides the motivation, definitions and algorithms. Experimental results are displayed in Section 4. Section 5 concludes this paper.

2. Related work

Section 2.1 outlines local expansion methods from the perspective of seeding and expansion methods. Section 2.2 outlines other overlapping community detection methods.

2.1. Local expansion methods

2.1.1. Seeding methods

The simplest seeding methods randomly select nodes as seeds; however, randomness may cause methods to generate low-quality seeds [15]. On the one hand, dense subgraphs such as the *k-clique*, *k-club*, *k-truss*, *k-plex*, *k-core*, *k-dense*, *k-clique-star*, etc. can serve as seeds [16]. On the other hand, the *degree centrality*, *k-shell centrality*, *coreness*, *Hirsch index*, etc. can be employed to generate opinion leaders as seeds [17]. In addition, various technologies have been developed to get high-quality seeds. Whang et al. [18] proposed to use low-conductance clusters as seeds. Long [19] suggested building communities on closely connected node pairs evaluated by the *edge intensity*. Rhouma et al. [20] introduced the *node importance* combining neighborhood connectivity with node influence.

2.1.2. Expansion methods

Most expansion methods optimize communities based on quality functions. Quality functions based on network models, internal connectivity, external connectivity, etc. can be used to guide expansion methods to generate highly clustered communities [21]. Usually, the expansion methods only performing node addition operations work well in maintaining the core position of seeds in detected communities [22], while the expansion methods also performing node removal operations are good at recognizing communities with specific local structures [23]. In addition, few similarity-based expansion methods optimize node memberships, which excel in identifying diversely structured communities [14].

2.2. Other overlapping community detection methods

2.2.1. Clique percolation methods

Cliques are mostly made up of links within communities rather than links between communities. CPM (Clique Percolation Method) [24] is designed to find connected components composed of adjacent k-cliques sharing $k-1$ common nodes. Cliques in different communities may share common nodes, so overlaps between communities are possible. CPM suffers from high time complexity [25]. To address this problem, Zhang et al. [26] introduced WCPM (Weak-Clique Percolation Method). Since a *weak-clique* is determined by the common neighbors of two adjacent nodes, the time complexity of WCPM is linear to the number of links of the network.

2.2.2. Link partitioning methods

A node in the network is an overlapping node if the links connected to it belong to multiple clusters. Ahn et al. [7] first introduced a link clustering algorithm, named Links, for overlapping community detection. In Links, the *Jaccard* index is used to compute the similarity between a pair of links sharing a common node, and the single-linkage hierarchical clustering is employed for handling large-scale networks. Enjoying the notion of the line graph, many disjoint community detection methods can be extended to detect overlapping communities [27].

2.2.3. Agent-based dynamical methods

A community can be regarded as a group of nodes that reach a consensus on a unique label. The label propagation algorithm [28] first initializes each node with a unique label, and then iteratively updates the label of each node to the label currently owned by most of its neighbors. By providing each node with a memory to store received labels, SLPA (Speaker-listener Label Propagation Algorithm) [29] can identify overlapping communities. SLPA converts the memory of a node into the probability distribution of labels defining the strength of association between nodes and communities. Technologies such as game theory, random walk, etc. are also commonly employed by agent-based dynamical methods [30,31].

2.2.4. Fuzzy detection methods

Overlapping community detection methods can be classified as fuzzy and crisp methods in terms of membership degree. Fuzzy methods calculate a soft membership vector that defines the membership degree with corresponding communities for each node. Crisp methods can be extended to fuzzy methods. Su and Havens [32] suggested several heuristics for soft modularity maximization. Zhang et al. [33] proposed a fuzzy label propagation algorithm that iteratively propagates membership degrees of all nodes. Biswas and Biswas [9] proposed a fuzzy agglomerative algorithm that iteratively updates membership degrees of nodes.

2.2.5. Multi-objective evolutionary methods

The community detection problem can be modeled as a multi-objective optimization problem [10]. Evolutionary computation can solve multi-objective optimization problems. An evolutionary method first initializes a population and then performs variation and selection operators to improve the value of certain criterion. For example, Liu et al. [34] proposed MEASSN (Multi-objective Evolutionary Algorithm for handling Signed Social Networks). In MEASSN, a gain function and a loss function are used to model the problem of community detection as a multi-objective problem, and a direct and indirect combined representation is suggested for overlapping community detection.

2.2.6. Learning-based methods

Deep learning technologies have been applied to overlapping community detection. Jin et al. [35] divided the learning-based methods into PGM-based (Probabilistic Graphical Model) methods and DL-based (Deep Learning) methods. Yang and Leskovec [36] presented BIGCLAM (Cluster Affiliation Model for Big Networks), which is a PGM-based method. First, BIGCLAM projects node memberships to a bipartite affiliation network where each node connects to the communities to which it belongs. Second, BIGCLAM sets a non-negative weight to each affiliated connection by assigning a non-negative latent factor to each node-community pair. Last, BIGCLAM combines the non-negative matrix factorization with block stochastic gradient descent to estimate the non-negative latent factors. Bakshi et al. [37] proposed Bespoke, which is a DL-based method. For node labeling, Bespoke first compresses the distribution of the *Jaccard* score of node pairs into feature vectors and then uses the *k-means* for clustering and labeling. For training, Bespoke first extracts training communities following the distribution of the size of communities and then extracts clique-like testing communities having the same representation with the training communities. Recently, Jabbour et al. [38] proposed to formulate the community detection problem into the Max-SAT (Maximum SATisfiability) problem and put forward CDSAT (SAT-based Community Detection) taking WPM3 [39] as a Max-SAT solver.

3. Methods

3.1. Motivation

We take the behavior of LFM (Local Fitness Maximization) [15] on Karate (the network of the members of a karate club) [40] as an example to show the poor fault tolerance problem of local expansion methods. Suppose the community C_{v_i} is identified by the seed v_i. Fig. 1 displays ground-truth communities of Karate, where C_{v_1} and $C_{v_{33}}$ are ground-truth communities formed around the administrator (v_1) and the instructor (v_{33}). Table 1 lists communities generated by LFM based on each node of Karate, where seeds in the same row result in the same community. We can see that the communities result from

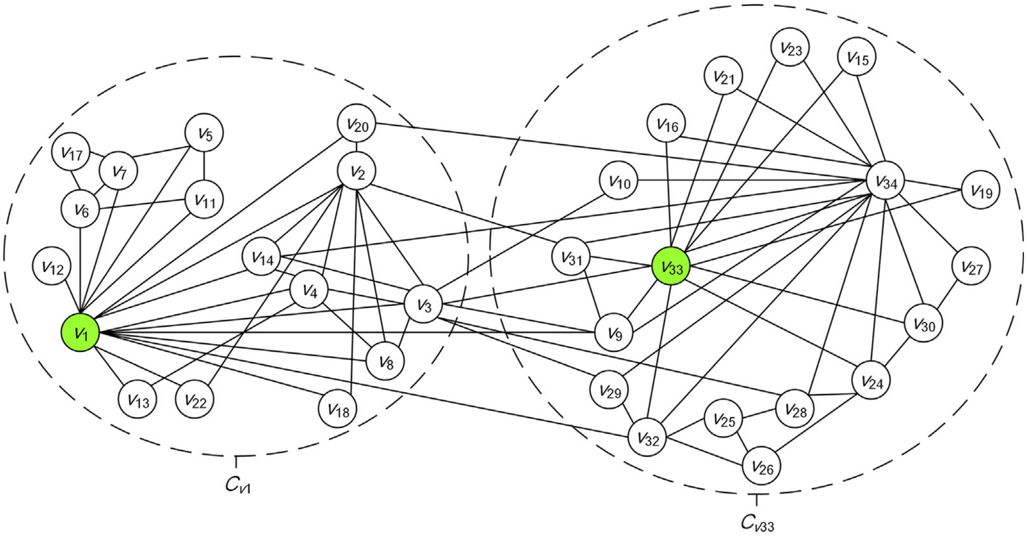

Fig. 1. The ground-truth communities of Karate.

Table 1
The communities generated by LFM based on each node of Karate.

Seeds	Resulting communities
$v_1, v_2, v_3, v_4, v_8, v_{10}, v_{12}, v_{13}, v_{14}, v_{18}, v_{20}, v_{22}$	$\{v_1, v_2, v_3, v_4, v_8, v_9, v_{10}, v_{12}, v_{13}, v_{14}, v_{18}, v_{20}, v_{22}, v_{31}\}$
$v_5, v_6, v_7, v_{11}, v_{17}$	$\{v_5, v_6, v_7, v_{11}, v_{17}\}$
$v_9, v_{15}, v_{16}, v_{19}, v_{21}, v_{23}, v_{31}, v_{33}, v_{34}$	$\{v_9, v_{10}, v_{15}, v_{16}, v_{19}, v_{21}, v_{23}, v_{24}, v_{27}, v_{28}, v_{30}, v_{31}, v_{33}, v_{34}\}$
v_{24}, v_{28}	$\{v_{24}, v_{25}, v_{26}, v_{28}, v_{29}, v_{32}\}$
$v_{25}, v_{26}, v_{29}, v_{32}$	$\{v_{25}, v_{26}, v_{29}, v_{32}\}$
v_{27}, v_{30}	$\{v_3, v_9, v_{10}, v_{15}, v_{16}, v_{19}, v_{21}, v_{23}, v_{24}, v_{25}, v_{26}, v_{27}, v_{28}, v_{29}, v_{30}, v_{31}, v_{32}, v_{33}, v_{34}\}$

the seeds in the first and third rows of Table 1 are highly similar to C_{v_1} and $C_{v_{33}}$ in Fig. 1. We regard the seeds and communities in the first and third rows of Table 1 as high-quality seeds and communities, whereas the others are low-quality seeds and communities. People expect an algorithm to generate high-quality communities based on high-quality seeds. However, low-quality seeds may cause the algorithm to generate low-quality communities, and low-quality communities may hinder the algorithm from identifying high-quality seeds. From Table 1, we can find that if LFM first generates $C_{v_{27}}$ based on v_{27}, then LFM cannot generate the high-quality community based on the seeds in the third row of Table 1 which have been assigned to $C_{v_{27}}$ and have no chance of being selected as seeds.

Various technologies have been developed to get high-quality seeds and communities; however no evidence shows that existing technologies never cause the algorithm to generate low-quality seeds and communities. Therefore, to solve the poor fault tolerance problem, we shift the focus of our study from identifying high-quality seeds and communities to discarding low-quality seeds and communities. Our motivation is as follows:

- Why do we discard low-quality seeds and communities? We think it is unreasonable that once the seeding process produces a seed, the expansion process should produce a fully expanded community. Since low-quality communities may hinder the algorithm from identifying high-quality seeds, a fully expanded low-quality community may cause more serious negative effects than an under-expanded one.
- How do we identify low-quality seeds and communities? We hold that low-quality seeds and communities should be discarded according to the following two rules. First, once a node is covered by some communities, the covered node becomes a low-quality seed that will be discarded. Second, once a community is completely covered by another community, the covered community becomes a low-quality community that will be discarded.
- Where do we discard low-quality seeds and communities? We believe that the formation of communities requires the following three processes. First, a construction process produces the bases of communities. Second, an expansion process produces communities worthy of further expansion. Third, an optimization process produces fully expanded communities.

Integrating our motivation, based on optimizing node memberships [14], we propose CEO which identifies and discards low-quality seeds and communities following the above two rules in the above three processes.

3.2. Definitions

$G = (V, E)$ denotes a simple graph with $n = |V|$ nodes and $m = |E|$ links. $\mathbf{A}_{\mathbf{n}\times\mathbf{n}}$ denotes the adjacency matrix of G. If $u, v \in V, \exists(u, v) \in E$, then u, v are neighbors of each other, $A_{uv} = 1$; otherwise, $A_{uv} = 0$. A community of G is a node set $C = \{v_i, \ldots, v_j\}, C \subseteq V$. Overlapping community detection aims to get a cover of the graph $\mathbf{C} = \{C_i, \ldots, C_j\}, \mathbf{C} \subseteq V$, where $\exists C_i \cap C_j \neq \emptyset, i \neq j$.

Definition 1 (*Node neighborhood [14]*). The neighborhood of v is a node set composed of v and the neighbors of v. The neighborhood of v denoted as $\Gamma(v)$ is defined as follows:

$$\Gamma(v) = \{v\} \cup \{u | u \in V, A_{uv} = 1\}, v \in V \tag{1}$$

In Fig. 2 (a), v_4, v_5 and v_8 are neighbors of v_9. Thus, $\Gamma(v_9) = \{v_9\} \cup \{v_4, v_5, v_8\} = \{v_4, v_5, v_8, v_9\}$. Eq. 1 is used to initialize communities.

Definition 2 (*Node expansibility [14]*). The expansibility of v is measured by the number of internal links of the derived subgraph of $\Gamma(v)$. The expansibility of v denoted as $ne(v)$ is defined as follows:

$$ne(v) = \frac{1}{2} \sum_{i,j \in \Gamma(v)} A_{ij}, v \in V \tag{2}$$

In Fig. 2 (b), the derived subgraph of $\Gamma(v_9)$ has five internal links $(v_4, v_5), (v_4, v_8), (v_4, v_9), (v_5, v_9)$ and (v_8, v_9). Thus, $ne(v_9) = 5$. Eq. 2 is used to determine the order in which seeds are selected to be processed.

Definition 3 (*Community boundaries [14]*). The boundaries of C are a set of nodes composed of members of C that have at least one neighbor outside C. The boundaries of C denoted as $B(C)$ are defined as follows:

$$B(C) = \{u | u \in C, \exists v \in V, v \notin C, A_{uv} = 1\}, C \subseteq V \tag{3}$$

In Fig. 2 (c), v_2 has a neighbor v_5 outside C_{v_7}, v_6 has a neighbor v_5 outside C_{v_7}, v_{10} has a neighbor v_5 outside C_{v_7}, while other members of C_{v_7} have no neighbors outside C_{v_7}. Thus, $B(C_{v_7}) = \{v_2, v_6, v_{10}\}$. Eq. 3 is used to get the members of a community that interact with the rest of the network.

Definition 4 (*Community neighbors [14]*). The neighbors of C are a set of nodes composed of non-members of C that have at least one neighbor in C. The neighbors of C denoted as $N(C)$ are defined as follows:

$$N(C) = \{u | u \in V, u \notin C, \exists v \in C, A_{uv} = 1\}, C \subseteq V \tag{4}$$

In Fig. 2 (d), v_5 has neighbors v_2, v_6 and v_{10} in C_{v_7}, while other non-members of C_{v_7} have no neighbors in C_{v_7}. Thus, $N(C_{v_7}) = \{v_5\}$. Eq. 4 is used to get the non-members of a community that interact with the community.

Definition 5 (*Node-community similarity [14]*). The similarity between v and C is measured by the number of internal links of the derived subgraph of $\Gamma(v) \cap C$. The similarity between v and C denoted as $ncs(v, C)$ is defined as follows:

$$ncs(v, C) = \frac{1}{2} \sum_{i,j \in (\Gamma(v) \cap C)} A_{ij}, v \in V, C \subseteq V \tag{5}$$

In Fig. 2 (e), the derived subgraph of $\Gamma(v_5) \cap C_{v_7}$ has one internal link (v_5, v_6). Thus, $ncs(v_5, C_{v_7}) = 1$. Eq. 5 is used to optimize node memberships.

Definition 6 (*Community expansibility*). The expansibility of C is measured by the number of external links of C. The expansibility of C denoted as $ce(C)$ is defined as follows:

$$ce(C) = \sum_{i \in B(C)} \sum_{j \in (V - C)} A_{ij}, C \subseteq V \tag{6}$$

In Fig. 2 (f), C_{v_7} has three external links $(v_2, v_5), (v_5, v_6)$ and (v_5, v_{10}). Thus, $ce(C_{v_7}) = 3$. If $C_{v_i} = \{v_i\}$, then $ce(C_{v_i}) = ne(v_i)$. Eq. 6 is used to determine the order in which communities are selected to be processed.

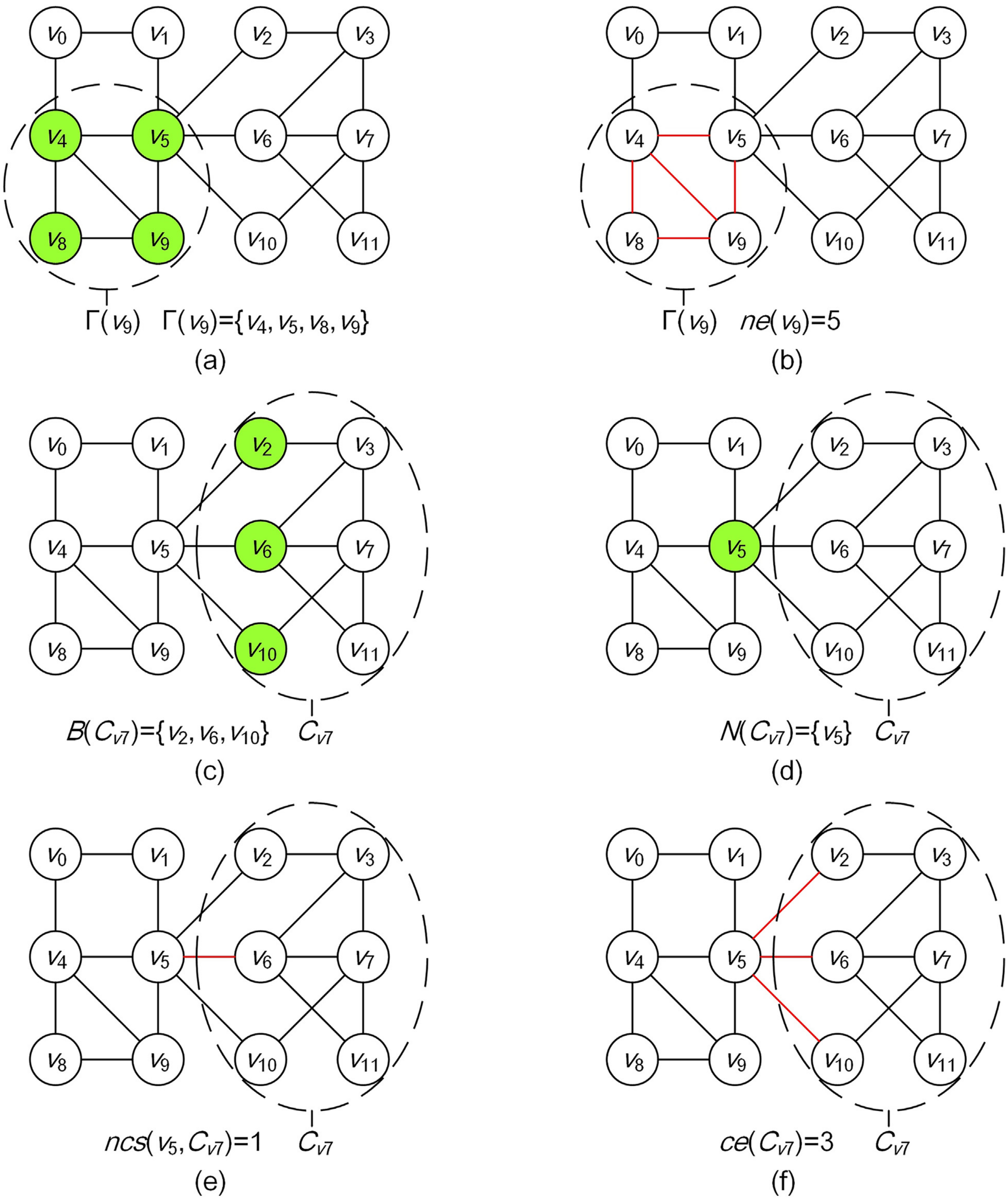

Fig. 2. Examples used to explain Definitions 1 to 6. Figures (a), (b), (c), (d), (e), (f) are used to explain Definitions 1–6, respectively.

3.3. Algorithms

Algorithm 1: The proposed algorithm (CEO).

Input: Network $G = (V, E)$, Node set V, Link set E.
Output: The cover of the network $\mathbf{C}$.
1: **The construction process:**
2: $V^{temp} = V$;
3: $\mathbf{C^{temp}} = \varnothing$;
4: For $\forall v \in V$, calculate $ne(v)$ based on Eq. 2;
5: **while** $V^{temp} \neq \varnothing$ **do**
6: $v^{seed} = \arg\max_{v \in V^{temp}} ne(v)$;
7: $C_{v^{seed}} = \Gamma(v^{seed})$;
8: Get $B(C_{v^{seed}})$ based on Eq. 3;
9: **while true do**
10: $U = \{v | v \neq v^{seed}, v \in B(C_{v^{seed}}), ncs(v, C_{v^{seed}}) < ncs(v, V - C_{v^{seed}})\}$;
11: **if** $U \neq \varnothing$
12: $C_{v^{seed}} = C_{v^{seed}} - U$;
13: Update $B(C_{v^{seed}})$ based on Eq. 3;
14: **else**
15: **break**;
16: **end while**
17: $\mathbf{C^{temp}} = \mathbf{C^{temp}} - \left\{C | C \in \mathbf{C^{temp}}, C \subseteq C_{v^{seed}}\right\}$;
18: $V^{temp} = V^{temp} - C_{v^{seed}}$;
19: $\mathbf{C^{temp}} = \mathbf{C^{temp}} \cup \{C_{v^{seed}}\}$;
20: **end while**
21: **The expansion process:**
22: $\mathbf{C} = \varnothing$;
23: For $\forall C \in \mathbf{C^{temp}}$, calculate $ce(C)$ based on Eq. 6;
24: **while** $\mathbf{C^{temp}} \neq \varnothing$ **do**
25: $C^{seed} = \arg\max_{C \in \mathbf{C^{temp}}} ce(C)$;
26: Get $N\left(C^{seed}\right)$ based on Eq. 4;
27: $C^{seed} = C^{seed} \cup \left\{v | v \in N\left(C^{seed}\right), \forall C \in \left(\mathbf{C} \cup \mathbf{C^{temp}}\right), ncs\left(v, C^{seed}\right) \geqslant ncs(v, C)\right\}$;
27: $\mathbf{C^{temp}} = \mathbf{C^{temp}} - \left\{C | C \in \mathbf{C^{temp}}, C \subseteq C^{seed}\right\}$;
29: $\mathbf{C} = \mathbf{C} - \left\{C | C \in \mathbf{C}, C \subseteq C^{seed}\right\}$;
30: $\mathbf{C} = \mathbf{C} \cup \left\{C^{seed}\right\}$;
31: **end while**
32: **The optimization process:**
33: Perform the boundary rechecking process in [14] for optimizing $\mathbf{C}$;
34: **return** $\mathbf{C}$;

This section provides CEO. First, the construction process constructs absolutely stable communities based seeds. Second, the expansion process expands absolutely stable communities into relatively stable communities. Finally, the optimization process optimizes relatively stable communities into fully expanded communities. The above three kinds of communities are defined by CEO via optimizing node memberships.

Algorithm 1 offers the pseudo-code of CEO, where V^{temp} is a sequence keeping nodes that have not been assigned to any community, $\mathbf{C^{temp}}$ is a sequence keeping absolutely stable communities, v^{seed} (C^{seed}) is the seed (community) selected to be processed. The explanation of **Algorithm** 1 is detailed in the following paragraphs.

The construction process (lines 2 to 20) In the beginning, V^{temp} is initialized to all nodes in the network (line 2), $\mathbf{C^{temp}}$ is initialized to empty (line 3), the expansibility of each node in the network is calculated based on Eq. 2 (line 4). Fig. 3 (a) shows the initialized V^{temp} and $\mathbf{C^{temp}}$, the expansibility of all nodes in the network, and the original network. At each iteration, the

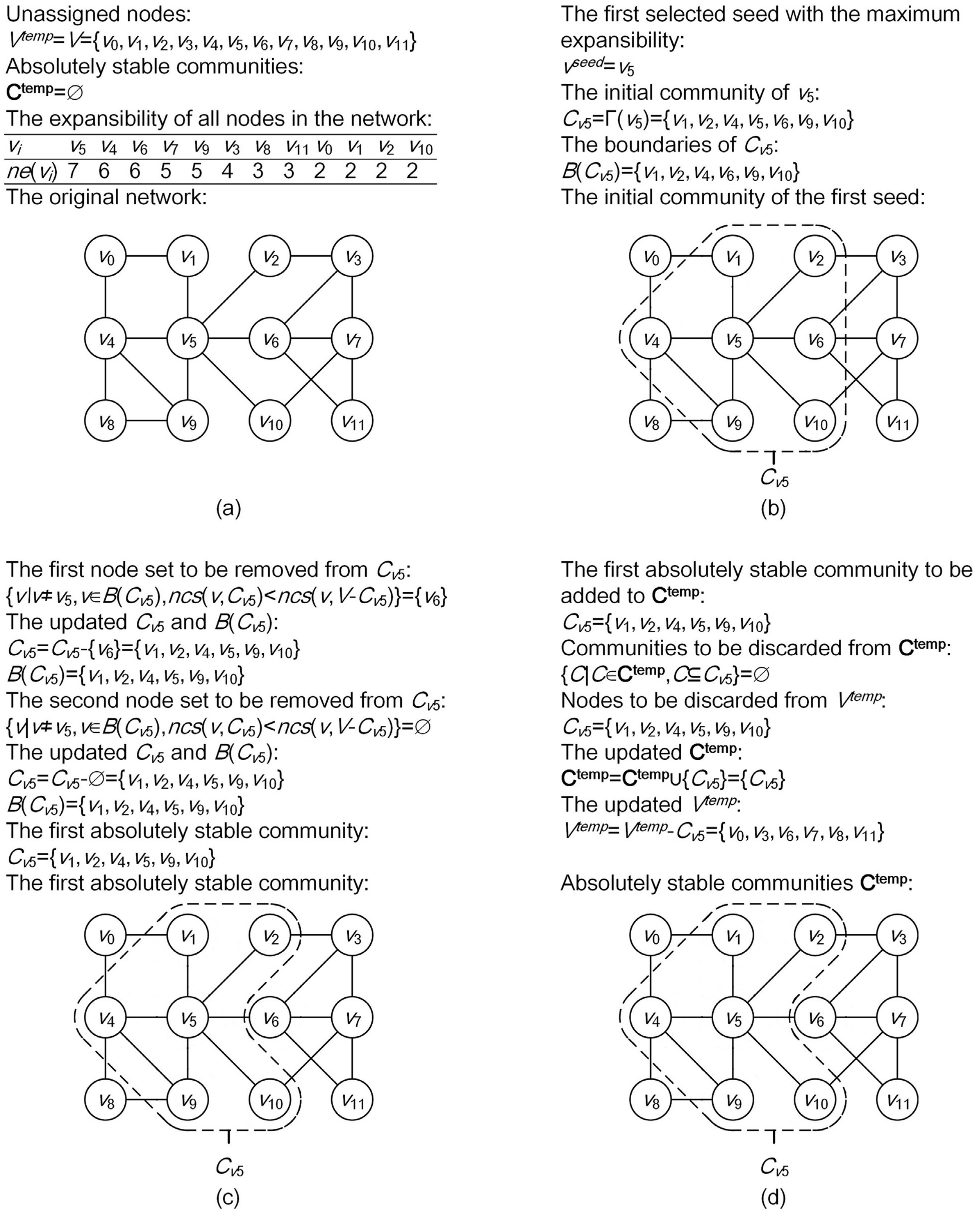

Fig. 3. Examples used to explain the construction process. Figures (a), (b), (c) and (d) show the changes in the state of communities from the beginning of the construction process to the generation of the first absolutely stable community.

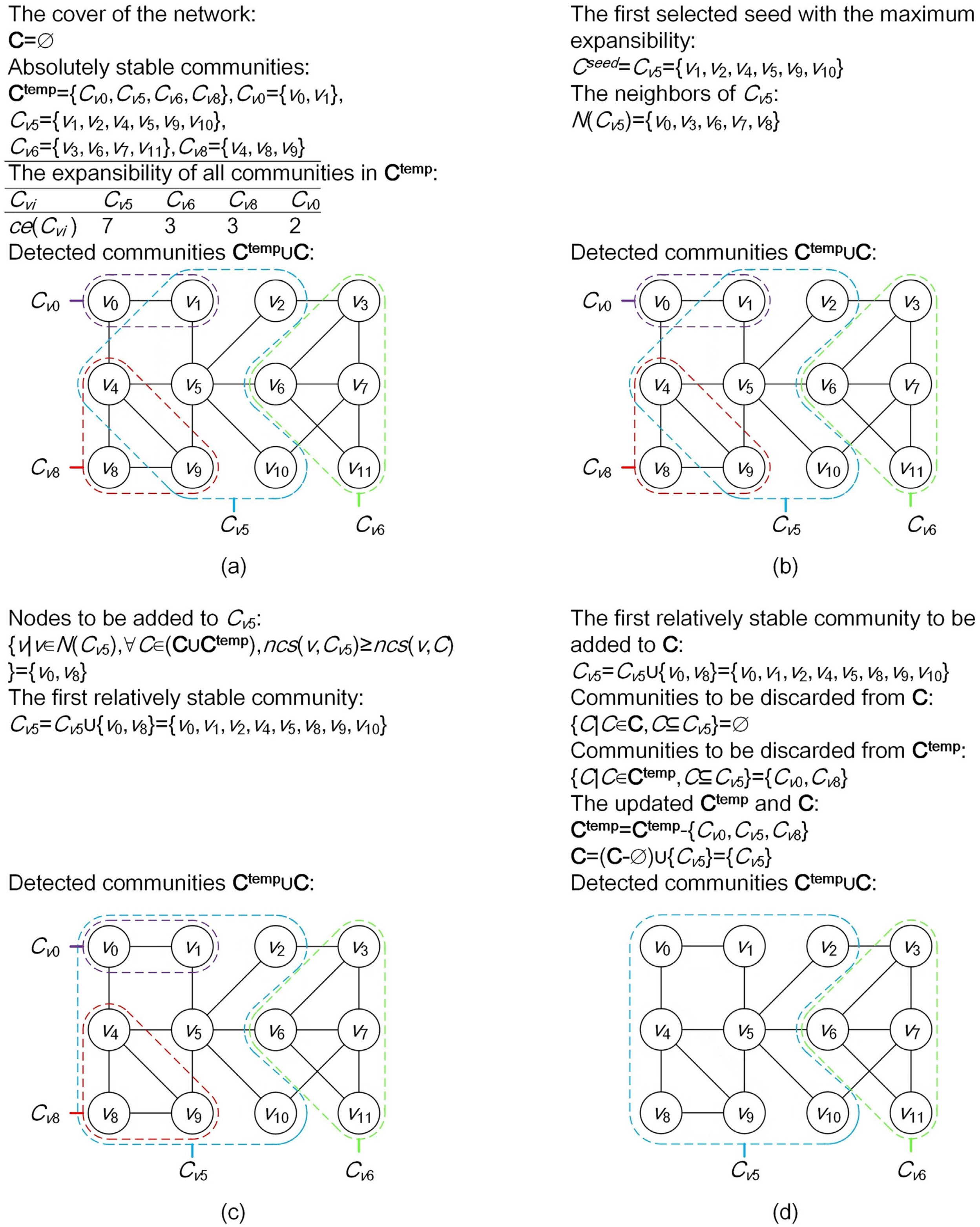

Fig. 4. Examples used to explain the expansion process. Figures (a), (b), (c) and (d) show the changes in the state of communities from the beginning of the expansion process to the generation of the first relatively stable community.

v^{seed} with the maximum expansibility is first selected from V^{temp} (line 6). Accordingly, $C_{v^{seed}}$ and $B(C_{v^{seed}})$ are initialized based on Eqs. 1 and 3 (lines 7 and 8). Fig. 3 (b) shows the first selected seed v^{seed}, the initialized community $C_{v^{seed}}$, and the initialized boundaries $B(C_{v^{seed}})$ of $C_{v^{seed}}$. Afterwards, based on Eq. 5, $C_{v^{seed}}$ shrinks into the absolutely stable community by iteratively removing its boundaries that are less similar to it than to the rest of the network (line 12). Accordingly, $B(C_{v^{seed}})$ has to be updated after each removal operation based on Eq. 3 (line 13). When no nodes other than v^{seed} can be removed from $C_{v^{seed}}$ an absolutely stable community is generated. Fig. 3 (c) shows the nodes to be removed from $C_{v^{seed}}$ at each iteration, the updated $C_{v^{seed}}$ and $B(C_{v^{seed}})$ after each removal, and the first generated absolutely stable community. Once a new community is generated, low-quality seeds and communities have to be discarded from $\mathbf{C}^{\mathbf{temp}}$ and V^{temp} (lines 17 and 18), and then the newly generated community has to be added to $\mathbf{C}^{\mathbf{temp}}$ (line 19). Fig. 3 (d) shows the first absolutely stable community to be added to $\mathbf{C}^{\mathbf{temp}}$, low-quality seeds and communities to be discarded, and the updated V^{temp} and $\mathbf{C}^{\mathbf{temp}}$. The construction process stops when V^{temp} becomes empty (line 5).

The expansion process (lines 22 to 31) In the beginning, the cover of the network $\mathbf{C}$ is initialized to empty (line 22), the expansibility of each community in $\mathbf{C}^{\mathbf{temp}}$ is calculated based on Eq. 6 (line 23). Fig. 4 (a) shows the initialized $\mathbf{C}$, $\mathbf{C}^{\mathbf{temp}}$ resulting from the construction process, and the expansibility of all communities in $\mathbf{C}^{\mathbf{temp}}$. At each iteration, the C^{seed} with the maximum expansibility is first selected from $\mathbf{C}^{\mathbf{temp}}$ (line 25), and $N\left(C^{seed}\right)$ is initialized based on Eq. 4 (line 26). Fig. 4 (b) shows the first selected community C^{seed}, the initialized $N\left(C^{seed}\right)$, and detected communities. Afterwards, based on Eq. 5, C^{seed} expands into the relatively stable community by absorbing its neighbors that are not less similar to it than to previously detected communities (line 27). Fig. 4 (c) shows the nodes to be added to C^{seed}, the first generated relatively stable community, and detected communities. Once a new community is generated, low-quality communities have to be discarded from $\mathbf{C}$ and $\mathbf{C}^{\mathbf{temp}}$ (lines 28 and 29), and then the newly generated community has to be added to $\mathbf{C}$ (line 30). Fig. 4 (d) shows the first relatively stable community to be added to $\mathbf{C}$, low-quality communities to be discarded, and the updated $\mathbf{C}^{\mathbf{temp}}$, $\mathbf{C}$ and detected communities. The expansion process stops when $\mathbf{C}^{\mathbf{temp}}$ becomes empty (line 24).

The optimization process (line 33) To optimize the cover of the network, Ding et al. [14] introduced the boundary rechecking process that works by iteratively moving nodes between neighboring communities based on optimizing node memberships. The boundary rechecking process stops when each node is not less similar to its community than the others in the cover of the network. CEO adopts the boundary rechecking process to optimize relatively stable communities into fully expanded communities (line 33).

$|\mathbf{C}|$ denotes the number of detected communities, $\overline{|C|}$ denotes the average size of the community, and $\bar{d}$ denotes the mean degree of the network. The time complexity of constructing V^{temp} and $\mathbf{C}^{\mathbf{temp}}$ based on Maximum Heap is $O(nlogn)$ and $O(|\mathbf{C}|log|\mathbf{C}|)$, respectively. The time cost of calculating node expansibility is $O\left(\bar{d}^2\right)$. The time cost of calculating community expansibility is $O\left(\bar{d}\overline{|C|}log\overline{|C|}\right)$. The time cost of calculating node-community similarity is $O\left(\bar{d}log\overline{|C|}+\bar{d}^2\right)$. The time complexity of the construction process is $O\left(nlogn+|\mathbf{C}|log|\mathbf{C}|+n\bar{d}^2+\overline{|C|}\left(\bar{d}log\overline{|C|}+\bar{d}^2\right)\right)$. The time complexity of the expansion process is $O\left(|\mathbf{C}|\overline{|C|}\left(\bar{d}log\overline{|C|}+\bar{d}^2\right)\right)$. The time complexity of the optimization process is $O\left(nlogn+|\mathbf{C}|mlog\overline{|C|}+m\bar{d}log\overline{|C|}+m\bar{d}^2\right)$ [14]. Since in large-scale networks $|\mathbf{C}| \gg \bar{d}$, the total time complexity of CEO is $O\left(nlogn+m\left(|\mathbf{C}|log\overline{|C|}+\bar{d}^2\right)\right)$.

4. Experiments

4.1. Experimental settings

4.1.1. Experimental environment

In the experiments, all algorithms were programmed in Java and performed on a computer with Intel(R) Core(TM) i7-8565U CPU, 1.80 GHz, 8 GB RAM. All algorithms were allowed to run on each network for 24 h before any results were gen-

Table 2
The characteristics of the proposed and compared algorithms.

Algorithms	Quality functions	Similarity indices	Time complexity
CEO	Not required	Required	$O\left(nlogn+m\left(\lvert\mathbf{C}\rvert log\overline{\lvert C\rvert}+\bar{d}^2\right)\right)$
LFM	Required	Not required	$O(\lvert\mathbf{C}\rvert nlogn)$
CFM	Required	Not required	$O(mlogm)$
TWD	Required	Not required	$O\left(\lvert\mathbf{C}\rvert nlogn+m\bar{d}\right)$
LECS	Required	Not required	$O(\lvert\mathbf{C}\rvert nlogn+m)$
LEBR	Not required	Required	$O\left(nlogn+m\left(\lvert\mathbf{C}\rvert log\overline{\lvert C\rvert}+\bar{d}^2\right)\right)$
OCLN	Not required	Required	$O(m)$

erated. In our experiment, there are three situations where algorithms may fail to generate communities: (1) OoT (Out of Time), (2) OoM (Out of Memory), (3) LoT (Lack of Training communities).

4.1.2. Compared algorithms

CEO was compared to six local-expansion-based algorithms, including LFM (Local Fitness Maximization) [15], CFM (Community Forest Model) [41], TWD (Three-Way Decision) [42], LECS (Local Expansion based on Core Similarity) [23], LEBR ($LEBR_{asc}$) (Local Expansion and Boundary Rechecking) [14] and OCLN (Overlapping Community detection algorithm using Local Neighborhood information) [43]. LFM is a typical local expansion method, so we use LFM as a baseline for quality analysis. OCLN is an approximately linear method, so we use OCLN as a baseline for efficiency analysis. TWD improves the seed-

Table 3
The parameter settings of artificial networks.

Networks	n	$\bar{d}$	d_{max}	$\|C\|_{min}$	$\|C\|_{max}$	μ	O_n	O_m
LFR-μ	10000	5	500	5	500	[0.1:0.1:0.8]	500	2
LFR-$\|C\|_{max}$	10000	5	500	5	[200:200:1000]	0.1	500	2
LFR-d_{max}	10000	5	[200:200:1000]	5	500	0.1	500	2
LFR-O_n	10000	5	500	5	500	0.1	[200:200:1000]	2
LFR-α_n	$5 \times 10^{[1:1:5]}$	5	$10 \times 5^{[1:1:5]}$	5	$20 \times 5^{[1:1:5]}$	0.1	$2 \times 5^{[1:1:5]}$	2

Table 4
The results on LFR-μ in terms of NMI.

μ	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
0.1	0.1303	0.1185	**0.6329**	0.3904	0.1063	0.302	0.0102	0.2723
0.2	0.0654	0.0569	0.3936	0.1796	0.0636	0.1432	0.0013	0.1709
0.3	0.0434	0.0369	0.2204	0.079	0.0342	0.0602	0.0002	0.1006
0.4	0.0153	0.0158	0.1555	0.0301	0.0104	0.0095	0.0001	0.037
0.5	0.0076	0.0077	**0.1339**	0.0054	0.0027	0.0012	0.0	0.0060
0.6	0.0032	0.0015	**0.0767**	0.0015	0.0008	0.0009	0.0	0.0007
0.7	0.0003	0.0001	**0.0255**	0.0003	0.0	0.0016	0.0	0.0
0.8	0.0	0.0	**0.0146**	0.0001	0.0001	0.0009	0.0	0.0
μ	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
0.1	0.441	0.1224	0.1271	0.5333	0.1645	OoT	0.2173	0.1538
0.2	**0.4305**	0.0248	0.0252	0.3072	0.0136	OoT	0.0493	0.0619
0.3	**0.3772**	0.0044	0.0	0.1045	0.0	OoT	0.0146	0.0237
0.4	**0.1755**	0.0041	0.0	0.0265	0.0	OoT	0.0056	0.0042
0.5	0.0544	0.0022	0.0	0.0058	0.0	OoT	0.0016	0.0002
0.6	0.0	0.0014	0.0	0.0019	0.0	OoT	0.0007	0.0
0.7	0.0	0.0016	0.0	0.0012	0.0	OoT	0.0	0.0
0.8	0.0	0.0010	0.0	0.0005	0.0	OoT	0.0	0.0

Table 5
The results on LFR-μ in terms of F-Score.

μ	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
0.1	0.5759	0.4868	**0.8648**	0.7644	0.6367	0.7359	0.3578	0.771
0.2	0.4719	0.3863	**0.7411**	0.6047	0.5426	0.5945	0.2209	0.6952
0.3	0.4151	0.3149	0.5306	0.4753	0.4398	0.4584	0.157	**0.5861**
0.4	0.3586	0.2353	0.3108	0.3889	0.3191	0.3436	0.1133	**0.4644**
0.5	0.3158	0.1702	0.1967	0.3169	0.2425	0.2448	0.1035	**0.3296**
0.6	**0.2764**	0.1129	0.0484	0.2715	0.1973	0.1851	0.0752	0.244
0.7	**0.2252**	0.0926	0.0365	0.2201	0.1526	0.1206	0.0729	0.1687
0.8	**0.1683**	0.0858	0.0206	0.1632	0.1262	0.0962	0.0591	0.1193
μ	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
0.1	0.2515	0.0622	0.0279	0.8577	0.4067	OoT	0.5029	0.6218
0.2	0.1372	0.0361	0.0233	0.709	0.2761	OoT	0.3979	0.5062
0.3	0.0717	0.0324	0.0208	0.525	0.1937	OoT	0.3169	0.4154
0.4	0.0173	0.0391	0.0217	0.3868	0.1362	OoT	0.2511	0.3313
0.5	0.0039	0.0423	0.021	0.2575	0.0966	OoT	0.2023	0.2587
0.6	0.0007	0.0475	0.0197	0.195	0.0714	OoT	0.1598	0.2001
0.7	0.0001	0.0481	0.0188	0.1416	0.0539	OoT	0.1291	0.1474
0.8	0.0001	0.0431	0.0206	0.1144	0.0438	OoT	0.0925	0.1064

ing method of LFM to get high-quality seeds, so we use TWD to present the impact of improving seeding technologies on algorithm quality. LECS improves the expansion method of LFM to get high-quality communities, so we use LECS to present the impact of improving expansion technologies on algorithm quality. CFM selects links as seeds and expands communities by absorbing their external links, so we use CFM as a representative of link-oriented local expansion methods. LEBR and CEO share the same definitions and optimization process, so we use LEBR as a baseline for analyzing the robustness of CEO. The characteristics of the six local-expansion-based algorithms are listed in Table 2. In addition, CEO was compared to seven algorithms of other overlapping community detection methods, including WCPM [26], Links [7], SLPA [29], MEASSN [34], BIGCLAM [36], Bespoke [37] and CDSAT ($CDSAT^k_{max}$) [38]. The characteristics of the seven algorithms of other overlapping community detection methods are outlined in Section 2.2.

4.1.3. Parameter settings

For OCLN, the parameter for controlling the size of communities is set to 2, and the parameter for filtering incorrectly identified nodes is set to 0.2. For WCPM, the threshold for merging weak-cliques is set to 0.6. For SLPA, the maximum iterations is set to 100, and the post-processing threshold is set to 0.05. For MEASSN, the population size is set to $\sqrt{n}$, the number of neighbors and max generation are set to $logn$, the verbose level is set to 2, the exponent of tightness is set to 1, and the probability of crossover and mutation is set to 0.7 and 1. For BIGCLAM, the number of threads for parallelization is set to 4, the parameters for backtracking line search are set to 0.3, and the number of communities c to be detected is set close to the number of ground-truth communities. Particularly, c is set to $|\mathbf{C}|$ for real-world networks, c is set to 100 for LFR-μ, LFR-$|C|_{max}$, LFR-d_{max} and LFR-O_n, and c is set to 5, 10, 50, 150 and 300 for LFR-α_n when α_n is set to 1, 2, 3, 4 and 5, respectively. For

Table 6
The results on LFR-μ in terms of EQ.

μ	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
0.1	0.4005	0.1855	**0.7558**	0.6196	0.2426	0.3872	0.3369	0.5055
0.2	0.3348	0.0928	**0.5838**	0.4368	0.1928	0.3144	0.2949	0.3959
0.3	0.3124	0.0573	**0.3661**	0.3247	0.1711	0.2781	0.2717	0.3242
0.4	0.2921	0.0271	0.1902	0.267	0.155	0.2387	0.2545	0.2711
0.5	0.2842	0.0133	0.1611	0.2413	0.1485	0.2173	0.2456	0.2423
0.6	0.2763	0.0042	0.0308	0.2283	0.1454	0.2095	0.2393	0.2232
0.7	0.2744	0.0018	0.0287	0.2235	0.144	0.1751	0.2348	0.2142
0.8	0.2731	0.0012	0.0	0.222	0.1426	0.1674	0.2346	0.2123
μ	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
0.1	0.0554	0.0071	0.0004	0.5354	0.3889	OoT	0.0878	0.3794
0.2	0.0153	0.0064	0.0001	0.4171	0.284	OoT	0.0589	0.345
0.3	0.0047	0.0086	0.0	0.3441	0.2487	OoT	0.053	0.329
0.4	0.0014	0.0111	0.0	0.3015	0.2285	OoT	0.0462	**0.3158**
0.5	0.0001	0.0151	0.0	0.2746	0.2263	OoT	0.045	**0.3108**
0.6	0.0	0.0165	0.0	0.2284	0.2201	OoT	0.0437	**0.303**
0.7	0.0	0.0176	0.0	0.2068	0.2196	OoT	0.0453	**0.3001**
0.8	0.0	0.0172	0.0	0.2119	0.2235	OoT	0.0448	**0.2994**

Table 7
The results on LFR-μ in terms of D-Score.

μ	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
0.1	26.5	82.04	3.05	13.56	21.28	6.3	6.58	7.66
0.2	29.67	94.43	7.63	23.08	24.35	7.52	7.47	10.64
0.3	28.36	92.36	11.43	29.02	24.14	8.73	7.27	12.57
0.4	30.5	100.78	9.4	35.0	26.36	9.0	7.51	15.23
0.5	30.0	99.41	-1.6	36.35	25.94	9.36	7.91	15.95
0.6	28.37	94.51	-61.38	35.0	24.49	7.73	7.29	15.5
0.7	26.82	90.45	-75.33	33.78	23.35	5.3	7.14	14.85
0.8	29.75	99.12	-89.25	37.26	25.23	5.25	7.83	16.01
μ	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
0.1	-2.09	-4.99	-67.63	12.8	**0.06**	OoT	3.22	4.43
0.2	-4.51	-3.12	-87.35	14.98	**0.08**	OoT	3.75	4.77
0.3	-9.38	-1.84	-98.6	14.76	**0.01**	OoT	3.84	4.46
0.4	-68.2	-1.08	-94.9	15.61	**0.06**	OoT	4.38	4.7
0.5	-88.75	-0.55	-98.1	14.93	**0.02**	OoT	4.42	4.61
0.6	-104.2	-0.48	-104.2	14.33	$-$**0.05**	OoT	4.23	4.29
0.7	-108.9	-0.47	-108.9	13.58	$-$**0.1**	OoT	4.39	3.97
0.8	-99.4	-0.35	-99.4	14.76	**0.0**	OoT	4.67	4.63

Bespoke, the number of labels is set to 4, and the number of subgraph patterns is set to 5, $|\mathbf{C}|/10$ ground-truth communities are extracted as training communities for all networks, and 2000 and $|\mathbf{C}| * 10$ communities are generated as testing communities for artificial and real-world networks, respectively. For CDSAT, the diameter of communities is set to 4.

4.1.4. Evaluation criteria

In our experiments, NMI (Normalized Mutual Information) [15], F-Score [18], EQ (Extended Modularity) [44] and D-Score [23] were used to evaluate the algorithm quality in detecting overlapping communities. NMI is used to evaluate the algo-

Table 8
The results on LFR-μ in terms of Time (s).

μ	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
0.1	6	2	13	12	5	15	192	7
0.2	7	2	16	13	4	21	238	8
0.3	8	2	16	13	4	21	271	9
0.4	8	2	17	13	4	46	276	11
0.5	9	2	21	14	5	82	360	12
0.6	11	3	26	16	7	109	364	15
0.7	19	2	45	12	6	208	413	13
0.8	21	2	48	13	5	232	390	16
μ	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
0.1	**0**	1	169	4	54	OoT	4	17
0.2	**0**	1	165	4	61	OoT	4	18
0.3	**0**	1	164	5	68	OoT	4	19
0.4	**0**	1	164	5	72	OoT	4	19
0.5	**0**	1	157	5	71	OoT	4	19
0.6	**0**	1	155	6	74	OoT	4	20
0.7	**0**	1	159	5	76	OoT	4	19
0.8	**0**	1	155	7	85	OoT	4	20

Table 9
The results on LFR-$|C|_{max}$ in terms of NMI.

$\lvert C\rvert_{max}$	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.2723	0.2389	**0.7142**	0.5654	0.2432	0.4449	0.0222	0.4516
400	0.17	0.1511	**0.6768**	0.4222	0.1325	0.3362	0.0115	0.3153
600	0.1147	0.1053	**0.6398**	0.3436	0.0857	0.2746	0.0099	0.2386
800	0.0891	0.0742	**0.5822**	0.3066	0.0747	0.2493	0.0095	0.2099
1000	0.0832	0.0758	**0.6473**	0.2514	0.0661	0.2345	0.0074	0.2025
$\lvert C\rvert_{max}$	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.4675	0.2425	0.2104	0.6083	0.1833	OoT	0.3	0.2954
400	0.4529	0.1537	0.1365	0.5436	0.1645	OoT	0.2464	0.1972
600	0.4346	0.085	0.1244	0.4921	0.1326	OoT	0.2062	0.1423
800	0.4064	0.0889	0.1108	0.4508	0.1305	OoT	0.1785	0.1083
1000	0.4414	0.0534	0.0785	0.4444	0.1089	OoT	0.1688	0.107

Table 10
The results on LFR-$|C|_{max}$ in terms of F-Score.

$\lvert C\rvert_{max}$	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.6842	0.6076	**0.8878**	0.8366	0.7462	0.7783	0.3797	0.8314
400	0.6164	0.5282	**0.8816**	0.7856	0.6697	0.7529	0.3498	0.7887
600	0.5712	0.4878	**0.8707**	0.7539	0.6204	0.7339	0.3555	0.7683
800	0.5538	0.4531	**0.8593**	0.7333	0.6099	0.7192	0.3074	0.7489
1000	0.5371	0.4569	**0.8822**	0.7029	0.5864	0.7214	0.3312	0.7479
$\lvert C\rvert_{max}$	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.3599	0.0763	0.0182	0.8568	0.3431	OoT	0.5698	0.6629
400	0.3018	0.0714	0.0236	0.8474	0.3798	OoT	0.5053	0.6393
600	0.2651	0.0755	0.0311	0.8539	0.3977	OoT	0.4899	0.6241
800	0.2405	0.0737	0.0382	0.8369	0.3998	OoT	0.4474	0.6103
1000	0.2368	0.0747	0.0405	0.8491	0.4042	OoT	0.4398	0.5965

rithm quality in identifying ground-truth communities in community-level. F-Score is used to evaluate the algorithm quality in identifying ground-truth communities in node-level. EQ is used to evaluate the algorithm quality in identifying highly clustered communities. D-Score is used to evaluate the algorithm quality in estimating the number of communities.

4.1.5. Experimental objective

The main purpose of our experiments is to verify the effectiveness of discarding low-quality seeds and communities in solving the poor fault tolerance problem of local expansion methods. In our experiments, we name CEO that only performs the construction and expansion processes as CEO1; we name CEO that performs the construction, expansion and optimization processes as CEO2; we name LEBR that only performs the local expansion process as LEBR1; we name LEBR that performs the local expansion and optimization processes as LEBR2. The quality of CEO2 represents that of CEO, and the quality of LEBR2 represents that of LEBR. CEOs refer to CEO1 and CEO2, and LEBRs refer to LEBR1 and LEBR2.

Table 11
The results on LFR-$|C|_{max}$ in terms of EQ.

$\|C\|_{max}$	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.5118	0.3446	**0.7908**	0.7278	0.3513	0.4994	0.399	0.6261
400	0.4304	0.2286	**0.7759**	0.6462	0.2619	0.4162	0.3496	0.5354
600	0.3821	0.1619	**0.7549**	0.5868	0.2265	0.3781	0.3256	0.4776
800	0.3581	0.1169	**0.7251**	0.5576	0.2121	0.348	0.3122	0.4497
1000	0.352	0.1119	**0.7438**	0.5041	0.2052	0.3373	0.3136	0.4285
$\|C\|_{max}$	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.1169	0.0132	0.0012	0.6062	0.4895	OoT	0.1398	0.4338
400	0.082	0.0091	0.0006	0.5502	0.4107	OoT	0.1022	0.3917
600	0.0507	0.0089	0.0004	0.5047	0.3605	OoT	0.0812	0.368
800	0.0201	0.0082	0.0005	0.4836	0.3482	OoT	0.074	0.3602
1000	0.0269	0.0079	0.0002	0.4689	0.3271	OoT	0.0642	0.3511

Table 12
The results on LFR-$|C|_{max}$ in terms of D-Score.

$\|C\|_{max}$	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	9.53	29.63	1.21	3.03	7.44	2.2	2.53	2.61
400	19.48	60.24	2.21	8.95	15.75	4.53	5.19	5.71
600	31.33	97.68	3.49	17.51	25.52	7.07	7.93	9.52
800	37.37	118.68	4.93	22.37	30.52	9.23	9.6	11.78
1000	47.55	150.79	5.55	33.73	39.03	12.52	11.83	16.15
$\|C\|_{max}$	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	-1.1	-9.58	-109.41	5.02	**−0.99**	OoT	1.83	1.54
400	-1.43	-5.69	-82.99	9.6	**−0.2**	OoT	2.32	3.26
600	-1.82	-2.43	-62.35	14.91	**0.22**	OoT	3.21	5.26
800	-1.9	-2.2	-54.18	17.94	**0.4**	OoT	3.06	6.26
1000	-2.22	-1.33	-49.85	23.34	**0.75**	OoT	3.72	8.07

Table 13
The results on LFR-$|C|_{max}$ in terms of Time (s).

$\|C\|_{max}$	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	5	1	8	8	5	7	143	5
400	6	1	12	11	5	10	189	6
600	7	2	14	13	5	16	203	7
800	7	2	15	14	4	16	232	8
1000	7	2	15	13	5	17	218	8
$\|C\|_{max}$	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	**0**	1	151	5	57	OoT	4	14
400	**0**	1	162	5	58	OoT	4	16
600	**0**	1	170	6	59	OoT	4	18
800	**0**	1	161	5	54	OoT	4	18
1000	**0**	1	162	5	60	OoT	4	18

We take LEBR1 as a baseline to examine the effectiveness of CEO1 in discarding low-quality seeds and communities, and take LEBR2 as a baseline to examine the robustness of CEO2 in solving the poor fault tolerance problem. Our motivation is as follows:

- CEO1 follows our motivation mentioned in Section 3.1, but LEBR1 follows the general framework of local expansion methods.
- CEO2 and LEBR2 are algorithms based on optimizing node memberships.
- CEO2 and LEBR2 share the same definitions and optimization process.

Once CEO1 has a better average quality than LEBR1, the effectiveness of CEO1 in discarding low-quality seeds and communities can be verified. Once CEO2 has a better average quality than LEBR2, the robustness of CEO2 in solving the poor fault tolerance problem can be verified.

Table 14
The results on LFR-d_{max} in terms of NMI.

d_{max}	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.1106	0.0937	**0.6405**	0.3334	0.1011	0.2831	0.0047	0.2399
400	0.1282	0.1136	**0.6277**	0.3788	0.0953	0.2892	0.0086	0.2564
600	0.1217	0.1047	**0.6221**	0.3534	0.1022	0.2959	0.0081	0.2655
800	0.1337	0.1162	**0.6277**	0.3548	0.1149	0.3032	0.0099	0.2688
1000	0.1443	0.1284	**0.6457**	0.4057	0.1064	0.3304	0.0079	0.2962
d_{max}	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.4313	0.1102	0.095	0.51	0.1502	OoT	0.2079	0.1337
400	0.4496	0.0937	0.0504	0.5175	0.1519	OoT	0.2213	0.1408
600	0.436	0.1047	0.0594	0.5155	0.1507	OoT	0.2114	0.1433
800	0.4101	0.0998	0.1297	0.5178	0.1755	OoT	0.2176	0.1625
1000	0.4367	0.0899	0.099	0.5522	0.1739	OoT	0.2352	0.1784

Table 15
The results on LFR-d_{max} in terms of F-Score.

d_{max}	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.5616	0.4664	**0.869**	0.7383	0.6204	0.7281	0.2955	0.761
400	0.5733	0.4897	**0.8655**	0.7593	0.6265	0.7318	0.3572	0.7629
600	0.5838	0.4992	**0.8683**	0.7576	0.6481	0.73	0.3182	0.7732
800	0.579	0.489	**0.8639**	0.7475	0.6466	0.7349	0.3622	0.7659
1000	0.5803	0.4896	**0.8655**	0.7688	0.6353	0.7431	0.3349	0.7757
d_{max}	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.2425	0.0684	0.0277	0.8446	0.3987	OoT	0.4861	0.6178
400	0.2648	0.0576	0.0238	0.8555	0.394	OoT	0.5065	0.6155
600	0.2688	0.0566	0.0244	0.8542	0.3885	OoT	0.4853	0.6164
800	0.2474	0.056	0.0262	0.851	0.4041	OoT	0.4915	0.6298
1000	0.2651	0.0527	0.0248	0.8593	0.4079	OoT	0.516	0.625

Table 16
The results on LFR-d_{max} in terms of EQ.

d_{max}	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.3798	0.1492	**0.7572**	0.5867	0.2413	0.3754	0.3214	0.4805
400	0.3945	0.1736	**0.7504**	0.614	0.2326	0.3945	0.3381	0.4942
600	0.3858	0.1626	**0.7522**	0.5975	0.2417	0.4	0.3381	0.4983
800	0.403	0.1841	**0.7554**	0.5985	0.2477	0.3925	0.3419	0.5007
1000	0.4132	0.2033	**0.7623**	0.637	0.2473	0.4156	0.3336	0.5274
d_{max}	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.0342	0.0081	0.0004	0.5117	0.396	OoT	0.0891	0.3769
400	0.0513	0.0073	0.0001	0.5209	0.3862	OoT	0.0907	0.3723
600	0.0478	0.0063	0.0003	0.5218	0.3832	OoT	0.0888	0.376
800	0.0407	0.0076	0.0005	0.5281	0.3967	OoT	0.0925	0.3827
1000	0.0598	0.0064	0.0003	0.5422	0.3928	OoT	0.0944	0.3787

4.2. Experimental results on artificial networks

4.2.1. Artificial networks

In our experiments, LFR (Lancichinetti-Fortunato-Radicchi) benchmark was used to produce artificial networks, which are characterized by the heterogeneous distribution of node degrees and community sizes [45]. Table 3 lists the parameter settings of artificial networks. A group of ten artificial networks with ground-truth communities were produced for each benchmark network. Detailed description of the artificial networks can be found in [14].

4.2.2. The results on LFR-μ

Each node has a fraction $1-\mu$ of links with its community and a fraction μ of links with the rest of the network. In LFR-μ, communities become less recognizable with the increase of μ. In Tables 4–6, the results in terms of NMI, F-Score and EQ become worse as μ increases in most cases. It can be seen from Table 7 that LFM, CFM and TWD have more stable results

Table 17
The results on LFR-d_{max} in terms of D-Score.

d_{max}	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	27.21	84.96	3.53	15.29	21.17	6.59	6.98	7.99
400	26.84	83.97	3.88	13.74	21.58	6.75	6.8	7.97
600	26.39	82.17	3.46	14.07	20.84	6.78	6.22	7.55
800	25.0	77.72	3.63	14.21	19.92	5.85	6.4	7.52
1000	24.22	75.58	3.0	11.66	19.95	5.68	6.06	7.01
d_{max}	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	-1.94	-3.87	-77.28	12.73	**0.05**	OoT	2.77	4.55
400	-1.81	-4.18	-84.65	13.04	**0.06**	OoT	3.03	4.45
600	-1.66	-5.23	-72.28	12.86	**0.04**	OoT	2.8	4.27
800	-1.8	-4.44	-65.72	12.26	**0.0**	OoT	2.71	4.21
1000	-1.73	-4.83	-79.1	12.12	**0.01**	OoT	3.1	4.02

Table 18
The results on LFR-d_{max} in terms of Time (s).

d_{max}	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	7	1	13	12	6	9	221	6
400	7	1	13	12	5	11	202	6
600	7	1	13	12	5	12	177	7
800	6	1	13	12	7	12	205	7
1000	6	1	12	11	5	14	178	6
d_{max}	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	**0**	1	142	5	54	OoT	4	17
400	**0**	1	156	4	71	OoT	4	17
600	**0**	1	149	4	77	OoT	4	17
800	**0**	1	157	5	56	OoT	4	17
1000	**0**	1	175	5	75	OoT	4	17

Table 19
The results on LFR-O_n in terms of NMI.

O_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.1286	0.1172	**0.695**	0.3785	0.1014	0.3055	0.0099	0.2804
400	0.1352	0.1196	**0.6795**	0.4086	0.1076	0.3256	0.0084	0.2977
600	0.1185	0.0981	**0.6065**	0.3515	0.0979	0.2792	0.0062	0.2506
800	0.1242	0.1068	**0.5673**	0.3662	0.1039	0.2772	0.0078	0.2641
1000	0.1221	0.1039	**0.5161**	0.3143	0.0934	0.2541	0.0050	0.2329
O_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.4541	0.1539	0.1532	0.572	0.1991	OoT	0.2562	0.1592
400	0.4368	0.1223	0.1101	0.5363	0.1744	OoT	0.242	0.1623
600	0.4366	0.0884	0.0848	0.5138	0.1485	OoT	0.2051	0.1372
800	0.4214	0.1238	0.1008	0.4664	0.141	OoT	0.1868	0.1399
1000	0.4236	0.0976	0.1354	0.4515	0.112	OoT	0.1765	0.1503

in terms of D-Score than CEO, LEBR, LECS and OCLN as μ increases. This is because quality functions lead expansion methods to generate communities with specified characteristics, while similarity indices lead expansion methods to generate diversely structured communities. Table 8 shows that the time cost of most algorithms increases with the increase of μ. In particular, we can see that the time cost of CEOs monotonously increases as the recognizability of communities decreases. This is because that CEOs need much more time to discard low-quality seeds and communities when node memberships become fuzzy. The main findings about the results of CEOs and LEBRs on LFR-μ are as follows. First, CEO1 outperforms LEBR1 in identifying ground-truth communities when $\mu \leq 0.3$ and $0.6 \leqslant \mu \leq 0.7$, whereas LEBR1 outperforms CEO1 in other cases; CEO1 outperforms LEBR1 in identifying highly clustered communities and estimating the number of communities. Second, CEO2 outperforms LEBR2 in identifying ground-truth communities; CEO2 outperforms LEBR2 in identifying highly clustered communities when $\mu \leqslant 0.3$, whereas LEBR2 outperforms CEO2 in other cases; CEO2 outperforms LEBR2 in estimating the number of communities when $\mu \leqslant 0.5$, whereas LEBR2 outperforms CEO2 in other cases.

Table 20
The results on LFR-O_n in terms of F-Score.

O_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.5887	0.5008	**0.8938**	0.7701	0.6471	0.7453	0.3662	0.7802
400	0.5965	0.5119	**0.8872**	0.7842	0.6508	0.7429	0.3464	0.7906
600	0.5699	0.4688	**0.8553**	0.7438	0.6275	0.7261	0.334	0.7637
800	0.5773	0.4961	**0.8485**	0.7524	0.6327	0.7228	0.3441	0.7736
1000	0.5569	0.4659	**0.8212**	0.7193	0.6096	0.6992	0.3203	0.7427
O_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.2745	0.0873	0.0307	0.8761	0.41	OoT	0.5129	0.6342
400	0.3004	0.0664	0.0254	0.8567	0.3964	OoT	0.5131	0.6303
600	0.2546	0.0614	0.0257	0.8456	0.4	OoT	0.4852	0.6164
800	0.251	0.068	0.0257	0.8259	0.3805	OoT	0.4804	0.6137
1000	0.2502	0.0675	0.0283	0.8142	0.3702	OoT	0.4893	0.6111

Table 21
The results on LFR-O_n in terms of EQ.

O_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	0.3957	0.1773	**0.7839**	0.6179	0.2421	0.3892	0.3403	0.5134
400	0.3985	0.1835	**0.7746**	0.6366	0.2421	0.4039	0.3469	0.5204
600	0.3861	0.1614	**0.7454**	0.6032	0.2396	0.3801	0.3318	0.4913
800	0.3917	0.1699	**0.727**	0.6095	0.2392	0.3806	0.3344	0.4942
1000	0.3932	0.1739	**0.7068**	0.5763	0.2355	0.3752	0.3348	0.4707
O_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	0.0498	0.0093	0.0009	0.5434	0.4066	OoT	0.0952	0.3785
400	0.0472	0.0076	0.0004	0.5305	0.3919	OoT	0.0928	0.3752
600	0.0397	0.0074	0.0004	0.5222	0.3883	OoT	0.0871	0.3739
800	0.0371	0.0081	0.0003	0.5048	0.3791	OoT	0.0872	0.3769
1000	0.0525	0.0072	0.0005	0.5036	0.3701	OoT	0.0858	0.3724

Table 22
The results on LFR-O_n in terms of D-Score.

O_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	26.13	80.93	2.96	13.65	20.73	5.94	6.76	7.26
400	24.87	77.18	2.98	11.69	19.77	5.58	5.88	6.84
600	26.73	83.75	3.66	14.03	21.16	6.32	6.89	7.95
800	25.65	80.37	4.34	13.11	20.65	6.04	6.47	7.8
1000	25.43	79.58	4.89	14.42	20.59	6.06	6.61	8.08
O_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	−1.75	−4.2	−65.06	12.22	**0.04**	OoT	2.93	4.35
400	−1.4	−5.24	−78.22	12.01	**0.0**	OoT	2.75	4.04
600	−1.92	−3.96	−77.08	12.55	**0.05**	OoT	2.78	4.45
800	−1.92	−4.29	−74.75	12.5	**0.01**	OoT	2.85	4.2
1000	−1.86	−4.01	−69.67	12.55	**0.01**	OoT	2.9	4.25

4.2.3. The results on LFR-$|C|_{max}$

The larger a community in the network, the more difficult it is for the community to grasp its members. For LFR-$|C|_{max}$, communities become more divisible with the increase of $|C|_{max}$. In Tables 9–11, the results in terms of NMI, F-Score and EQ become worse with the increase of $|C|_{max}$ in most cases. It can be seen from Table 12 that most algorithms generate more communities as $|C|_{max}$ increases. Table 13 shows that the time cost of most algorithms increases with the increase of $|C|_{max}$. From the results of CEOs and LEBRs on LFR-$|C|_{max}$, we can find that CEOs outperform LEBRs in identifying highly clustered ground-truth communities and estimating the number of communities.

4.2.4. The results on LFR-d_{max}

The denser the network, the more information can be used to assign nodes to the correct communities. For LFR-d_{max}, node memberships become clearer with the increase of d_{max}. In Tables 14–16, the results in terms of NMI, F-Score and EQ become better with the increase of d_{max} in most cases. It can be seen from Table 17 that most algorithms generate fewer communities

Table 23
The results on LFR-O_n in terms of Time (s).

O_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
200	7	1	13	12	6	11	203	7
400	7	1	13	12	8	11	163	7
600	7	1	13	12	4	11	214	7
800	7	1	13	12	4	13	199	7
1000	7	1	13	12	4	12	216	7
O_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
200	**0**	1	153	5	79	OoT	4	17
400	**0**	1	160	4	69	OoT	4	18
600	**0**	1	168	5	63	OoT	4	17
800	**0**	1	160	5	70	OoT	4	17
1000	**0**	1	171	5	74	OoT	4	18

Table 24
The results on LFR-α_n in terms of NMI.

α_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
1	0.6968	0.7417	**0.8604**	0.8684	0.9378	0.8784	0.7544	0.9067
2	0.3643	0.309	**0.7576**	0.6322	0.5044	0.5468	0.2718	0.6111
3	0.1348	0.1354	**0.6816**	0.3985	0.1165	0.3228	0.0193	0.28
4	0.0312	0.0252	**0.624**	0.1472	0.0323	0.1324	0.0014	0.0938
5	0.0090	0.0074	OoT	0.048	0.0053	OoT	OoT	0.0261
α_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
1	0.6641	0.168	0.035	0.7624	0.7347	0.6683	LoT	0.4348
2	0.4964	0.1504	0.0838	0.6475	0.5003	0.3359	LoT	0.4158
3	0.4605	0.103	0.135	0.5602	0.2251	0.1079	LoT	0.1752
4	0.4323	0.0226	0.2513	0.4694	0.0386	OoT	0.1139	0.038
5	**0.4235**	0.0016	OoM	0.2433	0.0048	OoT	0.0079	OoT

Table 25
The results on LFR-α_n in terms of F-Soce.

α_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
1	0.9022	0.9003	0.9548	0.9575	0.9846	0.967	0.8874	**0.9736**
2	0.7404	0.6657	**0.9048**	0.8667	0.8513	0.8394	0.6075	0.8751
3	0.589	0.5203	**0.8932**	0.7822	0.6551	0.7646	0.39	0.7864
4	0.481	0.3713	**0.8884**	0.6366	0.5274	0.6739	0.2765	0.6891
5	0.4177	0.3284	OoT	0.4976	0.4138	OoT	OoT	0.588
α_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
1	0.6487	0.4609	0.3901	0.8939	0.8607	0.8544	LoT	0.7298
2	0.3607	0.166	0.1385	0.8739	0.6014	0.7112	LoT	0.706
3	0.2975	0.0913	0.0507	0.8805	0.4307	0.6301	LoT	0.6249
4	0.2048	0.1008	0.0241	0.8511	0.288	OoT	0.4637	0.5529
5	0.177	0.0448	OoM	**0.7982**	0.1674	OoT	0.3793	OoT

as d_{max} increases. Table 18 shows that the time cost of most algorithms increases with the increase of d_{max}. From the results of CEOs and LEBRs on LFR-d_{max}, we can find that CEOs outperform LEBRs in identifying highly clustered ground-truth communities and estimating the number of communities.

4.2.5. The results on LFR-O_n

The more overlaps between communities, the more difficult it is to recognize community boundaries. For LFR-O_n, community boundaries become more fuzzy with the increase of O_n. In Tables 19–21, the results in terms of NMI, F-Score and EQ become worse with the increase of O_n in most cases. The overlapping nodes are usually far from the core members of communities. As a result, changes in the number of overlapping nodes are unlikely to affect the number of communities determined by selected seeds. It can be seen from Tables 22 and 23 that local expansion methods have stable results in terms of D-Score and efficiency as O_n increases. From the results of CEOs and LEBRs on LFR-O_n, we can find that CEOs outperform LEBRs in identifying highly clustered ground-truth communities and estimating the number of communities.

Table 26
The results on LFR-α_n in terms of EQ.

α_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
1	0.5221	0.5366	0.5988	0.6016	0.6006	0.5559	0.5586	**0.6146**
2	0.5196	0.3761	**0.7244**	0.6847	0.4926	0.4958	0.458	0.6416
3	0.3965	0.1921	**0.7613**	0.6177	0.251	0.3928	0.3383	0.4983
4	0.3118	0.0501	**0.7693**	0.4004	0.1766	0.3125	0.2914	0.3485
5	0.2811	0.0132	OoT	0.2824	0.1496	OoT	OoT	0.2406
α_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
1	0.3617	0.0873	0.0175	0.479	0.5267	0.5076	LoT	0.3954
2	0.1286	0.0179	0.0061	0.5986	0.6278	0.4951	LoT	0.4542
3	0.0712	0.0097	0.0008	0.5163	0.4147	0.3645	LoT	0.3802
4	0.0046	0.0112	0.0001	0.4433	0.2528	OoT	0.035	0.3258
5	0.0017	0.0158	OoM	**0.3557**	0.1777	OoT	0.0175	OoT

Table 27
The results on LFR-α_n in terms of D-Score.

α_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
1	0.6	1.32	0.19	0.17	0.07	0.13	0.03	0.06
2	5.41	16.69	0.61	1.43	3.29	1.09	1.31	1.43
3	26.16	80.23	3.28	13.25	20.52	6.29	6.73	7.63
4	119.45	378.49	18.38	109.6	98.64	43.02	33.11	48.24
5	514.92	1652.81	OoT	573.49	434.14	OoT	OoT	274.48
α_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
1	-0.39	-0.68	-3.35	0.28	**0.13**	0.02	LoT	-0.15
2	-1.27	-0.88	-12.65	2.3	−**0.62**	3.9	LoT	0.84
3	-1.61	-0.73	-35.05	12.95	**0.04**	16.98	LoT	4.33
4	-2.51	0.76	-60.5	60.15	**0.22**	OoT	6.02	20.67
5	-2.81	10.45	OoM	253.9	**0.01**	OoT	18.27	OoT

Table 28
The results on LFR-α_n in terms of Time (s).

α_n	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
3	4	**0**	8	3	2	3	26	1
4	236	49	463	410	61	751	62328	182
5	63148	4006	OoT	57196	5669	OoT	OoT	23732
α_n	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
1	**0**	**0**	**0**	**0**	**0**	**0**	LoT	**0**
2	**0**	**0**	**0**	**0**	**0**	2	LoT	**0**
3	**0**	**0**	32	2	20	33428	LoT	3
4	**0**	21	8320	28	253	OoT	22	744
5	**2**	1954	OoM	371	2063	OoT	261	OoT

4.2.6. The results on LFR-α_n

The latest research showed that the independence of detected communities and the dependence on quality functions hinder local expansion methods from identifying diversely structured communities [14]. For LFR-α_n, the structure of communities in the network becomes more diversified with the increase of α_n. In Tables 24–27, the results in terms of NMI, F-Score, EQ and D-Score become worse with the increase of α_n in most cases. Table 28 shows that the time cost of all

Table 29
The characteristics of real-world networks.

Networks	n	m	$\bar{d}$	$\lvert C\rvert$	$\overline{\lvert C\rvert}$	μ	O_n	O_m	EQ	Source
Karate	34	78	4.58	2	17.00	0.128	0	––	0.3715	[40]
Dolphin	62	159	5.12	2	31.00	0.038	0	––	0.3735	[46]
Football	115	613	10.66	12	9.58	0.357	0	––	0.554	[47]
Book	105	440	8.38	3	35.0	0.159	0	––	0.4148	[48]
Amazon	16716	48739	5.83	1163	15.16	0.005	867	2.06	0.9632	[21]
DBLP	93432	335520	7.18	4876	22.84	0.305	13439	2.33	0.6442	[21]
Youtube	39841	224235	11.26	4481	15.95	0.838	11935	3.65	0.1336	[21]
LiveJournal	84438	1521988	36.05	4090	29.33	0.204	22398	2.59	0.5573	[21]
DE	4027	8454	4.20	2257	7.59	0.530	2239	6.85	0.0227	[49]
EN	918	1081	2.36	560	4.18	0.266	507	3.80	0.2796	[49]
ES	1529	3253	4.26	978	6.96	0.470	958	6.51	0.0735	[49]
FR	2521	6417	5.09	1689	7.81	0.562	1670	7.39	0.0357	[49]
PT	1009	3725	7.38	811	9.51	0.609	809	9.29	0.0225	[49]
RU	896	1800	4.02	565	6.65	0.473	547	6.23	0.0944	[49]
Facebook	6858	36545	10.66	4858	6.69	0.650	4726	6.43	0.0619	[49]
GitHub	9848	15489	3.15	5134	6.12	0.439	5012	5.30	0.0534	[49]

Table 30
The results on real-world networks in terms of NMI.

Networks	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
Karate	**0.9185**	**0.9185**	**0.9185**	**0.9185**	0.3725	0.8253	0.7047	0.8318
Dolphin	0.409	0.2972	0.5153	0.4909	0.3183	0.3884	0.3638	0.3336
Book	0.4444	0.4263	0.4509	0.4558	0.431	0.4594	0.5139	0.3845
Football	0.7098	0.7877	0.6688	0.7632	0.7944	0.7989	0.519	**0.8339**
Amazon	0.6532	0.6745	0.6834	0.6915	0.6684	0.6553	0.7425	0.6916
DBLP	0.3174	0.3238	0.3217	0.3205	0.3289	0.2605	0.0608	0.3036
Youtube	0.233	0.2338	0.3271	0.3409	0.2423	0.063	0.0857	0.1136
LiveJournal	0.7793	0.771	**0.8215**	0.7991	0.7981	0.6858	0.2024	OoT
DE	0.3776	0.3507	0.3861	0.359	0.2836	0.2996	0.3134	0.287
EN	0.7458	0.6827	0.7258	0.6944	0.686	0.6707	0.5856	0.6648
ES	0.4728	0.4075	0.4335	0.4329	0.4077	0.3483	0.3323	0.368
FR	0.3359	0.2841	0.3304	0.3014	0.289	0.2498	0.2368	0.2782
PT	0.3166	0.199	0.3027	0.3174	0.1915	0.2236	0.1902	0.2905
RU	0.5184	0.4227	0.4918	0.4538	0.4304	0.4197	0.4019	0.4559
Facebook	0.3241	0.3069	0.3173	0.3041	0.3189	0.2847	0.2778	0.3088
GitHub	0.5266	0.4235	0.5002	0.5031	0.466	0.4254	0.3873	0.0296
Average	0.5052	0.4694	**0.5122**	0.5092	0.4392	0.4411	0.3699	0.4117

Networks	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
Karate	0.8368	0.0	0.2597	0.6151	0.2282	0.429	LoT	0.5656
Dolphin	0.2478	0.0	0.0	0.4829	0.3972	**0.5421**	LoT	0.3854
Book	**0.5453**	0.0	0.0	0.4281	0.2439	0.3386	LoT	0.4557
Football	0.7429	0.0	0.0	0.5559	0.7992	0.538	LoT	0.1083
Amazon	0.6895	0.8624	**0.9268**	0.7123	0.6755	OoT	0.8089	0.7365
DBLP	0.3128	**0.3604**	0.3515	0.2693	0.1832	OoT	0.0756	0.1379
Youtube	**0.4814**	0.3719	OoM	0.2901	0.0432	OoT	0.1083	0.0706
LiveJournal	0.7579	0.5401	OoT	0.7408	0.5984	OoT	0.4171	0.5079
DE	0.3586	0.3304	0.3689	0.2764	**0.5222**	0.3581	0.4073	0.2793
EN	0.6864	0.5666	0.5364	0.5978	**0.7697**	0.7219	0.7482	0.6603
ES	0.4321	0.3377	0.3363	0.3305	**0.539**	0.4412	0.4447	0.3685
FR	0.3503	0.317	0.3295	0.2398	**0.4956**	0.3294	0.354	0.2693
PT	0.3524	0.3046	0.3395	0.2184	**0.4958**	0.2849	0.4655	0.2193
RU	0.4381	0.3854	0.3573	0.3958	**0.5704**	0.4838	0.5282	0.4387
Facebook	0.3085	0.2698	0.284	0.295	**0.338**	OoT	0.336	0.2361
GitHub	0.4495	0.3889	0.3963	0.4114	**0.5714**	OoT	0.515	0.4654
Average	0.4994	0.3147	0.3204	0.4287	0.4669	0.4467	0.4341	0.3691

algorithms increases with the increase of α_n. The main findings about the results of CEOs and LEBRs on LFR-α_n are as follows. First, CEO1 outperforms LEBR1 in identifying ground-truth communities when $\alpha_n = 2$ and $\alpha_n \geqslant 4$, whereas LEBR1 outperforms CEO1 in other cases; CEO1 outperforms LEBR1 in identifying highly clustered communities when $\alpha_n \geqslant 2$, whereas LEBR1 outperforms CEO1 in other cases; CEO1 outperforms LEBR1 in estimating the number of communities. Second, CEO2 outperforms LEBR2 in identifying highly clustered ground-truth communities and estimating the number of communities when $2 \leqslant \alpha_n \leqslant 4$, whereas LEBR2 outperforms CEO2 in other cases.

From the results on artificial networks, we draw the following conclusions. OCLN performs the best in efficiency, followed by WCPM, SLPA and Bespoke. MEASSN with a time complexity of $O(n^2)$ cannot process networks with 10^5 nodes in 24 h. In terms of identifying overlapping communities, Links performs the worst, while CEOs perform the best. TWD, LECS, LEBR2 and CEO2 outperform LFM in detecting overlapping communities, which indicates that identifying high-quality seeds and communities and discarding low-quality seeds and communities both help to improve algorithm quality. LFM outperforms CFM in effectiveness and efficiency, which implies that node-oriented algorithms outperform link-oriented algorithms. TWD, OCLN, WCPM and Links performing the community merging process cannot get high-quality ground-truth communities when communities are highly fuzzy. The learning-based algorithms do not show significant advantages over other algorithms in identifying overlapping communities from artificial networks. Overall, LEBRs outperform CEOs in efficiency, while CEOs outperform LEBRs in detecting communities of diverse recognizability, size, density, overlap, and structure.

4.3. Experimental results on real-world networks

4.3.1. Real-world networks

Table 29 lists the characteristics of real-world networks with ground-truth communities. The real-world networks can be downloaded from http://www-personal.umich.edu/ mejn/netdata/ and http://snap.stanford.edu/data/. Detailed description of the real-world networks can be found in [14,21,49].

Table 31
The results on real-world networks in terms of F-Score.

Networks	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
Karate	0.9848	0.9848	0.9848	0.9848	0.8042	0.9571	0.9428	**0.9857**
Dolphin	0.8031	0.7226	0.8514	0.8104	0.7333	0.8535	0.8449	0.7232
Book	0.7357	0.7033	0.7346	0.7388	**0.7762**	0.7734	0.6878	0.7216
Football	0.8251	0.8698	0.8001	0.8379	0.8726	0.8752	0.6367	**0.8769**
Amazon	0.9298	0.9351	0.935	0.9386	0.9349	0.928	0.9481	0.939
DBLP	0.7656	0.7778	0.7606	0.7642	**0.8157**	0.7119	0.2489	0.7626
Youtube	**0.6446**	0.5294	0.343	0.3384	0.6332	0.2228	0.1795	0.4188
LiveJournal	0.9006	0.8882	0.903	0.888	**0.9266**	0.8615	0.5817	OoT
DE	0.2019	0.1226	0.1218	0.0939	0.2292	0.102	0.1171	0.1904
EN	0.7411	0.6926	0.7138	0.6841	0.6964	0.6679	0.5901	0.6725
ES	0.3795	0.3198	0.302	0.2864	0.3305	0.2532	0.26	0.3343
FR	0.2314	0.2108	0.1517	0.1316	0.2529	0.1321	0.1524	0.2049
PT	0.279	0.3181	0.1416	0.147	0.2486	0.1439	0.1411	0.1998
RU	0.4819	0.4442	0.3925	0.3629	0.3658	0.363	0.3656	0.4178
Facebook	0.3869	0.3591	0.3628	0.352	0.3845	0.3563	0.3366	0.3698
GitHub	0.2825	0.2619	0.233	0.2134	0.2849	0.1646	0.2043	0.0767
Average	0.5983	0.5713	0.5457	0.5358	0.5806	0.5229	0.4523	0.5263

Networks	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
Karate	0.9713	0.6662	0.6557	0.9436	0.4545	0.8235	LoT	0.8598
Dolphin	0.6069	0.6477	0.6477	**0.879**	0.7386	0.8273	LoT	0.8018
Book	0.6415	0.4793	0.4793	0.7347	0.5278	0.7342	LoT	0.6537
Football	0.6241	0.1533	0.1533	0.6155	0.8421	0.6233	LoT	0.2709
Amazon	0.9294	0.9668	**0.9773**	0.9509	0.6683	OoT	0.9569	0.9548
DBLP	0.4354	0.2575	0.0857	0.6941	0.5672	OoT	0.3172	0.5119
Youtube	0.2009	0.1711	OoM	0.4208	0.238	OoT	0.6139	0.3362
LiveJournal	0.7998	0.5472	OoT	0.856	0.7329	OoT	0.735	0.7609
DE	0.097	0.0491	0.0551	0.2662	**0.6765**	0.3043	0.5179	0.1646
EN	0.6657	0.5746	0.494	0.6864	0.7614	0.7478	**0.8337**	0.6871
ES	0.2603	0.1087	0.0816	0.3737	**0.6601**	0.4177	0.5612	0.2825
FR	0.1075	0.0582	0.0418	0.2078	**0.6763**	0.2808	0.4755	0.1693
PT	0.0705	0.0491	0.0243	0.1632	**0.6769**	0.2407	0.6153	0.1681
RU	0.3439	0.2137	0.2037	0.3799	0.683	0.487	**0.6984**	0.424
Facebook	0.3485	0.26	0.2432	0.3667	**0.6416**	OoT	0.5661	0.3581
GitHub	0.1984	0.1122	0.1087	0.314	**0.6776**	OoT	0.5681	0.1852
Average	0.4563	0.3322	0.3037	0.5533	**0.6389**	0.5487	0.6216	0.4743

4.3.2. The results on real-world networks

Tables 30–34 show the results on real-world networks in terms of NMI, F-Score, EQ, |D-Score| (the absolute value of D-Score) and Time. For the results on each real-world network, we rank algorithms in descending order of the values of NMI and EQ, and rank algorithms in ascending order of |D-Score|. For example, CEO1, LEBR1, CEO2, LEBR2, LFM, CFM, TWD, LECS, OCLN, WCPM, Links, SLPA, BIGCLAM, MEASSN, Bespoke and CDSAT rank 1, 1, 1, 1, 9, 4, 5, 3, 2, 12, 10, 6, 11, 8, 12 and 7 respectively on Karate in terms of NMI. Table 35 shows the average ranking of algorithms on NMI, F-Score, EQ and |D-Score|.

From the results on real-world networks, we draw the following conclusions. In terms of efficiency, OCLN performs the best, followed by WCPM, SLPA and Bespoke, whereas MEASSN performs the worst. In terms of effectiveness, Links performs the worst, while CEOs perform the best. For overlapping community detection, LEBR2 and CEO2 outperform LFM, while LFM outperforms TWD and LECS. For improving algorithm quality, the idea of discarding low-quality seeds and communities seems to be more effective than that of identifying high-quality seeds and communities. LFM outperforms CFM in effectiveness and efficiency, which is consistent with the results on artificial network. The learning-based algorithms outperform other algorithms in detecting highly clustered ground-truth communities from real-world networks where communities are highly fuzzy and overlapping. Overall, CEOs rank ahead of LEBRs in terms of NMI, F-Score and |D-Score|, while LEBRs rank ahead of CEOs in terms of EQ. In addition, LEBRs outperform CEOs in efficiency.

4.4. The in-depth analysis

The experimental phenomena supporting our view are as follows:

- LFM outperforms the algorithms (CFM, TWD and LECS) dedicated to improving seeding and expansion technologies in identifying overlapping communities from real-world networks. This is because existing seeding and expansion technologies unilaterally describing the structural characteristics of networks may result in low-quality seeds and communities. CEOs outperform compared local-expansion-based algorithms in identifying overlapping communities from artificial and real-world networks. This is because CEOs identify and discard low-quality seeds and communities following the two

Table 32
The results on real-world networks in terms of EQ.

Networks	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
Karate	0.3717	0.3717	0.3717	0.3717	**0.4021**	0.358	0.3456	0.3133
Dolphin	0.4884	0.4717	0.5153	**0.5261**	0.4661	0.4261	0.363	0.4938
Book	0.512	0.5094	0.5225	0.5151	0.4649	**0.5238**	0.4244	0.4959
Football	0.4872	0.5576	0.5332	0.5835	0.5478	0.5767	0.4739	**0.6005**
Amazon	0.9034	0.9242	0.9434	0.949	0.9134	0.8801	0.9331	0.939
DBLP	0.7366	0.7105	**0.7918**	0.7826	0.6133	0.6624	0.4056	0.6573
Youtube	0.3189	0.3636	**0.44**	0.4356	0.2012	0.1285	0.3712	0.114
LiveJournal	0.8907	0.9442	0.9585	**0.9611**	0.9219	0.8278	0.8418	OoT
DE	0.2081	0.2336	0.1979	0.1916	0.3908	0.286	0.2313	0.2298
EN	0.726	0.8547	0.8554	0.8765	0.8322	0.8672	0.8221	0.8842
ES	0.6158	0.7171	0.7309	**0.748**	0.6966	0.5965	0.6494	0.5686
FR	0.4903	**0.5369**	0.5249	0.5274	0.4247	0.4108	0.4259	0.4389
PT	0.3608	0.3384	0.3446	0.3457	**0.3642**	0.2961	0.2848	0.3462
RU	0.5668	0.589	0.6216	**0.6528**	0.5585	0.5225	0.5445	0.6208
Facebook	0.8457	0.9433	0.9476	**0.9511**	0.9027	0.8012	0.9333	0.8791
GitHub	0.513	0.5562	0.568	**0.5798**	0.5556	0.5397	0.5338	0.0538
Average	0.5647	0.6014	0.6167	**0.6249**	0.5785	0.544	0.5365	0.509
Networks	**OCLN**	**WCPM**	**Links**	**SLAP**	**BIGCLAM**	**MEASSN**	**Besopke**	**CDSAT**
Karate	0.3624	0.0	0.1578	0.2502	0.0943	0.3337	LoT	0.1518
Dolphin	0.329	0.0	0.0	0.4519	0.2688	0.3299	LoT	0.3731
Book	0.4408	0.0	0.0	0.4744	0.3103	0.5112	LoT	0.4309
Football	0.3794	0.0	0.0	0.5137	0.5643	0.4928	LoT	0.1899
Amazon	0.8662	0.9896	**0.9945**	0.906	0.8774	OoT	0.6429	0.8946
DBLP	0.3646	0.1816	0.0493	0.7413	0.6112	OoT	0.1352	0.6036
Youtube	0.0157	0.0175	OoM	0.0595	0.0888	OoT	0.0353	0.1505
LiveJournal	0.8583	0.4329	OoT	0.9591	0.5917	OoT	0.2942	0.7336
DE	0.1958	0.0492	0.0395	**0.452**	0.0647	0.3622	0.0616	0.1995
EN	0.8237	**0.9321**	0.8437	0.7368	0.4842	0.7418	0.5003	0.7757
ES	0.5978	0.2423	0.0878	0.6523	0.1423	0.6053	0.188	0.3856
FR	0.2817	0.0595	0.0333	0.4542	0.0727	0.4598	0.1162	0.2305
PT	0.1293	0.0504	0.0038	0.2829	0.0528	0.3251	0.0655	0.2241
RU	0.5211	0.3814	0.4127	0.5268	0.1731	0.5547	0.2373	0.4114
Facebook	0.9215	0.8938	0.717	0.9245	0.1376	OoT	0.2463	0.6716
GitHub	0.517	0.1351	0.1133	0.4624	0.1053	OoT	0.098	0.2479
Average	0.4753	0.2728	0.2466	0.553	0.29	0.4717	0.2184	0.4171

Table 33
The results on real-world networks in terms of |D-Score|.

Networks	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
Karate	**0.0**	**0.0**	**0.0**	**0.0**	1.0	**0.0**	**0.0**	0.5
Dolphin	2.0	4.5	1.0	1.0	2.5	1.0	1.0	2.0
Book	0.67	1.33	0.33	0.67	1.33	0.33	0.5	1.0
Football	0.25	0.67	0.17	**0.0**	0.17	**0.0**	0.33	**0.0**
Amazon	0.7	0.71	0.47	0.41	0.55	0.49	0.26	0.45
DBLP	1.52	2.29	0.74	0.8	2.01	0.63	1.24	0.9
Youtube	1.9	1.53	1.97	2.08	0.26	3.39	4.73	0.85
LiveJournal	0.32	0.59	0.2	0.22	0.12	0.2	0.74	OoT
DE	4.52	11.13	16.63	25.24	8.48	24.36	24.65	6.95
EN	1.62	1.8	1.81	2.18	2.03	2.31	3.75	2.27
ES	3.37	4.23	6.82	8.14	6.19	9.75	11.54	5.27
FR	3.15	4.76	15.56	20.65	7.66	22.46	21.82	8.6
PT	2.57	1.65	26.03	26.03	10.11	31.44	37.62	11.87
RU	2.69	2.49	5.01	6.34	5.73	6.24	7.43	4.59
Facebook	6.41	8.01	8.51	9.31	7.58	9.04	10.38	8.01
GitHub	5.68	3.88	7.96	9.79	6.33	13.22	13.63	**0.12**
Average	2.3356	3.0981	5.8256	7.0538	3.8781	7.8037	8.7263	3.5587
Networks	**OCLN**	**WCPM**	**Links**	**SLAP**	**BIGCLAM**	**MEASSN**	**Besopke**	**CDSAT**
Karate	**0.0**	1.0	**0.0**	1.0	**0.0**	0.5	LoT	**0.0**
Dolphin	0.5	1.0	1.0	2.0	**0.0**	2.0	LoT	1.5
Book	0.5	2.0	2.0	1.67	**0.0**	1.33	LoT	0.5
Football	0.5	11.0	11.0	0.71	**0.0**	0.71	LoT	2.0
Amazon	0.36	0.11	0.01	0.89	**0.0**	OoT	0.74	0.32
DBLP	0.43	2.35	10.98	1.58	**0.0**	OoT	0.64	0.1
Youtube	4.02	4.99	OoM	0.47	**0.0**	OoT	0.53	0.5
LiveJournal	0.44	1.01	OoT	0.09	**0.0**	OoT	0.4	0.21
DE	23.8	42.4	25.55	2.31	**0.42**	4.57	3.25	12.36
EN	2.33	3.15	3.41	**0.51**	1.2	1.01	0.58	2.03
ES	9.52	21.23	20.73	1.24	**0.73**	3.02	1.92	5.48
FR	21.52	37.39	41.22	1.74	**0.42**	5.16	3.44	9.11
PT	56.93	89.11	201.75	3.69	**0.2**	7.72	1.94	11.48
RU	6.64	10.08	9.87	1.3	0.88	2.14	**0.74**	3.91
Facebook	9.42	12.35	13.04	4.16	**0.15**	OoT	1.63	7.9
GitHub	10.51	17.47	14.8	1.85	0.91	OoT	2.69	4.97
Average	9.2137	16.04	25.3829	1.5756	**0.3069**	2.816	1.5417	3.8981

rules in the three processes mentioned in Section 3.1. The above phenomena support our view that existing seeding and expansion technologies cannot avoid producing low-quality seeds and communities, and local expansion methods commonly have poor fault tolerance to low-quality seeds and communities.

- CEO1 (CEO2) outperforms LEBR1 (LEBR2) in identifying overlapping communities from artificial and real-world networks. This is because the construction and expansion processes enable CEO1 (CEO2) to discard low-quality seeds and communities, whereas LEBR1 (LEBR2) cannot discard low-quality seeds and communities in the local expansion process that commonly employed by most local expansion methods. The above phenomenon supports our view that discarding low-quality seeds and communities is effective in solving the poor fault tolerance problem of local expansion methods.

The experimental phenomena reflecting the defects of CEO are as follows:

- LEBR2 outperforms CEO2 in identifying highly clustered communities from LFR-μ and real-world networks. This is because CEO2 performs the optimization process on relatively stable communities produced by CEO1, whereas LEBR2 performs the optimization process on fully expanded communities produced by LEBR1.
- LEBRs outperform CEOs in efficiency. The reasons for this phenomenon are as follows. First, CEO1 implements a two-step node allocation strategy, but LEBR1 implements a one-step node allocation strategy. Second, the communities processed by CEO1 must not be less than those processed by LEBR1. Third, CEOs have to discard low-quality seeds and communities, but LEBRs do not need to discard low-quality seeds and communities.
- CEO1 outperforms CEO2 in identifying highly clustered communities from LFR-μ. CEO1 ranks ahead of CEO2 for identifying overlapping communities from real-world networks. This is because the optimization process is employed to identify diversely structured communities rather than highly clustered communities, and the optimization process has limited effectiveness in optimizing node memberships.

Some compared algorithms outperform CEO on certain networks in terms of certain evaluation criteria. However, this phenomenon does not violate the experimental conclusions of this study.

Table 34
The results on real-world networks in terms of Time (s).

Networks	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
Karate	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Dolphin	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Book	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Football	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Amazon	5	**0**	6	2	2	5	1136	9
DBLP	405	20	644	265	70	630	3594	462
Youtube	761	310	1140	428	9451	24068	4342	38762
LiveJournal	537	191	1351	302	52715	26041	54988	OoT
DE	8	4	9	5	2330	958	8	28
EN	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ES	**0**	**0**	**0**	**0**	17	2	1	1
FR	3	1	3	1	80	38	3	7
PT	1	1	1	1	18	7	**0**	1
RU	**0**	**0**	**0**	**0**	19	1	**0**	**0**
Facebook	2	1	4	1	163	14	61	33
GitHub	37	29	45	32	16580	5114	51	8757
Average	110	35	200	65	5090	3555	4012	3204

Networks	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
Karate	**0**	**0**	**0**	**0**	**0**	**0**	LoT	**0**
Dolphin	**0**	**0**	**0**	**0**	**0**	**0**	LoT	**0**
Book	**0**	**0**	**0**	**0**	**0**	**0**	LoT	**0**
Football	**0**	**0**	**0**	**0**	3	**0**	LoT	**0**
Amazon	**0**	1	503	7	52	OoT	8	22
DBLP	**2**	46	46653	54	1409	OoT	46	1685
Youtube	**0**	92	OoM	30	9229	OoT	27	1067
LiveJournal	95	**74**	OoT	168	4885	OoT	143	7791
DE	**1**	2	179	**1**	4762	36056	3	**1**
EN	**0**	**0**	**0**	**0**	4	58	**0**	**0**
ES	**0**	**0**	4	**0**	126	436	1	**0**
FR	**0**	**0**	30	1	1170	1965	2	2
PT	**0**	**0**	6	**0**	407	53	1	**0**
RU	**0**	**0**	1	**0**	51	44	**0**	0
Facebook	**0**	**0**	576	4	1397	OoT	8	12
GitHub	5	12	1711	**2**	8331	OoT	9	348
Average	**6**	14	3547	17	1989	3861	21	683

Table 35
The average ranking of algorithms in terms of each evaluation criterion.

Criteria	CEO1	LEBR1	CEO2	LEBR2	LFM	CFM	TWD	LECS
NMI	5.125	8.5625	**4.8125**	5.3125	8.25	9.625	11.4375	9.1875
F-Score	**4.625**	6.125	7.0625	8.1875	5.1875	8.9375	10.5	7.4375
EQ	6.4375	4.3125	3.3125	**2.5625**	5.625	7.0625	7.6875	6.875
|D-Score|	5.5	6.6875	6.8125	7.875	6.625	8.0625	9.625	6.5
Average	**5.4219**	6.4219	5.5	5.9844	6.4219	8.4219	9.8125	7.5

Criteria	OCLN	WCPM	Links	SLAP	BIGCLAM	MEASSN	Besopke	CDSAT
NMI	6.125	10.25	10.5625	10.3125	5.375	10.25	6.9375	10.375
F-Score	11.5625	13.1875	13.8125	6.25	5.875	9.0625	6.25	8.8125
EQ	10.0	11.625	12.6875	7.125	12.5	10.375	13.75	10.75
|D-Score|	8.5	11.5	11.5	4.625	**1.3125**	8.6875	5.3125	5.8125
Average	9.0469	11.6406	12.1406	7.0781	6.2656	9.5938	8.0625	8.9375

5. Conclusions

Various technologies have been developed to get high-quality seeds and communities; however, the poor fault tolerance problem is still a blind spot in the study of local expansion methods. To solve the poor fault tolerance problem, we propose CEO based on our previous work, where the construction, expansion and optimization processes discard low-quality seeds and communities and reallocating community boundaries. CEO was compared to thirteen noted algorithms by examining the performance on five groups of artificial networks and sixteen real-world networks with ground-truth communities. Experimental results showed CEO performs the best in identifying overlapping communities, which verifies the effectiveness of discarding low-quality seeds and communities in solving the poor fault tolerance problem.

Experimental results support the contribution of this study. However, the quality of CEO can be further improved. On the one hand, seeding and expansion technologies can be used to get high-quality seeds and communities. On the other hand, community merging process can be used to get highly clustered communities. Besides, the efficiency of CEO is mainly limited by the size of networks and the recognizability of communities. Synchronous update strategy can be used to improve the efficiency of CEO.

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.

Acknowledgements

National Key Research and Development Program of China under Grant (No. 2016YFB1000901), National Natural Science Foundation of China (Nos. 61672179, 61370083, 61402126), Natural Science Foundation of Heilongjiang Province (No. F2015030), Natural Science Foundation of Chongqing under Grant (No. cstc2019jcyj-cxttX0002), Chongqing Excellent Scientist Project (No. cstc2021ycjh-bgzxm0013), Science Foundation for Youths of Heilongjiang Province (No. QC2016083), Heilongjiang Postdoctoral Science Foundation (No. LBH-Z14071), University Nursing Program for Young Scholars with Creative Talents in Heilongjiang Province (No. UNPYSCT-2017094) support this paper.

References

[1] S. Fortunato, Community detection in graphs, CoRR abs/0906.0612. arXiv:0906.0612..
[2] S. Fortunato, D. Hric, Community detection in networks: A user guide, CoRR abs/1608.00163. arXiv:1608.00163..
[3] S. Rani, M. Mehrotra, Community detection in social networks: Literature review, J. Inform. Knowl. Manage. 18 (2) (2019) 1950019.
[4] M.A. Javed, M.S. Younis, S. Latif, J. Qadir, A. Baig, Community detection in networks: A multidisciplinary review, J. Network Comput. Appl. 108 (2018) 87–111.
[5] M. Huang, G. Zou, B. Zhang, Y. Liu, Y. Gu, K. Jiang, Overlapping community detection in heterogeneous social networks via the user model, Inf. Sci. 432 (2018) 164–184.
[6] O. Doluca, K. Oğuz, Apal: Adjacency propagation algorithm for overlapping community detection in biological networks, Inf. Sci. 579 (2021) 574–590.
[7] Y.-Y. Ahn, J.P. Bagrow, S. Lehmann, Link communities reveal multiscale complexity in networks, Nature 466 (7307) (2010) 761–764.
[8] K. Nath, R. Shanmugam, V. Varadaranjan, ma-code: A multi-phase approach on community detection in evolving networks, Inf. Sci. 569 (2021) 326–343.
[9] A. Biswas, B. Biswas, Fuzag: Fuzzy agglomerative community detection by exploring the notion of self-membership, IEEE Trans. Fuzzy Syst. 26 (5) (2018) 2568–2577.
[10] S. Tahmasebi, P. Moradi, S. Ghodsi, A. Abdollahpouri, An ideal point based many-objective optimization for community detection of complex networks, Inf. Sci. 502 (2019) 125–145.
[11] Y. Xie, X. Wang, D. Jiang, R. Xu, High-performance community detection in social networks using a deep transitive autoencoder, Inf. Sci. 493 (2019) 75–90.
[12] X. Huang, L.V.S. Lakshmanan, J. Xu, Community Search over Big Graphs, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2019.
[13] R. Liu, H. Wang, X. Yu, Shared-nearest-neighbor-based clustering by fast search and find of density peaks, Inf. Sci. 450 (2018) 200–226.
[14] X. Ding, J. Zhang, J. Yang, Node-community membership diversifies community structures: An overlapping community detection algorithm based on local expansion and boundary re-checking, Knowl.-Based Syst. 198 (2020) 105935.
[15] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New J. Phys. 11 (3) (2009) 033015.
[16] S. Jabbour, N. Mhadhbi, B. Raddaoui, L. Sais, Pushing the envelope in overlapping communities detection, in: Advances in Intelligent Data Analysis XVII - 17th International Symposium, IDA 2018, 's-Hertogenbosch, The Netherlands, October 24–26, 2018, Proceedings, Vol. 11191, Springer, 2018, pp. 151–163..
[17] L. Lü, T. Zhou, Q.-M. Zhang, H.E. Stanley, The h-index of a network node and its relation to degree and coreness, Nat. Commun. 7 (1) (2016) 10168.
[18] J.J. Whang, D.F. Gleich, I.S. Dhillon, Overlapping community detection using seed set expansion, 22nd ACM International Conference on Information and Knowledge Management, 1, San Francisco, CA, USA, November 2013, pp. 2099–2108.
[19] H. Long, Overlapping community detection with least replicas in complex networks, Inf. Sci. 453 (2018) 216–226.
[20] D. Rhouma, L.B. Romdhane, An efficient algorithm for community mining with overlap in social networks, Expert Syst. Appl. 41 (9) (2014) 4309–4321.
[21] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowl. Inf. Syst. 42 (1) (2015) 181–213.
[22] X. Ding, J. Zhang, J. Yang, A robust two-stage algorithm for local community detection, Knowl.-Based Syst. 152 (2018) 188–199.
[23] J. Zhang, X. Ding, J. Yang, Revealing the role of node similarity and community merging in community detection, Knowl.-Based Syst. 165 (2019) 407–419.
[24] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814–818.
[25] J.M. Kumpula, M. Kivelä, K. Kaski, J. Saramäki, Sequential algorithm for fast clique percolation, Phys. Rev. E 78 (2) (2008) 026109.
[26] X. Zhang, C. Wang, Y. Su, L. Pan, H. Zhang, A fast overlapping community detection algorithm based on weak cliques for large-scale networks, IEEE Trans. Comput. Soc. Syst. 4 (4) (2017) 218–230.
[27] Y. Kim, H. Jeong, Map equation for link communities, Phys. Rev. E 84 (2011) 026110.
[28] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (2007) 036106.
[29] J. Xie, B.K. Szymanski, X. Liu, SLPA uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process, Data Mining Workshops (ICDMW) 2011 IEEE 11th International Conference on, 11, Vancouver, BC, Canada, December 2011, pp. 344–349.
[30] Y. Wang, Z. Bu, H. Yang, H. Li, J. Cao, An effective and scalable overlapping community detection approach: Integrating social identity model and game theory, Appl. Math. Comput. 390 (2021) 125601.
[31] T. Ma, Z. Xia, F. Yang, An ant colony random walk algorithm for overlapping community detection, in: Intelligent Data Engineering and Automated Learning - IDEAL 2017–18th International Conference, Guilin, China, October 30 - November 1, 2017, Proceedings, Vol. 10585, 2017, pp. 20–26..
[32] J. Su, T.C. Havens, Quadratic program-based modularity maximization for fuzzy community detection in social networks, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1356–1371.

[33] H. Zhang, X. Chen, J. Li, B. Zhou, Fuzzy community detection via modularity guided membership-degree propagation, Pattern Recogn. Lett. 70 (2016) 66–72.
[34] C. Liu, J. Liu, Z. Jiang, A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks, IEEE Trans. Cybern. 44 (12) (2014) 2274–2287.
[35] D. Jin, Z. Yu, P. Jiao, S. Pan, P.S. Yu, W. Zhang, A survey of community detection approaches: From statistical modeling to deep learning, CoRR abs/2101.01669..
[36] J. Yang, J. Leskovec, Overlapping community detection at scale: a nonnegative matrix factorization approach, in: S. Leonardi, A. Panconesi, P. Ferragina, A. Gionis (Eds.), Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4–8(2013), 2013, pp. 587–596.
[37] A. Bakshi, S. Parthasarathy, K. Srinivasan, Semi-supervised community detection using structure and size, in: IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17–20, 2018, IEEE Computer Society, 2018, pp. 869–874..
[38] S. Jabbour, N. Mhadhbi, B. Raddaoui, L. Sais, Sat-based models for overlapping community detection in networks, Computing 102 (5) (2020) 1275–1299.
[39] C. Ansótegui, J. Gabàs, WPM3: an (in)complete algorithm for weighted partial maxsat, Artif. Intell. 250 (2017) 37–57.
[40] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[41] Y. Xu, H. Xu, D. Zhang, Y. Zhang, Finding overlapping community from social networks based on community forest model, Knowl.-Based Syst. 109 (2016) 238–255.
[42] H. Yu, P. Jiao, Y. Yao, G. Wang, Detecting and refining overlapping regions in complex networks with three-way decisions, Inf. Sci. 373 (2016) 21–41.
[43] F. Cheng, C. Wang, X. Zhang, Y. Yang, A local-neighborhood information based overlapping community detection algorithm for large-scale complex networks, IEEE/ACM Trans. Networking (2020) 1–14.
[44] H. Shen, X. Cheng, K. Cai, M.-B. Hu, Detect overlapping and hierarchical community structure in networks, Physica A 388 (8) (2009) 1706–1712.
[45] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Phys. Rev. E 80 (1) (2009) 016118.
[46] 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: Can geographic isolation explain this unique trait?, Behav. Ecol. Sociobiol. 54 (4) (2003) 396–405.
[47] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Nat. Acad. Sci. 99 (2002) 7821–7826.
[48] V. Krebs, Books about us politics,http://www.orgnet.com/ (2004)..
[49] B. Rozemberczki, C. Allen, R. Sarkar, Multi-scale attributed node embedding, CoRR abs/1909.13021. arXiv:1909.13021..