PAPER • OPEN ACCESS

Self-falsifiable hierarchical detection of overlapping communities on social networks

To cite this article: Tianyi Li and Pan Zhang 2020 *New J. Phys.* **22** 033014

View the article online for updates and enhancements.

New Journal of Physics

The open access journal at the forefront of physics

CrossMark

PAPER

OPEN ACCESS

Self-falsifiable hierarchical detection of overlapping communities on social networks

Tianyi Li[1] and **Pan Zhang**[2]

1 System Dynamics Group, Sloan School of Management, Massachusetts Institute of Technology, United States of America
2 CAS Key Laboratory of Theoretical Physics, Institute of Theoretical Physics, Chinese Academy of Sciences, People's Republic of China

**E-mail:** tianyil@mit.edu

Abstract

No community detection algorithm can be optimal for all possible networks, thus it is important to identify whether the algorithm is suitable for a given network. We propose a multi-step algorithmic solution scheme for overlapping community detection based on an advanced label propagation process, which imitates the community formation process on social networks. Our algorithm is parameter-free and is able to reveal the hierarchical order of communities in the graph. The unique property of our solution scheme is *self-falsifiability*; an automatic quality check of the results is conducted after the detection, and the fitness of the algorithm for the specific network is reported. Extensive experiments show that our algorithm is self-consistent, reliable on networks of a wide range of size and different sorts, and is more robust than existing algorithms on both sparse and large-scale social networks. Results further suggest that our solution scheme may uncover features of networks' intrinsic community structures.

Community detection is a central topic in network science. Pioneered by works represented by Palla *et al* [1], in recent years more and more studies focus on the detection of overlapping communities as opposed to exhaustive communities, a division also regarded as soft-partitioning versus hard-partitioning. Besides relying on the optimization of certain metrics, e.g. modularity [2], conductance [3], fitness [4], or aggregative metric based on link prediction algorithms [5] etc, as the extensively-studied traditional approach for detecting exhaustive communities, for overlapping communities, a lot of new tools are designed based on various ideas, including link communities [6, 7], clique percolation [8], seed set expansion [3, 9–12], label propagation [13–16], local spectral clustering method [17], and methods based on statistical inference, such as Infomap [18], the stochastic block model (SBM, [19, 20]; in particular, methods adopting the belief propagation algorithm [21, 22]), and other generative models [23, 24].

Despite the success of different solution schemes on various application fronts, some weak points of existing community detection algorithms could be pinned down in practice, which we believe might be problematic in certain cases (see appendix A for a detailed discussion). These weaknesses include: (1) many solution schemes are over-parameterized, and in some cases the tuning of parameters depends largely on unwarranted heuristics; (2) many scalable methods based on the seed set expansion process [2–4, 9, 25–28] may lack well-designed seeding strategies [10, 11, 29] and often rely on ad-hoc strategies; (3) some algorithms that claim to be local, as opposed to methods based on an optimization over the entire graph, in fact still optimize on the community level and thus do not guarantee complete locality; (4) the number of communities in the graph is often pre-determined in certain algorithms, which might not be a good treatment, despite its claimed advantage [30] and the possible determination by the non-backtracking matrix [31]; (5) the overlapping communities revealed by some algorithms are in fact still exhaustive in their corresponding link communities [6], which should not be an implicit constraint imposed by algorithms; (6) in many cases, the revealed communities do not follow any order and instead are treated as of equal significance to the graph ('blended' [30]), which may deviate from realistic situations; (7) most algorithms assume that all nodes in the graph should belong to at least one community, without taking care of those isolated nodes that do not have any community membership [32–35]; (8) finally, a

Figure 1. Illustration of the integrated belief underlying the proposed detection scheme.
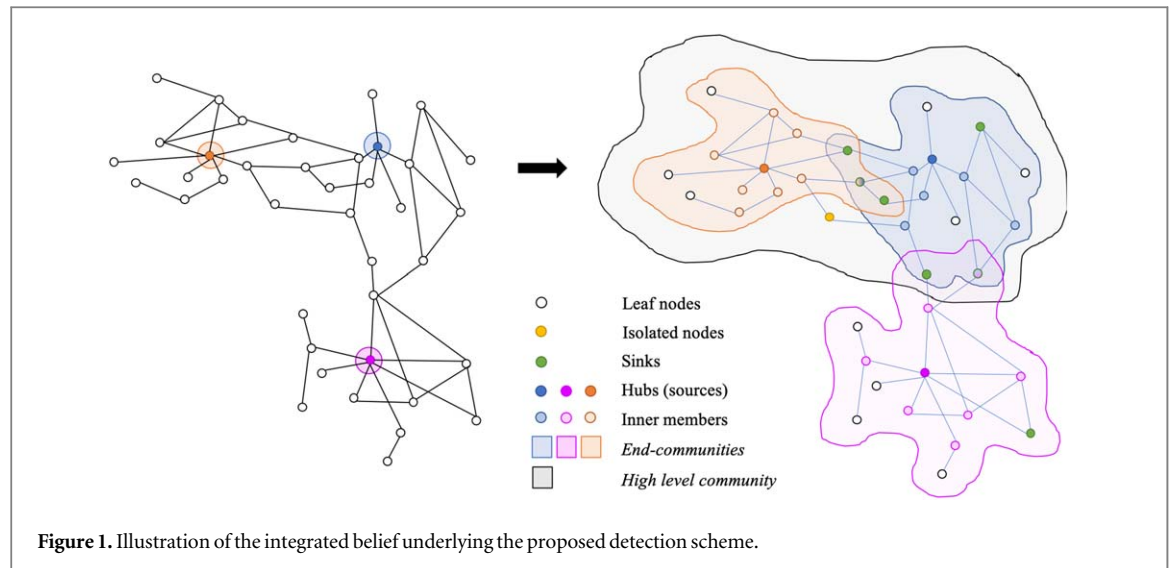
notification of the quality of detection results is not incorporated in most algorithms, failing to indicate the inevitable limited applicability of the method.

In a recent study, Peel *et al* [36] showed that, community detection is such an ill-defined problem that intrinsically no algorithm could be the optimal solution for all tasks, essentially a variant of the No-Free-Lunch theorem. Although this result seems to make the probe of community detection algorithms less meaningful, we argue that various streams of community detection ideas have embodied valuable beliefs for solving this problem and it is still useful to devise new approaches that inherit and combine the successful ideas of previous attempts. However, the most important lesson from [36] is that, as noted by point (8) in the above discussion, when implemented on an arbitrary graph, a reliable community detection scheme should be able to indicate the extent of its applicability on the specific network, i.e. the extent of imperfectness of its detection results. We believe that this property of *self-falsifiability* is an important missing piece in most existing algorithms.

Aiming at circumventing these weaknesses, correspondingly, we formulate our integrated belief for the overlapping community detection problem, with an emphasis on social networks where nodes represent human beings. This emphasis implies that the determination of community membership should incorporate behavioral features, rather than being a completely mechanical process. We conclude important insights from multiple streams of existing algorithms [30] and integrate them into our belief; some extra attention to propagation-based approaches is paid, which are missing in [30] yet play a key role to account for the dynamic nature of social networks. The integrated belief consists of six aspects and is sketched in figure 1.

(i) *Overlappedness*. One node could be able to belong to multiple communities, and its 'strength' in different communities, e.g. in terms of the degree of attachment, should not be assumed to be homogeneous over its multiple memberships. Meanwhile, the corresponding link communities derived from the nodes' overlapping community assignment, should not be assumed to be exhaustive. One link could belong to different communities, such that the overlapping of communities should allow two communities to share a finite part of their components (e.g. [37]), consisting of both nodes and edges, as opposed to the case in [6, 7].

(ii) *Different roles of nodes*. Depending on the roles in communities, nodes in a typical social network may fall into five categories: hubs (sources), inner members of communities, boundaries (sinks), leaf nodes, and isolated nodes. Communities are initiated by hubs, but are finalized by sink nodes who set the boundaries, which are nodes that belong to more than one communities. Edges are not natural boundaries of communities, as implied by (i). Isolated nodes belong to no communities; leaf nodes have only one neighbor and thus play a trivial role in the detection process.

(iii) *Behavioral locality*. In social networks, it is difficult for nodes to be acknowledged with information regarding the entire graph, even information regarding the other part of their communities. Therefore, in human networks the decision of (elementary, as apposed to aggregated) community membership should be *local*, following behavioral rules on nodes, instead of being derived from any optimization standpoint.

(iv) *Propagatory formulation of communities*. On social networks, communities emerge along the propagation of information and action, hence methods imitating the propagation process (e.g. the gradient flow [38]) have the advantage in revealing community structures. Many nodes could be the source (seeds) of the propagation, while some of them are dominated by others and only a few could be successfully identified as hubs. During the propagation, each node should be associated with a finite memory [39], recording the history

of infections it receives from multiple communities. The determination of the community membership as well as the strength of the membership emerge from the infection history.

(v) *Order of communities*. Communities on graphs should follow a hierarchical order [4, 26, 40–43]: iteratively, the aggregation of small communities gives rise to bigger communities, and the entire graph is the single ultimate community.

(vi) *Self-falsifiability*. The applicability of any community detection algorithm is limited [36]. When implemented on graphs with an arbitrary topology, detection algorithms should be able to quantitatively indicate the quality of the detection results, due to their varied applicability on specific graphs. In particular, a reliable detection algorithm is supposed to notify its potential failure on certain networks.

Based on our integrated belief, we proposed a multi-step [12, 35] algorithmic solution scheme for overlapping community detection. Our approach is in line with the DBSCAN algorithm [44–46], the SHRINK algorithm [34], and the idea of gradient flow in the hierarchical landscape of complex networks [38], but having a more transparent and better quantified workflow, along with two new features: *parameter-free*, and *self-falsifiable*. The framework consists of four steps. First, nodes are identified with different roles in the graph based on their centrality scores, among which nodes having local centrality peaks are detected as the hubs (sources) of end-communities. Next, a diffusive label propagation process is initiated, starting simultaneously from all hubs, and spread on the entire graph. The determination (expansion) of end-communities converges at the end of the propagation process. These two steps echo with the gradient flow and the identification of nodes' role in [38]. Third, the distance matrix of end-communities is calculated, which facilitates the construction of the community hierarchy by aggregating end-communities in an upward fashion along the distance matrix. The entire graph becomes the ultimate community on the top of the hierarchy. In the end, the quality of the obtained community hierarchy is automatically checked and quantitatively indicated after the detection, and suggestions for the cutoff levels of the community hierarchy are provided. Details of the four steps of the detection scheme are discussed in the next section. Performance of the algorithm is studied and discussed in the following sections.

Multi-step detection algorithm

Step 1: *Identification of nodes' roles*. Assume a graph with N nodes and E edges. Given the connectivity matrix $A = \{a_{ij}\}$ of the graph, first we calculate the centrality scores c_i of each node i, and find the set of nodes whose centrality score is local peak, i.e. whose centrality is *no less than* all its neighbors. In theory, different kinds of centrality measures could be used. Path-based centralities such as betweenness centrality or closeness centrality may not be suitable for our current setting; density-based measures such as degree centrality and eigenvector centrality are more appropriate to apply. Among these nodes that are local peaks, those whose centrality is *strictly greater* than at least one neighbor are identified as source nodes (hubs); in the rare case, nodes having the same centrality score as all its neighbors are considered as isolated nodes[3]. S is the set of hubs, each of which is the core of an *end-community*, and $|S|$ is the number of end-communities.

Correspondingly, find the set of nodes whose centrality score is local trough, i.e. whose centrality score is smaller than all its neighbors. Among these nodes, those that are not leaf nodes (only having one neighbor) are identified as sinks; each node in this category has at least two neighbors that have great centrality scores who could pass the end-community label to it (see *Step* 2). Since these nodes decide the stopping of the label propagation, some of them determine the boundary of end-communities, while other sinks lie strictly in the community. The remaining nodes in the graph are inner members of communities who are neither the local maximum nor the minimum of the score. Note that the above definitions should be distinguished with nodes's overlappedness: sinks are not always belong to more than one community, and only those sinks lying in the boundary of communities have multiple community membership (i.e. 'cross-overs'); in the similar sense, inner members could also belong to more than one communities.

Therefore, based on the centrality measure relative to its neighbors, each node is identified with one of the five roles: hubs (sources), sinks, inner members, isolated nodes and leaf nodes (figure 1). Each hub defines an *end-community*. Isolated nodes are very rare and sparsely distributed in the graph and we assume that they do not have community membership; they could always be allocated to neighboring communities if one seeks to eliminate this category.

---

[3] Except for the specific situation of 'centrality cliques'. As an extended concept from the common 'cliques', this corresponds to a situation where a group of fully connected nodes have the same centrality scores. In this case, under our definition all nodes would be identified as isolated nodes. However, although none of them is a strict peak in centrality scores, such a clique of nodes should essentially constitute an end-community. Therefore, in such situations one node from the centrality clique is identified as a hub, and a small value is added to its centrality score to make sure the propagation in the following step is successful (Step 2); this treatment makes no further impact. The rest of the clique are identified as inner members of this end-community.

Step 2: *Determination of end-communities.* Assign a different community label s on each hub, and initiate a diffusive label propagation process *simultaneously* starting from all hubs in S. The membership of a specific end-community s is represented by a tuple $x_s = \{(i, t)\}$, which records that node i joins the community at time t. Correspondingly, every node i is associated with an infection history (memory) tuple $h_i = \{(s, t)\}$ that records the label s it receives at time t. The two tuples $X = \{x_s\}$ and $H = \{h_i\}$ are updated in the propagation process. Note that the synchronization of label propagation is guaranteed in our algorithm by using t to record the timestamps of the label infection, instead of only recording the incident source of infection, as in [39].

To the first order, we assume that nodes only infect their immediate neighbors. The propagation rule is: at time t, starting from node i with *current* community label s, for a different node j, if $a_{ij} = 1$ and $c_i > c_j$, add (s, t) to h_j and (j, t) to x_s, when $(s, \forall t) \notin h_j$ (same as $(j, \forall t) \notin x_s$). In other words, there will be a successful infection of the community label, if and only if the incident node's centrality score is greater than the target node, an immediate neighbor of it, and the infection will be recorded when this is the first time the target node received this community label. The label propagation will not take place between two nodes having the same centrality score, which is consistent with our definition of isolated nodes (they are insulated from any infection).

At each time step, the label propagation will spread to all neighbors of the newly infected nodes (except the infector of the previous step, since its centrality score is higher). The propagation of a certain label will stop at those directions where the neighbors to be infected have a higher centrality score, or the neighbors are already infected by that label (hence the infection history H is non-repetitive). In the most extreme case where the graph has a strict tree structure of centrality scores, the label propagation will take at most q time steps, where q is the longest path length of the graph, and the length of the infection memory h is at most $N - 1$. In practice, the propagation could stop after only a few time steps, when all the community memberships $X = \{x_s\}$ (or equivalently, the infection history of all nodes $H = \{h_i\}$) do not change, i.e. no new community label is assigned to any node. The propagation process is sketched in algorithm 1.

H records the information of nodes' overlapping community membership. For node i, if $|H_i| = 1$, it only belongs to one community; if $|H_i| > 1$, it belongs to more than one community and is thus a 'cross-over'. From X and H, we could qualify the overlappedness of communities in the graph by two metrics: (1) the average community membership of all nodes (average length of the non-repetitive infection history) m_h, and (2) the average size of end-communities m_x. The two metrics are related by:

$$m_h = \frac{|S|}{N} m_x. \tag{1}$$

Notably, one advantage of recording the infection history H is that, for nodes belonging to multiple communities, their strength to different communities could possibly be indicated by the infection time t: a small t in (s, t) means that the node is near to the hub s, thus having a large strength to this community, and vice versa.

Step 3: *Aggregation of small communities.* Communities join each other and form bigger ones when they are close enough. We assume that the distance between end-communities is represented by the distance of their hubs, and thus formulate the distance matrix R_0 over end-communities, where each entry r_{pq}^0 is the shortest path between two hubs p and q:

$$R_0 = \{r_{pq}^0\} = \{d_{\text{shortest path}}(p, q)\}, \quad p, q \in S. \tag{2}$$

If no path exists between p and q (in the case of sub-components; the graph is not fully connected), we set $r_{pq}^0 = d_{\text{inf}}$ (in practice, d_{inf} is an extremely large number). The distance matrix R_0 is $|S| \times |S|$, whose diagonal elements are all 0 and off-diagonal elements are positive integers. Now we aggregate end-communities by iteratively rearrange the distance matrix R_0. Find two hubs (two rows in the matrix) with distance $\epsilon = 1$ and replace them with a single merged node in the matrix (not on the graph); for any unaddressed node, take the maximum of the two original distances in the matrix R_0 as the new distance between the node and the aggregated new node. Repeat until all $\epsilon = 1$ elements in the R_0 matrix are detected and replaced. This procedure iteratively reduces the size of R_0 and update the matrix; in the end, larger communities (ϵ_1-communities) are formed out of end-communities and we obtain the new distance matrix R_1. Using the same approach, we then formulate ϵ_2-communities and R_2, up to the final $\epsilon_{d^{\max}}$-community and $R_{d^{\max}} = [0, d^{\max}; d^{\max}, 0]$, where $d^{\max}$ is the largest shortest path distance between the hubs in the graph, i.e. the largest element in R_0. For conveniences, we write $\epsilon_{d^{\max}}$ as $\epsilon_{\max}$ and $R_{d^{\max}}$ as $R_{\max}$. In the end, a hierarchy of communities is obtained through this upward iterative aggregation of small communities, whose distances are represented by a series of matrices R_0, $R_1, \ldots R_{\max}$ of gradually reduced sizes. This step is summarized in algorithm 2 and illustrated in figure 3.

Note that our algorithm naturally takes care of input graphs that are not fully connected through d_{inf}: during the iterative reduction of R, once we find that at a certain stage, all off-diagonal elements of R equal d_{inf}, it suggests that the remaining communities are the sub-components and could not be further combined, and thus the aggregation process stops, with $d_{\max}$ indicating the largest diameter of the subcomponents of the unconnected graph.

For integer ϵ ranging from ϵ_0 to $\epsilon_{\max}$, a different number of communities (the size of $|R_\epsilon|$) remain at each value of ϵ. The $\epsilon \leftrightarrow |R_\epsilon|$ relationship demonstrates the nature of the community hierarchy; turning points of the slope of the $\epsilon \leftrightarrow |R_\epsilon|$ curve could be used in practice to suggest the cutoff level of the obtained community hierarchy (see Results and discussion).

Step 4: *Quality check of the hierarchy*. Calculate the Jaccard index matrix J between each pair of end-communities whose hubs are p, q: $J = \{j_{pq}\}$, which is an index characterizing the similarity of two groups of nodes. The automatic quality check of the detection results is carried out relying on the Jaccard matrix J. In the previous step, we aggregate small communities based on the distance of their source nodes; conceptually, one may propose an alternative aggregation rule: iteratively aggregate small communities that have the most overlap, indicated by the Jaccard index. However, one problem arises with this plausible treatment: it may almost always lead to a strictly *binary* hierarchy that at each step, only one big community will be formulated out of exactly two small communities, since the Jaccard index is a real number. By contrast, our (integer-valued) distance-based aggregation rule allows that at each stage a few mergings take place. In other words, an overlap-based merging rule will almost always result in a hierarchy with $|S| - 1$ stages and therefore formulate a dendrogram, which is a binary-tree structure [40], whereas our distance-based merging rule is more flexible and may be able to yield a much tighter hierarchy (k-ary tree).

Algorithm 1. Propagatory formulation of end-communities (Step 2).

1: Initialize X, H.
2: Input adjacency matrix $A = \{a_{ij}\}$ and calculate nodes' centrality scores $C = \{c_i\}$.
3: Identify the set of hubs for end-communities S (Step 1).
4: $t = 0$
5: **while** $t < t_{\max}$ **do**
6:      $t = t + 1$
7:      **for** $\forall$ community $s \in S$ **do**
8:              **for** $\forall$ node i s.t. $(i, t-1) \in x_s$ **do**
9:                      **for** $\forall$ node j s.t. $a_{ij} = 1$ and $c_i > c_j$ **do**
10:                             **if** j not in community s **then**
11:                                     Add (j, t) to x_s
12:                                     Add (s, t) to h_j
13:                             **end if**
14:                     **end for**
15:             **end for**
16:     **end for**
17:     **if** H (or X) not change **then**
18:             **break**
19:     **end if**
20: **end while**
21: X, H, t_{fin} obtained. Calculate m_h, m_x.

Nevertheless, we could utilize the Jaccard index to check the quality of our distance-based mergings. Under our distance-based rule, at each merging, i.e. combing two communities p, q into one $p + q$, the Jaccard index j_{pq} is not necessarily the largest element in the matrix J (i.e. p and q do not necessarily have the most overlap among the pairing of all communities); however, to be considered a *good* merging, one idea is that it should be satisfied that the Jaccard index between the two communities p, q to be merged, must be larger than the index between one community out of p, q and any other community at the current stage that is not going to be further merged with p and q (i.e. whose two distances to p, q are not *both* the same as r_{pq}). This means that, the merging of p and q will be considered *good* (i.e. consistent with overlap-based heuristics), if and only if all the other communities that have more overlap with p *or* with q are going to be further merged with p and q at this ϵ stage, or equivalently, no community that has more overlap is not to be merged. We call this condition as J–D *consistency*, which is formally stated as:

$$\text{J–D consistency: } j^\epsilon_{pq} > j^\epsilon_{pz} \text{ and } j^\epsilon_{pq} > j^\epsilon_{qz}, \quad \forall z \in HR_\epsilon \;\; s.t. \;\; r^\epsilon_{pz} > r^\epsilon_{pq} \text{ or } r^\epsilon_{qz} > r^\epsilon_{pq} \text{ (for a certain } \epsilon), \quad (3)$$

where HR_ϵ denotes the set of communities obtained at a certain ϵ level of the hierarchy (i.e. in the beginning, $HR_0 = S$; in the end, $|HR_{\epsilon_{\max}}| = 1$ for connected graphs). If the above J–D consistency is satisfied, the merging at this step is considered as a good merging. Hence, by this means, we are able to indicate the quality of the obtained community hierarchy (thus the quality of our detection workflow) by a J–D consistency factor Φ, which is the number of good mergings (condition (3) satisfied) normalized by the total number of mergings $|S| - 1$. Since the merging events in the last round (round $\epsilon_{\max}$) are always J–D consistent, they are subtracted from both the

numerator and the denominator of the ratio, and we have:

$$\Phi = \frac{\sum_{\epsilon=1}^{\epsilon_{\max}-1} \text{card}(\text{J–D consistent})_\epsilon}{|HR_0| - |HR_{\epsilon_{\max}-1}|}. \tag{4}$$

Note that Φ is applicable only when $|HR_0| \neq |HR_{\epsilon_{\max}-1}|$, i.e. there are at least 3 level of communities in the hierarchy. During the merging process, the Jaccard index matrix J is recalculated at each stage, and the dimension reduction of J is in accordance with the dimension reduction of R (algorithm 2). In practice, each merging event is associated with a Boolean variable indicating its J–D consistency, and therefore at each ϵ, we are able to calculate the Φ_ϵ at that level, which is a component of the final Φ. The curve $\epsilon \leftrightarrow \Phi_\epsilon$ also helps the determination of the cutoff level of the community hierarchy (see Results).

The metric Φ provides the algorithm with the desired property of self-falsifiability. If $\Phi = 1$, our distance-based merging rule is perfectly consistent with the overlap-based rule. A large Φ indicates that the establishment of the community hierarchy obtained from the detection workflow embodies a large proportion of good mergings, in the sense that two communities whose hubs are shortest-distanced are also having the largest overlap, so that the aggregation of them is double credited. By contrast, a small Φ implies that the aggregation process mostly consists of bad mergings, in which the two merging heuristics often do not coincide; therefore, our detection scheme may not be suitable for the specific graph.

It is discrete to argue that, however, the utility of metric Φ should not be overstated. It is not universally applicable, as at least 3 level of communities in the hierarchy (with the last level being the whole graph) should exist for the metric to be useful; this condition asks that the distances between all hubs should have at least two values. Moreover, the metric is more useful in signaling good detection results than denoting bad ones: while a high Φ could largely suggest great detection results according to J–D consistency, a small Φ does not necessarily indicate a bad detection, as J–D consistency is only characterizing a single aspect of the results.

Computational complexity

Consider the graph with N nodes and E edges. The time complexity for calculating the centrality measures is $O(N)$ for degree centrality and $O(N \log(N))$ for eigenvector centrality. After the centrality scores of all nodes have been obtained, at Step 1, the identification of nodes' roles is realized by comparing each node's centrality score to all its neighbors; this procedure incurs a time cost $O(E)$. At Step 2 (algorithm 1), during the last iteration, every node in every community is visited, with each visit accessing all the node's neighbors. This corresponds to $2E$ visits at this (last) iteration, and therefore the entire time complexity of Step 2 is $O(t_{\text{fin}}E) = O(E)$. Intuitively, the propagation is always along the descent of centrality scores, so it is one-way; since labels propagate simultaneously from all hubs, one could thus identify a topological order of all nodes (with time $O(N+E)$) and define accordingly a directed acyclic graph (DAG). Along this DAG, the propagation could be performed in a synchronized one-pass following the topological order, and each node is visited exactly once. Overall, the time cost is $O(E)$. Steps 3 and 4 are carried out at the same time in algorithm 2. The time complexity of algorithm 2 depends on the dimension of the matrix R_0, which is determined by the number of end-communities (i.e. number of hubs) $|S|$. The $|S| \times |S|$ matrix R_0 gradually degenerates into the 2×2 matrix $R_{\max}$, with the minimum element in the R matrix detected at each stage; therefore the time is upper bounded by $O(|S|^2)$, as is also the time complexity for calculating the shortest distance between the hubs of end-communities in the formulation of R_0.

After all, the computational complexity of the entire algorithm t_{algo} is (using degree centrality at Step 1; assuming $|E| > N$):

$$t_{\text{algo}} = O(N) + O(E) + O(t_{\text{fin}}E) + O(|S|^2) = O(t_{\text{fin}}E) + O(|S|^2). \tag{5}$$

In practice (see Results), t_{fin} is always very small, and one could often set up a small $t_{\max}$ to let the detection finish early by cutting off nodes' membership to remote communities; this treatment will not influence the final detection results in most cases. $|S|$ is also very small, normally a tiny fraction of N, and $|S|^2$ is unlikely to exceed N. Therefore, the computational complexity of our algorithm is effectively $O(E)$ when using degree centrality as the measure (figure B1), which is fast on real networks where node connections are often not dense.

Computational superiority

A few advantages of our algorithm could be highlighted in computation. First, by recording the timestamps of the infections, the label propagation process in our framework is synchronized. This advantage prevents the numerical error incurred by unsynchronized algorithms (e.g. the original label propagation [15]). Next, Steps 2 and 3 of the algorithm could run in parallel after Step 1, although Step 4 and the determination of the cutoff level of communities still need to be carried out after Steps 2 and 3 are finished. Last, as demonstrated, the computational time of our algorithm is linear with the number of edges, which is a desirable feature for its application on massive real-world social networks. This is achieved by (1) the one-way propagation (hubs to

surroundings) of labels with stops at centrality sinks (Step 2), which is notably faster than existing label propagation algorithms without a one-way formulation (quasi-linear with the number of edges), and (2) the iterative dimension reduction of the distance matrix of communities, which in practice often takes only a few steps to degenerate into the final matrix.

Results

We applied our community detection scheme to networks of a wide range of size and various sorts (table 1). Degree centrality is used as the centrality measure in the detection; tests show that using eigenvector centrality often fails to identify enough hubs for end-communities, and the computational cost for calculating eigenvector centrality for large scale networks is often prohibitive. In our experiments, the propagation process (Step 2) converges after a few number of iterations ($t_{fin} < 22$) on all tested networks. Besides the proportion of nodes of different roles, we calculate the proportion of nodes that belong to multiple end-communities, i.e. the 'cross-overs'. Notably, these cross-overs are only with respect to end-communities; expectedly, with respect to aggregated communities up in the hierarchy, the number of cross-overs would be smaller, as some of them might no longer have multiple memberships once small communities are aggregated. We used small real networks (Karate club network [47], Dolphin network [48]) and synthetic networks (LFR benchmark network [49] and Erdös Rényi (ER) random network [50]) to demonstrate the detailed procedures of our detection scheme and show the important $\epsilon \leftrightarrow \Delta|R_\epsilon|$ and $\epsilon \leftrightarrow \Phi_\epsilon$ relationships constructed along the formulation of the community hierarchy (figures 2–4). We then apply the algorithm to a panel of large real networks to carry out horizontal discriminative analysis. A number of notable features emerged from the detection results, which demonstrated the self-consistency and robustness of our algorithms; meanwhile, a few unexpected interesting phenomena regarding the intrinsic structure of networks are uncovered (figures 5–6).

Algorithm 2. Determination of the community hierarchy (Steps 3 and 4).

1: Calculate R_0, J_0. Obtain ϵ_{max} from R_0.
2: $\epsilon = 0$, $\Phi = 0$, $HR_0 = S$.
3: **while** $\epsilon < \epsilon_{max}$ **do**
4:     $\epsilon = \epsilon + 1$
5:     **while** true **do**
6:         Find the ϵ-element of R, whose position is (p, q)
7:         **if** no ϵ-element found **then**
8:             **break**
9:         **end if**
10:        Update the community hierarchy HR:
11:            Remove the communities indexed by p and q from $HR_{\epsilon-1}$.
12:            Merge the two communities into a larger community $p + q$ and add to the hierarchy.
13:        Update the distance matrix R:
14:            Remove the columns and rows $\{p, q\}$ from $R_{\epsilon-1}$.
15:            Add a new row and a new column denoting the merged community $p + q$.
16:            $d(r, p + q) = \max(d(r, p), d(r, q))$
17:        **if** J–D consistency satisfied **then**
18:            $\Phi = \Phi + 1$
19:            Record the J–D consistent merging event.
20:        **end if**
21:        Update the Jaccard matrix J.
22:     **end while**
23:     $R_{\epsilon-1} \rightarrow R_\epsilon$, $HR_{\epsilon-1} \rightarrow HR_\epsilon$, $J_{\epsilon-1} \rightarrow J_\epsilon$
24:     Calculate Φ_ϵ.
25:     **if** all off-diagonal elements of R_ϵ equal d_{inf} **then**
26:         **break**
27:     **end if**
28: **end while**

Karate club network. The two centers (node #0, Mr. Hi; node #33, the officer) in the network are successfully identified as the only two hubs (blue nodes; figure 2, left top), around which two end-communities (obviously, the only non-trivial communities in the 2-level hierarchy) are determined (figure 2, left bottom). Although our algorithm detects overlapping communities while the ground truth communities of the Karate club network are disjoint, the detection results recover the ground truth to a great extent (figure 2, right). First,

Table 1. Summary of detection results. 'ISO' stands for 'isolates'; 'CO' stands for 'cross-overs'. Networks with star marks are not fully connected. On each network, self-edges and nodes with degree 0 are removed, a trivial modification to the original graph in all cases.

| Network	#Node	#Edge	#Hub	#Sink	#ISO	#Leaf	#Inner	#CO	t_{fin}	m_h	m_x	ϵ_{max}	Φ
Karate Club	34	78	2	16	0	1	15	17	5	1.5	25.5	2	...
Dolphin	62	159	5	12	0	9	36	38	5	2.08	25.8	4	1
LFR (3, 1.2, 0.1)	1000	2153	79	324	1	5	591	637	8	2.05	25.9	10	0.25
Facebook users	4039	88 234	5	621	0	75	3338	3081	13	2.00	1617.2	6	0.333
Enron email*	36 692	183 831	483	8640	530	11 211	15 828	32 173	16	3.65	277.2	11	0.431
Brightkite*	58 228	214 078	682	12 259	49	21 157	24 081	55 610	12	2.91	248.5	16	0.262
CA-GrQc*	5241	14 484	298	851	185	1197	2710	3470	13	3.74	65.8	13	0.308
CA-HepTh*	9875	25 973	341	2123	184	2109	5118	8197	14	8.00	231.6	15	0.232
CA-HepPh*	12 006	118 489	172	2605	170	1493	7566	5383	17	1.59	110.9	11	0.473
CA-AstroPh*	18 771	198 050	185	3909	281	1282	13 114	18 054	8	1.94	196.8	11	0.4*
CA-CondMat*	23 133	93 439	442	4717	447	2373	15 154	21 134	15	9.38	490.9	13	0.286
Deezer-RO	41 773	125 826	1051	9221	5	5430	26 066	38 621	18	31.10	1236.3	17	0.119
Deezer-HU	47 538	222 887	450	10 494	0	2701	33 893	46 506	20	112.31	11 864.7	12	0.054
Deezer-HR	54 573	498 202	64	11 035	1	2330	41 143	54 455	19	40.43	34 473.1	10	0.145
FB-artist	50 515	819 090	30	14 570	0	3124	32 791	50 456	11	3.96	6673.7	10	0.214
FB-new sites	27 917	205 964	179	7762	0	2137	17 839	27 351	18	14.33	2234.4	12	0.250
FB-company	14 113	52 126	341	3602	3	2358	7809	12 758	18	25.38	1050.4	13	0.200
FB-athletes	13 866	86 811	43	4715	0	1240	7868	13 623	19	16.72	5391.7	8	0.105
FB-government	7057	89 429	15	1894	0	355	4793	7021	15	4.15	1951.8	9	0.385
FB-politician	5908	41 706	60	1845	0	600	3403	5534	14	8.03	790.3	12	0.155
FB-public figure	11 565	67 038	129	3239	0	1912	6285	11 101	16	6.83	612.7	13	0.268
FB-tv show	3892	17 239	153	997	0	611	2131	3036	11	3.62	92.2	17	0.291
Gowalla	196 591	950 327	1266	49 295	9	49 452	96 569	186 616	22	3.97	616.5	14	0.156
Amazon	334 863	925 872	17 837	120 277	71	25 709	170 969	180 783	14	2.30	43.2	16	0.193
DBLP	317 080	1 049 866	2965	68 403	1	43 181	202 530	294 878	25	56.1	5999.2	19	0.156
ER ($p = 0.1$)	50	124	5	12	0	1	32	29	5	1.94	19.4	3	0.333
ER ($p = 0.01$)	500	1241	47	109	0	17	327	382	7	4.03	42.9	6	0.089
ER ($p = 0.001$)	4956	12 301	487	1159	0	190	3120	3851	9	4.78	48.7	8	0.050
ER ($p = 0.0001$)	49 661	124 959	4638	11 643	1	1732	31 647	38 737	10	4.87	52.1	11	0.044

Figure 2. Detection results on the Karate club network [47]. Left top: nodes' different roles. Blue, red, green nodes are hubs, inner members and sinks, respectively. Left bottom: two end-communities detected. Right: detection results compared with the ground truth.

Figure 3. Detection results on the Dolphin network [48]. Leftmost: 5 end-communities. Rightmost: iterative reduction of the distance matrix R and aggregation of small communities. Middle top: nodes' different roles and the two ϵ_2-communities. Blue, red, green nodes are hubs, inner members and sinks, respectively. Middle bottom: the obtained community hierarchy and the $\epsilon \leftrightarrow |R_\epsilon|$ relationship. Tick marks indicate the satisfaction of J–D consistency of merging events; all four mergings are J–D consistent.

the two overlapping communities of our results (second and fourth column) strictly contain the ground truth (first and third column). Second, as in our detection process each community assignment is associated with a timestamp, one may decide that the earlier the node joins the community, the larger its strength to this community is. Therefore, by abandoning the nodes that have lower strength to the communities (i.e. nodes having large values in the timestamp), it is possible to further compare the (truncated) overlapping communities
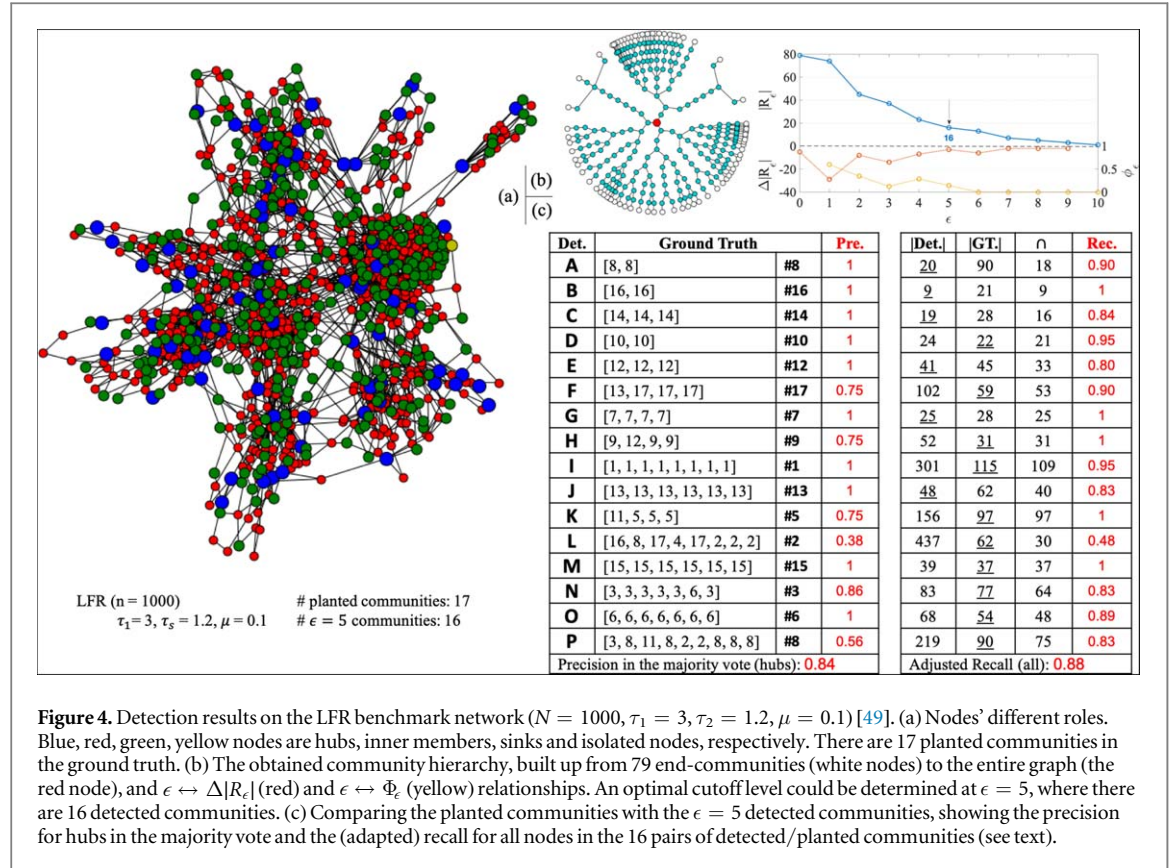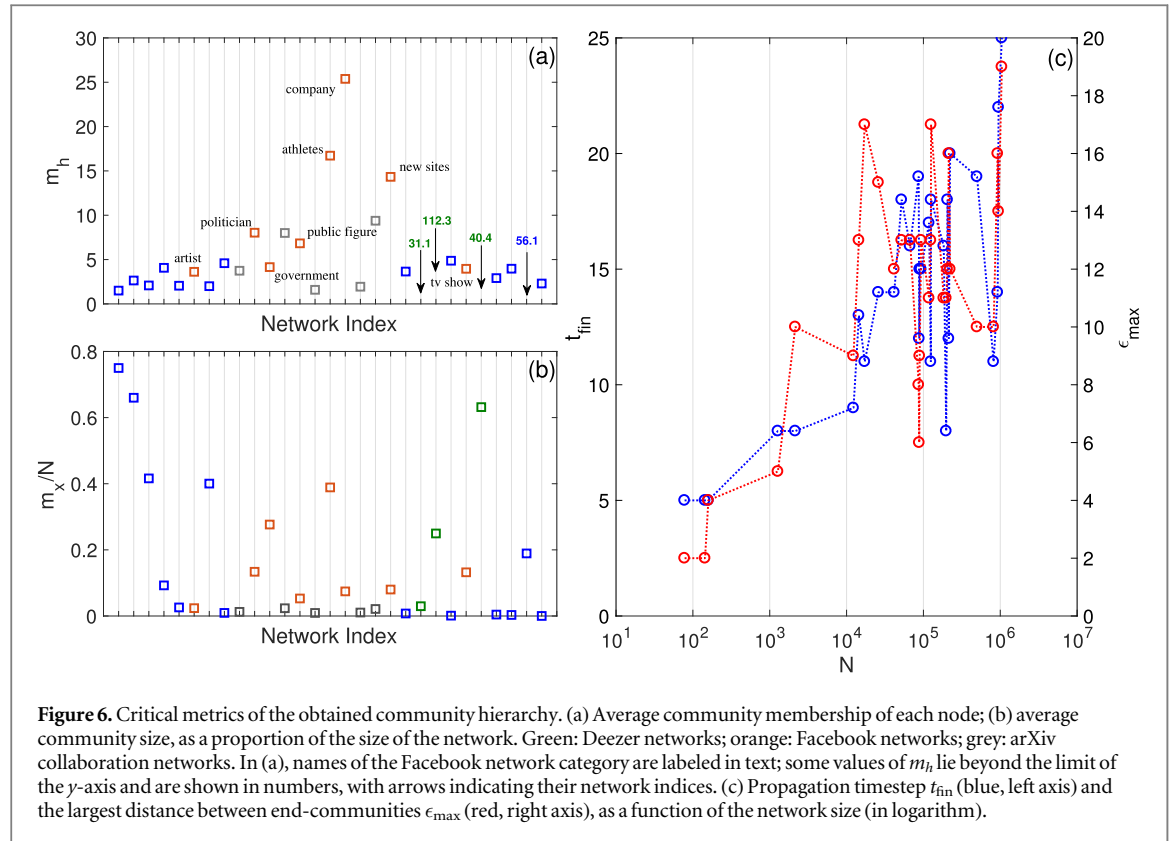
Figure 4. Detection results on the LFR benchmark network ($N = 1000$, $\tau_1 = 3$, $\tau_2 = 1.2$, $\mu = 0.1$) [49]. (a) Nodes' different roles. Blue, red, green, yellow nodes are hubs, inner members, sinks and isolated nodes, respectively. There are 17 planted communities in the ground truth. (b) The obtained community hierarchy, built up from 79 end-communities (white nodes) to the entire graph (the red node), and $\epsilon \leftrightarrow \Delta|R_\epsilon|$ (red) and $\epsilon \leftrightarrow \Phi_\epsilon$ (yellow) relationships. An optimal cutoff level could be determined at $\epsilon = 5$, where there are 16 detected communities. (c) Comparing the planted communities with the $\epsilon = 5$ detected communities, showing the precision for hubs in the majority vote and the (adapted) recall for all nodes in the 16 pairs of detected/planted communities (see text).
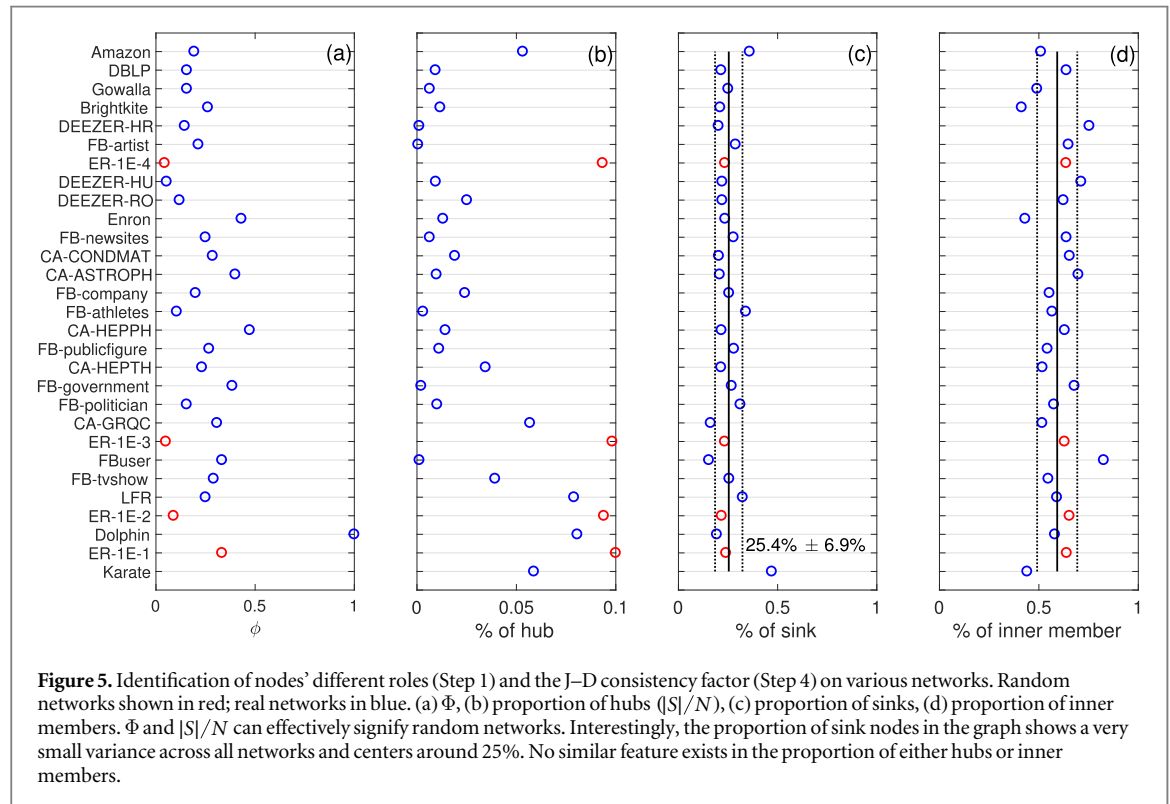
Figure 5. Identification of nodes' different roles (Step 1) and the J–D consistency factor (Step 4) on various networks. Random networks shown in red; real networks in blue. (a) Φ, (b) proportion of hubs ($|S|/N$), (c) proportion of sinks, (d) proportion of inner members. Φ and $|S|/N$ can effectively signify random networks. Interestingly, the proportion of sink nodes in the graph shows a very small variance across all networks and centers around 25%. No similar feature exists in the proportion of either hubs or inner members.

with the ground truth. Specifically, when only considering the nodes that join the community before time $t = 2$, our detected communities deviate from the ground truth by a small margin (entries in red; figure 2, right). When compared with the community results on this Karate club network in [38], whose ideas have common grounds with the current study, our results demonstrated a better recovery of the ground truth. Also note that the need to

Figure 6. Critical metrics of the obtained community hierarchy. (a) Average community membership of each node; (b) average community size, as a proportion of the size of the network. Green: Deezer networks; orange: Facebook networks; grey: arXiv collaboration networks. In (a), names of the Facebook network category are labeled in text; some values of m_h lie beyond the limit of the y-axis and are shown in numbers, with arrows indicating their network indices. (c) Propagation timestep t_{fin} (blue, left axis) and the largest distance between end-communities ϵ_{max} (red, right axis), as a function of the network size (in logarithm).

add random perturbations to the energy function (i.e. the centrality score in our case) for the gradient flow in [38] is dismissed in our study.

Dolphin network. The Dolphin network contains 62 nodes, among which 5 are identified as hubs (blue nodes; figure 3, middle top) for the corresponding 5 end-communities (figure 3, leftmost panel). The iterative reduction of the 5×5 distance matrix R_0 and the sequential aggregation of small communities are demonstrated in detail (figure 3, rightmost panel). Red marks show the communities that are aggregated at each stage of ϵ. The community hierarchy is obtained at the end of this iterative process (figure 3, middle bottom). In each of the four merging event, the J–D consistency condition is satisfied (right marks; figure 3, middle bottom), and thus the consistency factor $\Phi = 1$. From the $\epsilon \leftrightarrow \Delta|R_\epsilon|$ relationship, one can see that a proper cutoff level for communities is $\epsilon = 2$, and the corresponding two ϵ_2-communities are shown. Such a cutoff is chosen because the community membership does not change at the following $\epsilon = 3$ level, indicating that $\epsilon = 2$ may be a characteristic distance between communities. This example shows that local peaks on the $\epsilon \leftrightarrow \Delta|R_\epsilon|$ curve could be considered as the cutoff level on the final community hierarchy.

LFR benchmark network. We applied our detection algorithm to a LFR benchmark network of 1000 nodes ($\tau_1 = 3, \tau_2 = 1.2, \mu = 0.1$) with 17 planted communities. Our algorithm identified 79 end-communities (blue nodes; figure 4(a)) during a propagation process of 8 time steps. The formulated complete community hierarchy demonstrates the gradual build-up of large communities from smaller ones (figure 4(b)). The $\epsilon \leftrightarrow \Delta|R_\epsilon|$ and $\epsilon \leftrightarrow \Phi_\epsilon$ relationships are obtained along the formulation of the hierarchy. It shows that from $\epsilon = 4$ to $\epsilon = 5$, there is a trough in the change of the community hierarchy (red curve), and the J–D consistency factor arrives at a local peak around $\epsilon = 4$ and $\epsilon = 5$ (yellow curve). They suggest that $\epsilon = 5$ is a good cutoff level, at which stage there are 16 communities detected, indicating a good recovery of the synthetic ground truth (17 communities). Then we compared the ground-true LFR communities ('GT.') with the detected communities at the $\epsilon = 5$ level ('Det.'). We conducted the comparison in two ways (figure 4(c)). First, for each aggregated $\epsilon = 5$ community with a few hubs, we identify the planted LFR community label of each hub, and regard the majority vote of these labels as the ground-true community label (with the # marker, figure 4(c)) of this $\epsilon = 5$ community; for the 16 detected $\epsilon = 5$ communities, we thus discover 16 corresponding planted communities out of the 17 ground truth labels. It is shown that in most cases different hubs from a certain $\epsilon = 5$ community belong to the same ground-true LFR community, and the precision for all 79 hubs with respect to the majority vote is as high as 0.84. Second, we compare the planted LFR communities with the complete aggregated communities at $\epsilon = 5$. For each pair of the detected $\epsilon = 5$ community and the corresponding ground-true community, we calculate the recall factor, dividing the intersection of the two sets ($\cap$, figure 4(c)) by the minimum size of the two sets (underlined, figure 4(c)); in the end, the overall recall factor for all 16 pairs of communities is 0.88, suggesting a

very good overlap between the detected and the ground-truth structures. Although comparing two sets of communities of different numbers (17 versus 16) and different conditions of overlappedness, nevertheless, this detailed comparison and the large values of the two factors indicate that our algorithm is reliable in revealing the LFR planted structures. During the experiments, a number of different LFR networks were synthesized and tested, including some with overlapping communities, although the overlappedness of these generated LFR communities are very trivial (e.g. ~10 nodes out of 500 nodes are overlapping). Various tests show that the option of overlapping or not makes very little difference to the detection results, and our algorithm's performance is in general invariant at both cases. During extensive experiments on LFR networks, our algorithm yielded similar performances to the one demonstrated in this example; notably, the two cutoff criteria always agree with each other and the suggested number of communities is always close to the ground truth.

ER random network. We tested our detection algorithm on ER networks; a reasonable community detection method should be able to discover that these network do not contain significant community structures. Multiple ER networks are synthesized, with (n, p) selected such that the number of nodes and edges of the synthetic networks are close to the magnitude of the real networks in use, in order to make fair comparisons (table 1). Results show that, as desired, our detection scheme clearly separates random networks from real networks, which presumably have certain community structures embedded (figure 5). First, repeated tests show that the proportion of hubs identified among all nodes (i.e. $|S|/N$) is always significantly smaller for real networks than random networks of similar sizes (figure 5(b)), as one would expect, since random networks have a relatively flat structure and thus many nodes would be identified as 'plain hubs'. No similar distinction emerges in the proportion of sinks and inner members, where random networks and real networks are indistinguishable from each other (figures 5(c), (d)). Second, for random networks the J–D consistency condition is poorly matched; Φ is clearly small compared with real networks of similar size (figure 5(a)). This suggests that, unsurprisingly, on random networks, not only is the identification of end-communities (hubs) unwarranted, but also the merging of these end-communities not self-consistent. One might also be able to spot random networks during the propagation process in Step 2; real networks typically show an S-shape in the cumulative iteration time plot, while random networks have a flatter running time growth (figure B1). The two quantities $|S|/N$ and Φ could thus be used as the self-falsifiability benchmarks for detection results: for an arbitrary network, issue a random network of similar size and carry out detection on the two networks; if either $|S|/N$ or Φ in the detection result of the original network falls below the value of that on the issued random network, one should realize that the detection is not valid and the algorithm should be considered as not suitable for this specific graph.

Large real networks. We also tested our algorithm on a number of large real networks across a wide range of magnitude, including the DBLP network and Amazon product network [51], the Enron email network [52], the Facebook user network [53], the five Arxiv collaboration networks [54], the recent data from two digital platforms (Deezer, 3 networks; Facebook, 8 networks) [55], and the Gowalla network and the Brightkite network, both location-based social networks [56]. Detection results are summarized in table 1, and a few critical metrics are visualized in figure 5 and 6. The cutoff level of the community hierarchy for large real networks could be determined from the $\epsilon \leftrightarrow \Delta |R_\epsilon|$ and $\epsilon \leftrightarrow \Phi_\epsilon$ relationships, in the same way as for small networks (figure B2). Yet It is difficult to make further discussion on the hierarchy cutoffs based on the current information; hence we focus on the horizontal discriminative analysis of the detection results on various networks.

One very interesting result is that the proportion of sinks, identified in Step 1, exhibits a very small variance across all real networks that we have studied, with an average value 25.4% $\pm$ 6.9% (figure 5(c)). In our definition, sinks are those nodes that stop the propagation of labels; therefore, this result may imply that, on average, around 1/4 people are 'mutes', who do not pass on the action or information, on many kinds of real (social) networks. No similar phenomenon could be seen in the proportion of either hubs (figure 5(b)) or inner members (figure 5(d)), although the proportion of inner members show some clustering features as well. Moreover, although this result is surprisingly robust across various real networks, tests show that it does not always hold true (as one would expect) for ER random networks of different (n, p) and may depend on their $|E|/|N|$ values. At the current stage, however, no structural explanation could be warranted for this observation, and further analysis need to be carried out to better understand this phenomenon.

The categorical data facilitate the comparison of our algorithm's performance on (digital) social networks (3 Deezer networks, 8 Facebook networks) and on traditional (communication) networks (5 arXiv collaboration networks). Results show that (figure 6), the average size of community membership (m_h) and the average size of each community (m_x/N, as a proportion of the network size) of social networks (green and orange) are both clearly greater than that of traditional networks (grey). This is consistent with empirical considerations: on digital social networks, nodes have more access to different communities and thus it is easier to join multiple groups online than offline. Comparisons between different facebook groups are further indicative (figure 6(a)): the average size of community membership is significantly smaller on artist, government and tv-show networks than on politician, athlete, company and public-figure networks, which is close to what one would imagine in real-world situations. As mentioned, it is expected that our algorithm will be more suitable for social networks
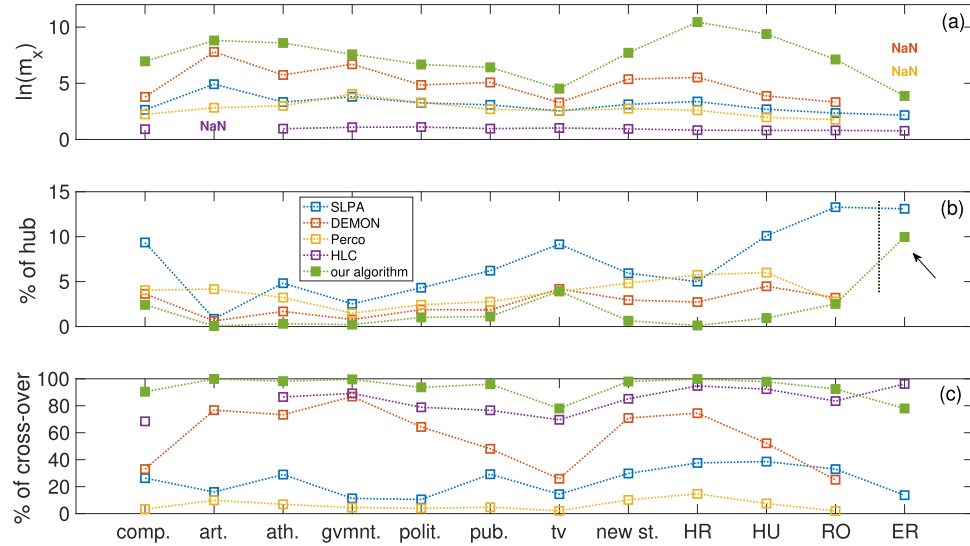
Figure 7. Comparison with other detection methods. 8 Facebook networks and 3 Deezer networks plus a sparse random network ER(5e3, 1e−3) are used to demonstrate the performance of different algorithms, showing (a) $\ln(m_x)$, (b) proportion of hubs (i.e. number of detected communities normalized by network size), and (c) proportion of cross-overs (i.e. proportion of nodes that have multiple community memberships). Four well-known algorithms are studied besides our algorithm: clique percolation (Perco) [1, 28], link community (HLC) [6], SLPA [16], and DEMON [14]. Perco and DEMON could not process properly on the sparse random network, and HLC did not generate result on the large-scale network (FB-artist); corresponding detection results are missing (NaN in the figure). Given its special nature, link community (HLC) discovered more communities than the number of nodes in the network (i.e. $|S|/E < 1$, but $|S|/N > 1$) and thus the results are omitted in (b). Our algorithm successfully separates the random network from real networks, a feature not retained by SLPA (black arrow in (b)). In general, our algorithm detected fewer but larger communities in the graph.

than traditional networks, i.e. the quality factor Φ on social networks will be larger; unfortunately, while results clearly do not show the other way around, the winning margin is relatively vague (table 1 and figure 5(a)). Last, for Steps 2 and 3 of the detection scheme, results show that both the propagation time t_{fin} and the largest distance between end-communities ϵ_{max} (and the running time as well, figure B1) are in general positively correlated with the network size (figure 6(c)), which is consistent with our expectations.

Comparison with other detection methods. We compare our detection results with the results of a few well-known algorithms for overlapping community detection, including the clique percolation (Perco) method [1, 28], the link community (HLC) method [6], the SLPA algorithm [16], and the DEMON algorithm [14]; both SLPA and DEMON adopt the label propagation process, which our detection scheme relies on as well. Recommended parameters are used for these reference algorithms: for SLPA, the iteration timestep is 20 and $r = 0.1$; for DEMON, $\epsilon_{DEMON} = 0.25$ and the minimum community size is 3; for Perco, $k = 4$ (4-clique); for HLC, the dendrogram is cut at the threshold of the maximum partition density for each experimented network (from left to right on the *x*-axis of figure 7, the thresholds are: 0.25, 0.29, 0.34, 0.33, 0.27, 0.48, 0.29, 0.21, 0.21, 0.20, 0.13). Facebook networks (8 networks) and Deezer networks (3 networks) are used to carry out the comparison; these two groups of networks are from the same data source. A random network ER(5e3, 1e−3) is also initiated for the experiments.

The performance of different algorithms are shown and compared in figure 7. Generally speaking, our algorithm detects fewer but larger communities: among all, its results contain the smallest number of communities with the largest average community size m_x. Note that here we plot the number of end-communities (hubs) in our detected hierarchy; high-level communities ($\epsilon > 0$) are even larger and more scarce. This suggests that our detection scheme identifies much denser community structures on networks than the other four algorithms. As for the proportion of cross-overs among all nodes (figure 7(c)), our algorithm generally identifies more cross-overs than the other algorithms, with respect to end-communities; as discussed earlier, such proportions are expected to drop when we calculate the cross-overs with respect to high-level communities obtained at a certain cutoff level of the hierarchy.

Tests suggest a few advantages of our detection scheme. Perco and DEMON could not process properly on the sparse random ER network, and HLC did not generate result on the large size network (facebook-artist) even after a long computational time; corresponding detection results are missing (figure 7). By contrast, our algorithm is robust on both sparse and large-scale networks. SLPA is not deterministic, and detection results from multiple runs differ to a non-trivial extent; it is also not able to clearly separate random networks from real networks, at least by the number of communities detected as a fraction of the number of nodes (% of hubs),

which is considered as an important metric in the detection results of our algorithm (figure 7(b)). Perco often assigns no community to a large portion of nodes, given the sparse existence of cliques in real networks; even so, it found more communities than our algorithm, most of which are small-scale. Given its special nature, the link community method (HLC) always discovered more communities than the number of nodes in the network (i.e. $|S|/E < 1$, but $|S|/N > 1$) and thus corresponding results are omitted in figure 7(b). Both Perco and HLC determined communities far smaller than our algorithm; they also do not exhibit consistent $O(E)$ time complexity, unlike SLPA and DEMON (figure B2). In a further test, we examined the performance of these algorithms on the Karate club network, which clearly shows that our detection results are the most reliable on this classic small network (figure D1, appendix D). In general, DEMON yields detection results closest to the results of our algorithm, yet its inability on sparse networks (figure 7) and relatively insufficient coverage of nodes in communities (e.g. figure D1) highlight the advantage of our new detection scheme. Finally, none of these reference algorithms has the ability to self-indicate its effectiveness on different networks and they all rely on certain parameter-tuning efforts in practice. The two novel features (parameter-free and self-falsifiable) of our solution scheme stand out.

Discussion and concluding remarks

In this study, we formulated an integrated belief for the algorithmic design of the community detection problem, consisting of six aspects: *overlappedness*, *different roles of nodes*, *behavioral locality*, *propagatory formulation of communities*, *order of communities*, and *self-falsifiability*. Based on the belief, we proposed a multi-step detection scheme that tries to incorporate successful ideas of existing algorithms as well as to obviate their exposed weaknesses. Our solution scheme relies on nodes' centrality scores to determine their different roles in the graph, especially the hubs and boundaries of end-communities, and initiates a diffusive label propagation process that tries to simulate the formation of communities on social networks. Small communities are iteratively aggregated into large communities and at the end of the detection, a hierarchical order of overlapping communities is established, with the entire graph sitting on the top of the hierarchy. Since there is in fact no concrete and general definition of a community structure on graphs and communities could then only be defined in the relative sense [36], we are attached to the belief that the old problem of finding the best partition of communities could be replaced by the new problem of finding the best cutoff level on a community hierarchy, which could be constructed on any given graph. With this idea in mind, in this study our solution scheme makes a tentative attempt.

The label propagation process initiated in this study essentially resembles the gradient flow in [38] (with centrality scores as the energy function), and the identification of end-communities echoes with the identification of attraction basins in that study. However, in [38], the focus is the classification of nodes into high-level roles and the construction of the corresponding hierarchical structure, whereas in our study, the focus is the bottom-up aggregation and the hierarchical representation of communities. As mentioned before, the two studies have common grounds while highlighting different aspects of the problem; the current study could be viewed as complementary to [38].

Our detection algorithm is parameter-free, and therefore as a trade-off, it is not fully decisive. While consolidated detection results of community structures are not produced by our completely objective algorithm, we adopt a few sophisticated measures that provide useful information for the determination of communities, specifically, the cutoff level of the community hierarchy. A peak on the $\epsilon \leftrightarrow \Delta|R_\epsilon|$ curve (or equivalently, a plateau on the $\epsilon \leftrightarrow |R_\epsilon|$ curve) means that across a certain ϵ stage the community hierarchy barely changes, which implies that such an ϵ level might be an appropriate candidate for the cutoff. Similarly, a peak on the $\epsilon \leftrightarrow \Phi_\epsilon$ curve suggests that across such ϵ level the merging of small communities into big ones is well-conditioned, in terms of the defined J–D consistency (equation (2)); thus this ϵ level is also a desired cutoff. By taking into account these two aspects, which often agree on the same ϵ, we may be able to decide an appropriate cutoff level of the community hierarchy. However, it should be noted that despite the proposed solution, the determination of the cutoff level is far from being consolidated; in many cases, subjective heuristics still need to be called for in making the decision.

An important feature of our detection scheme is the automatic indication of the goodness of detection results. As discussed in Peel *et al* [36], any community detection algorithm has only a limited power in application, inevitably not being able to conduct successful detections on networks with certain topologies. Therefore, we believe that a reliable detection scheme should be able to notify implementers with the quality of the detection results it yields; in particular, the scheme should be able to indicate its potential failures. Such an automatic self-check procedure is embedded in our algorithm. By defining the concept of J–D consistency which indicates the quality of the mergings of small communities into big ones during the formation of the community hierarchy, we invented a robust metric Φ that quantitatively indicates the quality of detection results (which may

also facilitate the determination of the best cutoff level on the community hierarchy). Although this metric is originated not from rooted mathematical theories but rather from engineering buildups, it is shown that the metric is quite robust, especially in the successful separation of random networks from real networks; upon this construction, it is expected that more advanced metrics in this direction would be invented in future works. Self-falsifiability and the parameter-free property are not emphasized by existing algorithms and may be considered as novel features of our detection scheme.

We tested our algorithm on networks of various sizes and kinds (figures 2–6). On small real networks (e.g. Karate Club network, Dolphin network), our algorithm yielded very good detection results. On LFR networks, our result reveals the ground truth to a great extent, and it shows that our heuristics for determining the cutoff level of the community hierarchy are reliable. ER random networks could be effectively distinguished by our algorithm. On large-scale real networks, horizontal analysis further demonstrate the self-consistency of our detection scheme, and a few interesting phenomena emerge from the results, which exhibits extra values of this study beyond the algorithmic design. Specifically, an unexpected observation emerged, showing that under our identification scheme there are always around 1/4 nodes in the graph that are mutes in the propagation of information or action, on various types of real networks. Although this phenomenon is significant in our results, more work needs to be done before it could be verified and generalize.

Advantages of our algorithms over existing overlapping community detection algorithms could be identified (figure 7). Unlike the clique percolation method, the DEMON algorithm and the link community method, our detection scheme is robust on both sparse networks and large-scale networks; it yields deterministic detection results and successfully separates random networks from real networks, two superior features over the SLPA algorithm. In general, our algorithm generates fewer but larger communities than all the above algorithms, capturing the dense community structures on the network. The comparison of different algorithms' performance on the Karate club network provides unambiguous evidence in favor of our algorithm's reliability.

In our detection results, the strength of nodes' membership in different communities is not assumed to be homogenous and could possibly be indicated by utilizing the timestamp t in their infection history, which records the first time the node gets exposed to community labels. The hierarchical order of communities are maintained throughout the workflow, thus the whole detection process is fully transparent. We believe that transparency is an important feature of this detection scheme, and the inclusion of timestamps in the finite memory associated with each node makes the algorithm easy to be extended to temporal networks or high-order networks (e.g. [57]), possibly with a refined centrality measure for these advanced networks [58]. Another line of extension for this study is to replace some flexible components of the algorithm and test with alternatives, for example, different centrality measures (Step 1) and alternative graph distance measures (Step 3). In the current scheme we used the most common measures (degree centrality, shortest path distance), but under the rapid development of network sciences, it would be interesting to apply and test alternative ideas under our general solution scheme in future studies.

A number of limitations exist in this study, besides what have been discussed. First, in theory, our algorithm is not able to identify communities that are *strictly contained* in larger communities. Spectral clustering on the connectivity matrix of each determined community needs to be performed in order to find sub-communities strictly lying within big communities. Second, although we claim that the algorithm is parameter-free, a few quantitative constraints are still implied in our solution scheme, although they are not represented by explicit parameters. For example, we assume that nodes could only directly propagate the labels to their *immediate* neighbors; this could be viewed as a dummy parameter $d_{prop} = 1$ (distance of infections). The choice of centrality measures may also be viewed as a tuning procedure. Third, besides comparing on some general metrics of the detected communities, we found it a bit difficult to compare our detection results (a hierarchy of communities) with results obtained from other algorithms (a certain community partition) or more importantly, with the ground truth, although people argue that comparing detection results with ground truths may not be always desirable since the ground truth does not always reflect the real community structures of the network [59]. It is plausible that we could compare the determined communities at the cutoff level of the community hierarchy with the singular detection result of other algorithms or the ground truth, as we did with LFR networks, but it is possible that multiple cutoffs could be identified in the hierarchy and therefore the comparison becomes less straightforward. Sophisticated metrics need to be invented to address this comparison, or in general, to better characterize the performance of our proposed solution scheme.

Data and code availability

All network datasets used in this study could be found on Networkx (https://networkx.github.io) and SNAP (Stanford Network Analysis Project; http://snap.stanford.edu/index.html). A Python package of the detection algorithm is available at https://github.com/TimothyLi0123/LZ-cd.git.

Acknowledgments

Appendix A. Overview of overlapping community detection methods

The topic of community detection on graphs is extensively studied over the time and a numerous set of algorithms have been proposed to deal with the problem. Although the research history for community detection is not long, there has seen multiple generations of views and ideas for this topic, and traditional methods are quickly surpassed by more advanced approaches. Important transitions of ideas include the transition from detecting exhaustive (disjoint) communities to overlapping communities, the transition from a deterministic definition of communities to a probabilistic definition of communities, and the transition from relying on synthetic data and data of small real networks without explicit community structures to test the algorithm, to utilizing networks with ground truth community structures, and then to further realizing the limits of ground truth constraints in evaluating community detection results [30]. The active evolution of community detection methods reflects the unconsolidated nature of the problem.

As discussed in the main text, existing algorithms offer a great number of important aspects for the algorithmic design on overlapping community detection. From an evolutionary perspective, those ideas constitute a transitional logic line for thinking about the problem, and one could identify multiple stages in the development of solutions. Here we present an overview of overlapping community detection methods, trying to establish conceptual links connecting different ideas and to point out their successful insights as well as shortcomings.

A.1. Link communities
The idea of link communities, detected by a hierarchical clustering of edges [6, 7], is based on the assumption that vertex communities may be overlapped but the corresponding link communities are always disjoint. In other words, it implies that the boundaries of communities are not determined by nodes, as traditionally assumed, but by the edges connecting them. Despite being an advanced view over hard-partitioning of nodes, this idea is still subject to improvements since it is possible that edges also belong to different communities and hard-partitioning on edges is still an imposed assumption. The overlapping of communities, in a broader sense, should allow communities to share a finite part of their components, consisting of both nodes and edges. This view of the overlappedness of communities is related to the recent discussion of 'dominant communities' versus 'hidden communities' [37], which emphasizes that detected communities are not of the same significance to the graph and may demonstrate different strength, essentially embodying the idea of hierarchical community structures (see below).

A.2. Seed set expansion
Alongside the abandoning of hard-partitions, which inspired a lot of metric-based optimization methods that directly deal with the entire graph, people gradually adopted the new belief that *locality* matters in the determination of communities. In particular, a local determination is more consistent with the logic behind the formulation of communities in real social networks, where nodes often do not have a clear sense of the entire network and groups mostly emerge from local commonalities. Adopting this modern view, a new category of algorithms for community detection, termed as the seed set expansion process, has been gaining more and more attention. The idea is to start with finite seed sets and expand them into communities by adding/removing nodes to/from the set if a certain measure of the community is improved, such as modularity [2], conductance [3], outwardness [25], fitness [4], significance (OSLOM, [26]), or aggregative metric based on link prediction algorithms [5]. One important line of seed set expansion algorithms originate from the PageRank algorithm and expand the seed set based on a random walk process, as pioneered by the work of Andersen and Lang [3] and Andersen *et al* [9]. Li *et al* [27] proposed an algorithm in which the seed set is determined based on clique-detection methods, as cliques could essentially be viewed as communities cores [1, 28]. Kloumann and Kleinberg [11] studied different seed set expansion algorithms through a comparative analysis, focusing on the determination of a good seed set. More recently, Gialampoukidis *et al* [29] proposed a core identification strategy, an algorithm based on the DBSCAN method [44, 45] where two parameters are adopted: (1) ϵ defines the radius of the neighborhood of a node that is considered; (2) *MinPts* is the minimum number of neighbors of a node's ϵ-neighborhood; nodes are defined as cores if they have more than *MinPts* neighbors in their

ϵ-neighborhood. Similarly, Bai *et al* [10] proposed an algorithm for overlapping community detection using the nodes that are density peaks as community cores, an idea borrowed from clustering analysis [60]. Nodes with high local density ρ and large distance δ from other density peaks are identified in the ρ–δ plot as community cores, around which other nodes are classified.

We notice that, among existing seed sed expansion methods, a few problems arise. First, many existing algorithms make ad-hoc decisions on the seed set or the community core (e.g. cliques), which often consists of an arbitrary number of nodes. Clique percolation methods use cliques as the seed sets, while the size of the cliques is experimentally decided [8]. Kloumann and Kleinberg [11] shows that in fact a random seed set may yield better performance than a seed set selecting high-degree nodes. Lancichinetti *et al* [4] invented the notion of the 'natural community' of nodes, which essentially serves as the community cores. There is little agreement on how many nodes a seed set should consist, and what is the order for these seeds to join the set, if the set has multiple nodes. We argue that it is more natural to assume that in most cases initially each seed set only contains a single node, and all other nodes sequentially joining the set should follow a hierarchical order; only for the rare case that neighboring nodes have completely identical topological features, could a seed set consists more than one node. The second problem is that in most existing algorithms, the expansion process is in fact still non-local: it does not allow each node *itself* to decide whether it should join a community, and in many cases the stopping criterion for expansion is still from an optimization standpoint. As we mentioned, in social network settings, nodes themselves are often ignorant about the nature of the entire network, which leads to the idea of seed set expansion; moreover, most likely nodes are also unaware of the situation of the rest of their belonged communities: they do not know if their joining or leaving the group will maximize some metrics of the community, and even they do, this may not be the factor that influences their decision. Therefore, we believe that the stopping criterion for seed set expansion is supposed to follow a more behavioral rule when dealing with human networks.

A.3. Label propagation

The second problem for the abovementioned seed set expansion algorithms, that they assume subsequent nodes are attached to the communities in a static and non-local fashion, could be resolved by an advanced idea, that the community assignment of non-core nodes is determined from a propagatory standpoint. This brings in the idea of another line of community detection methods, known as label propagation algorithms, first proposed by Raghavan *et al* [15] and having seen a lot of variants thereafter (for example, the speaker-listener SLPA [16], the DEMON algorithm [14] and more recent designs e.g. [13]). The idea of label propagation is simple: iteratively each node sends the label of its community membership to its neighbors, and at each time step the node's community membership is updated based on the information it receives from all neighbors, according to certain decision rules (e.g. a majority vote [15] or a listener-speaker scenario [16]); and eventually, the algorithm will stop at convergence, i.e. there is no more update of community membership on any node during the propagation.

We believe that this dynamic and propagatory point of view for community detection is important in social network settings: nodes compete with each other trying to expand their influence, and finally the winners will be able to establish their communities. It is more advanced than the traditional view that community membership is *a priori* determined from a global optimization standpoint. We agree that community assignments should definitely rely on the graph's topology, but instead of regressing the community membership to simplified metrics of the topology, it may be more organic (especially for social networks) to set up the propagation and let the dynamics decide the equilibrium convergence. By this means, label propagation algorithms successfully highlight *complete* locality in the determination of community memberships, as no optimization at any non-individual level is assumed. However, one significant problem for this approach is that the propagation could follow arbitrary rules, and thus each proposal of a different rule for community decisions will possibly end up with a new algorithm, which suggests that the label propagation idea essentially consists of an unlimited algorithmic space. Inevitably, this triggers debates on a good decision rule that processes a node's information received from different neighbors. Moreover, decision rules in the first generation of label propagation algorithms often select one community label for each node from all candidates and thus result in hard-partitioning; to apply label propagation in overlapping community detection, improved designs are to be invented.

A.4. Nodes with memory

The above difficulty could be overcome by a new generation of label propagation algorithms that introduce a finite memory associated with each node. With the memory kernel storing the information during the propagation (infection) process, detection algorithms are now able to carry out overlapping communities results [39]. The idea of nodes with memories is aligned with the term 'fuzzy detection' [16]; it retains more information of the propagation process than simplistic decision rules leading to hard-partitioning (e.g. the majority vote),

although in existing designs some infection information is still compromised [39], such as the receiving order of labels. From the node's memory, the finite infection history it experienced could be revealed and then used to decide its multiple community membership. Given these considerations, we argue that the memory of nodes is an important feature for effective overlapping community detection methods based on the propagation process. Moreover, as a side note, another problem of the algorithm in Gregory [39] is that it requires a pre-determined number of communities in the graph in order to set the dimension of the memory vector, which we believe is not necessary.

A.5. Multi-step detection and hierarchical structures

While seed set expansion based on label propagation process is a modern and arguably successful heuristic for community detection, one should note that a complete seed set expansion scheme is multi-step, and it requires specific algorithm design for each step of the workflow. Unfortunately, most previous studies focused on one stage of seed set expansion and few efforts have been made on designing the workflow of the expansion process.

Li *et al* [12] proposed a multi-step community discovery scheme for textual data where each node is a piece of text. First, the seeding cores are identified using the *Apriori* algorithm; then the detected cores are merged based on similarity; after the determination of cores, all other nodes are assigned to communities relying on their connectivity conditions; and finally, a classification step is applied to make sure that each node belongs to the right community and false assignments are removed. Whang *et al* [35] proposed another multi-step detection algorithm based on seed set expansion. The algorithm consists of four stages: filtering, seeding, seed set expansion and propagation. At the first stage the graph is pruned to core components that are densely connected, and the peripheral structures are omitted. The seed set is determined in the next stage, around which communities are formulated, using the spectral method based on the optimization of conductance, originated from Andersen *et al* [9]. The omitted peripheral structures are reinstalled to the detected communities at the final stage. Multi-step algorithms extensively appear in the detection of hierarchical community structures (e.g. [4, 26, 40]), which has been drawing more and more attention recently. The idea of hierarchical communities is that the detection of communities should associate the partitioning with an order of significance, possibly through a hierarchy, instead of treating all detected communities equally, as most existing methods do. Sales-Pardo *et al* [42] proposed a method uncovering the hierarchical organizations of nodes based on a new node-affinity metric and on searching for the local maxima of modularity. Shen *et al* [43] designed a multi-step algorithm named EAGLE to detect hierarchical and overlapping community structures, where maximal cliques in the graph are used as the seed set and an agglomerative process relying on modularity maximization helps establish the hierarchy. A similar multi-step algorithm named SHRINK was proposed by Sun *et al* [34], where each node is assigned with an initial label and the (multi-ary, as opposed to binary) hierarchical community structure is gradually established by measuring the modularity gain of merging end-communities. Peixoto [41] studied the hierarchical structure of the SBM and proposed an inference algorithm to select the best multi-level hierarchical model, which facilitates the formation of benchmark hierarchical SBM graphs for testing detection algorithms. Recently, a recursive bi-partitioning algorithm is devised with a top-down partition workflow [61], as opposed to the agglomerative process (e.g. [34, 43]). Overall, multi-step algorithms based on certain propagation processes that consider the hierarchical structure of communities, as emerged in this evolutionary discussion, may contribute the modernest ideas to current community detection methodologies.
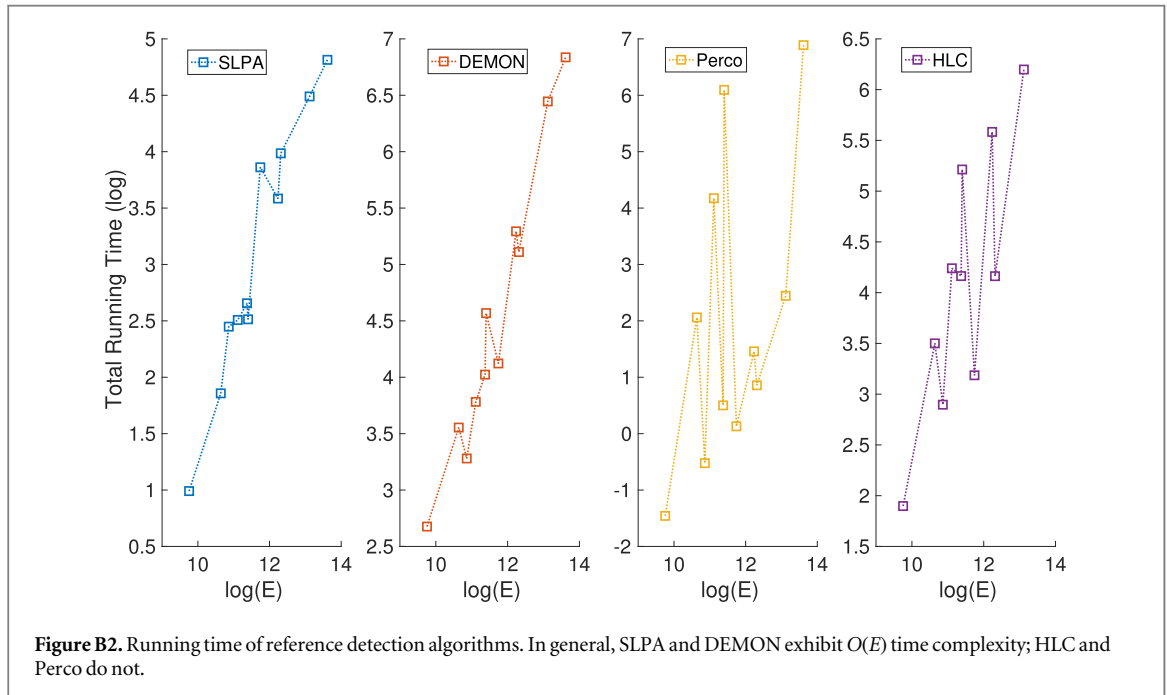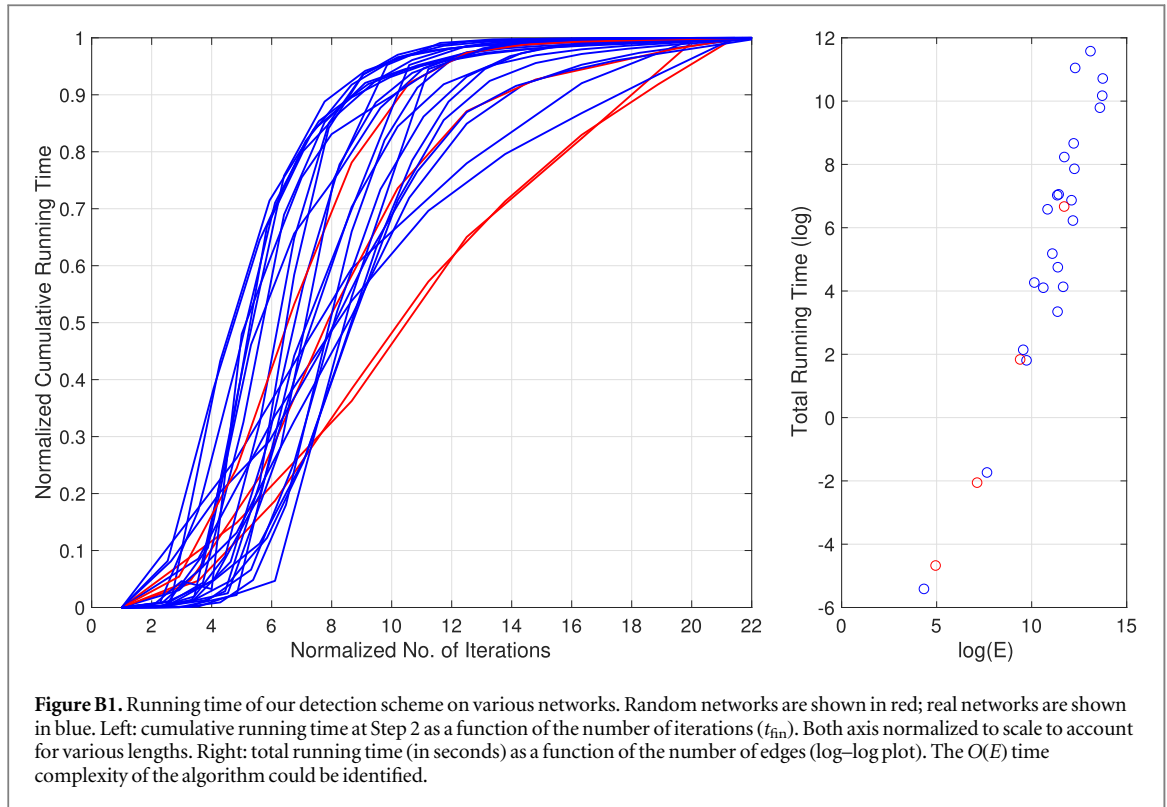
A.6. Isolated nodes

As a last note, it should be pointed out that the attention to the peripheral structures of the graph, besides the densely connected cores, is non-trivial, which is relevant to the idea of ungrouped isolated nodes [35]. The belief is that, not all nodes belong to communities; isolated nodes (noises) do exist. Gfeller *et al* [32] regarded them as 'unstable nodes' and discussed the determination of these nodes through essentially a Monte-Carlo approach by imposing random noises on edge weights. Gui *et al* [33] proposed a seed-set-based label propagation algorithm that discovers 'boundary nodes' as opposed to 'core nodes', whose basic idea is similar. Sun *et al* [34] also discussed the 'hubs' and 'outliers' among 'homeless' nodes identified in the detection process. Nevertheless, in general the notion of isolated nodes is often neglected by existing works and people tend to assign community memberships to all nodes in the graph.

Appendix B. Running time analysis

Comparing the running time of our algorithm on various networks, it shows that random networks could possibly be spotted during the label propagation process. The cumulative running time curve for a random network generally does not follow an S-shape, as the case on real networks, and instead demonstrates a more gradual growth (figure B1, left). This is because on real networks the depth of propagation, i.e. the reachability of

Figure B1. Running time of our detection scheme on various networks. Random networks are shown in red; real networks are shown in blue. Left: cumulative running time at Step 2 as a function of the number of iterations (t_{fin}). Both axis normalized to scale to account for various lengths. Right: total running time (in seconds) as a function of the number of edges (log–log plot). The $O(E)$ time complexity of the algorithm could be identified.

Figure B2. Running time of reference detection algorithms. In general, SLPA and DEMON exhibit $O(E)$ time complexity; HLC and Perco do not.
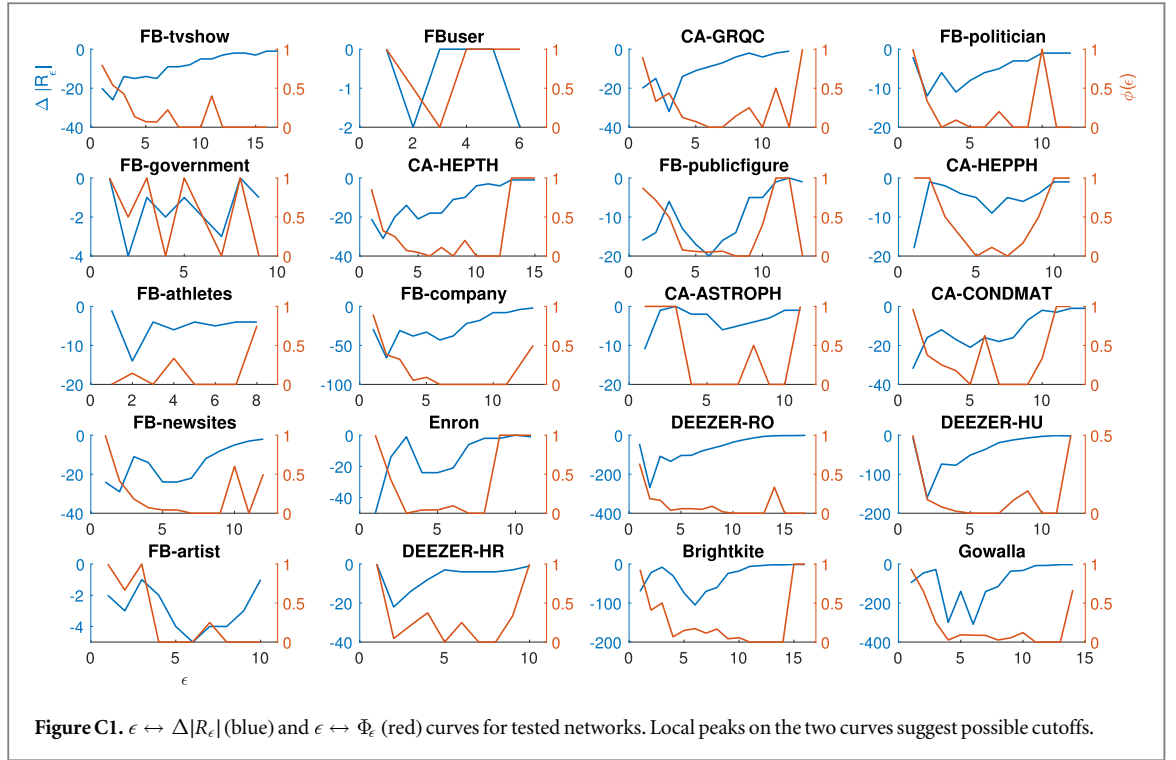
end-communities (hubs) is often heterogeneously distributed with a small tail, while on random networks all end-communities tend to have the same topological features and thus the simultaneous propagation from different hubs is more gradual and synchronized. However, this criterion may lead to a wrong catch since the curves for some real (traditional) networks are also not well S-shaped.

It shows that the total running time of the detection scheme has a quasi-linear relationship with the network size (figure B1, right), demonstrating the $O(E)$ time complexity of our algorithm (equation (5)).

For other detection methods, tests show that SLPA and DEMON demonstrate a good $O(E)$ time complexity, similar to our algorithm, while the running time of HLC or Perco does not exhibit a consistent dependence on the scale of the network (figure B2). As is acknowledged, a highly variant computational time undermines the applicability of the detection algorithm.
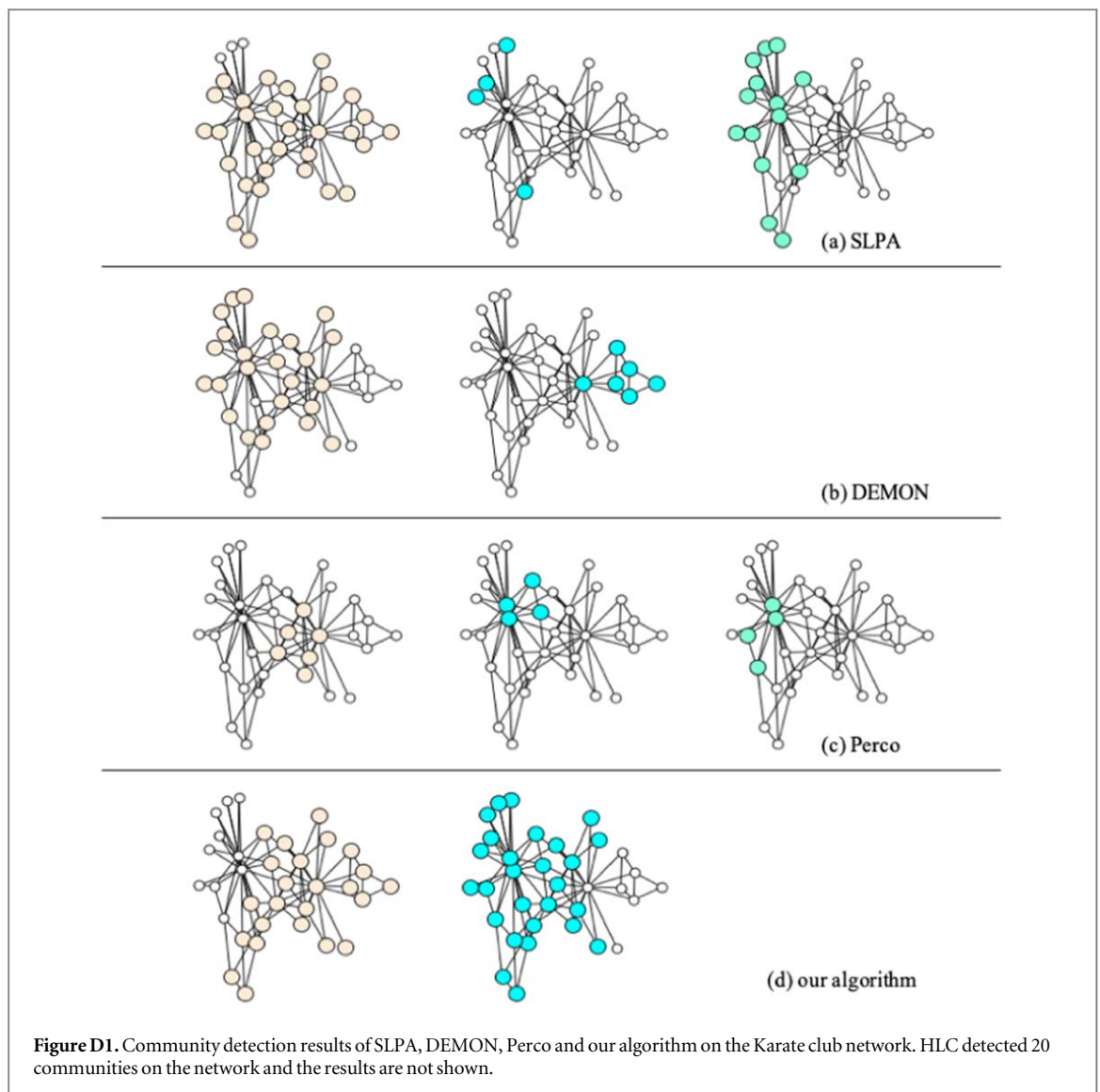
Figure C1. $\epsilon \leftrightarrow \Delta|R_\epsilon|$ (blue) and $\epsilon \leftrightarrow \Phi_\epsilon$ (red) curves for tested networks. Local peaks on the two curves suggest possible cutoffs.

Appendix C. Cutoff level of the community hierarchy

The determination of the cutoff level of the community hierarchy on large real networks could refer to the $\epsilon \leftrightarrow \Delta|R_\epsilon|$ and $\epsilon \leftrightarrow \Phi_\epsilon$ curves (figure C1), following the same heuristics explained on the LFR network (figure 4). The local peaks on these two curves indicate potential cutoff levels of ϵ. In practice, two peaks often agree on the same value, which implies a good cutoff.

Appendix D. Performance of different detection algorithms on the karate club network

Different detection algorithms are tested on the Karate club network (figure D1). The SLPA algorithm is not deterministic even on the small-scale network, and one detection result is shown. HLC detected 20 communities from 34 nodes and 78 edges, which is clearly not satisfactory and hence the results are not shown. DEMON and Perco yielded deterministic and meaningful results. Compared with other algorithms, our detection scheme won by a large margin on the Karate club network; the results are reliable and match the ground truth to a great extent (figure 2).

Figure D1. Community detection results of SLPA, DEMON, Perco and our algorithm on the Karate club network. HLC detected 20 communities on the network and the results are not shown.

ORCID iDs

Tianyi Li ⬤ https://orcid.org/0000-0002-4654-9862

References

[1] Palla G, Derényi I, Farkas I and Vicsek T 2005 Uncovering the overlapping community structure of complex networks in nature and society *Nature* **435** 814

[2] Clauset A 2005 Finding local community structure in networks *Phys. Rev.* E **72** 026132

[3] Andersen R and Lang K J 2006 Communities from seed sets *Proc. 15th Int. Conf. on World Wide Web* (New York: ACM) pp 223–32

[4] Lancichinetti A, Fortunato S and Kertész J 2009 Detecting the overlapping and hierarchical community structure in complex networks *New J. Phys.* **11** 033015

[5] Shang K, Small M, Wang Y, Yin D and Li S 2019 A novel metric for community detection arXiv:1909.12467

[6] Ahn Y Y, Bagrow J P and Lehmann S 2010 Link communities reveal multiscale complexity in networks *Nature* **466** 761

[7] Evans T S and Lambiotte R 2009 Line graphs, link partitions, and overlapping communities *Phys. Rev.* E **80** 016105

[8] Xie J, Kelley S and Szymanski B K 2013 Overlapping community detection in networks: the state-of-the-art and comparative study *ACM Comput. Surv.* **45** 43

[9] Andersen R, Lang K and Chung F 2006 Local graph partitioning using pagerank vectors *2006 47th Annual IEEE Symp. on Foundations of Computer Science (FOCS'06)(FOCS)* pp 475–86

[10] Bai X, Yang P and Shi X 2017 An overlapping community detection algorithm based on density peaks *Neurocomputing* **226** 7–15

[11] Kloumann I M and Kleinberg J M 2014 Community membership identification from small seed sets *Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (New York: ACM) pp 1366–75

[12] Li H, Nie Z, Lee W C, Giles L and Wen J R 2008 Scalable community discovery on textual data with relations *Proc. 17th ACM Conf. on Information and Knowledge Management* (New York: ACM) pp 1203–12

[13] Chin J H and Ratnavelu K 2017 A semi-synchronous label propagation algorithm with constraints for community detection in complex networks *Sci. Rep.* **7** 45836

[14] Coscia M, Rossetti G, Giannotti F and Pedreschi D 2012 Demon: a local-first discovery method for overlapping communities *Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (New York: ACM) pp 615–23

[15] Raghavan U N, Albert R and Kumara S 2007 Near linear time algorithm to detect community structures in large-scale networks *Phys. Rev.* E **76** 036106

[16] Xie J, Szymanski B K and Liu X 2011 Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process *Data Mining Workshops (ICDMW)* (Piscataway, NJ: IEEE) pp 344–9

[17] Li Y, He K, Kloster K, Bindel D and Hopcroft J 2018 Local spectral clustering for overlapping community detection *ACM Trans. Knowl. Discovery Data* **12** 17

[18] Rosvall M and Bergstrom C T 2008 Maps of random walks on complex networks reveal community structure *Proc. Natl Acad. Sci.* **105** 1118–23

[19] Karrer B and Newman M E 2011 Stochastic blockmodels and community structure in networks *Phys. Rev.* E **83** 016107

[20] Shen H W, Cheng X Q and Guo J F 2011 Exploring the structural regularities in networks *Phys. Rev.* E **84** 056111

[21] Zhang P and Moore C 2014 Scalable detection of statistically significant communities and hierarchies, using message passing for modularity *Proc. Natl Acad. Sci.* **111** 18144–9

[22] Zhang P, Moore C and Newman M E J 2016 Community detection in networks with unequal groups *Phys. Rev.* E **93** 012303

[23] Ball B, Karrer B and Newman M E 2011 Efficient and principled method for detecting communities in networks *Phys. Rev.* E **84** 036103

[24] Chen Y, Wang X, Xiang X, Tang B, Chen Q, Fan S and Bu J 2017 Overlapping community detection in weighted networks via a bayesian approach *Physica* A **468** 790–801

[25] Bagrow J P 2008 Evaluating local community methods in networks *J. Stat. Mech: Theory Exp.* P05001

[26] Lancichinetti A, Radicchi F, Ramasco J J and Fortunato S 2011 Finding statistically significant communities in networks *PLoS One* **6** e18961

[27] Li J, Wang X and Eustace J 2013 Detecting overlapping communities by seed community in weighted complex networks *Physica* A **392** 6125–34

[28] Reid F, McDaid A and Hurley N 2012 Percolation computation in complex networks *2012 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining* (Piscataway, NJ: IEEE) pp 274–81

[29] Gialampoukidis I, Tsikrika T, Vrochidis S and Kompatsiaris I 2016 Community detection in complex networks based on DBSCAN* and a Martingale process *2016 11th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)* (Piscataway, NJ: IEEE) pp 1–6

[30] Fortunato S and Hric D 2016 Community detection in networks: a user guide *Phys. Rep.* **659** 1–44

[31] Krzakala F, Moore C, Mossel E, Neeman J, Sly A, Zdeborová L and Zhang P 2013 Spectral redemption in clustering sparse networks *Proc. Natl Acad. Sci.* **110** 20935–40

[32] Gfeller D, Chappelier J C and De Los Rios P 2005 Finding instabilities in the community structure of complex networks *Phys. Rev.* E **72** 056135

[33] Gui Q, Deng R, Xue P and Cheng X 2018 A community discovery algorithm based on boundary nodes and label propagation *Pattern Recognit. Lett.* **109** 103–9

[34] Sun H, Huang J, Han J, Deng H and Sun Y 2010 SHRINK: a structural clustering algorithm for detecting hierarchical communities in networks *ACM Int. Conf. on Information & Knowledge Management* (New York: ACM) pp 219–28

[35] Whang J J, Gleich D F and Dhillon I S 2013 Overlapping community detection using seed set expansion *Proc. 22nd ACM Int. Conf. on Information & Knowledge Management* (New York: ACM) pp 2099–108

[36] Peel L, Larremore D B and Clauset A 2017 The ground truth about metadata and community detection in networks *Sci. Adv.* **3** e1602548

[37] He K, Li Y, Soundarajan S and Hopcroft J E 2018 Hidden community detection in social networks *Inf. Sci.* **425** 92–106

[38] E W, Lu J and Yao Y 2013 The landscape of complex networks? Critical nodes and a hierarchical decomposition *Methods Appl. Anal.* **20** 383–404

[39] Gregory S 2010 Finding overlapping communities in networks by label propagation *New J. Phys.* **12** 103018

[40] Clauset A, Moore C and Newman M E 2008 Hierarchical structure and the prediction of missing links in networks *Nature* **453** 98

[41] Peixoto T P 2014 Hierarchical block structures and high-resolution model selection in large networks *Phys. Rev.* X **4** 011047

[42] Sales-Pardo M, Guimerà Roger, Moreira André A and Nunes Amaral Luìs A 2007 Extracting the hierarchical organization of complex systems *Proc. Natl Acad. Sci.* **104** 15224–9

[43] Shen H, Cheng X, Cai K and Hu M B 2008 Detect overlapping and hierarchical community structure in networks *Physica* A **388** 1706–12

[44] Campello R J, Moulavi D and Sander J 2013 Density-based clustering based on hierarchical density estimates *Pacific-Asia Conf. on Knowledge Discovery and Data Mining* (Berlin: Springer) pp 160–72

[45] Ester M, Kriegel H P, Sander J and Xu X 1996 A density-based algorithm for discovering clusters in large spatial databases with noise *Kdd* vol 96, pp 226–31

[46] Schubert E, Sander J, Ester M, Kriegel H P and Xu X 2017 DBSCAN revisited, revisited: why and how you should (still) use DBSCAN *ACM Trans. Database Syst.* **42** 19

[47] Zachary W W 1977 An information flow model for conflict and fission in small groups *J. Anthropol. Res.* **33** 452–73

[48] Lusseau D, Schneider K, Boisseau O J, Haase P, Slooten E and Dawson M 2003 *Behav. Ecol. Sociobiol.* **54** 396

[49] Lancichinetti A, Fortunato S and Radicchi F 2008 Benchmark graphs for testing community detection algorithms *Phys. Rev.* E **78** 046110

[50] Erdös P and Rényi A 1960 On the evolution of random graphs *Publ. Math. Inst. Hung. Acad. Sci.* **5** 17–60

[51] Yang J and Leskovec J 2015 Defining and evaluating network communities based on ground-truth *Knowl. Inf. Syst.* **42** 181–213

[52] Leskovec J, Lang K J, Dasgupta A and Mahoney M W 2009 Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters *Internet Math.* **6** 29–123

[53] Leskovec J and Mcauley J J 2012 Learning to discover social circles in ego networks *Conf. Proc. Advances in Neural Information Processing Systems* pp 539–47

[54] Leskovec J, Kleinberg J and Faloutsos C 2007 Graph evolution: densification and shrinking diameters *ACM Trans. Knowl. Discovery Data* **1** 2

[55] Rozemberczki B, Davies R, Sarkar R and Sutton C 2019 Gemsec: Graph embedding with self clustering *Proceedings 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* pp 65–72

[56] Cho E, Myers S A and Leskovec J 2011 Friendship and mobility: user movement in location-based social networks *Proc. 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (New York: ACM) pp 1082–90

[57] Lambiotte R, Rosvall M and Scholtes I 2018 Understanding complex systems: from networks to optimal higher-order models arXiv:1806.05977

[58] Grindrod P, Parsons M C, Higham D J and Estrada E 2011 Communicability across evolving networks *Phys. Rev.* E **83** 046120

[59] Hric D, Peixoto T P and Fortunato S 2016 Network structure, metadata, and the prediction of missing nodes and annotations *Phys. Rev. X* **6** 031038

[60] Rodriguez A and Laio A 2014 Clustering by fast search and find of density peaks *Science* **344** 1492–6

[61] Li T, Bhattacharyya S, Sarkar P, Bickel P and Levina E 2018 Hierarchical community detection by recursive bi-partitioning arXiv:1810.01509v1

[62] Evans T S 2010 Clique graphs and overlapping communities *J. Stat. Mech: Theory Exp.* P12037

[63] Girvan M and Newman M E 2002 Community structure in social and biological networks *Proc. Natl Acad. Sci.* **99** 7821–6

[64] Lancichinetti A and Fortunato S 2009 Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities *Phys. Rev.* E **80** 016118

[65] Zhang S, Wang R S and Zhang X S 2007 Uncovering fuzzy community structure in complex networks *Phys. Rev.* E **76** 046103