Knowledge-Based Systems 198 (2020) 105935

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Node-community membership diversifies community structures: An overlapping community detection algorithm based on local expansion and boundary re-checking

Xiaoyu Ding, Jianpei Zhang *, Jing Yang

College of Computer Science and Technology, Harbin Engineering University, Harbin, Heilongjiang 150001, China

A R T I C L E I N F O

Article history:
Received 15 August 2019
Received in revised form 23 March 2020
Accepted 17 April 2020
Available online 23 April 2020

Keywords:
Overlapping community detection
Local expansion
Boundary re-checking
Node-community membership

A B S T R A C T

Local expansion methods excel in efficiency for mining overlapping communities in real-world networks. However, two problems prevent such methods from identifying diversely structured communities. First, local expansion methods generate independent communities only. Second, local expansion methods depend heavily on quality functions. This work provides a solution for local expansion methods to identify diversely structured communities. The proposed overlapping community detection algorithm performs local expansion and boundary re-checking sub-processes in order. The local expansion process first gets a cover of the network, and then the boundary re-checking process optimizes the cover of the network resulting from the local expansion process. To solve the first problem, the proposed algorithm establishes associations between boundaries of adjacent communities via the boundary re-checking process. To solve the second problem, the proposed algorithm expands and optimizes communities based on node-community membership optimization. We compared the proposed algorithm to seven state-of-the-art algorithms by examining their performance on five groups of artificial networks and sixteen real-world networks. Experimental results showed that the proposed algorithm outperforms compared algorithms in identifying diversely structured communities.

1. Introduction

Real-world systems in nature and society can be modeled as networks or graphs [1]. In the network, nodes or vertices represent objects, and links or edges represent the interactions between the objects [2]. This paper deals with a topological property of networks, community structure, which is essential for understanding the function and organization of real-world systems. A commonly accepted intuitive description of the community is: A community is a group of nodes that are tightly connected with each other but well separated from the rest of the network [3].

Community detection has gained growing attention for decades. Most previous research on community detection focuses on identifying disjoint communities, where each node in the *partition* of the network belongs to only one community [4–14]. However, there is growing evidence that many real-world networks are characterized by well-defined statistics of overlapping communities [15–18]. Therefore, more and more methods have been developed to identify overlapping communities, where each node in the *cover* of the network belongs to at least one community [19–25]. This paper divides mainstream overlapping community detection algorithms into the following eight categories: clique percolation methods, line graph partitioning methods, agent-based dynamical methods, fuzzy detection methods, statistical inference methods, non-negative matrix factorization methods, evolutionary methods and local expansion methods. This work focuses on the study of local expansion methods, while a comprehensive discussion of other overlapping community detection methods is beyond the scope of this paper.

Clique percolation methods. Clique percolation methods assume that the community consists of overlapping complete subgraphs [26]. Palla et al. [27] proposed CPM (Clique Percolation Method) to find *k-clique* communities. Kumpula et al. [28] sped up CPM via sequential clique percolation. Maity and Rath [29] improved CPM to ensure all nodes in the network belong to at least one community. Clique percolation methods work like pattern recognition methods that intend to find specific local structures in the network [30].

Line graph partitioning methods. In a simple graph, each node is attached to at least one link. By converting simple graphs to line graphs, disjoint community detection methods can be extended to identify overlapping communities [31]. For instance,

* Corresponding author.
E-mail addresses: B615060001@hrbeu.edu.cn (X. Ding), zhangjianpei@hrbeu.edu.cn (J. Zhang), yangjing@hrbeu.edu.cn (J. Yang).

https://doi.org/10.1016/j.knosys.2020.105935
0950-7051/

Kim and Jeong [32] extended the map equation method [33] to line graphs. The proposed algorithm encodes random walk paths on line graphs under the principle of minimum description length. Most existing line graph partitioning methods are based on link clustering [34–37].

Agent-based dynamical methods. Gregory [38] extended the label propagation algorithm [6] to identify overlapping communities by allowing a node to hold multiple labels. Later, Xie et al. [39] proposed a speaker-listener based information propagation algorithm. The proposed algorithm provides a memory for each node to store the received information such as labels. There are other well-known agent-based dynamical methods based on game-theoretic framework [40], particles walk [41], Potts model [42], Kuramoto model [43], and so on.

Fuzzy detection methods. Fuzzy detection methods use soft membership vectors to describe the strength of associations between nodes and communities [44]. Zhang et al. [45] introduced a spectral clustering algorithm [46], where fuzzy c-means is used to get the soft assignment of nodes. Nepusz et al. [47] modeled overlapping community detection as a nonlinear constrained optimization problem, which can be solved by a simulated annealing approach. Several heuristics for fuzzy modularity maximization can be found in [48].

Statistical inference methods. Graph clustering can be modeled as statistical inference problems. SBM (Stochastic Block Model) [49] can be applied to describe the formation of communities, which assumes that the connection probability between any two nodes is a function of block membership [50]. Bayesian inference can also be applied to overlapping community detection [51]. Yang et al. [52] proposed an SBM-based algorithm, where Bayesian inference is used to calculate the posterior distribution of all unknown parameters.

Non-negative matrix factorization methods. NMF (Non-negative Matrix Factorization) is a machine learning approach, which helps improve the interpretability of data analysis. NMF decomposes a given feature matrix to reveal the features of a given structure [53]. NMF can be used to identify the structural features of networks with overlapping communities. Several NMF-based algorithms for overlapping community detection can be found in [54–58].

Evolutionary methods. Evolutionary computation is a powerful optimization technique inspired by natural evolutionary processes, applied to solve real-world problems such as community detection [59]. Evolutionary methods first get an initial population and then iteratively perform variation and selection sub-processes that optimize a single objective or multiple objectives [60,61]. As advantages, evolutionary methods are naturally parallel and can avoid obtaining local optimal solutions [62,63].

Local expansion methods. The working principle of local expansion methods is described as follows. In a local expansion algorithm, a seed node or node set is first selected as an initial community or seed community. Then the community expands by absorbing its neighboring nodes. The expansion process stops when the joining of neighboring nodes can no longer improve the quality of the community. The algorithm stops when each node in the network is assigned to at least one community. Otherwise, the algorithm selects a new seed from nodes that have not been assigned to any community, and expands a new community in the same way as previously described. For local expansion methods, the quality of seeds and communities is usually guaranteed by centrality indices and quality functions. Since local topology information can be used to calculate centrality indices and quality functions, the time complexity of a well-designed local expansion algorithm can be linearly related to the number of nodes or links in a sparse network.

Other methods. Besides the above eight mainstream overlapping community detection methods, some work has been done to extract overlapping communities from disjoint communities. Gregory [64] extended the GN (Girvan and Newman) algorithm [65] based on splitting betweenness to find overlapping communities. Hajiabadi et al. [66] introduced an integrated approach in which internal and external associations are used to achieve disjoint and overlapping communities. Chakraborty et al. [67] adopted ensemble methods to discover hidden overlapping nodes from disjoint community structures.

Two problems prevent local expansion methods from identifying diversely structured communities. First, it can be seen from the working principle of local expansion methods that any community that is being expanded will neither change other previously expanded communities nor use them as prior information. In other words, local expansion methods cannot establish associations between different communities. As a result, local expansion methods generate independent communities only. Second, it is a fact that real-world networks consist of diversely structured communities, but there is no universally accepted definition that can be used to describe all community structures. However, one local expansion algorithm typically optimizes one quality function. Consequently, local expansion methods depend heavily on quality functions.

The main contributions of this paper are as follows:

- This work extends the working principle of local expansion methods, which provides a solution for local expansion methods to identify diversely structured communities.
- A two-step overlapping community detection algorithm based on node-community membership optimization is originally proposed by this paper, which performs local expansion and boundary re-checking sub-processes in order.
- To eliminate the independence of detected communities, the proposed algorithm establishes associations between boundaries of adjacent communities via the boundary re-checking process.
- To eliminate the dependence on quality functions, the proposed algorithm expands and optimizes communities by optimizing one-to-one and one-to-many node-community membership.

The rest of this paper is organized as follows. Section 2 outlines related work on local expansion methods. Section 3 presents the proposed algorithm. Experimental results are shown in Section 4. Section 5 concludes this paper and outlines our future work.

2. Related work

Overlapping communities are ubiquitous in real-world systems. A person usually plays different roles in different social groups; a researcher may be active in several fields; a protein could have multiple biological functions. Overlapping community detection is significant for understanding the function and organization of real-world systems. An overview on mainstream overlapping community detection algorithms is provided in Section 1. This section outlines related work on local expansion methods in terms of seed selection and community expansion.

2.1. Seed selection

Seeds are critical for local expansion methods [68]. Random seeds may cause local expansion algorithms to generate low-quality communities [69]. The quality of seeds is usually guaranteed by centrality indices [15]. Freeman et al. [70] defined *degree*, *closeness* and *betweenness*, which contribute greatly to the study of node centrality. Taking advantage of node topology location information, Kitsak et al. [71] introduced *k-shell*, which presents better performance *degree* and *betweenness* in identifying influential spreaders in the network. Unfortunately, *k-shell* decomposition [72] fails to get well-distinguishable seeds because a large number of nodes share the same shell. Later, Bae et al. [73] provided *coreness* to achieve a monotonic ranking of node centrality. However, the calculation of *coreness* is more time-consuming than that of *k-shell*. Recently, Lu et al. [74] proved that the convergence of Hirsch index to *k-shell* can be guaranteed under an asynchronous update process. Thus, Hirsch index enables decentralized local approaches to get node shell values from large-scale dynamical networks. Rhouma et al. [75] offered *node importance* that measures the centrality of a node by the product of its *degree* and *clustering coefficient*. Ding et al. [76] introduced *node mass* that measures the centrality of a node by the number of links in its neighborhood. Whang et al. [77] recommended two schemes to identify seeds located in clusters with low conductance. One scheme performs Graclus [78] to get low-conductance clusters of which the centroid nodes are selected as seeds. Another scheme performs PageRank [79] to achieve a ranking of node centrality in the form of a personalized PageRank vector, which has been confirmed to be closely related to graph partitioning [80].

In addition, there are other excellent centrality indices. Xu et al. [81] recommend to select pairs of closely connected nodes as seeds and introduced *backbone degree* to measure the strength of links in the network. Yu et al. [82] put forward *node local importance* based on a hypothesis that a node is more important if more of its neighbors have a lower weight than it. Zhang et al. [83] fully considered the similarity between nodes in global and local scenes and modified *degree* based on *core similarity*.

2.2. Community expansion

Quality functions define the characteristics of the community. Previous work has developed numerous well-known quality functions such as *ratio cut* [84], *normalized cut* [85], *conductance* [86], *triangle participation ratio* [87], and so on. Regrettably, one local expansion algorithm typically optimizes one quality function, which may cause the algorithm to generate similarly structured communities. First, one may also find that a quality function causes the local expansion algorithm to generate communities with specific characteristics. Yang and Leskovec [88] examined the behavior of thirteen quality functions on four goodness metrics. They found that *conductance* excels in identifying highly separated communities, whereas *triangle participation ratio* excels in identifying highly cohesive communities. Second, one may find some configurations for which a quality function performs unsatisfactory. Kanawati [89] applied different ensemble methods to combine multiple quality functions. They found that the ensemble-ranking methods provide better results than the methods that optimize a single quality function.

Several schemes have been developed to address the above two issues. Lancichinetti et al. [69] introduced *fitness*, where a positive real-valued parameter α is used to control the size of the community. Later, Yu et al. [82] extended *fitness* by introducing thresholds which are used to derive the overlapping regions of the community. Similarly, Guo et al. [90] extended *fitness* based on *internal force* between nodes to get high-quality communities from the network. Rhouma and Romdhane [75] offered *membership degree* for filtering the neighboring nodes absorbed by the community. Xu et al. [81] provided *backbone degree* to determine the order in which the neighboring nodes join the community. Lancichinetti et al. [91] presented an order statistics local optimization method to express the statistical significance of communities regarding random fluctuations. Ding et al. [76] recommended expanding the community by optimizing node-community membership, which describes the characteristics of the community from the perspective of the node.

3. Algorithm

3.1. Motivation

As can be seen from Section 2, local expansion methods have been well developed in terms of seed selection and community expansion. However, two problems still prevent local expansion methods from identifying diversely structured communities, as discussed in Section 1. First, local expansion methods generate independent communities only. Second, local expansion methods depend heavily on quality functions.

Our motivation for solving the above two problems is as follows. To solve the first problem, a local expansion algorithm should describe the associations between communities in a global scene. However, it is impossible for a local expansion algorithm to describe such associations based on the working principle described in Section 1. Therefore, we introduce a post-processing approach, boundary re-checking, to establish associations between boundaries of adjacent communities in the cover of the network. One way to solve the second problem is to give a community definition that can be used to describe all community structures. However, it is difficult to achieve such a universal definition based on people's one-sided understanding of real-world networks. Therefore, we recommend expanding and optimizing communities based on node-community membership optimization, where node-community membership enables the proposed algorithm to describe the characteristics of the community from the perspective of the node.

3.2. Problem formulation

This paper considers a simple graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ links. G can be represented as an adjacency matrix $\mathbf{A}$, where A_{ij} denotes the element in the ith row and jth column of $\mathbf{A}$. If $u, v \in V$ and $(u, v) \in E$, then u and v are neighbors and $A_{uv} = 1$. If $u, v \in V$ and $(u, v) \notin E$, then $A_{uv} = 0$. A community is a group of nodes $C = \{v_i, \dots, v_j\}$ $(C \subseteq V)$. The purpose of overlapping community detection is to get a cover of the graph $\mathbf{C} = \{C_1, C_2, \dots, C_k\}$ $(C_1 \cup C_2 \cup, \dots, \cup C_k \subseteq V)$, where $\exists C_i \cap C_j \neq \emptyset$, $C_i, C_j \in \mathbf{C}$ and $i \neq j$. If $\exists (u, v) \in E$, $u \in C_i$, $v \in C_j$, $C_i, C_j \in \mathbf{C}$ and $i \neq j$, then C_i and C_j are adjacent communities.

3.3. Basic definitions

Definition 1 (*Node Neighborhood*)**.** The neighborhood of node v is defined as follows:

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

The neighborhood of a node consists of the node and the neighbors of the node. In Fig. 1, v_5 has neighbors v_1, v_2, v_4, v_6, v_9 and v_{10}, $\Gamma(v_5) = \{v_1, v_2, v_4, v_5, v_6, v_9, v_{10}\}$. In the proposed algorithm, the neighborhood of a seed is the initial community of the seed.

Definition 2 (*Node Centrality*)**.** The centrality of node v is defined as follows:

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

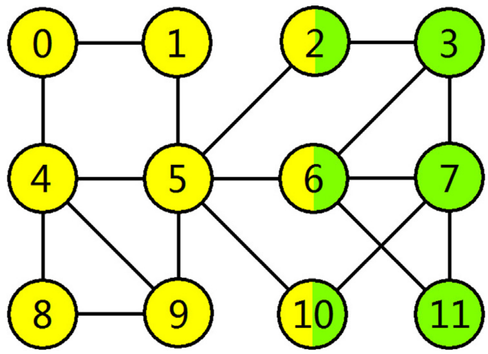


Fig. 1. A toy graph. Nodes with the same color belong to the same community. C_{v_i} denotes the community identified by node v_i. In the toy graph, $C_{v_5} = \{v_0, v_1, v_2, v_4, v_5, v_8, v_9, v_{10}\}$, $C_{v_6} = \{v_2, v_3, v_6, v_7, v_{10}, v_{11}\}$ and v_2, v_6, v_{10} are overlapping nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The centrality of a node is measured by the number of internal links in the neighborhood of the node. In Fig. 1, $\Gamma(v_5)$ has seven internal links (v_1, v_5), (v_2, v_5), (v_4, v_5), (v_4, v_9), (v_5, v_6), (v_5, v_9) and (v_5, v_{10}), $nc(v_5) = 7$. In the proposed algorithm, the centrality of the nodes determines the order in which seeds are selected and the order in which nodes are re-checked.

Definition 3 (*Community Boundaries*)**.** The boundaries of community C are defined as follows:

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

The boundaries of a community consist of nodes inside the community, which have at least one neighbor outside the community. In Fig. 1, v_5 inside C_{v_5} has a neighbor v_6 outside C_{v_5}, $v_5 \in B(C_{v_5})$, $B(C_{v_5}) = \{v_2, v_5, v_{10}\}$. In the proposed algorithm, the boundaries of a community play a role in interacting with the rest of the graph.

Definition 4 (*Community Neighbors*)**.** The neighbors of community C are defined as follows:

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

The neighbors of a community consist of nodes outside the community, which have at least one neighbor inside the community. In Fig. 1, v_6 outside C_{v_5} has a neighbor v_5 inside C_{v_5}, $v_6 \in N(C_{v_5})$, $N(C_{v_5}) = \{v_3, v_6, v_7\}$. In the proposed algorithm, the neighbors of a community play a role in interacting with the community.

Definition 5 (*Node-subgraph Similarity*)**.** The similarity between node v and subgraph S is defined as follows:

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

where subgraph S is a node set $S = \{v_i, \ldots, v_j\}$ ($S \subseteq V$). The similarity between a node and a subgraph is measured by the number of internal links in the intersection of the neighborhood of the node and the subgraph. In Fig. 1, the intersection of $\Gamma(v_5)$ and C_{v_6} has three internal links (v_2, v_5), (v_5, v_6) and (v_5, v_{10}), $nss(v_5, C_{v_6}) = 3$. In the proposed algorithm, node-subgraph similarity is used to determine the membership of the node.

Algorithm 1 The proposed algorithm.

Input: Graph $G = \{V, E\}$, Node set V, Link set E.
Output: The cover of the graph $\mathbf{C}$.
1: **Initialization:**
2: Initialize the cover of the graph, $\mathbf{C} = \emptyset$;
3: Initialize an unassigned-node sequence U, $U = V$;
4: Calculate the centrality of each node $v \in V$ based on Eq. (2);
5: **Local expansion process:**
6: **while** $U \neq \emptyset$ **do**
7: **Seeding procedure:**
8: Select the node v_s with the maximum centrality from U as the seed;
9: Initialize the seed community C_{v_s} based on Eq. (1), $C_{v_s} = \Gamma(v_s)$;
10: **Cleanup procedure:**
11: Get boundaries $B(C_{v_s})$ of C_{v_s} based on Eq. (3);
12: **while** $V^{del} = \{v|v \in B(C_{v_s}), nss(v, C_{v_s}) < nss(v, V - C_{v_s})\} \neq \emptyset$ **do**
13: Update the members of the community, $C_{v_s} = C_{v_s} - V^{del}$;
14: Update the boundaries of the community based on Eq. (3);
15: **end while**
16: **Expansion procedure:**
17: Get neighbors $N(C_{v_s})$ of C_{v_s} based on Eq. (4);
18: **while** $V^{add} = \{v|v \in N(C_{v_s}), nss(v, C_{v_s}) \geq nss(v, V - C_{v_s})\} \neq \emptyset$ **do**
19: Update the members of the community, $C_{v_s} = C_{v_s} \cup V^{add}$;
20: Update the neighbors of the community based on Eq. (4);
21: **end while**
22: Update the cover of the graph, $\mathbf{C} = \mathbf{C} \cup \{C_{v_s}\}$;
23: Update the unassigned-node sequence, $U = U - C_{v_s}$;
24: **end while**
25: **Boundary re-checking process:**
26: Initialize a dubious-node sequence D, $D = \{v|v \in B(C), C \in \mathbf{C}\}$;
27: **while** $D \neq \emptyset$ **do**
28: Pop the node v_t with the maximum (or minimum) centrality from D as a target node, $D = D - \{v_t\}$;
29: Get current communities $\mathbf{C}^{cur}_{v_t}$ of v_t, $\mathbf{C}^{cur}_{v_t} = \{C|C \in \mathbf{C}, v_t \in C\}$;
30: Get the fittest communities $\mathbf{C}^{fit}_{v_t}$ of v_t based on Eq. (5), $\mathbf{C}^{fit}_{v_t} = \{C_i|C_i \in \mathbf{C}, \nexists C_j \in \mathbf{C}, nss(v_t, C_i) < nss(v_t, C_j)\}$;
31: **if** $\mathbf{C}^{cur}_{v_t} \neq \mathbf{C}^{fit}_{v_t}$ **then**
32: Move v_t from $\mathbf{C}^{cur}_{v_t}$ to $\mathbf{C}^{fit}_{v_t}$, $\mathbf{C}^{cur}_{v_t} = \{C - \{v_t\}|C \in \mathbf{C}^{cur}_{v_t}\}$, $\mathbf{C}^{fit}_{v_t} = \{C \cup \{v_t\}|C \in \mathbf{C}^{fit}_{v_t}\}$;
33: Update the dubious-node sequence, $D = D \cup \{u|(u, v_t) \in E\}$;
34: **end if**
35: **end while**
36: **return** $\mathbf{C}$;

3.4. The proposed algorithm

The proposed algorithm named LEBR performs LE (*Local Expansion*) and BR (*Boundary re-checking*) sub-processes in order. LEBR is displayed in Algorithm 1. Examples of LE and BR are shown in Figs. 2 and 3. The *initialization* and the sub-processes of LEBR are detailed in the following paragraphs.

Initialization (lines 1–3). The cover of the graph is initialized to empty (line 2). An unassigned-node sequence storing the nodes that have not been assigned to any community is first initialized to the nodes in the graph (line 3). The centrality of each node in the graph is calculated based on Eq. (2) (line 4). In the implementation of LEBR, a structure is used to maintain the ID and centrality of each node, and a structure is used to maintain the members, boundaries and neighbors of each community.

Local expansion (lines 4–26). LE aims to get a cover of the graph. After that, each iteration of LE performs *seeding*, *cleanup*, and *expansion* sub-procedures in order and generates an independent community. Once a community is obtained, the community has to be added to the cover of the graph (line 24), and the members

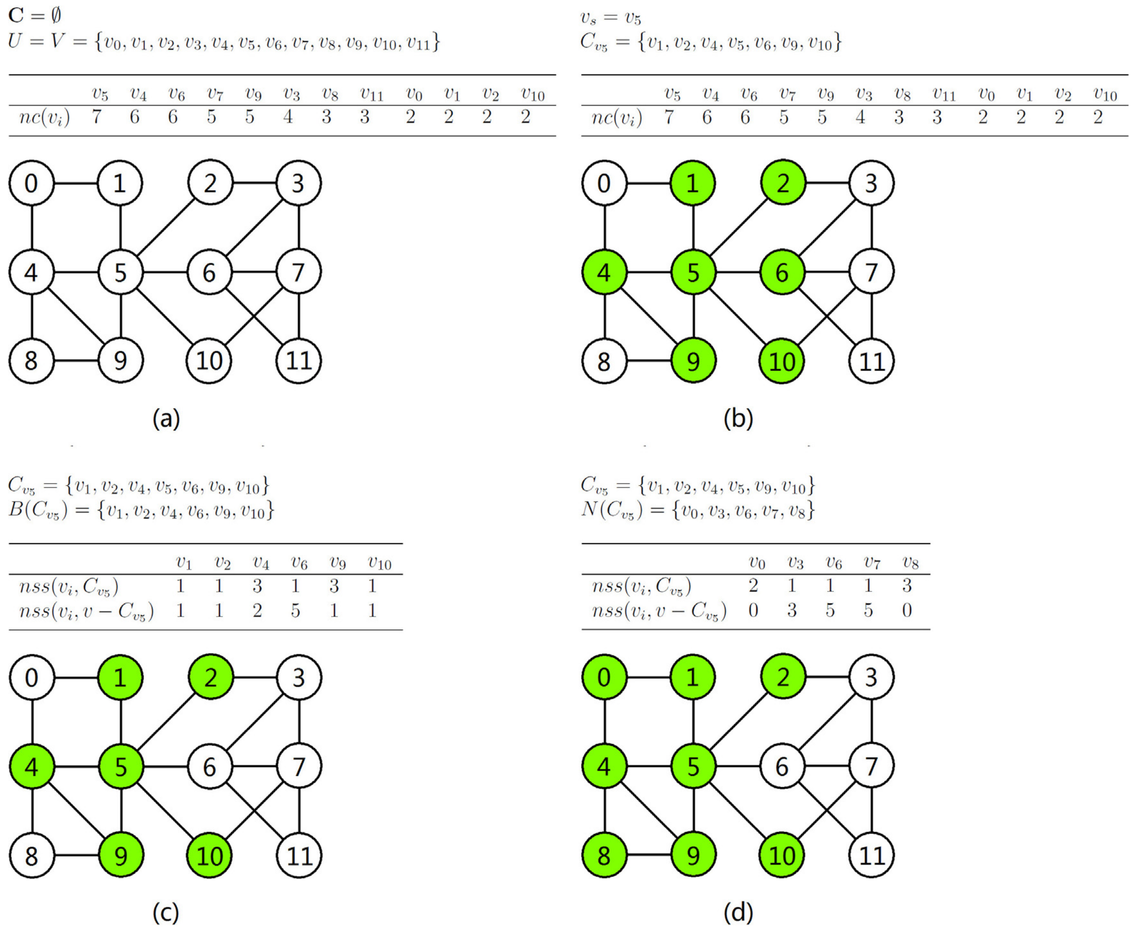


Fig. 2. Examples of LE. (a) is used to explain the initialization of LEBR. (a) shows a toy graph, the initial cover of the graph $\mathbf{C}$ unassigned-node sequence U and the centrality of each node $nc(v_i)$. An example of the seeding procedure of LE is displayed in (b), where node v_5 with the maximum centrality is selected as the seed, and the seed community C_{v_5} is initial as the neighborhood of v_5. An example of the cleanup procedure of LE is displayed in (c), where nodes v_0 and v_8 in community neighbors $N(C_{v_5})$ that are more similar to community C_{v_5} than to the rest of the graph $V - C_{v_5}$ is added to C_{v_5}. An example of the expansion procedure of LE is displayed in (d), where node v_6 in community boundaries $B(C_{v_5})$ that is less similar to community C_{v_5} than to the rest of the graph $V - C_{v_5}$ is deleted from C_{v_5}.

of the community have to be removed from the unassigned-node sequence (line 25). Finally, LE generates the cover of the graph when each node in the graph is assigned to at least one community (line 6).

Seeding (lines 7–9) The seeding procedure generates an initial community. In the seeding procedure, the node with the maximum centrality in the unassigned-node sequence is selected as the seed (line 8). The seed forms an initial community with its neighbors (line 9).

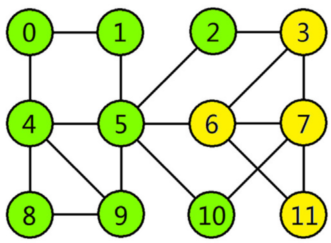
Cleanup (lines 10–16) The cleanup procedure refines the initial community generated by the seeding procedure. The cleanup procedure first gets the boundaries of the community (line 11). The nodes which are less similar to the community than to the rest of the graph have to be removed from the community. If there are nodes that meet the condition to be removed, then the cleanup procedure removes such nodes from the community (line 13) and updates the boundaries of the community (line 14). The cleanup procedure stops when there are no nodes can be removed from the community (line 12).

Expansion (lines 17–23) The expansion procedure expands the refined community generated by the cleanup procedure. The expansion procedure first gets the neighbors of the community (line 18). The nodes which are more similar to the community than to the rest of the graph have to be added to the community. If there are nodes that meet the condition to be added, then the expansion procedure adds such nodes to the community (line 20) and updates the neighbors of the community (line 21). The expansion procedure stops when there are no nodes can be added to the community (line 19).

Boundary re-checking (lines 27–37). BR aims to optimize the cover of the graph resulting from LE. A dubious-node sequence storing the nodes with dubious membership is first initialized to the union of the boundaries of communities in the cover of the graph (line 28). Note that the membership of nodes that interact with multiple communities is dubious and needs to be re-checked. At each iteration, the dubious-node sequence first pops up the node with the maximum (or minimum) centrality as a target node (line 30). After that, BR obtains the *current communities* and the *fittest communities* of the target node from the cover of the graph (lines 31 and 32). The *current communities* of the target node consist of the communities to which the target node currently belongs. The *fittest communities* of the target node consist of the communities that are more similar to the target node than other communities. If the current communities of the target node are different from the fittest communities of the target node (line 33), then BR moves the target node from its current communities to its fittest communities (line 34) and adds its neighbors to the dubious-node sequence (line 35). Finally, BR generates the optimal cover of the graph when the dubious-node sequence becomes empty (line 29).

Supplement. In LE a community has to compete with the rest of the graph for ownership of a node. And we make use of

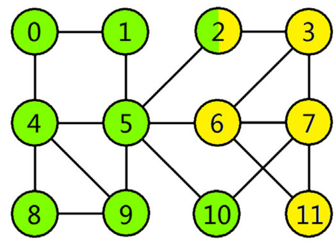
$\mathbf{C} = \{C_{v_5}, C_{v_6}\}$
$C_{v_5} = \{v_0, v_1, v_2, v_4, v_5, v_8, v_9, v_{10}\}$
$C_{v_6} = \{v_3, v_6, v_7, v_{11}\}$
$B(C_{v_5}) = \{v_2, v_5, v_{10}\}$
$B(C_{v_6}) = \{v_3, v_6, v_7\}$
$D = B(C_{v_5}) \cup B(C_{v_6}) = \{v_2, v_3, v_5, v_6, v_7, v_{10}\}$



(a)

	v_5	v_6	v_7	v_3	v_2
$nc(v_i)$	7	6	5	4	2
$C_{v_i}^{cur}$	$\{C_{v_5}\}$	$\{C_{v_6}\}$	$\{C_{v_6}\}$	$\{C_{v_6}\}$	$\{C_{v_5}\}$
$C_{v_i}^{fit}$	$\{C_{v_5}\}$	$\{C_{v_6}\}$	$\{C_{v_6}\}$	$\{C_{v_6}\}$	$\mathbf{C}$

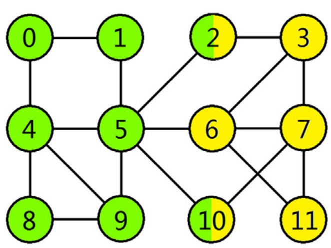
$C_{v_5} = \{v_0, v_1, v_2, v_4, v_5, v_8, v_9, v_{10}\}$
$C_{v_6} = \{v_2, v_3, v_6, v_7, v_{11}\}$
$D = \{v_3, v_5, v_{10}\}$
$\mathbf{C} = \{C_{v_5}, C_{v_6}\}$



(b)

	v_5	v_3	v_{10}
$nc(v_i)$	7	4	2
$C_{v_i}^{cur}$	$\{C_{v_5}\}$	$\{C_{v_6}\}$	$\{C_{v_5}\}$
$C_{v_i}^{fit}$	$\{C_{v_5}\}$	$\{C_{v_6}\}$	$\mathbf{C}$

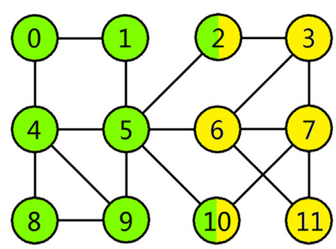
$C_{v_5} = \{v_0, v_1, v_2, v_4, v_5, v_8, v_9, v_{10}\}$
$C_{v_6} = \{v_2, v_3, v_6, v_7, v_{10}, v_{11}\}$
$D = \{v_5, v_7\}$
$\mathbf{C} = \{C_{v_5}, C_{v_6}\}$



(c)

	v_5	v_7
$nc(v_i)$	7	5
$C_{v_i}^{cur}$	$\{C_{v_5}\}$	$\{C_{v_6}\}$
$C_{v_i}^{fit}$	$\{C_{v_5}\}$	$\{C_{v_6}\}$

$C_{v_5} = \{v_0, v_1, v_2, v_4, v_5, v_8, v_9, v_{10}\}$
$C_{v_6} = \{v_2, v_3, v_6, v_7, v_{10}, v_{11}\}$
$D = \emptyset$
$\mathbf{C} = \{C_{v_5}, C_{v_6}\}$



(d)

Fig. 3. Examples of BR. The cover of the graph **C** resulting from LE is displayed in (a), where the dubious-node sequence D is initialized to the union of community boundaries $B(C_{v_5})$ and $B(C_{v_6})$. BR pops and re-checks the membership of nodes in D in descending of node centrality. As displayed in (b), v_2 is the first node uncovered by BR whose current communities are not equal to its fittest communities. BR updates the membership of v_2 by moving v_2 from C_{v_5} to C_{v_5} and C_{v_6} and updates the dubious-node sequence by adding the neighbors of v_2 in D. As displayed in (c), v_{10} is the second node uncovered by BR whose current communities are not equal to its fittest communities. BR updates the membership of v_{10} by moving v_{10} from C_{v_5} to C_{v_5} and C_{v_6} and updates the dubious-node sequence by adding the neighbors of v_{10} in D. (d) displays the cover of the graph resulting from BR.

such competition to determine the one-to-one membership of the node. LE generates communities where each node is more similar to the communities to which it belongs than to the rest of the graph. In BR many communities may compete with each other for ownership of a node. And we make use of such competition to determine the one-to-many membership of the node. BR generates communities where each node is more similar to the communities to which it belongs than to other communities in the cover of the graph.

3.5. Time complexity analysis

Suppose $\overline{d}$ is the mean degree of the graph, $\overline{|C|}$ is average size of the community, and $|\mathbf{C}|$ is the number of communities or detected communities. The time complexity of constructing the unassigned-node sequence and the dubious-node sequence based on Maximum Heap is $O(nlogn)$. The time complexity of calculating the centrality of the node is $O(\overline{d}^2)$. The time complexity of calculating the similarity between the node and the community is $O(\overline{d}log\overline{|C|} + \overline{d}^2)$. The time complexity of LE is $O(nlogn + |\mathbf{C}|\overline{|C|}(\overline{d}log\overline{|C|} + \overline{d}^2)) = O(nlogn + m(log\overline{|C|} + \overline{d}))$. The time complexity of BR is $O(nlogn + n(|\mathbf{C}|\overline{d}log\overline{|C|} + (\overline{d}^2 log\overline{|C|} + \overline{d}^3))) = O(nlogn + |\mathbf{C}|mlog\overline{|C|} + m\overline{d}log\overline{|C|} + m\overline{d}^2)$. Since in most cases $|\mathbf{C}| > \overline{d}$, the total time complexity of LEBR is $O(nlogn + m(|\mathbf{C}|log\overline{|C|} + \overline{d}^2))$.

4. Experiments and analysis

4.1. Experiment settings

In this section, the proposed algorithm that performs only the local expansion process is named LE; the proposed algorithm that re-checks nodes in ascending order of node centrality is named LEBR$_{asc}$; the proposed algorithm that re-checks nodes in descending order of node centrality is named LEBR$_{desc}$.

The proposed algorithm was compared to the following seven state-of-the-art algorithms: LFM (Local Fitness Maximization) [69], DOCN (Detecting Overlapping Communities in Networks) [75], CFM (Community Forest Model) [81], TWD (Three-Way Decision) [82], LERS (Local Expansion based on Relation Strength) [76], LECS (Local Expansion based on Core Similarity) [83] and the Louvain algorithm [7]. Note that all compared algorithms except the Louvain algorithm perform a local expansion process. In addition, LERS was extended to identify overlapping communities based on the working principle of local expansion methods.

In the experiments, LE was used as a baseline to examine the effectiveness of the boundary re-checking process in optimizing the cover of the network. Note that only the experimental results of LEBR_{asc} and LEBR_{desc} reflect the performance of the proposed algorithm. LFM, DOCN, CFM, TWD, LERS and LECS were used to examine the effectiveness of the proposed algorithm in identifying diversely structured communities. The Louvain algorithm, which is a standard and approximately linear disjoint community detection algorithm, was only used for efficiency analysis. The parameter settings of compared algorithms can be found in [83]. The time complexity of the proposed and compared algorithms is listed in Table 1. All experiments were programmed in Java and performed on a computer with Intel (R) Core (TM) i5-6402P CPU, 2.80 GHz, 16 GB RAM. In addition, all algorithms were allowed to run on each network for 24 h before any results were generated.

The proposed and compared algorithms were tested on the following three evaluation criteria: NMI [69] (Normalized Mutual Information), EQ [92] (Expended Modularity) and D-Score [83]. In the experiments, NMI was used to test the algorithm performance in identifying ground-truth communities; EQ was used to test the algorithm performance in identifying highly clustered communities; D-Score was used to test the algorithm performance in estimating the number of communities. A detailed description of the above three evaluation criteria can be found in [69,83,92].

4.2. Experimental results on artificial networks

4.2.1. Artificial networks

LFR [93] (Lancichinetti Fortunato Radicchi) benchmark was employed to produce artificial networks. We produced ten artificial networks with ground-truth communities for each benchmark network in the experiments. The parameter settings of the benchmark networks are listed in Table 2, where d_{max} is the maximum degree of the node, $|C|_{min}$ is the minimum size of the community, $|C|_{max}$ is the maximum size of the community, μ is the mixing parameter, O_n is the number of overlapping nodes in the network, O_m is the overlap degree of each overlapping nodes, and $[x : i : y]$ means the value of the parameter ranges from x to y with a span of i. LFR-μ was used to examine the effect of community identifiability on algorithm performance. LFR-$|C|_{max}$ was used to examine the effect of community size on algorithm performance. LFR-d_{max} was used to examine the effect of node degree on algorithm performance. LFR-O_n was used to examine the effect of the number of overlapping nodes on algorithm performance. LFR-α_n was used to examine the effect of community diversity on algorithm performance.

4.2.2. Algorithm results on LFR-μ

From Tables 3 and 4, we can see that the algorithm performance in identifying ground-truth and highly clustered communities becomes worse with the increase of μ. The reason for this outcome is as follows. For LFR benchmark networks, each node shares a fraction $1 - \mu$ of links with its community and a fraction μ of links with the rest of the network. As μ increases, communities in the network become less identifiable.

From Table 3, we can see that LEBR_{asc} and LEBR_{desc} outperform other algorithms in identifying ground-truth communities. There are two reasons for this outcome. First, LEBR_{asc} and LEBR_{desc} establish associations between boundaries of adjacent communities via the boundary re-checking process, but other algorithms generate independent communities only. Second, LEBR_{asc} and LEBR_{desc} expand and optimize communities by optimizing one-to-one and one-to-many node-community membership, but other algorithms cannot describe one-to-many node-community membership by optimizing quality functions.

From Table 4, it can be found that LEBR_{asc} and LEBR_{desc} show the best and worst average performance in identifying highly clustered communities, respectively. Further, it can be found that LEBR_{asc} outperforms LEBR_{desc} when $\mu \geq 0.3$, while LEBR_{desc} outperforms LEBR_{asc} when $\mu < 0.3$. When we say that communities are highly clustered, we mean that the communities are well-connected internally and well-separated externally. However, highly clustered communities are not necessarily highly similar to ground-truth communities in the network. As conclusive evidence of this statement, it can be found from Table 3 that LEBR_{desc} performs the best in identifying ground-truth communities.

From Table 5, we can observe that LE, LEBR_{asc}, LEBR_{desc} and LERS perform worse than other algorithms in estimating the number of communities. This outcome is because LE, LEBR_{asc}, LEBR_{desc} and LERS are based on node-community membership optimization, which describes the characteristics of the community from the perspective of the node rather than the community.

In Table 6, LE and LFM seem to be more efficient than the Louvain algorithm on LFR-μ. The time performance of LEBR_{asc}, LEBR_{desc} and LECS is comparable to that of the Louvain algorithm.

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

From Tables 7 and 8, we can see that the algorithm performance in identifying ground-truth and highly clustered communities becomes worse with the increase of $|C|_{max}$. This outcome is because the expansion of a community usually ends before the complete structure of the community is obtained. As conclusive evidence of this statement, it can be seen from Table 9 that all algorithms tend to get more communities with the increase of $|C|_{max}$.

From Tables 7 and 8, we can find that LEBR_{asc} and LEBR_{desc} outperform compared algorithms in identifying ground-truth and highly clustered communities. This outcome is because the boundary re-checking process helps LEBR_{asc} and LEBR_{desc} to generate fully structured communities. Two pieces of evidence support this statement. First, it can be found from Table 9 that LEBR_{asc} and LEBR_{desc} always get fewer communities than LE. Second, it can be found from Tables 7 and 8 that LEBR_{asc} and LEBR_{desc} outperform other algorithms in identifying ground-truth and highly clustered communities.

In Table 10, LE seems to be more efficient than the Louvain algorithm on LFR-$|C|_{max}$. The time performance of LFM and LECS is comparable to that of the Louvain algorithm. LEBR_{asc}, LEBR_{desc} and CFM show similar average time costs.

4.2.4. Algorithm results on LFR-d_{max}

From Tables 11–13, we can find that the algorithm performance in identifying ground-truth and highly clustered communities and estimating the number of communities improves with the increase of d_{max}. The reason for this outcome is as follows. In the network links are the basic elements that enable nodes to form communities. As d_{max} increases, the network becomes denser and communities become more identifiable. In Table 14, LE seems to be more efficient than the Louvain algorithm on LFR-d_{max}. The time performance of LFM and LECS is comparable to that of the Louvain algorithm. LEBR_{asc}, LEBR_{desc} and CFM show similar time costs.

Table 1
The time complexity of the proposed and compared algorithms.

Algorithms	Time complexity
LEBR	$O(nlogn + m(\|\mathbf{C}\|log\overline{\|C\|} + \overline{d}^2))$
LFM	$O(\|\mathbf{C}\|nlogn)$
DOCN	$O(n^2)$
CFM	$O(mlogm)$
TWD	$O(\|\mathbf{C}\|nlogn + m\overline{d})$
LERS	$O(m(\overline{d}^3 + log\overline{\|C\|}))$
LECS	$O(\|\mathbf{C}\|nlogn + m)$
Louvain	$O(n)$

Table 2
The parameter settings of LFR benchmark networks.

Networks	n	$\overline{d}$	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 3
NMI on LFR-μ.

μ	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
0.1	0.1185	0.3904	**0.7235**	0.1063	0.2633	0.3020	0.0102	0.2244	0.2723
0.2	0.0569	0.1796	**0.5184**	0.0636	0.1432	0.1432	0.0013	0.1413	0.1709
0.3	0.0369	0.0790	**0.3452**	0.0342	0.0876	0.0602	0.0002	0.0954	0.1006
0.4	0.0158	0.0301	**0.2851**	0.0104	0.0386	0.0095	0.0001	0.0319	0.0370
0.5	0.0077	0.0054	**0.2576**	0.0027	0.0119	0.0012	0.0000	0.0280	0.0060
0.6	0.0015	0.0015	**0.0973**	0.0008	0.0042	0.0009	0.0000	0.0064	0.0007
0.7	0.0001	0.0003	**0.0221**	0.0000	0.0001	0.0016	0.0000	0.0004	0.0000
0.8	0.0000	0.0001	**0.0146**	0.0001	0.0000	0.0009	0.0000	0.0001	0.0000

Table 4
EQ on LFR-μ.

μ	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
0.1	0.1855	0.6196	**0.7821**	0.2426	0.2883	0.3872	0.3369	0.3680	0.5055
0.2	0.0928	0.4368	**0.6282**	0.1928	0.2257	0.3144	0.2949	0.2420	0.3959
0.3	0.0573	**0.3247**	0.2827	0.1711	0.2054	0.2781	0.2717	0.1954	0.3242
0.4	0.0271	0.2670	0.0255	0.1550	0.1899	0.2387	0.2545	0.1635	**0.2711**
0.5	0.0133	0.2413	0.0029	0.1485	0.1846	0.2173	**0.2456**	0.1837	0.2423
0.6	0.0042	0.2283	0.0002	0.1454	0.1810	0.2095	**0.2393**	0.1145	0.2232
0.7	0.0018	0.2235	0.0001	0.1440	0.1792	0.1751	**0.2348**	0.1209	0.2142
0.8	0.0012	0.2220	0.0000	0.1426	0.1784	0.1674	**0.2346**	0.1556	0.2123

Table 5
D-Score on LFR-μ.

μ	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
0.1	82.04	13.56	1.12	21.28	14.45	6.30	6.58	**0.44**	7.66
0.2	94.43	23.08	2.52	24.35	18.76	7.52	7.47	**0.06**	10.64
0.3	92.36	29.02	**0.90**	24.14	19.40	8.73	7.27	−0.96	12.57
0.4	100.78	35.00	−5.11	26.36	21.94	9.00	7.51	−**4.88**	15.23
0.5	99.41	36.35	−24.31	25.94	21.95	9.36	**7.91**	−15.04	15.95
0.6	94.51	35.00	−69.25	24.49	20.73	7.73	**7.29**	−49.89	15.50
0.7	90.45	33.78	−93.00	23.35	19.76	**5.30**	7.14	−57.08	14.85
0.8	99.12	37.26	−89.25	25.23	21.38	**5.25**	7.83	−43.81	16.01

Table 6
Time (s) on LFR-μ.

μ	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS	Louvain
0.1	**2.10**	12.16	15.85	5.63	751.83	15.12	192.96	35.78	7.15	2.69
0.2	**2.19**	13.52	20.38	4.65	278.34	21.45	238.77	54.71	8.85	3.41
0.3	**2.17**	13.39	22.57	4.93	268.81	21.48	271.99	61.65	9.21	6.65
0.4	**2.49**	13.68	18.49	4.55	323.58	46.92	276.61	77.15	11.27	8.14
0.5	**2.69**	14.09	18.02	5.53	319.24	82.08	360.22	77.08	12.81	8.22
0.6	**2.68**	14.13	18.47	7.64	255.45	109.24	364.50	76.22	13.45	8.94
0.7	**2.88**	14.72	18.71	6.59	668.63	208.24	413.02	81.26	16.37	11.32
0.8	**2.83**	14.56	19.10	5.19	770.50	232.00	390.41	75.47	18.67	10.39

Table 7
NMI on LFR-$|C|_{max}$.

$\|C\|_{max}$	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	0.2389	0.5654	**0.7365**	0.2432	0.4518	0.4449	0.0222	0.2937	0.4516
400	0.1511	0.4222	**0.7255**	0.1325	0.3143	0.3362	0.0115	0.2666	0.3153
600	0.1053	0.3436	**0.7340**	0.0857	0.2147	0.2746	0.0099	0.2403	0.2386
800	0.0742	0.3066	**0.7506**	0.0747	0.1679	0.2493	0.0095	0.2173	0.2099
1000	0.0758	0.2514	**0.7013**	0.0661	0.1544	0.2345	0.0074	0.2002	0.2025

Table 8
EQ on LFR-$|C|_{max}$.

$\|C\|_{max}$	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	0.3446	0.7278	**0.7934**	0.3513	0.4110	0.4994	0.3990	0.5580	0.6261
400	0.2286	0.6462	**0.7863**	0.2619	0.3127	0.4162	0.3496	0.4230	0.5354
600	0.1619	0.5868	**0.7843**	0.2265	0.2776	0.3781	0.3256	0.3613	0.4776
800	0.1169	0.5576	**0.7814**	0.2121	0.2627	0.3480	0.3122	0.3939	0.4497
1000	0.1119	0.5041	**0.7580**	0.2052	0.2508	0.3373	0.3136	0.3750	0.4285

Table 9
D-Score on LFR-$|C|_{max}$.

$\|C\|_{max}$	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	29.63	3.03	0.75	7.44	4.44	2.20	2.53	**0.15**	2.61
400	60.24	8.95	1.11	15.75	10.32	4.53	5.19	**0.45**	5.71
600	97.68	17.51	1.12	25.52	17.92	7.07	7.93	**0.61**	9.52
800	118.68	22.37	1.11	30.52	21.78	9.23	9.60	**0.99**	11.78
1000	150.79	33.73	1.38	39.03	29.17	12.52	11.83	**1.14**	16.15

Table 10
Time (s) on LFR-$|C|_{max}$.

$\|C\|_{max}$	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS	Louvain
200	1.40	8.42	9.81	5.81	146.91	7.07	143.60	19.92	5.55	**1.07**
400	**1.75**	11.12	14.12	5.32	268.65	10.69	189.13	30.43	6.74	2.10
600	**2.09**	13.35	16.97	5.56	1034.67	16.74	203.11	36.36	7.75	2.51
800	**2.11**	14.46	18.36	4.88	557.83	16.32	232.84	35.63	8.54	3.36
1000	**2.12**	13.89	18.57	5.46	256.58	17.07	218.55	39.07	8.69	3.84

Table 11
NMI on LFR-d_{max}.

d_{max}	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	0.0937	0.3334	**0.6981**	0.1011	0.2092	0.2831	0.0047	0.2128	0.2399
400	0.1136	0.3788	**0.7241**	0.0953	0.2223	0.2892	0.0086	0.2152	0.2564
600	0.1047	0.3534	**0.7593**	0.1022	0.2388	0.2959	0.0081	0.2560	0.2655
800	0.1162	0.3548	**0.7139**	0.1149	0.2514	0.3032	0.0099	0.2176	0.2688
1000	0.1284	0.4057	**0.7190**	0.1064	0.2735	0.3304	0.0079	0.2531	0.2962

Table 12
EQ on LFR-d_{max}.

d_{max}	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	0.1492	0.5867	**0.7735**	0.2413	0.2927	0.3754	0.3214	0.4554	0.4805
400	0.1736	0.6140	**0.7817**	0.2326	0.2859	0.3945	0.3381	0.4568	0.4942
600	0.1626	0.5975	**0.7935**	0.2417	0.2846	0.4000	0.3381	0.4216	0.4983
800	0.1841	0.5985	**0.7812**	0.2477	0.2990	0.3925	0.3419	0.4178	0.5007
1000	0.2033	0.6370	**0.7819**	0.2473	0.2988	0.4156	0.3336	0.3458	0.5274

4.2.5. Algorithm results on LFR-O_n

From Tables 15–18, we cannot observe significant variation in the algorithm performance in identifying ground-truth and highly clustered communities, estimating the number of communities and efficiency, as O_n increases. One can image that overlapping nodes are located at the boundaries of the community, while the seeds selected by local expansion algorithms are located in the core area of the community. Therefore, local expansion algorithms are unlikely to select overlapping nodes as seeds and consequently get low-quality communities.

4.2.6. Algorithm results on LFR-α_n

From Tables 19 and 20, we can find that the algorithm performance in identifying ground-truth and highly clustered communities becomes worse with the increase of α_n. This outcome confirms our discussion in Section 1 that local expansion methods are prevented from identifying diversely structured communities. Further, we can find that LEBR$_{asc}$ and LEBR$_{desc}$ show much better average performance than compared algorithms in identifying ground-truth and highly clustered communities. This outcome validates the effectiveness of the proposed algorithm in identifying diversely structured communities.

Table 13
D-Score on LFR-d_{max}.

d_{max}	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	84.96	15.29	1.47	21.17	15.12	6.59	6.98	**1.03**	7.99
400	83.97	13.74	1.20	21.58	14.94	6.75	6.80	**0.88**	7.97
600	82.17	14.07	0.95	20.84	14.39	6.78	6.22	**0.53**	7.55
800	77.72	14.21	1.39	19.92	13.78	5.85	6.40	**0.81**	7.52
1000	75.58	11.66	1.19	19.95	13.20	5.68	6.07	**0.31**	7.01

Table 14
Time (s) on LFR-d_{max}.

d_{max}	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS	Louvain
200	**1.77**	12.95	16.43	6.01	74.65	9.60	221.76	25.72	6.86	3.47
400	**1.88**	12.57	15.79	5.66	269.86	11.64	202.17	29.54	6.84	2.41
600	**1.94**	12.90	16.02	5.70	348.20	12.25	177.40	32.31	7.29	3.43
800	**1.84**	12.09	15.63	7.48	333.38	12.07	205.45	31.74	7.20	2.79
1000	**1.95**	11.90	14.70	5.89	514.57	14.06	178.92	36.15	6.84	2.34

Table 15
NMI on LFR-O_n.

O_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	0.1172	0.3785	**0.7835**	0.1014	0.2301	0.3055	0.0099	0.2527	0.2804
400	0.1196	0.4086	**0.7562**	0.1076	0.2492	0.3256	0.0084	0.2540	0.2977
600	0.0981	0.3515	**0.6960**	0.0979	0.2313	0.2792	0.0062	0.2272	0.2506
800	0.1068	0.3662	**0.6758**	0.1039	0.2388	0.2772	0.0078	0.2235	0.2641
1000	0.1039	0.3143	**0.6192**	0.0934	0.2259	0.2541	0.0050	0.2178	0.2329

Table 16
EQ on LFR-O_n.

O_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	0.1773	0.6179	**0.8094**	0.2421	0.3097	0.3892	0.3403	0.3889	0.5134
400	0.1835	0.6366	**0.7940**	0.2421	0.2976	0.4039	0.3469	0.4686	0.5204
600	0.1614	0.6032	**0.7723**	0.2396	0.2873	0.3801	0.3318	0.4131	0.4913
800	0.1699	0.6095	**0.7626**	0.2392	0.2857	0.3806	0.3344	0.3800	0.4942
1000	0.1739	0.5763	**0.7418**	0.2355	0.2734	0.3752	0.3348	0.3853	0.4707

Table 17
D-Score on LFR-O_n.

O_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
200	80.93	13.65	1.09	20.73	14.10	5.94	6.76	**0.59**	7.26
400	77.18	11.69	1.02	19.77	13.29	5.58	5.88	**0.74**	6.84
600	83.75	14.03	1.41	21.16	14.86	6.32	6.89	**0.73**	7.95
800	80.37	13.11	1.39	20.65	14.53	6.04	6.47	**0.55**	7.80
1000	79.58	14.42	1.61	20.59	14.58	6.06	6.61	**0.60**	8.08

Table 18
Time (s) on LFR-O_n.

O_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS	Louvain
200	**1.80**	12.33	15.47	6.21	243.84	11.04	203.80	30.58	7.03	2.24
400	1.87	12.16	15.01	8.17	399.94	11.57	163.05	29.61	7.17	**1.81**
600	**1.86**	12.72	16.34	4.93	255.87	11.36	214.97	32.85	7.22	2.97
800	**1.96**	12.82	16.14	4.99	464.07	13.81	199.28	35.47	7.59	2.21
1000	**1.89**	12.91	17.19	4.48	340.91	12.96	216.11	38.16	7.80	2.87

From Table 21, we can find that the number of communities generated by each algorithm increases with the increase of α_n. As discussed in Section 4.2.3, the increase in community size is the main reason for this outcome.

In Table 22, LE and LFM seem to be more efficient than the Louvain algorithm on LFR-α_n. The time cost of LEBR$_{asc}$ and LECS seems to be linearly related to the size of the network.

4.3. Experimental results on real-world networks

4.3.1. Real-world networks

Table 23 lists the characteristics of the real-world networks used in the experiments. The data sets of networks without ground-truth communities (DE, EN, ES, FR, PT, RU, Facebook and GitHub) can be found from http://snap.stanford.edu/data/. We carried out a three-step pre-processing procedure on these data sets to extract community-like structures (or uncertain communities). The three-step pre-processing procedure is detailed as follows.

Table 19
NMI on LFR-α_n.

α_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
1	0.7417	0.8684	0.8788	**0.9378**	0.9115	0.8784	0.7544	0.7204	0.9067
2	0.3090	0.6322	**0.7722**	0.5044	0.6076	0.5468	0.2718	0.4621	0.6111
3	0.1354	0.3985	**0.8193**	0.1165	0.2454	0.3228	0.0193	0.2823	0.2800
4	0.0252	0.1472	**0.7891**	0.0323	0.0714	0.1324	0.0014	0.1462	0.0938
5	0.0074	**0.0480**	–	0.0053	–	–	–	–	0.0261

Table 20
EQ on LFR-α_n.

α_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
1	0.5366	0.6016	0.6040	0.6006	0.5847	0.5559	0.5586	0.5411	**0.6146**
2	0.3761	0.6847	**0.7289**	0.4926	0.4854	0.4958	0.4580	0.5324	0.6416
3	0.1921	0.6177	**0.8006**	0.2510	0.3093	0.3928	0.3383	0.3685	0.4983
4	0.0501	0.4004	**0.8263**	0.1766	0.2075	0.3125	0.2914	0.2492	0.3485
5	0.0132	**0.2824**	–	0.1496	–	–	–	–	0.2406

Table 21
D-Score on LFR-α_n.

α_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
1	1.32	0.17	0.15	0.07	**0.00**	0.13	0.03	0.01	0.06
2	16.69	1.43	0.53	3.29	2.24	1.09	1.31	**0.00**	1.43
3	80.23	13.25	0.92	20.52	14.35	6.29	6.73	**0.44**	7.63
4	378.49	109.6	**0.84**	98.64	79.35	43.02	33.11	1.69	48.24
5	1652.81	573.49	–	434.14	–	–	–	–	**274.48**

Table 22
Time (s) on LFR-α_n.

α_n	LE	LEBR$_{asc}$	LEBR$_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS	Louvain
1	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
2	0.03	0.08	0.08	0.18	0.23	0.07	0.05	0.06	0.05	**0.01**
3	**0.66**	3.47	4.39	2.90	44.09	3.93	26.55	7.44	2.24	**0.66**
4	**46.77**	424.67	541.10	61.25	18692.33	751.28	62328.20	1497.50	281.80	152.62
5	**4006.00**	57196.23	–	5669.48	–	–	–	–	24014.44	15009.31

Table 23
The characteristics of real-world networks.

Networks	n	m	$\bar{d}$	$\lvert\mathbf{C}\rvert$	$\overline{\lvert C\rvert}$	μ	O_n	O_m	Ground-truth	Source
Karate	34	78	4.58	2	17.00	0.128	0	–	yes	[94]
Dolphin	62	159	5.12	2	31.00	0.038	0	–	yes	[95]
Football	115	613	10.66	12	9.58	0.357	0	–	yes	[65]
Book	105	440	8.38	3	35.0	0.159	0	–	yes	[96]
Amazon	16716	48739	5.83	1163	15.16	0.005	867	2.06	yes	[88]
DBLP	93432	335520	7.18	4876	22.84	0.305	13439	2.33	yes	[88]
Youtube	39841	224235	11.26	4481	15.95	0.838	11935	3.65	yes	[88]
LiveJournal	84438	1521988	36.05	4090	29.33	0.204	22398	2.59	yes	[88]
DE	4027	8454	4.20	2257	7.59	0.530	2239	6.85	no	[97]
EN	918	1081	2.36	560	4.18	0.266	507	3.80	no	[97]
ES	1529	3253	4.26	978	6.96	0.470	958	6.51	no	[97]
FR	2521	6417	5.09	1689	7.81	0.562	1670	7.39	no	[97]
PT	1009	3725	7.38	811	9.51	0.609	809	9.29	no	[97]
RU	896	1800	4.02	565	6.65	0.473	547	6.23	no	[97]
Facebook	6858	36545	10.66	4858	6.69	0.650	4726	6.43	no	[97]
GitHub	9848	15489	3.15	5134	6.12	0.439	5012	5.30	no	[97]

1 Get the projection (or extracted subgraph) of the network on each node feature.
2 Remove links that have an HPI (Hub Promoted Index) [98] value less than 0.5 from the extracted subgraphs generated by step 1.
3 Output connected components (or uncertain communities) that have at least 3 nodes generated by step 2.

Notes. If $C_i \subset C_j$, $i \neq j$, then C_i is a sub-community of C_j. If $C_i \subseteq C_j$, $C_j \subseteq C_i$ and $i \neq j$, then C_i and C_j are duplicate communities of each other. Sub-communities are not allowed to participate in the calculation of the evaluation criteria. Only one sample from each group of duplicate communities is allowed to participate in the calculation of the evaluation criteria. The communities that are neither sub-communities nor duplicate communities are all required to participate in the calculation of the evaluation criteria.

4.3.2. Algorithm results on real-world networks

From Tables 24 and 25, we can find that LEBR$_{asc}$ and LEBR$_{desc}$ show better average performance than LE in identifying ground-truth, uncertain and highly clustered communities. This outcome validates the effectiveness of the boundary re-checking process in optimizing the cover of the network. Further, we can find that LEBR$_{asc}$ and LEBR$_{desc}$ show better average performance than compared algorithms in identifying ground-truth, uncertain and highly clustered communities. This outcome validates the effectiveness of the proposed algorithm in identifying diversely structured communities.

Table 24
NMI on real-world networks.

Networks	LE	$LEBR_{asc}$	$LEBR_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
Karate	0.9185	0.9185	0.9185	0.3725	0.5274	0.8253	0.7047	**1.0000**	0.8318
Dolphin	0.2972	0.4909	0.5153	0.3183	**0.6704**	0.3884	0.3638	0.5573	0.3336
Football	0.7877	0.7632	0.7632	0.7944	0.5223	0.7989	0.5190	0.5885	**0.8339**
Book	0.4263	0.4558	0.4558	0.4310	0.4555	0.4594	**0.5139**	0.4978	0.3845
Amazon	0.6745	0.6915	0.6932	0.6684	0.7141	0.6553	**0.7425**	0.6981	0.6916
DBLP	0.3238	0.3205	0.3214	**0.3289**	0.3133	0.2605	0.0608	0.1650	0.3036
Youtube	0.2338	**0.3409**	0.3248	0.2423	–	0.0630	0.0857	0.3015	0.1136
LiveJournal	0.7710	0.7991	0.8027	0.7981	**0.8116**	0.6858	0.2024	0.5404	–
DE	0.3507	**0.3590**	0.3562	0.2836	0.2624	0.2996	0.3134	0.3339	0.2870
EN	0.6827	**0.6944**	0.6906	0.6860	0.6195	0.6707	0.5856	0.5567	0.6648
ES	0.4075	**0.4329**	0.4229	0.4077	0.3728	0.3483	0.3323	0.3493	0.3680
FR	0.2841	**0.3014**	0.2928	0.2890	0.2690	0.2498	0.2368	0.2796	0.2782
PT	0.1990	**0.3174**	0.3025	0.1915	0.2231	0.2236	0.1902	0.2255	0.2905
RU	0.4227	0.4538	0.4520	0.4304	0.4192	0.4197	0.4019	0.3947	**0.4559**
Facebook	0.3069	0.3041	0.3052	**0.3189**	0.2731	0.2847	0.2778	0.2281	0.3088
GitHub	0.4235	**0.5031**	0.4908	0.4660	–	0.4254	0.3873	0.3599	0.3401

Table 25
EQ on real-world networks.

Networks	LE	$LEBR_{asc}$	$LEBR_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
Karate	0.3717	0.3717	0.3717	**0.4021**	0.3313	0.3580	0.3456	0.3715	0.3133
Dolphin	0.4717	**0.5261**	0.5153	0.4661	0.3713	0.4261	0.3630	0.4907	0.4938
Football	0.5576	0.5835	0.5835	0.5478	0.4877	0.5767	0.4739	0.5632	**0.6005**
Book	0.5094	0.5151	0.5151	0.4649	0.4942	**0.5238**	0.4244	0.4527	0.4959
Amazon	0.9242	0.9490	**0.9498**	0.9134	0.9263	0.8801	0.9331	0.9336	0.9390
DBLP	0.7105	0.7826	0.7897	0.6133	0.6090	0.6624	0.4056	**0.8397**	0.6573
Youtube	0.3636	**0.4356**	0.3886	0.2012	–	0.1285	0.3712	0.0801	0.1140
LiveJournal	0.9442	0.9611	**0.9620**	0.9219	0.9134	0.8278	0.8418	0.9353	–
DE	0.2336	0.1916	0.1922	**0.3908**	0.2112	0.2860	0.2313	0.1344	0.2298
EN	0.8547	0.8765	0.8806	0.8322	0.8051	0.8672	0.8221	0.8295	**0.8842**
ES	0.7171	**0.7480**	0.7464	0.6966	0.5316	0.5965	0.6494	0.7298	0.5686
FR	**0.5369**	0.5274	0.5276	0.4247	0.4049	0.4108	0.4259	0.4749	0.4389
PT	0.3384	0.3457	0.3303	**0.3642**	0.3183	0.2961	0.2848	0.3089	0.3462
RU	0.5890	**0.6528**	0.6480	0.5585	0.5292	0.5225	0.5445	0.5695	0.6208
Facebook	0.9433	0.9511	**0.9513**	0.9027	0.8788	0.8012	0.9333	0.9291	0.8791
GitHub	0.5562	0.5798	**0.5805**	0.5556	–	0.5397	0.5338	0.5176	0.3363

Table 26
D-Score on real-world networks.

Networks	LE	$LEBR_{asc}$	$LEBR_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS
Karate	**0.00**	**0.00**	**0.00**	1.00	0.50	**0.00**	**0.00**	**0.00**	0.50
Dolphin	4.50	1.00	1.00	2.50	**0.50**	1.00	1.00	**0.50**	2.00
Football	0.67	**0.00**	**0.00**	0.17	−0.33	0.00	−0.33	−0.71	**0.00**
Book	1.33	0.67	0.67	1.33	**0.00**	0.33	−0.50	−0.50	1.00
Amazon	0.71	0.41	0.41	0.55	0.41	0.49	**0.26**	0.89	0.45
DBLP	2.29	0.80	0.74	2.01	1.23	**0.63**	−1.24	−0.67	0.90
Youtube	1.53	−2.08	−2.21	**0.26**	–	−3.39	−4.73	−4.32	−0.85
LiveJournal	0.59	−0.22	−0.24	0.12	**−0.09**	−0.20	−0.74	0.60	–
DE	−11.13	−25.24	−25.55	−8.48	−14.05	−24.36	−24.65	−9.09	**−6.95**
EN	−1.80	−2.18	−2.24	−2.03	−2.24	−2.31	−3.75	**−1.60**	−2.27
ES	**−4.23**	−8.14	−8.23	−6.19	−6.47	−9.75	−11.54	−10.64	−5.27
FR	**−4.76**	−20.65	−21.82	−7.66	−14.78	−22.46	−21.82	−28.12	−8.60
PT	**−1.65**	−26.03	−31.44	−10.11	−22.17	−31.44	−37.62	−37.62	−11.87
RU	**−2.49**	−6.34	−6.43	−5.73	−5.01	−6.24	−7.43	−6.06	−4.59
Facebook	−8.01	−9.31	−9.34	−7.58	−9.16	−9.04	−10.38	**−6.00**	−8.01
GitHub	−3.88	−9.79	−10.04	−6.33	–	−13.22	−13.63	−9.99	**−2.77**

From Table 26, we can find that the boundary re-checking process fails to improve the results of LE on Youtube, DE, EN, ES, FR, PT, RU, Facebook and GitHub. Observing Table 23, we can find that Youtube, DE, EN, ES, FR, PT, RU, Facebook and GitHub either have a high value of the mixing parameter or a high proportion of overlapping nodes. These findings indicate that the boundary re-checking process may cause the proposed algorithm to fail to distinguish between highly mixed or overlapped communities.

From Table 27, we can observe that $LEBR_{asc}$ and $LEBR_{desc}$ show better average time performance than compared algorithms in efficiency. The efficiency of $LEBR_{asc}$ and $LEBR_{desc}$ is second only to that of the Louvain algorithm.

5. Conclusions

Two problems prevent local expansion methods from identifying diversely structured communities in the network. First, local expansion methods generate independent communities only. Second, local expansion methods depend heavily on quality functions. This work provides a solution for local expansion methods to identify diversely structured communities. This paper proposed an overlapping community detection algorithm that performs local expansion and boundary re-checking sub-processes in order. The local expansion process first gets a cover of the network, and then the boundary re-checking process optimizes the

Table 27
Time (s) on real-world networks.

Networks	LE	$LEBR_{asc}$	$LEBR_{desc}$	LFM	DOCN	CFM	TWD	LERS	LECS	Louvain
Karate	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.02	0.03	**0.00**
Dolphin	**0.02**	**0.02**	**0.02**	**0.02**	0.03	**0.02**	**0.02**	0.05	0.03	**0.02**
Football	0.03	0.03	0.03	0.03	0.05	0.03	0.02	0.09	0.09	**0.00**
Book	0.02	0.02	0.02	0.08	0.06	0.05	0.03	0.06	0.09	**0.00**
Amazon	**0.48**	2.12	2.11	2.15	1.17	5.06	1136.37	1.62	12.58	1.30
DBLP	**20.89**	265.25	272.11	70.19	438.11	630.41	3594.51	3165.43	462.96	37.82
Youtube	310.66	428.13	504.86	9451.4	–	24068.06	4342.21	3116.65	38762.76	**9.81**
LiveJournal	191.81	302.1	312.78	52715.83	69066.57	26041.73	54988.22	85754.44	–	**55.71**
DE	4.90	5.16	5.41	2330.18	67280.80	958.93	8.38	22.19	15.36	**0.13**
EN	0.03	0.03	0.03	0.05	0.09	0.05	0.59	0.03	0.06	**0.02**
ES	0.19	0.20	0.25	17.19	27.99	2.12	1.17	1.05	0.86	**0.02**
FR	1.26	1.69	2.10	80.30	756.79	38.67	3.00	12.45	4.26	**0.05**
PT	1.06	1.33	1.42	18.68	122.76	7.43	0.59	2.83	0.95	**0.02**
RU	0.20	0.23	0.28	19.92	7.05	1.10	0.29	0.78	0.24	**0.02**
Facebook	1.14	1.26	1.12	163.75	245.46	14.07	61.55	19.69	19.49	**0.25**
GitHub	29.22	32.25	36.72	16580.07	–	5114.72	51.56	1350.86	13637.28	**0.38**

cover of the network resulting from the local expansion process. To solve the first problem, the proposed algorithm establishes associations between boundaries of adjacent communities via the boundary re-checking process. To solve the second problem, the proposed algorithm expands and optimizes the community based on node-community membership optimization.

In the experiments, the proposed algorithm was compared to seven state-of-the-art algorithms by examining their performance on five groups of artificial networks and sixteen real-world networks. Two positive conclusions can be drawn from the experimental results. First, the boundary re-checking process is effective in optimizing the cover of the network. And this verifies that establishing associations between adjacent communities helps the proposed algorithm to optimize independent communities. Second, the proposed algorithm outperforms compared algorithms in identifying diversely structured communities. And this verifies that optimizing one-to-one and one-to-many node-community membership helps the proposed algorithm to identify diversely structured communities in the network.

Nevertheless, the experimental results uncover that the boundary re-checking process may cause the proposed algorithm to fail to distinguish between highly mixed or overlapped communities. Therefore, in future work, we may introduce adaptive constraints into the boundary re-examination process to clearly distinguish adjacent communities.

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.

Acknowledgments

National Natural Science Foundation of China (Nos. 61672179, 61370083, 61402126), Natural Science Foundation of Heilongjiang Province, China (No. F2015030), Science Foundation for Youths of Heilongjiang Province, China (No. QC2016083) and Heilongjiang Postdoctoral Science Foundation, China (No. LBH-Z14071) support this paper.

References

[1] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.
[2] M.A. Javed, M.S. Younis, S. Latif, J. Qadir, A. Baig, Community detection in networks: A multidisciplinary review, J. Netw. Comput. Appl. 108 (2018) 87–111.
[3] J. Zhang, Z. Ma, Q. Sun, J. Yan, Research review on algorithms of community detection in complex networks, J. Phys. Conf. Ser. (2018) 012124.
[4] M. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.
[5] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.
[6] U. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 73 (3) (2007) 036106.
[7] V. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) P10008.
[8] J.Q. Jiang, A.W. Dress, G. Yang, A spectral clustering-based framework for detecting community structures in complex networks, Appl. Math. Lett. 22 (9) (2009) 1479–1482.
[9] Y. Xu, H. Xu, D. Zhang, A novel disjoint community detection algorithm for social networks based on backbone degree and expansion, Expert Syst. Appl. 42 (21) (2015) 8349–8360.
[10] L. Bai, X. Cheng, J. Liang, Y. Guo, Fast graph clustering with a new description model for community detection, Inform. Sci. 388–389 (2017) 37–47.
[11] K.R. Žalik, B. Žalik, Memetic algorithm using node entropy and partition entropy for community detection in networks, Inform. Sci. 445–446 (2018) 38–49.
[12] J. Sánchez-Oro, A. Duarte, Iterated greedy algorithm for performing community detection in social networks, Future Gener. Comput. Syst. 88 (2018) 785–791.
[13] F. Cheng, T. Cui, Y. Su, Y. Niu, X. Zhang, A local information based multi-objective evolutionary algorithm for community detection in complex networks, Appl. Soft Comput. 69 (2018) 357–367.
[14] B. Tian, W. Li, Community detection method based on mixed-norm sparse subspace clustering, Neurocomputing 275 (2018) 2150–2161.
[15] R. Belfin, G. E., P. Bródka, Overlapping community detection using superior seed set selection in social networks, Comput. Electr. Eng. 70 (2018) 1074–1083.
[16] C. Zhou, L. Feng, Q. Zhao, A novel community detection method in bipartite networks, Physica A 492 (2018) 1679–1693.
[17] F. Zhang, A. Ma, Z. Wang, Q. Ma, B. Liu, L. Huang, Y. Wang, A central edge selection based overlapping community detection algorithm for the detection of overlapping structures in protein-protein interaction networks, Molecules 23 (10) (2018) 2633.
[18] B. Tripathi, S. Parthasarathy, H. Sinha, K. Raman, B. Ravindran, Adapting community detection algorithms for disease module identification in heterogeneous biological networks, Front. Genet. 10 (MAR) (2019) 00164.
[19] T. Ma, Y. Wang, M. Tang, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, LED: A fast overlapping communities detection algorithm based on structural clustering, Neurocomputing 207 (2016) 488–500.
[20] H. Liu, C. Zhao, Y. Tian, J. Yang, Density peaks based clustering algorithm for overlapping community detection, in: Proceedings - 2016 12th International Conference on Semantics, Knowledge and Grids, SKG 2016 7815070, 2017, pp. 1–8.
[21] C. Shang, S. Feng, Z. Zhao, J. Fan, Efficiently detecting overlapping communities using seeding and semi-supervised learning, Int. J. Mach. Learn. Cybern. 8 (2) (2017) 455–468.
[22] G. Liu, K. Meng, H. Guo, L. Pan, J. Li, Automatic threshold calculation based label propagation algorithm for overlapping community, in: Proceedings - 2016 IEEE 1st International Conference on Data Science in Cyberspace, DSC 2016 7866155, 2017, pp. 382–387.
[23] X. Bai, P. Yang, X. Shi, An overlapping community detection algorithm based on density peaks, Neurocomputing 226 (2017) 7–15.

[24] W. Li, S. Jiang, Q. Jin, Overlap community detection using spectral algorithm based on node convergence degree, Future Gener. Comput. Syst. 79 (2018) 408–416.
[25] M. Huang, G. Zou, B. Zhang, Y. Liu, Y. Gu, K. Jiang, Overlapping community detection in heterogeneous social networks via the user model, Inform. Sci. 432 (2018) 164–184.
[26] I. Farkas, D. Ábel, G. Palla, T. Vicsek, Weighted network modules, New J. Phys. 9 (2007) 108.
[27] 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.
[28] J.M. Kumpula, M. Kivelä, K. Kaski, J. Saramäki, Sequential algorithm for fast clique percolation, Phys. Rev. E 78 (2) (2008) 026109.
[29] S. Maity, S.K. Rath, Extended Clique percolation method to detect overlapping community structure, in: 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014, Delhi, India, September 24-27, 2014, 2014, pp. 31–37.
[30] T. Deng, D. Ye, R. Ma, H. Fujita, L. Xiong, Low-rank local tangent space embedding for subspace clustering, Inform. Sci. 508 (2020) 1–21.
[31] T. Evans, R. Lambiotte, Line graphs, link partitions, and overlapping communities, Phys. Rev. E 80 (1) (2009) 016105.
[32] Y. Kim, H. Jeong, Map equation for link communities, Phys. Rev. E 84 (2) (2011) 026110.
[33] M. Rosvall, C. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. USA 105 (4) (2008) 1118–1123.
[34] Y.-Y. Ahn, J. Bagrow, S. Lehmann, Link communities reveal multiscale complexity in networks, Nature 466 (7307) (2010) 761–764.
[35] D. He, D. Jin, C. Baquero, D. Liu, Link community detection using generative model and nonnegative matrix factorization, PLoS One 9 (1) (2014) e86899.
[36] J. Kim, S. Lim, J. Lee, B.S. Lee, Linkblackhole*: Robust overlapping community detection using link embedding, IEEE Trans. Knowl. Data Eng. 31 (11) (2019) 2138–2150.
[37] H. Tao, Z. Li, Z. Wu, J. Cao, Link communities detection: an embedding method on the line hypergraph, Neurocomputing 367 (2019) 46–54.
[38] S. Gregory, Finding overlapping communities in networks by label propagation, New J. Phys. 12 (2010) 103018.
[39] J. Xie, B.K. Szymanski, X. Liu, SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process, in: Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, December 11, 2011, 2011, pp. 344–349.
[40] W. Chen, Z. Liu, X. Sun, Y. Wang, A game-theoretic framework to identify overlapping communities in social networks, Data Min. Knowl. Discov. 21 (2) (2010) 224–240.
[41] F.A. Breve, L. Zhao, M.G. Quiles, Uncovering overlap community structure in complex networks using particle competition, in: Artificial Intelligence and Computational Intelligence, International Conference, AICI 2009, Shanghai, China, November 7-8, 2009. Proceedings, 2009, pp. 619–628.
[42] Q. Lu, G. Korniss, B.K. Szymanski, The naming game in social networks: Community formation and consensus engineering, J. Econ. Interact. Coord. 4 (2) (2010) 221–235.
[43] D. Li, I. Leyva, J. Almendral, J.M.B. I. Sendiña Nadal, S. Havlin, S. Boccaletti, Synchronization interfaces and overlapping communities in complex networks, Phys. Rev. Lett. 101 (16) (2008) 168701.
[44] J. Xie, S. Kelley, B.K. Szymanski, Overlapping community detection in networks: The state-of-the-art and comparative study, ACM Comput. Surv. 45 (4) (2013) 2501657.
[45] S. Zhang, R.-S. Wang, X.-S. Zhang, Identification of overlapping community structure in complex networks using fuzzy c-means clustering, Physica A 374 (1) (2007) 483–490.
[46] M. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (3) (2006) 036104.
[47] T. Nepusz, A. Petróczi, L. Négyessy, F. Bazsó, Fuzzy communities and the concept of bridgeness in complex networks, Phys. Rev. E 77 (1) (2008) 016107.
[48] 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.
[49] E. Abbe, Community detection and stochastic block models: Recent developments, J. Mach. Learn. Res. 18 (2017) 177:1–177:86.
[50] A.F. McDaid, N.J. Hurley, T.B. Murphy, Overlapping stochastic community finding, in: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2014, Beijing, China, August 17-20, 2014, 2014, pp. 17–20.
[51] M. Ghorbani, H.R. Rabiee, A. Khodadadi, Bayesian overlapping community detection in dynamic networks, 2016, CoRR abs/1605.02288.
[52] T. Yang, Y. Chi, S. Zhu, Y. Gong, R. Jin, Detecting communities and their evolutions in dynamic social networks - a Bayesian approach, Mach. Learn. 82 (2) (2011) 157–189.
[53] D. Lee, H. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[54] M. Zarei, D. Izadi, K. Samani, Detecting overlapping community structure of networks based on vertex–vertex correlations, J. Stat. Mech. Theory Exp. 2009 (11) (2009) P11013.
[55] I. Psorakis, S. Roberts, M. Ebden, B. Sheldon, Overlapping community detection using Bayesian non-negative matrix factorization, Phys. Rev. E 83 (6) (2011) 066114.
[56] F. Wang, T. Li, X. Wang, S. Zhu, C.H.Q. Ding, Community discovery using nonnegative matrix factorization, Data Min. Knowl. Discov. 22 (3) (2011) 493–521.
[57] Y. Zhang, D. Yeung, Overlapping community detection via bounded nonnegative matrix tri-factorization, in: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012, 2012, pp. 606–614.
[58] J. Yang, J. Leskovec, Overlapping community detection at scale: a nonnegative matrix factorization approach, in: Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013, 2013, pp. 587–596.
[59] C. Pizzuti, Evolutionary computation for community detection in networks: A review, IEEE Trans. Evol. Comput. 22 (3) (2018) 464–483.
[60] Y. Feng, H. Chen, T. Li, C. Luo, A novel community detectionmethod based on whale optimization algorithm with evolutionary population, Appl. Intell. (2020) http://dx.doi.org/10.1007/s10489-020-01659-7, URL https://link.springer.com/article/10.1007/s10489-020-01659-7.
[61] X. Zhang, K. Zhou, H. Pan, L. Zhang, X. Zeng, Y. Jin, A network reduction-based multiobjective evolutionary algorithm for community detection in large-scale complex networks, IEEE Trans. Cybern. 50 (2) (2020) 703–716.
[62] L. Zhang, H. Pan, Y. Su, X. Zhang, Y. Niu, A mixed representation-based multiobjective evolutionary algorithm for overlapping community detection, IEEE Trans. Cybern. 47 (9) (2017) 2703–2716.
[63] Z. Li, J. Liu, K. Wu, A multiobjective evolutionary algorithm based on structural and attribute similarities for community detection in attributed networks, IEEE Trans. Cybern. 48 (7) (2018) 1963–1976.
[64] S. Gregory, An algorithm to find overlapping community structure in networks, in: 18th European Conference on Machine Learning (ECML 2007)/11th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2007, Vol. 4702, pp. 91–102.
[65] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 (12) (2002) 7821–7826.
[66] M. Hajiabadi, H. Zare, H. Bobarshad, IEDC: An integrated approach for overlapping and non-overlapping community detection, Knowl.-Based Syst. 123 (2017) 188–199.
[67] T. Chakraborty, S. Ghosh, N. Park, Ensemble-based overlapping community detection using disjoint community structures, Knowl.-Based Syst. 163 (2019) 241–251.
[68] W. Zhi-Xiao, L. Ze-Chao, D. Xiao-Fang, T. Jin-Hui, Overlapping community detection based on node location analysis, Knowl.-Based Syst. 105 (2016) 225–235.
[69] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New J. Phys. 11 (3) (2009) 033015.
[70] L. Freeman, D. Roeder, R. Mulholland, Centrality in social networks: ii. experimental results, Social Networks 2 (2) (1979) 119–141.
[71] M. Kitsak, L. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. Stanley, H. Makse, Identification of influential spreaders in complex networks, Nat. Phys. 6 (11) (2010) 888–893.
[72] S. Dorogovtsev, A. Goltsev, J. Mendes, K-core organization of complex networks, Phys. Rev. Lett. 96 (4) (2006) 040601.
[73] J. Bae, S. Kim, Identifying and ranking influential spreaders in complex networks by neighborhood coreness, Physica A 395 (2014) 549–559.
[74] L. Lü, T. Zhou, Q.-M. Zhang, H. Stanley, The H-index of a network node and its relation to degree and coreness, Nature Commun. 7 (2016) 10168.
[75] D. Rhouma, L.B. Romdhane, An efficient algorithm for community mining with overlap in social networks, Expert Syst. Appl. 41 (9) (2014) 4309–4321.
[76] X. Ding, J. Zhang, J. Yang, A robust two-stage algorithm for local community detection, Knowl.-Based Syst. 152 (2018) 188–199.
[77] J.J. Whang, D.F. Gleich, I.S. Dhillon, Overlapping community detection using seed set expansion, in: 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013, 2013, pp. 2099–2108.
[78] I.S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, IEEE Trans. Pattern Anal. Mach. Intell. 29 (11) (2007) 1944–1957.
[79] R. Andersen, F.R.K. Chung, K.J. Lang, Local graph partitioning using pagerank vectors, in: 47th Annual IEEE Symposium on Foundations of Computer Science, (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings, 2006, pp. 475–486.
[80] D.F. Gleich, C. Seshadhri, Vertex neighborhoods, low conductance cuts, and good seeds for local community methods, in: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012, 2012, pp. 597–605.

[81] 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.
[82] H. Yu, P. Jiao, Y. Yao, G. Wang, Detecting and refining overlapping regions in complex networks with three-way decisions, Inform. Sci. 373 (2016) 21–41.
[83] 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.
[84] Y.A. Wei, C. Cheng, Towards efficient hierarchical designs by ratio cut partitioning, in: IEEE International Conference on Computer-Aided Design, 1989, pp. 298–301.
[85] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[86] B. Bollobas, Modern Graph Theory, Springer Verlag, New York, USA, 1998.
[87] D. Watts, Small Worlds: The Dynamics of Networks Between Order and Randomness, Princeton University Press, Princeton, USA, 2003.
[88] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowl. Inf. Syst. 42 (1) (2015) 181–213.
[89] R. Kanawati, Empirical evaluation of applying ensemble methods to ego-centred community identification in complex networks, Neurocomputing 150 (2015) 417–427.
[90] K. Guo, L. He, Y. Chen, W. Guo, J. Zheng, A local community detection algorithm based on internal force between nodes, Appl. Intell. 50 (2) (2020) 328–340.
[91] A. Lancichinetti, F. Radicchi, J. Ramasco, S. Fortunato, Finding statistically significant communities in networks, PLoS One 6 (4) (2011) e18961.
[92] H. Shen, X. Cheng, K. Cai, M.-B. Hu, Detect overlapping and hierarchical community structure in networks, Physica A 388 (8) (2009) 1706–1712.
[93] 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.
[94] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[95] 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.
[96] V. Krebs, Social network of political books, 2004, www.visualcomplexity.com.
[97] B. Rozemberczki, C. Allen, R. Sarkar, Multi-scale attributed node embedding, 2019, arXiv:1909.13021.
[98] E. Ravasz, A. Somera, D. Mongru, Z. Oltvai, A.-L. Barabási, Hierarchical organization of modularity in metabolic networks, Science 297 (5586) (2002) 1551–1555.