OPEN ACCESS Freely available online

PLOS ONE

Active Semi-Supervised Community Detection Based on Must-Link and Cannot-Link Constraints

Jianjun Cheng*, Mingwei Leng, Longjie Li, Hanhai Zhou, Xiaoyun Chen*

School of Information Science and Engineering, Lanzhou University, Lanzhou, Gansu Province, China

Abstract

Community structure detection is of great importance because it can help in discovering the relationship between the function and the topology structure of a network. Many community detection algorithms have been proposed, but how to incorporate the prior knowledge in the detection process remains a challenging problem. In this paper, we propose a semi-supervised community detection algorithm, which makes full utilization of the *must-link* and *cannot-link* constraints to guide the process of community detection and thereby extracts high-quality community structures from networks. To acquire the high-quality *must-link* and *cannot-link* constraints, we also propose a semi-supervised component generation algorithm based on active learning, which actively selects nodes with maximum utility for the proposed semi-supervised community detection algorithm step by step, and then generates the *must-link* and *cannot-link* constraints by accessing a noiseless oracle. Extensive experiments were carried out, and the experimental results show that the introduction of active learning into the problem of community detection makes a success. Our proposed method can extract high-quality community structures from networks, and significantly outperforms other comparison methods.

Citation: Cheng J, Leng M, Li L, Zhou H, Chen X (2014) Active Semi-Supervised Community Detection Based on Must-Link and Cannot-Link Constraints. PLoS ONE 9(10): e110088. doi:10.1371/journal.pone.0110088

Editor: Rongrong Ji, Xiamen University, China

Received June 15, 2014; **Accepted** September 16, 2014; **Published** October 17, 2014

Data Availability: The authors confirm that all data underlying the findings are fully available without restriction. All relevant data are within the paper.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities, grant id: lzujbky-2014-54. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* Email: chengjianjun@lzu.edu.cn (JC); chenxy@lzu.edu.cn (XC)

Introduction

Community structures are significant features observed in many complex networks, meaning that the nodes in a network can be divided naturally into groups, within which connections are relatively dense but between which connections are much sparser. Communities may correspond to the sets of topic-related Web pages in Web graphs [1–3], the papers on certain scientific research subjects in article citation networks [4,5], the real social groupings in social networks [6–9], or the basic reaction modules or other functional units in metabolic networks and protein-protein interaction networks [3,10–15]. Thus, community structure detection is of great importance because it can shed light on the relationships between the structural and functional characteristics of networks. Furthermore, a number of research results have provided evidence that networks may have quite different properties when considered from a community perspective rather than from the perspective of individual nodes or a whole network [3,11], and therefore, many interesting network features may be revealed through detecting the community structures from networks.

Community detection has therefore attracted significant interests from researchers, and a large number of community detection methods and algorithms have been developed during the last decade. For example, the GN algorithm [6,7] is a divisive community detection algorithm, the Fast$Q$ algorithm [8] and the CNM algorithm [16] are agglomerative algorithms. Also, Fast$Q$ and CNM are modularity-optimization based algorithms, which take the modularity [7] as the optimization objective, and try to maximize the modularity over all possible community structures of a network. In addition to these, all of the methods in [10–12,17–19] are based on the modularity-maximization strategies. Spectral methods based on the eigenvalue spectra of various types of matrices associated with networks have also yielded fruitful results [10,11,20–26] in discovering community structures from networks. The LPA algorithm [27] exploits a label propagation mechanism to make the densely connected groups of nodes to reach consensuses on node labels to form communities, and a series of variants and improvements [28–30] have been derived from LPA owing to its simplicity and near linear-time complexity. Methods based on random walk utilize the tendency of a random walker to identify community structures from networks, the walker tends to be trapped in communities rather than walks across community boundaries within a limited number of steps. Such methods have also been applied in many applications successfully [31–38]. In which, the Infohiermap (abbreviation for Hierarchical Infomap [36]) algorithm [37], which reveals the best hierarchical community structures in networks by finding the shortest multilevel descriptions of the random walker, and the PPC (acronym for Personalized PageRank Clustering) algorithm [38], which combines the random walks and the modularity to efficiently identify the community structures of networks, are two representatives of the state-of-the-art algorithms based on random walk.

All of these algorithms and methods are in essence a kind of unsupervised learning, meaning that they identify community structures from networks using only topological information of the

networks, without using any prior knowledge of the nodes. However, in many real-world applications, there exists usually some background information, which can be used as the guidance in detecting the communities from networks. The *must-link* and *cannot-link* constraints are one type of such background information, which are also known as the pair-wise constraints that specify whether the nodes involved must or cannot be classified into the same communities. If the relationship between two nodes is "*must-link*", the two nodes must be assigned to the same community. If the relationship is "*cannot-link*", the two nodes cannot be classified into the same community, and they must be allocated into different communities. The *must-link* and *cannot-link* constraints are generally adopted as a type of semi-supervised information and have been successfully integrated in many clustering algorithms to improve their performance. To some extent, the essence of community detection is node clustering in networks. Therefore, it is a natural idea to introduce these constraints to guide the process of community detection. However, this remains a challenge, and the first problem to be addressed is how to obtain the high-quality semi-supervised components. In general, the semi-supervised components are acquired by annotating the data points involved by an oracle (e.g. a domain expert). In order to maximize the utilities of the semi-supervised components at the minimum cost, strategies based on active learning [39] are used to actively select those data points to annotate, such that the clustering algorithm can achieve as a high performance as possible compared with random selection. Most active learning algorithms are pool-based [40,41] or stream-based [41–43], and most work with data represented by attribute vectors [44–46]. However, for the problem of community detection, the nodes in the networks have no other attributes except for the topological information, thus these algorithms cannot be utilized directly. As datasets with intrinsic graph structures become ubiquitous, substantial efforts have been devoted in recent years to the problem of active learning on graphs, and many algorithms [47–50] have been proposed.

The main contributions of this paper are threefold. First, we propose a semi-supervised community detection algorithm, which fully utilizes the *must-link* and *cannot-link* constraints to guide the procedure of community detection to extract high-quality community structures from networks. Next, for being used in the proposed semi-supervised community detection algorithm, we propose an algorithm based on active learning to actively select the nodes with maximum utilities from the networks to generate the *must-link* and *cannot-link* constraints. This active learning algorithm takes both the informative nodes and the nodes with least certainty into account, and thus it can select the nodes with local maximal degrees and the nodes located at the boundaries between the ground truth communities step by step to access a noiseless oracle for generating the *must-link* and *cannot-link* constraints. Finally, we carried out extensive experiments on several real-world networks to evaluate the performance of our proposed method, the experimental results demonstrate that the method can extract high-quality community structures from networks, and outperforms other comparison methods significantly.

Definitions

To facilitate the description of our algorithms, the following notations are given in definition form:

Definition 1 *A network is a graph $G=(V,E)$, where V and E are the node set and the edge set, respectively, and $|V|=n$, $|E|=m$.*

In this paper, we only consider the simple networks as what are involved in the conventional problem of community detection, which means that all of the networks involved are undirected and unweighted graphs, and every edge must connect two different nodes.

Definition 2 *The community structure of a network is a partition $C=\{C_1,C_2,\cdots,C_k\}$ of the network, subject to the conditions $\bigcup_{i=1}^{k} C_i=V$ and $C_i \cap C_j=\phi$, $i\neq j$, where C_i represents the node set of community i ($i=1,2,\cdots,k$), and k is the number of communities.*

Compared with the general concept of a partition in graph theory, another condition, $\sum_{i=1}^{k}|\{(u,v)|\ (u,v)\in E, u\in C_i, v\in C_i\}| \gg \sum_{j=1}^{k}\left|\{(u,v)|(u,v)\in E, u\in C_i, v\in C_j, i\neq j\}\right|$, must be attached to the *community structure*, which indicates that the connections between intra-community nodes are much denser than those between inter-community nodes.

Definition 3 *The must-link constraint set, C_{ML}: $\forall v_i,v_j\in V$, $(v_i,v_j)\in C_{ML}$ indicates that two nodes v_i and v_j must belong to the same community.*

Definition 4 *The cannot-link constraint set, C_{CL}: $\forall v_i,v_j\in V$, $(v_i,v_j)\in C_{CL}$ indicates that two nodes v_i and v_j cannot be classified into the same community, and they must be allocated into different communities.*

As only undirected and unweighted networks are considered in this paper, the tuples in C_{ML} and C_{CL} are order-independent, i.e., $\forall v_i,v_j\in V$, $(v_i,v_j)\in C_{ML}\Rightarrow(v_j,v_i)\in C_{ML}$, and $(v_i,v_j)\in C_{CL}\Rightarrow(v_j,v_i)\in C_{CL}$ also.

Definition 5 *$d(v)$ is the degree of node v, that is, the number of edges associated with node v.*

Definition 6 *For a given node u, $N(u)=\{v|(u,v)\in E\}$ is a set containing all neighbors of node u.*

Definition 7 *$sim(u,v)$ is the similarity measure between two nodes, u and v.*

Definition 8 *The similarity measure between community C_i and node v, denoted as $Sim(C_i,v)$, is formulated as follows:*

$$Sim(C_i,v)=\max_{u\in C_i} sim(u,v),$$

which means that it is defined as the maximal value of similarity between every node in community C_i and node v.

Methods

Semi-supervised community detection algorithm

As mentioned above, the proposal is a semi-supervised algorithm, which makes full utilization of the *must-link* and *cannot-link* constraints to guide the process of community detection. The pseudo-code outlining the procedure of our algorithm is shown as Algorithm 1 in Table 1.

A set of *must-link* constraints define a transitive relation over the nodes involved, and permit additional *must-link* constraints to be derived from the original set, e.g., $(v_i,v_j)\in C_{ML}$, and $(v_j,v_k)\in C_{ML}\ \Rightarrow\ (v_i,v_k)\in C_{ML}$. The *cannot-link* constraints themselves do not have the transitive property, but the combination of *cannot-link* constraints and *must-link* constraints also permits additional *cannot-link* constraints to be inferred, e.g., $(v_i,v_j)\in C_{CL}$ and $(v_i,v_k)\in C_{ML}\Rightarrow(v_k,v_j)\in C_{CL}$.

Thus, in Algorithm 1, we start with the derivations of *must-link* constraints, and enlarge set C_{ML} by adding all derived constraints, which are the functions of *Transitive_Augment*(). Then, from the combination of *cannot-link* constraints and enlarged *must-link* constraints, all additional *cannot-link* constraints are inferred, and

Table 1. Algorithm 1: Semi-supervised community detection algorithm based on *must-link* and *cannot-link* constraints.

Input: $G(V,E)$, the network; C_{ML}, the *must-link* constraint set; C_{CL}, the *cannot-link* constraint set

Output: C, a partition of the network corresponding to the community structure

1: Augment the *must-link* and *cannot-link* sets utilizing the transitive property of *must-link*:

$C_{ML} \leftarrow Transitive\_Augment(C_{ML})$

$C_{CL} \leftarrow Combined\_Augment(C_{CL},C_{ML})$

/ Construct the initial skeleton of the community structure from the cannot-link and must-link constraints */*

2: Initialize set C corresponding to the community structure, and set V_u used to record the unclassified nodes:

$C \leftarrow \phi$

$V_u \leftarrow V$

3: Take every node involved in each *cannot-link* constraint $(v_i,v_j) \in C_{CL}$ as a separate community:

foreach $(v_i,v_j) \in C_{CL}$ **do**

$C \leftarrow C \cup \{\{v_i\},\{v_j\}\}$

$V_u \leftarrow V_u \backslash \{v_i,v_j\}$

4: If some nodes contained in different communities C_i, C_j ($i=1, 2, \cdots, |C|$, $j=1, 2, \cdots, |C|$, $i \neq j$) are involved in some *must-link* constraints, then merge community C_j into community C_i:

for $v_i \in C_i, v_j \in C_j$ **do**

if $\exists (v_i,v_j) \in C_{ML}$ **then**

$C_i \leftarrow C_i \cup C_j$

$C \leftarrow C \backslash \{C_j\}$

/ Expand the communities to obtain the final community structure */*

5: For each community $C_i \in C$, select those unclassified nodes that have *must-link* and transitive *must-link* relationships with the nodes contained in C_i, and insert them into C_i, repeatedly:

foreach $C_i \in C$ **do**

repeat

$M \leftarrow \{v | v \in V_u, \exists u \in C_i, (u,v) \in C_{ML}\}$

$C_i \leftarrow C_i \cup M$

$V_u \leftarrow V_u \backslash M$

until $M = \phi$

6: Among all communities and unclassified nodes, find the most similar pair (C_i,v) from the network greedily globally, and insert the node v into the community C_i first:

$(C_i,v) \leftarrow arg\,max_{C_j \in C, u \in V_u}(Sim(C_j,u))$

$C_i \leftarrow C_i \cup \{v\}$

$V_u \leftarrow V_u \backslash \{v\}$

and then insert the nodes that have *must-link* and transitive *must-link* relationships with node v into community C_i:

$M \leftarrow \{u | u \in V_u, (v,u) \in C_{ML}\}$

repeat

$C_i \leftarrow C_i \cup M$

$V_u \leftarrow V_u \backslash M$

$M' \leftarrow \{u | u \in V_u, \exists u' \in M, (u,u') \in C_{ML}\}$

$M \leftarrow M'$

until $M = \phi$

7: Repeat step 6, until all nodes in the network are processed, i.e., until $V_u = \phi$

8: **return** C

doi:10.1371/journal.pone.0110088.t001

set C_{CL} is augmented by adding all of the inferences, which is conducted using the *Combined_Augment*() function.

For any node pair in the *cannot-link* constraints, the two nodes involved must be classified into different communities, and we therefore use the *cannot-link* constraints to construct the initial skeleton of the community structure. For any *cannot-link* constraint node pair, we create two new communities and insert the two nodes into each of the communities, respectively, i.e., for any node pair $(v_i,v_j) \in C_{CL}$, two communities $\{v_i\}$, $\{v_j\}$ are created. In this way, we obtain many communities with a sole member in each of them. Among all of the communities, two members across two communities may have a *must-link* relationship, e.g., for node $v_i \in C_i$ and node $v_j \in C_j$, $(v_i,v_j) \in C_{ML}$ may exist. According to the definition of the *must-link* constraints, nodes v_i and v_j should be in the same community. Therefore, we merge the two communities involved into one — in Algorithm 1,

community C_j is merged into community C_i using the operation $C_i \leftarrow C_i \cup C_j$. After all cases of this type are processed, we obtain the initial skeleton of the community structure, and those nodes in the initial communities are intended to be seeds or initiators of the corresponding communities.

Then, based on the skeleton of the community structure, we begin to expand the communities. First, if some unclassified nodes (nodes that have not been allocated to any community yet) have *must-link* or transitive *must-link* relationships with some classified nodes (nodes that have already been assigned to communities), the unclassified nodes are allured into joining the communities in which their buddies belong. Concretely, for every community C_i and any node $u \in C_i$, the algorithm selects the unclassified nodes that have *must-link* and transitive *must-link* relationships with u, and inserts them into community C_i.

After all of this type of *must-link* node pairs are processed, a greedy strategy is employed in the next steps: the (*community, unclassified node*) pair (C_i,v) with the largest value of similarity between the community and unclassified node is chosen from all (*community, unclassified node*) pairs, and the algorithm inserts node v into the corresponding community C_i, which means that in each iteration, a global optimal node is selected and assigned to the corresponding community. In the next steps, we find all nodes that have *must-link* and transitive *must-link* relationships with node v, and insert them into community C_i as well. These greedy operations are repeated until every node in the network is classified into the corresponding community, and we finally obtain the resulting community structure.

Active learning algorithm

In this subsection, we present the idea of the proposed semi-supervised component generation algorithm based on active learning. Generally, the semi-supervised components are obtained by annotating the nodes involved by a noiseless oracle. However, in real-world applications, annotating the nodes in networks is a time-consuming job, and it is also very costly. Therefore, the goal of the proposed algorithm is to select those nodes with the maximum utilities for Algorithm 1 to generate the semi-supervised components.

In Algorithm 1, the initial skeleton of the community structure is constructed purely from the *must-link* and *cannot-link* constraints. The nodes involved in the constraint pairs are taken as the seeds or initiators of the communities, and the communities are then expanded by pulling the most similar nodes to join the corresponding community iteratively. From this perspective, the selected nodes should cover all of the ground truth communities and have relative larger degrees, such that the accuracy of community assignments of the nodes can be ensured during the expansion process. However, most of the nodes having a larger degree are the internal nodes of the ground truth communities, and are unlikely to be assigned to a wrong community. The nodes located at the community boundaries tend to be misclassified, but their selection does not facilitate the expansion of the communities. We make a compromise to select those nodes with a relative larger degree and the boundary nodes to generate the *must-link* constraints and the *cannot-link* constraints by accessing the oracle. The basic idea of this active learning algorithm is to extract some nodes with larger degrees in local area into a set and partition the set into some clusters quickly, then to select the nodes having the maximal degree values in each cluster and the nodes having connections with other nodes in other clusters to access the oracle to query the relationship between some pairs of the selected nodes. Which means we try to maximize the utilities of the semi-supervised components by taking both the informative nodes (the nodes with a relative larger degree) and the nodes with least certainty (the boundary nodes) into account during the process of node selection.

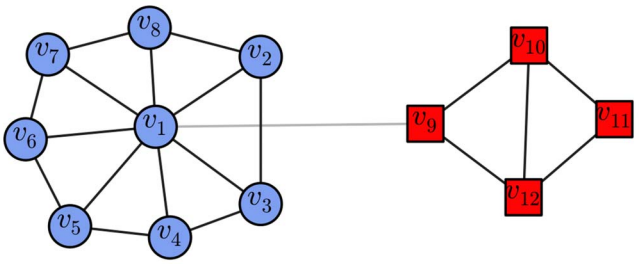


Figure 1. A simple two-community network. If the nodes are selected according to their degree values, only node v_1 will be selected, and community $\{v_9,v_{10},v_{11},v_{12}\}$ will be ignored. However, using the *score* value in conjunction with degree value of every node in the network as the condition, we will select node v_1,v_{10} (or v_{12}) from the network at least, which means that the selected nodes can cover all of the ground truth communities. (The different node shapes and shades indicate different communities, the black lines are the edges within communities, and the light-gray connections represent the edges across different communities. This illustration style is also applied in the following figures.)
doi:10.1371/journal.pone.0110088.g001

Although Algorithm 1 needs nodes with larger degrees to be taken as community seeds to facilitate the expansion of the communities, if we select nodes using only their degrees as a condition, the nodes in small communities will necessarily be ignored. For example, in the simple two-community network illustrated in Figure 1, only node v_1 will be selected according to the values of the node degrees. It is obviously that the selected nodes do not cover all of the ground truth communities. To solve this problem, we calculate a degree-related *score* for every node v in the network using the following formula:

$$score(v) = \sum_{u \in N(v)} \sum_{w \in N(v) \cap N(u)} \frac{1}{d(w)},$$

and the *score* values of nodes are used in conjunction with the degree values of nodes as a condition for node selection.

Concretely, the nodes whose *score* values are larger than a given threshold, ξ, are extracted into a set, *cand*, as candidates firstly, and *cand* is then partitioned into some clusters by calling the function *partition_into_clusters*(). From every cluster, the node with the maximal degree is selected as the representative of that cluster, and the ties are broken by selecting the node with both the maximal degree and the maximal *score* value. In this way, at least nodes v_1 and v_{10} (or v_{12}) will be selected from the network illustrated in Figure 1 after these steps. Using these operations coupled with the following steps, we can ensure that the selected nodes distribute over all of the ground truth communities.

For the selected representatives, we access the oracle to query the relationship between each pair of them, and generate *must-link* constraints or *cannot-link* constraints according to the query results. There may exist other nodes having the same maximal degree with the representative in each cluster, thus we process these nodes in descending order of the degree values of the cluster representatives. From each cluster, we draw out every one of such nodes and access the oracle to query the relationship between the node and the representative of that cluster. If the query result indicates that the relationship is "*cannot-link*", then the relation-

ships between the node and other representatives are queried. If some of the results show that the node and certain representatives have *must-link* relationships, we insert the node into the corresponding clusters. And if all the results are "*cannot-link*"s, then a new cluster is constructed by taking the node as its representative. During this process, the *must-link* constraint set or the *cannot-link* constraint set is updated according to the result of each query. This process is repeated until all nodes having the same maximal degrees with the representative nodes are processed, or until certain user-specified termination criteria, such as the query number limit, *etc.*, are reached. The initial *must-link* and *cannot-link* constraints are then obtained, and the nodes with the maximal degrees in all clusters will cover all of the ground truth communities.

If more constraints are needed, the boundary nodes of the clusters are considered in order of the numbers of nodes contained in the clusters alternately, where the boundary nodes are those having edges connected with nodes located in other clusters. From each cluster, the boundary node with the maximal degree is selected, and the algorithm accesses the oracle to query the relationship between the boundary node and the representative of that cluster. If the relationship is "*cannot-link*", then the relationships between the boundary node and other representatives are queried. If some of the results show the *must-link* relationships between the boundary node and certain representatives, we insert the boundary node into the corresponding clusters. As with the process for nodes with the maximal degrees, during the process for each boundary node, the *must-link* constraint set or the *cannot-link* constraint set is updated after each query. This process is repeated until all boundary nodes are selected, or certain user-specified termination criteria are met. Finally, all *must-link* and *cannot-link* constraints generated are returned and utilized in Algorithm 1.

The steps of the entire procedure are listed as Algorithm 2 in Table 2.

The function *partition_into_clusters*() in Algorithm 2 is responsible for partitioning the candidate node set, *cand*, into some clusters. In this function, we take every node in set *cand* as a cluster first, then merge some clusters repeatedly to obtain the resulting clusters. The logic of this function is described as Algorithm 3 in Table 3.

To achieve the goals efficiently, in Algorithm 3, we first calculate a value, S, for each pair of nodes (u,v) ($u \in cand, v \in cand$) using the following formula:

$$S(u,v) = \begin{cases} \sum_{w \in N(u) \cap N(v)} \frac{1}{d(w)} & (u,v) \in E \\ 0 & \text{otherwise} \end{cases},$$

the value of S takes the role of the *local similarity* between the pair of nodes in *partition_into_clusters*(). Next, for every node in set *cand*, the *most similar neighbors* are identified according to the value of local similarity S. Each node in set *cand* is then taken as a cluster, and that node is the sole member of the corresponding cluster. In the next steps, two clusters are merged into one iteratively, until all nodes in set *cand* are processed. In each merge operation, the nodes contained in one of the two clusters are some of the *most similar neighbors* of the nodes contained in the other cluster. Finally, the set of clusters is returned and used in Algorithm 2.

Similarity measure computation algorithm based on random walk

In Algorithm 1, we expand the communities by selecting the most similar unclassified nodes and inserting them into the corresponding community iteratively. In general, the selected unclassified nodes fall into two categories: nodes having *must-link* relationships with the classified nodes, and nodes having the largest similarity values with the corresponding communities among all of the community and unclassified node pairs. Because of the small number of *must-link* constraints, the vast majority of nodes are pulled to join the communities for the latter reason.

Thus, the similarity between a community and a node plays an important role in our algorithm. According to Definition 8, $Sim(C_i,v)$ is defined as the maximal similarity value between every node in community C_i and node v, and thus we need to compute the similarity $sim(u,v)$ between every pair of nodes, (u,v), in the network beforehand, where $u \in V$ and $v \in V$.

Adapting Algorithm 1, we need the similarity to provide a quantitative metric to measure the closeness between two nodes from the global perspective of the entire network. When the length of the random walks is set properly, a random walker starting from any node can walk through the whole network, and thus the idea of random walk can be used to compute the global similarity between any pair of nodes. Most of the methods based on random walk implicitly utilize the tendency of the walker being trapped in a group of densely connected nodes corresponding to a community by using the probabilistic theory knowledge and matrix operations. In [35], the authors implemented a method directly applying the idea of random walk by actually simulating the process of random walk in a network to compute the similarities between nodes. In this paper, we directly utilize such method to compute the similarity values used in Algorithm 1, the operations of this random walk method are listed as Algorithm 4 in Table 4.

The operations are almost self-explanatory. First, all elements of the similarity matrix *sim* are initialized to be 0. We then take every node in the network as the *start* node to carry out a random walk. During each random walk, we keep track of the visited nodes into set *path*, and at the end of each walk, the similarity value between each pair of nodes in *path* is increased. After all random walks are completed, we finally obtain and return the similarity matrix, *sim*.

Clearly, Algorithm 4 applies to undirected networks only, because we need the walker starting from any node can walk through the whole network in principle. In many directed networks, it is impossible. In addition, the networks should be unweighted networks, or the walker have to consider the influences of the edge weights in each jump. Because the edge weights in different networks have different meanings, this will increase the complexity of the similarity computation. For simplicity, Algorithm 4 does not touch upon the edge weights at all. Therefore, Algorithm 4 applies to unweighted networks only also. This is also the major reason why we only consider the undirected and unweighted networks in this paper.

Evaluation metrics

Although the algorithm can consistently produce a partition of a network, how do we know whether the partition is acceptable as a community structure or not? We need some metrics to measure the quality of the community structure extracted by the algorithm. The *modularity* [7] is the de facto standard at present to measure the strength of a community structure, the *accuracy* and *NMI* (*Normalized Mutual Information*) [51] are two metrics frequently used to assess the performance of clustering algorithms in the fields

Table 2. Algorithm 2: Active approach to generate the *must-link* and *cannot-link* constraints.

Input: $G(V,E)$, the network; ξ, the *score* threshold
Output: C_{ML}, C_{CL}, the sets of *must-link* constraints and *cannot-link* constraints
1: For each node $v \in V$, calculate a *score*:
$score(v) \leftarrow \sum_{u \in N(v)} \sum_{w \in N(v) \cap N(u)} \frac{1}{d(w)}$
2: Extract the nodes whose *score* values are larger than the given threshold, ξ, from V into set *cand* as candidates
3: $K \leftarrow partition\_into\_clusters(G, cand)$
4: Select the node with the maximal degree in each cluster $K_i \in K (i = 1,2,\cdots,\lvert K\rvert)$ as its representative
$r_i \leftarrow arg\ max_{v \in K_i} d(v)$
if more than one node having the same maximal degree exist, the node with the maximal *score* value is chosen
5: Initialize the sets of *must-link* constraints and *cannot-link* constraints:
$C_{ML} \leftarrow \phi$; $C_{CL} \leftarrow \phi$
6: For any two representatives r_i, r_j ($i,j = 1,2,\cdots,\lvert K\rvert$, and $i \neq j$), access the oracle to query their relationship:
$rs \leftarrow query\_from\_oracle(G, r_i, r_j)$
if $rs =$ "*must-link*" then $C_{ML} \leftarrow C_{ML} \cup \{(r_i, r_j)\}$
if $rs =$ "*cannot-link*" then $C_{CL} \leftarrow C_{CL} \cup \{(r_i, r_j)\}$
7: Check each cluster $K_i \in K$ in descending order of the degree values of the representative nodes, and select each node $u \in K_i$ that has the same maximal degree with r_i to query the relationships between u and r_i:
$rs \leftarrow query\_from\_oracle(G, u, r_i)$
• if $rs =$ "*must-link*" then update the *must-link* set: $C_{ML} \leftarrow C_{ML} \cup \{(r_i, u)\}$
• if $rs =$ "*cannot-link*" then update the *cannot-link* set and K_i first:
$C_{CL} \leftarrow C_{CL} \cup \{(r_i, u)\}$; $K_i \leftarrow K_i \setminus \{u\}$
then query the relationships between u and other representatives r_j ($j = 1,2,\cdots,\lvert K\rvert$, $j \neq i$):
$rs \leftarrow query\_from\_oracle(G, u, r_j)$
• if the query result is "*must-link*", update C_{ML} and K_j: $C_{ML} \leftarrow C_{ML} \cup \{(u, r_j)\}$; $K_j \leftarrow K_j \cup \{u\}$
• if the result is "*cannot-link*", update C_{CL}: $C_{CL} \leftarrow C_{CL} \cup \{(u, r_j)\}$
• for all r_j ($j = 1,2,\cdots,\lvert K\rvert$, $j \neq i$), if all query results are "*cannot-link*"s, create a new cluster by taking u as its representative:
$r_{\lvert K\rvert+1} \leftarrow u$; $K \leftarrow K \cup \{\{u\}\}$
8: Repeat step 7, until all nodes having the same maximal degrees in every cluster are processed, or until certain user-specified termination criteria are met
9: If more *must-link* constraints and *cannot-link* constraints are needed, consider the boundary nodes in each cluster $K_i \in K (i = 1,2,\cdots,\lvert K\rvert)$ alternately in order of the number of nodes contained in K_i
• From the remainder nodes in K_i, extract the boundary nodes into set B, where the boundary nodes are those nodes having edges connected with nodes in other clusters
• if $B \neq \phi$, select the node with the maximal degree, denoted as b, from B to query the relationship between b and the representative of K_i:
$rs \leftarrow query\_from\_oracle(G, b, r_i)$
• if $rs =$ "*must-link*", update C_{ML}:
$C_{ML} \leftarrow C_{ML} \cup \{(r_i, b)\}$
• if $rs =$ "*cannot-link*", update C_{CL} and cluster K_i first:
$C_{CL} \leftarrow C_{CL} \cup \{(r_i, b)\}$; $K_i \leftarrow K_i \setminus \{b\}$
then query the relationships between b and other representatives r_j ($j = 1,2,\cdots,\lvert K\rvert$, $j \neq i$):
$rs \leftarrow query\_from\_oracle(G, b, r_j)$
• if $rs =$ "*cannot-link*", update C_{CL}:
$C_{CL} \leftarrow C_{CL} \cup \{(r_j, b)\}$
• if $rs =$ "*must-link*" to certain r_j, update C_{ML} and cluster K_j:
$C_{ML} \leftarrow C_{ML} \cup \{(r_j, b)\}$; $K_j \leftarrow K_j \cup \{b\}$
10: Repeat step 9, until certain user-specified criteria are met
11: **return** C_{CL}, C_{ML}

doi:10.1371/journal.pone.0110088.t002

of data mining and machine learning. To some extent, the essence of detecting a community structure from a network is node clustering, thus using the *accuracy* and NMI to measure the ability of the community detection algorithms also makes sense. Therefore, in this paper, we take all of the three metrics to evaluate the ability of the algorithms.

Table 3. Algorithm 3: *partition_into_clusters*(G,*cand*).

Input: $G(V,E)$, the network; *cand*, the set of candidate nodes
Output: K, the set of clusters
1: For each node pair (u,v), where $u \in cand$, $v \in cand$, calculate the value of S:
$S(u,v) \leftarrow \begin{cases} \sum_{w \in N(u) \cap N(v)} \frac{1}{d(w)} & (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$
2: For each node $u \in cand$, identify its *most similar neighbors* (abbreviated as *msn*):
$maxS \leftarrow \max_{v \in N(u)} (S(u,v))$
$msn(u) \leftarrow \{v \mid S(u,v) = maxS, v \in N(u)\}$
3: Construct the initial clusters K by taking each node $v \in cand$ as a cluster:
$K = \{\{v\} \mid v \in cand\}$
4: Select the node with the largest degree from the unprocessed nodes in *cand* as the *start* node
5: Merge the cluster containing the *start* node and every cluster containing any node in $msn(start)$
6: Take each node in $msn(start)$ as a new *start* node alternately, and repeat step 5 until every new *start* node and $msn(start)$ are in the same cluster
7: Repeat steps 4–6 until all of the nodes in *cand* are processed
8: **return** K

doi:10.1371/journal.pone.0110088.t003

Modularity

As mentioned above, The *modularity*, denoted as Q, is the actual metric at present to measure the quality of a community structure. Let us assume that a network is partitioned into k communities, and define a $k \times k$ symmetric matrix **e**, whose element e_{ij} is the proportion of edges in the network that connect the nodes in community i with the nodes in community j. Further, let us define the row sum of **e** as a_i, i.e., $a_i = \sum_{j=1}^{k} e_{ij}$, which represents the proportion of edges that are incident to nodes in community i. Based on the assumption and definitions, the metric *modularity* is defined as:

$$Q = \sum_{i=1}^{k} (e_{ii} - a_i^2) = \mathrm{Tr}\, \mathrm{e} - \|\mathrm{e}^2\|, \quad (1)$$

where the first term, $\sum_{i=1}^{k} e_{ii}$, is the proportion of edges inside the communities, and the second term, $\sum_{i=1}^{k} a_i^2$, represents the expected value of the same quantity in a random network constructed by keeping the same node set and node degree distribution, but connecting the edges between nodes randomly.

Table 4. Algorithm 4: Similarity computation algorithm based on random walk.

Input: $G(V,E)$, the network; l, the length of the random walks
Output: *sim*, the similarity matrix whose elements are the similarity values between nodes in the network
1: Initialize the similarity matrix:
for $u \in V$, $v \in V$ **do**
$sim(u,v) \leftarrow 0$
2: Take any node in the network as the *start* node to simulate a random walk, and record the visited nodes during the walk in set *path*:
$path \leftarrow \{start\}$
$current \leftarrow start$
for $number\_of\_steps = 1$ **to** l **do**
$next \leftarrow random\_select\_from(N(current))$
$path \leftarrow path \cup \{next\}$
$current \leftarrow next$
3: Increase the similarity value between every pair of nodes recorded in set *path*
for $u \in path$ **do**
for $v \in path\, (v \neq u)$ **do**
$sim(u,v) \leftarrow sim(u,v) + 1$
4: Take any other node as the *start* node, repeat the random walk and similarity-increase operations depicted in steps 2 and 3, until every node in the network is taken as the *start* node to simulate a random walk once
5: **return** *sim*

doi:10.1371/journal.pone.0110088.t004

Such randomness is generally accepted as a network having no significant community structure.

The *modularity* Q measures the quality of a community structure from the perspective of how far it deviates from a random network: the more the value of Q is close to 0, the more the term $\sum_{i=1}^{k} e_{ii}$ is close to $\sum_{i=1}^{k} a_i^2$, which means that the network more approaches a random network, and thus the strength of the community structure is weaker. In contrast, the larger the value of Q is, the further the community structure deviates from a random network, and thus the strength of the community structure is stronger. In practice, values greater than about 0.3 have already indicated significant community structures, and typically fall within the range of [0.3,0.7]. Higher values of Q are rare.

The *modularity* can be computed using only the topological connectivity of the network, without requiring any other information. However, some disadvantages of the *modularity* exist: in [52], the authors found that optimizing the *modularity* in large networks would fail to identify communities that are smaller than a scale, even when the smaller communities is well defined. This is the so-called resolution limit problem. Furthermore, the *modularity* formalizes the goal of attaining high intra-community connectivity and low inter-community connectivity, and is an *internal criterion* for measuring the quality of a community structure. Regarding the internal criterion, it is well known that a good score does not necessarily translate into a good effectiveness in practice. For the *modularity*, a high value of Q does not necessarily correspond to a real community structure, which will be verified through the experimental results.

Therefore, in addition to the *modularity*, we use the *accuracy* and *NMI* to measure the ability of the community detection algorithms.

Accuracy

Compared with the *modularity*, the *accuracy*, denoted as A, is an *external criterion* for evaluating the ability of the community detection algorithms, and is defined as the ratio of the number of nodes classified into the correct communities to the total number of nodes in the network. As mentioned above, community detection is equivalent to node clustering in the network to some extent, thus it is a basic requirement that the nodes be classified into the correct communities. The *accuracy* takes the ground truth community structure as a baseline, and utilizes the ratio to measure the proximity between the extracted community structure and the ground truth community structure, and to measure the ability of the algorithm.

Let us denote the ground truth community structure and the extracted community structure as $C^T=\{C_1^T,C_2^T,\cdots,C_k^T\}$ and $C=\{C_1,C_2,\cdots,C_{k'}\}$, respectively. To compute the *accuracy*, we assign every community $C_i^T \in C^T$ a unique label, which is also assigned to each node $v \in C_i^T$ concurrently as its true label, denoted as *v.label*. We then resolve which community $C_j^T \in C^T$ matches with community $C_i \in C$. To do so, for each community $C_i \in C$, we scan all of the nodes in C_i to count the occurrences of each label in C_i, and take the label occurring most frequently in C_i as the label of community C_i. After this process, some communities may have the same labels. For these communities, we keep the community with the largest number of nodes with the same label, and for each of the other communities, if the nodes in the community have no other labels, that community is removed from C, and all nodes in that community are taken as misclassified nodes; otherwise, we take the next label whose node number is the next-largest in the community as the label of that community. If some communities still have the same labels, this procedure is repeated until every community has a unique label. Then, community $C_i \in C$ and community $C_j^T \in C^T$ with the same label match with each other, and we assign the label of community C_i to each node $v \in C_i$ as its predicted label, denoted as $v.label^*$. Based on the above description, A is defined as

$$A=\frac{\sum_{v \in \bigcup_{i=1}^{|C|} C_i} \delta(v.label,v.label^*)}{n}, \quad (2)$$

where $\delta()$ is the Kronecker delta function.

The *accuracy* A measures how the extracted community structure approaches the ground truth community structure. Obviously, the value of A falls within the range of [0,1], and the more it is close to 1, the more the extracted community structure is close to the ground truth community structure. The ideal scenario is $A=1$, which is corresponding to the result that all nodes in the network are classified into the corresponding communities correctly, so that the extracted community structure is identical to the ground truth community structure.

NMI

NMI is an information-theory based metric, which measures the quality of the extracted community structure from the perspective of the agreement between the extracted community structure and the ground truth community structure, i.e., it also takes the ground truth community structure as a baseline, and thus is also an *external criterion* for measuring the ability of the community detection algorithm.

Taking the frequency counts as approximations of the probabilities, the entropies of the ground truth community structure and the extracted community structure can be represented as $H(C^T)=-\sum_{j=1}^{k}\frac{n_j^{C^T}}{n}\log(\frac{n_j^{C^T}}{n})$ and $H(C)=-\sum_{i=1}^{k'}\frac{n_i^C}{n}\log(\frac{n_i^C}{n})$, respectively, where $n_j^{C^T}=|C_j^T|$, $n_i^C=|C_i|$. The joint entropy of them can be expressed as $H(C,C^T)=-\sum_{i=1}^{k'}\sum_{j=1}^{k}\frac{n_{ij}}{n}\log(\frac{n_{ij}}{n})$, where $n_{ij}=|C_i \cap C_j^T|$, which is the number of shared nodes in C_i and C_j^T. The agreement between the extracted community structure C and the ground truth community structure C^T is measured by the mutual information $I(C,C^T)$, which is defined as follows:

$$\begin{aligned} I(C,C^T) &= H(C)+H(C^T)-H(C,C^T) \\ &= \sum_{i=1}^{k'}\sum_{j=1}^{k}\frac{n_{ij}}{n}\log\left(\frac{\frac{n_{ij}}{n}}{\frac{n_i^C}{n}\cdot\frac{n_j^{C^T}}{n}}\right). \end{aligned}$$

In practice, it is the normalized version of the mutual information that is frequently used to measure the agreement between the extracted community structure and the ground truth community structure, rather than the mutual information itself. It is easy to prove that $I(C,C^T) \le \frac{H(C)+H(C^T)}{2}$, therefore, the normalized mutual information, *NMI*, is defined as follows:

Table 5. Statistical information of the networks.

network	nodes	edges	communities
karate	34	78	2
dolphin	62	159	4
Risk map	42	83	6
collaboration	118	197	6

doi:10.1371/journal.pone.0110088.t005

$$NMI(C,C^T)=\frac{I(C,C^T)}{\frac{H(C)+H(C^T)}{2}} =\frac{-2\sum_{i=1}^{k'}\sum_{j=1}^{k} n_{ij}\log(\frac{n_{ij}\cdot n}{n_i^C\cdot n_j^{C^T}})}{\sum_{i=1}^{k'} n_i^C\log(\frac{n_i^C}{n})+\sum_{j=1}^{k} n_j^{C^T}\log(\frac{n_j^{C^T}}{n})}. \quad (3)$$

Clearly, the value of *NMI* also falls within the range of [0, 1], and the larger the value of *NMI* is, the more the extracted community structure agrees with the ground truth community structure, whereas the smaller the value of *NMI* is, the farther they differentiate from each other, and vice versa.

Results and Discussion

Datasets

In our experiments, we need to evaluate the results both qualitatively and quantitatively, thus the networks used for the evaluation have to satisfy certain criteria: their ground truth community structures must be known a priori, their scales must be sufficiently small to facilitate the interpretation and visualization of the results, and the networks should be publicly available to facilitate the verification of the methods or algorithms. These restrictions resulted in the selection of four real-world networks, i.e., Zachary's karate club network [6–8,53], Lusseau's bottlenose dolphin social network [54], a map used in the board game Risk [35], and a collaboration network of scientists working at the Santa Fe Institute, which is an interdisciplinary research center in Santa Fe, New Mexico [6]. The statistical information of these networks is listed in Table 5.

Using these networks, we carried out two types of experiments: one for testifying the ability of the semi-supervised community detection algorithm based on the *must-link* and *cannot-link* constraints, and the other for demonstrating the utility of the semi-supervised component-generation algorithm based on active learning.

Parameter settings

In Algorithm 2, the *score* threshold, ξ, works as a parameter to control the number of nodes extracted into the candidate set, *cand*. Too large ξ will filter out too many nodes with larger degrees, this will lead to the result that the selected nodes cannot distribute over all of the ground truth communities. On the contrary, too small ξ will extract too many nodes into set *cand*, this will influence the efficiency of partitioning set *cand* into some clusters. In the following experiments, we controlled the value of ξ, so that the nodes whose *score* are among the top 50% of *score* values were extracted into set *cand*, and then *cand* was quickly partitioned into some clusters.

In Algorithm 4, the length of the random walks, l, is also a parameter. In our experiments, we accepted the setting, $l=n$, as what is used in [35], so that the walker starting from any node can reach any other node in the network, theoretically. Therefore, the similarity between any two nodes in the network can be computed.

Experiments on the ability of semi-supervised community detection algorithm

To test the ability of our semi-supervised community detection algorithm, we ran the proposed algorithm on the four networks described above, and compared the results with those of four unsupervised community detection algorithms, Fast*Q*, LPA, Infohiermap, and PPC. For our proposal, the initial skeleton of the community structure is constructed from the *must-link* and *cannot-link* constraints, and as the minimum requirement, the nodes that are selected to generate these constraints should distribute over all of the ground truth communities. Thus, to accommodate this minimum requirement, in these experiments, we controlled the termination criteria of the active node selection approach in Algorithm 2, and selected only the nodes with the maximal degrees in the corresponding clusters to query their relationships. As for LPA, it is a non-deterministic algorithm, running the algorithm on a given network many times may incur different results. We therefore took the method originated in [27] to run the LPA 30 times on every network, and then aggregated these community structures to obtain the resulting structure. But to be frank, the aggregated structure on each network is still non-deterministic, and in the experiments described below, we therefore performed the aggregation operations 20 times on every network, and the aggregated community structure occurring most frequently was taken as the resulting structure of that network.

Zachary's karate club network. This is a well-known benchmark network for testing community detection algorithms. The network is made up of 34 nodes and 78 edges, where every node represents a member of a karate club at an American university. If two members are observed to have social interactions within or away from the karate club, they are connected by an edge. Later, because of a dispute arising between the club's administrator and instructor, the club is eventually split into two factions centered on the administrator and the instructor, respectively. Matched with these two factions, the ground truth community structure is illustrated in Figure 2-(a). Feeding this network into the proposed and comparison algorithms, we obtained the results illustrated in Figures 2-(b), 2-(c), 2-(d), 2-(e), and 2-(f), respectively. The comparison results of the three metrics are listed in Table 6.

To obtain the illustrated results, we controlled the termination criteria in Algorithm 2, such that only nodes "1" and "34" were selected to generate the *must-link* and *cannot-link* constraints by

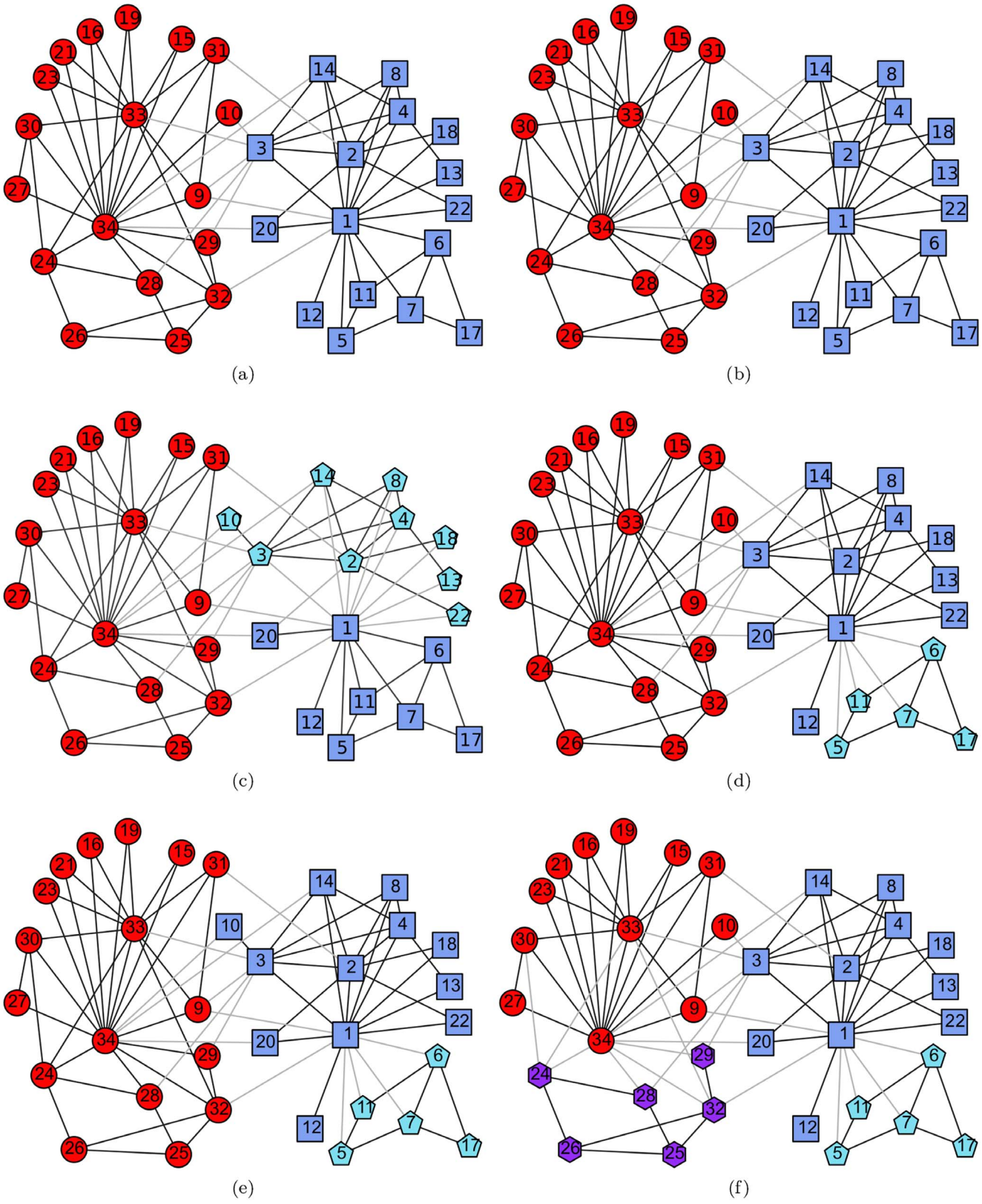


Figure 2. Zachary's karate club network. (a) The ground truth community structure; (b) The community structure extracted by the proposed algorithm; (c) The community structure extracted by Fast*Q*; (d) The community structure aggregated from 30 community structures extracted by LPA; (e) The community structure detected by Infohiermap; (f) The community structure identified by PPC.
doi:10.1371/journal.pone.0110088.g002

Table 6. Comparisons of the 3 metrics: A rank (number in parentheses) is attached to the value of each metric for each network, and the value with the highest rank for each metric on each network is shown in bold.

network	algorithm	*Q*	*A*	*NMI*	rank score	rank
karate	ground truth	0.371	1.00	1.00		
	Fast*Q*	0.381 (4)	0.735 (4)	0.692 (4)	4.00	5
	LPA	0.399 (3) (max: 0.416 average: 0.397)	0.853 (2)	0.826 (2)	2.33	**1**
	Infohiermap	0.402 (2)	0.824 (3)	0.699 (3)	2.67	3
	PPC	**0.42** (1)	0.676 (5)	0.687 (5)	3.67	4
	proposal	0.371 (5)	**1.00** (1)	**1.00** (1)	2.33	**1**
dolphin	ground truth	0.519	1.00	1.00		
	Fast*Q*	0.491 (5)	0.839 (4)	0.733 (5)	4.67	5
	LPA	0.503 (4) (max: 0.526 average: 0.506)	0.823 (5)	0.837 (3)	4.00	4
	Infohiermap	0.525 (2)	0.887 (2)	**0.898** (1)	1.67	2
	PPC	0.519 (3)	0.871 (3)	0.812 (4)	3.33	3
	proposal	**0.526** (1)	**0.935** (1)	0.85 (2)	1.33	**1**
Risk map	ground truth	0.621	1.00	1.00		
	Fast*Q*	0.625 (2)	0.929 (2)	0.894 (3)	2.33	3
	LPA	0.624 (3) (max: 0.634 average: 0.619)	0.81 (4)	0.848 (4)	3.67	4
	Infohiermap	**0.634** (1)	0.857 (3)	0.945 (2)	2.00	**1**
	PPC	0.621 (4)	0.81 (4)	0.803 (5)	4.33	5
	proposal	0.621 (4)	**1.00** (1)	**1.00** (1)	2.00	**1**
collaboration	ground truth	0.739	1.00	1.00		
	Fast*Q*	0.749 (2)	0.831 (3)	0.867 (3)	2.67	3
	LPA	0.681 (5) (max: 0.726 average: 0.678)	0.627 (5)	0.799 (5)	5.00	4
	Infohiermap^{1st}	0.651 (6)	0.636 (4)	0.764 (6)	5.33	6
	Infohiermap^{2nd}	0.704 (4)	0.602 (6)	0.805 (4)	4.67	5
	PPC	**0.751** (1)	0.847 (2)	0.876 (2)	1.67	**1**
	proposal	0.72 (3)	**0.873** (1)	**0.877** (1)	1.67	**1**

Infohiermap^{1st} and Infohiermap^{2nd} represent the first-level and the second-level community structures extracted by the Infohiermap algorithm, respectivly.
doi:10.1371/journal.pone.0110088.t006

accessing the oracle. Clearly, the relationship between this pair of nodes is "*cannot-link*". Based on this constraint, our method identified the correct community structure from this network easily, the result of which is identical to the ground truth community structure. Compared with this, all of the community structures extracted by Fast*Q*, LPA, Infohiermap, and PPC have some deviations from the ground truth. This means that by introducing only the minimum semi-supervised components, we can obtain the best community structure.

It is worth noting that the output of Fast*Q* herein is different from the counterpart described in [8]. In [8], when the value of modularity Q reaches its peak ($Q = 0.381$), the dendrogram agglomerated by the algorithm is cut into two communities correspondingly. However, in our experiments, we carried out the algorithm using a variety of implementations, including conducting the programming ourselves, compiling the source code downloaded from a Web site [55], running the executable file, and calling the function implemented in **igraph** package [56]. All outputs are consistent with that presented herein, i.e., when $Q = 0.381$, the corresponding structure contains three communities, as illustrated in Figure 2-(c), rather than two.

In Table 6, all of the values of Q obtained by Fast*Q*, Infohiermap, PPC, and the maximal, the average and the aggregated values of Q acquired by LPA are larger than that of the ground truth community structure, but all of the corresponding community structures deviate from the ground truth community structure more or less, which confirms one of the shortcomings of the modularity mentioned before.

Lusseau's bottlenose dolphin social network. This is also a famous network widely used as a benchmark to validate community detection algorithms. It contains 62 nodes that represent bottlenose dolphins living in Doubtful Sound, New Zealand, and 159 edges that represent associations between dolphin pairs observed to co-occur more often than expected occasionally. The nodes in this network can be partitioned into four groups, which corresponds to the ground truth community structures illustrated in Figure 3-(a). Running our proposed algorithm and the comparison algorithms on this network, we obtained the results illustrated in Figures 3-(b), 3-(c), 3-(d), 3-(e), and 3-(f). The comparison results of the three metrics are listed in Table 6.

In this network, the nodes with the maximal degrees in the clusters selected by Algorithm 2 are nodes "grin", "topless", "web", "jet", "tr77", and "double". Among them, nodes "grin" and "double" belong in the same ground truth community, as do the pair of nodes "web" and "jet". To meet the minimum requirement that the selected nodes simply cover all of the ground truth communities, we interfered manually to select the node whose degree is larger than the other node in the pair. When the two nodes had the same degree value, the one with the larger

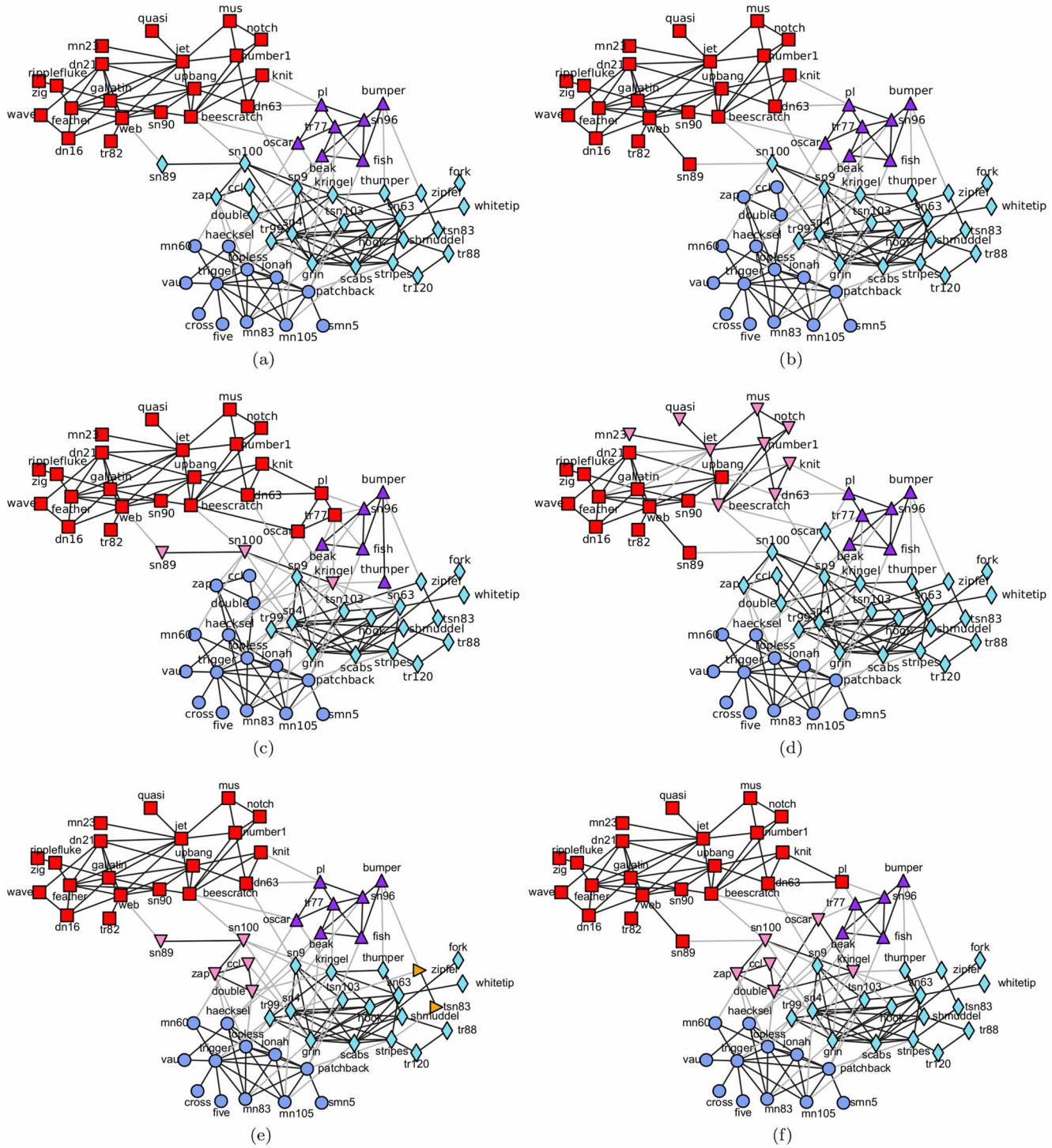


Figure 3. Lusseau's bottlenose dolphin social network. (a) The ground truth community structure; (b) The community structure extracted by the proposed algorithm; (c) The community structure identified by Fast*Q*; (d) The community structure aggregated from 30 outputs of LPA; (e) The community structure detected by Infohiermap; (f) The community structure identified by PPC.
doi:10.1371/journal.pone.0110088.g003

score value was selected. Thus, the semi-supervised components were generated from nodes "grin", "topless", "web", and "tr77" in this experiment, and it is clear that the relationships between their pairs are all "*cannot-link*"s. Compared with the ground truth community structure shown in Figure 3-(a), in the result of our proposed method illustrated in Figure 3-(b), nodes "sn89", "zap", "double", and "ccl" were classified into the wrong communities. The first 3 of them are all located at the community boundaries, and they tend to be classified erroneously. For the misclassifications of nodes "double" and "zap", node "ccl" also becomes a boundary node. Thus, it is easy to understand why they were classified into the wrong communities.

Despite this, from the perspective of the proximity of the community structures identified by the algorithms and the ground truth community structure, the proposed algorithm performs better than the other algorithms. Both the values of Q and A are larger in the proposed algorithm than in the comparison algorithms, and the value of NMI of the proposed algorithm is only smaller than that of the Infohiermap algorithm, but still larger than those of the others. Additionally, along with the increase in the number of selected nodes participating in generating the semi-supervised components, some of the misclassifications will be eliminated, the value of Q will approach that of the ground truth community structure, the values of A and NMI will increase further, all of which are verified in the next type of experiments.

Risk map network. This network is a map of the popular board game Risk, which was invented by Albert Lamorisse and released in 1957 originally. The game can be played by two to six players on a board representing a political map of the Earth, which is divided into 42 territories grouped into 6 continents. Hence, this network is composed of 42 nodes and 83 edges, and all nodes can be partitioned naturally into 6 communities. To eliminate any political sensitivity, we assigned each of the nodes a continuous number instead of the name of the country or territory, the ground truth community structure of which is shown in Figure 4-(a). Taking this network as an input in the proposed and comparison algorithms, the outputs are demonstrated in Figures 4-(b), 4-(c), 4-(d), 4-(e), and 4-(f), individually, and the comparison results of the three metrics are enumerated in Table 6.

In this network, the minimum number of nodes with the maximal degrees in the clusters selected by Algorithm 2 are nodes "5", "36", "24", "31", "40", and "16", and all relationships between every pair of them are *"cannot-link"*s. Based on these constraints, the proposed algorithm yielded the result shown in Figure 4-(b), which is identical to the ground truth community structure. This means that, by utilizing the minimum number of semi-supervised components to guide the community detection procedure, we obtain the best result for this network.

The 6 communities in this network are well separated, but because of the existence of some special nodes, some mistakes tended to be introduced into the results of many algorithms. For instance, node "26" is such a special node, which has 6 edges, but only 2 of them are intra-community connections. For 4 other inter-community edges, 2 of them connect nodes in another community, and 2 other of them are incident to nodes in the third community. Thus, it is hard to say which one of the three communities the node is more intimate with. Similar scenarios occur for nodes "12", "16", and "33". It seems rational that they be classified into any one of the communities that they are associated with, if we do not consider the physical meaning of the nodes in this network. The results produced by the comparison algorithms have certain biases from the ground truth community structure, and most of the mistakes occur around these nodes.

For our proposed method, a special node, "16", was selected to participate in the generation of the semi-supervised components. Because the relationship between nodes "16" and "36" is "*cannot-link*", as is the relationship between nodes "16" and "24", and the similarity values computed by Algorithm 3 indicated that nodes "33" and "34" were more intimate with node "36" than with node "16", and that node "26" was closer to node "24" than to node "16" or node "36", and thus the misclassifications of these nodes were eliminated. The resulting structure identified by our proposed method is already identical to the ground truth community structure, and naturally, the values of the three metrics of our algorithm are superior to those of the comparison algorithms. In fact, if more semi-supervised components are needed, nodes "23", "26", "12", "33", "18", *etc*. will be selected by Algorithim 2 individually to generate the semi-supervised components.

Scientist collaboration network. This network is the largest component of a collaboration network of scientists in residence at Santa Fe Institute. Here, the nodes represent the scientists, and the edges connect those scientists who have coauthored at least one article. This network contains 118 nodes and 197 edges, and can be divided into 6 partitions as its ground truth community structure, which is as presented in Figure 5-(a). Feeding this network into the algorithms, we achieved the final results visualized in Figures 5-(b), 5-(c), 5-(d), 5-(e), 5-(f), and 5-(g), separately. The comparison results of the three evaluation metrics are listed in Table 6.

In this network, nodes "78", "42", "7", "65", "109", "33", "111", and "75" were chosen by Algorithm 2. In the ground truth community structure, nodes "75" and "65" belong to the same community, and as do nodes "109" and "111". To meet the minimum requirement that the selected nodes simply cover all of the ground truth communities, we also manually interfered and selected from the two node pairs the node with the larger degree, i.e., in this experiment, nodes "78", "42", "7", "65", "109", and "33" were selected to generate the semi-supervised components. Apparently, all of the relationships between each pair of nodes are "*cannot-link*"s. Utilizing these constraints, the proposed algorithm extracted the community structure shown in Figure 5-(b).

Compared with the ground truth community structure shown in Figure 5-(a), 15 nodes ("39", "40", "41", "102", "103", "104", "105", "106", "107", "108", "110", "111", "112", "117", and "118") were classified into incorrect communities. Some of them, including nodes "39", "40", "102", "103", "104", and "105", are located at the community boundaries. Among the neighbors of each of them, there exists a node with a very large influence who acts like a center of gravitation. For the first two boundary nodes, node "42" plays this role; and for the latter four nodes, node "78" is the authority. In the random walks passing through those boundary nodes, the walker is more likely to be attracted by these two centers to depart from the communities where the boundary nodes originally belonged, and to be trapped in opposite communities, thus these boundary nodes tend to be misclassified into the opposite communities. Owing to the mistakes this introduces, misclassifications of the other nodes ("41", "107", "108", "106", "110", "111", "117", and "118") are inevitable. Along with the increase in the number of selected nodes, most of these boundary nodes will be taken as the nodes with least certainty and be selected to generate the *must-link* and *cannot-link* constraints, thus the vast majority of their misclassifications will be eliminated, which is verified by the next type of experiments we conducted.

Although, these nodes were misclassified by the proposed algorithm, the resulting structure of the proposed algorithm is the closest to the ground truth community structure compared with the other algorithms. Fast$Q$ took apart two small groups of nodes from two larger communities, and took them as two additional communities; in addition, 8 other nodes ("108", "107", "102", "103", "105", "104", "106", and "112") were also misclassified into the incorrect communities. For LPA, its resulting structure is quite poor, in addition to some nodes being assigned to the incorrect communities, many small groups of nodes were separated from the larger communities, and the resulting structure deviates far from the ground truth community structure. Infohiermap extracted two levels of community structures from this network, the first level contains 3 communities, which is shown in Figure 5-(e), and the second level consists of 16 communities,

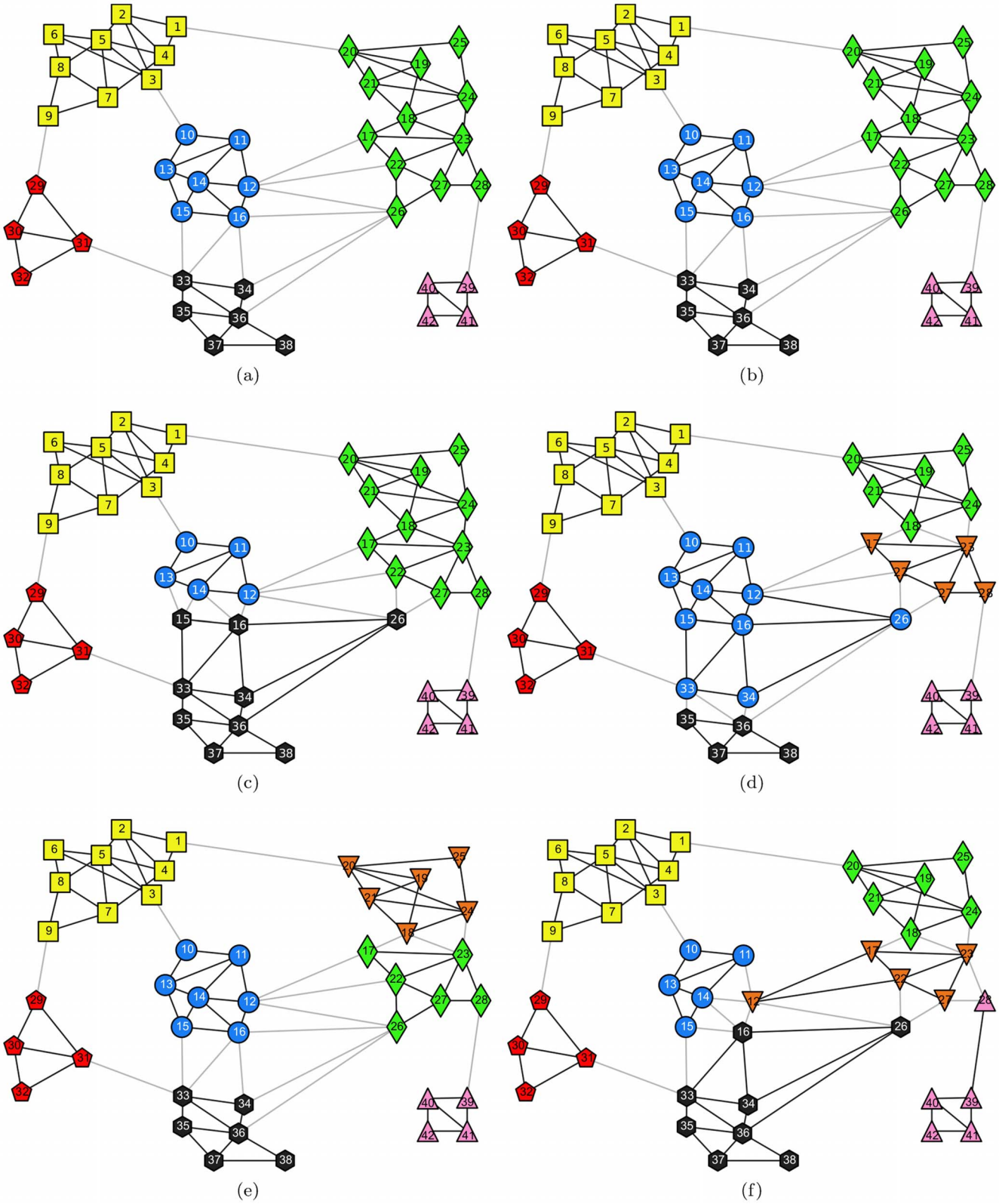


Figure 4. Risk map network. (a) The ground truth community structure; (b) The community structure identified by the proposed algorithm; (c) The community structure extracted by FastQ; (d) The community structure aggregated from 30 outputs of LPA; (e) The community structure detected by Infohiermap; (f) The community structure extracted by PPC.
doi:10.1371/journal.pone.0110088.g004

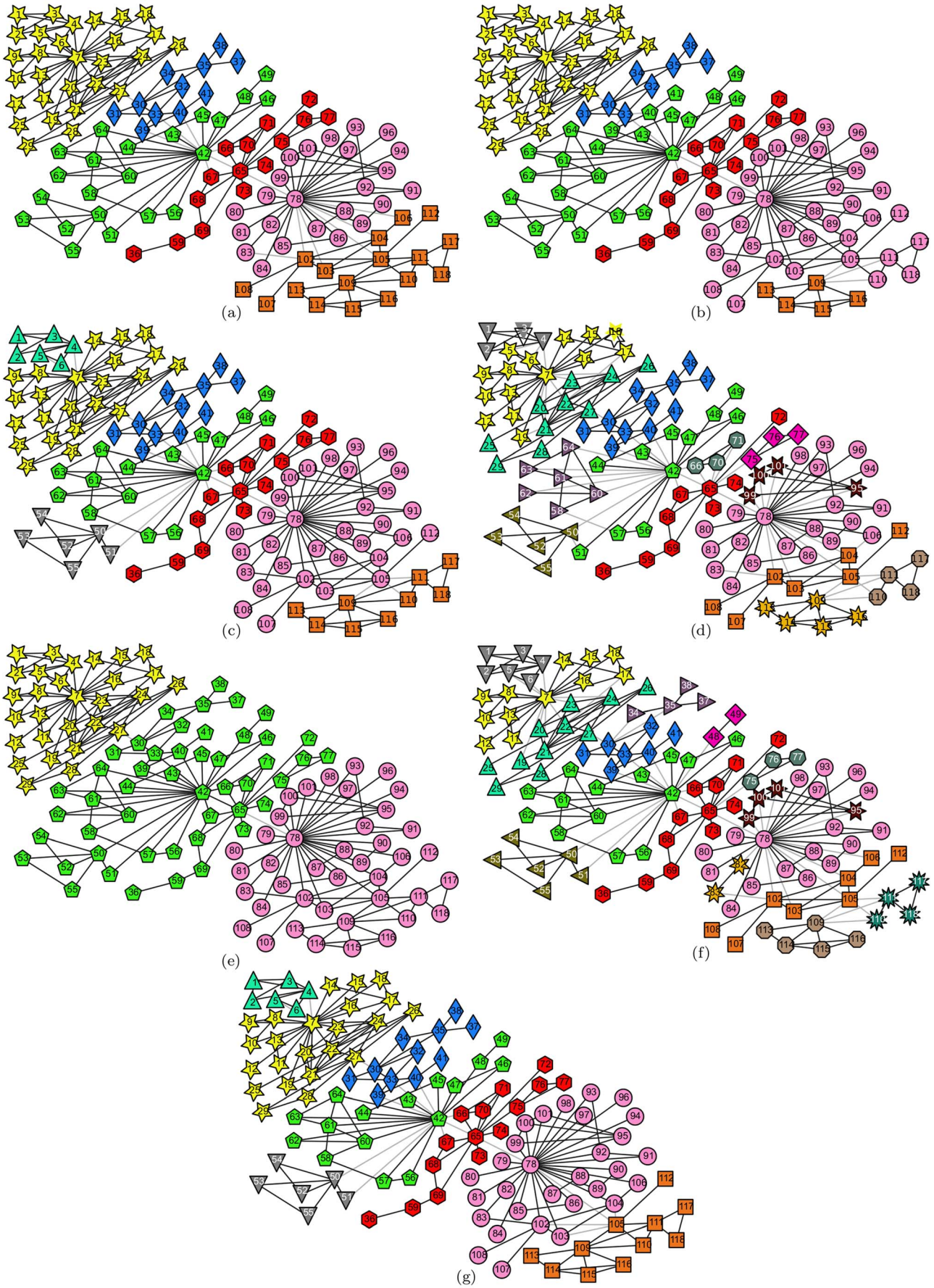

(a)
(b)
(c)
(d)
(e)
(f)
(g)

Figure 5. Collaboration network of scientists at the Santa Fe Institute. (a) The ground truth community structure; (b) The community structure detected by the proposed algorithm; (c) The community structure obtained by Fast*Q*; (d) The community structure aggregated from 30 results of LPA; (e) The first-level community structure extracted by Infohiermap; (f) The second-level community structure extracted by Infohiermap; (g) The community structure identified by PPC.
doi:10.1371/journal.pone.0110088.g005

which is illustrated in Figure 5-(f). Both of them depart far from the ground truth community structure. For PPC, the resulting structure is somewhat similar with that of Fast*Q*, except for the community assignments of nodes "105", and "112". Therefore, it is still not an ideal result.

All values of the three evaluation metrics of the community structures extracted by the proposed algorithm and the comparison algorithms are listed in Table 6. Here, PPC obtained the largest *modularity* twice (on the karate club network and scientist collaboration network), both Infohiermap and the proposed algorithm obtained the largest *modularity* once (on the Risk map network and on the dolphin social network, respectively). However, as discussed above, all of the community structures corresponding to these largest modularities have certain deviations from the ground truth community structures, which verifies one of the previously mentioned shortcomings of the *modularity*. However, the proposed algorithm achieved the largest *accuracy* on all four networks, got the largest *NMI* on three networks and the second largest *NMI* on the other network. When considering the meanings of the *accuracy* and *NMI*, this result indicates that the community structure extracted by the proposed algorithm is the closest to the ground truth community structure, i.e., by introducing only the minimum semi-supervised components, we can obtain the best results. This indicates the effectiveness and significant ability of our proposed semi-supervised community detection algorithm. For another perspective, we attached a rank (the number in the parentheses) to each of the metrics of each network, calculated a score by averaging these ranks for every algorithm, and used the score to rank the algorithms. From the ranks listed in the last column of Table 6, we can confirm that the proposed semi-supervised algorithm is superior to the comparison algorithms in its ability to detect the community structures from networks.

Experiments on the utility of the semi-supervised component generation strategy

In this subsection, we demonstrate the utility of our proposed semi-supervised component generation strategy based on active learning. In Algorithm 2, we loosened the termination criteria step by step, thus the number of selected nodes and then the number of generated semi-supervised components increased gradually. Each time the semi-supervised components were generated, we integrated them in Algorithm 1 as constraints to guide the community detection process. Meanwhile, we applied a random-selection strategy to select an equal number of nodes to generate the semi-supervised components, and then incorporated them in Algorithm 1 to detect the community structure from the network as a comparison. Here, two kinds of random-selection strategies were employed: selecting the nodes from the network completely at random (denoted as "random 1"), and selecting the nodes randomly but ensuring that the selected nodes cover all of the ground truth communities (denoted as "random 2"). When the community structures were extracted from each network, we applied comparisons using Q, A, and NMI to determine which strategy can produce the result closest to the ground truth community structure. In this way, we demonstrated that the proposed semi-supervised component generation strategy based on active learning can actively acquire the *must-link* and *cannot-link* constraints with the maximum utility for the proposed community detection algorithm, thus showing that our proposal is an effective method for extracting high-quality community structures from networks.

As described in the first type of experiments, incorporating only the minimum number of semi-supervised components, the community structures detected from the karate club network and the Risk map network by the proposed method are identical to the ground truth community structures. This means that the experiments effectively demonstrated the utility of the proposed active semi-supervised component generation strategy on these networks, and it was unnecessary to further increase the number of selected nodes. Thus, we conducted the following experiments only on the dolphin social network and the scientist collaboration network.

There are 4 and 6 communities in the ground truth community structures of these two networks, respectively, but as described in the previous subsection, the minimum numbers of nodes selected from these networks by Algorithm 2 were 6 and 8, respectively. In the first type of experiments, to accommodate the minimum requirement that the selected nodes distribute simply over all of the ground truth communities, we interfered manually to choose 4

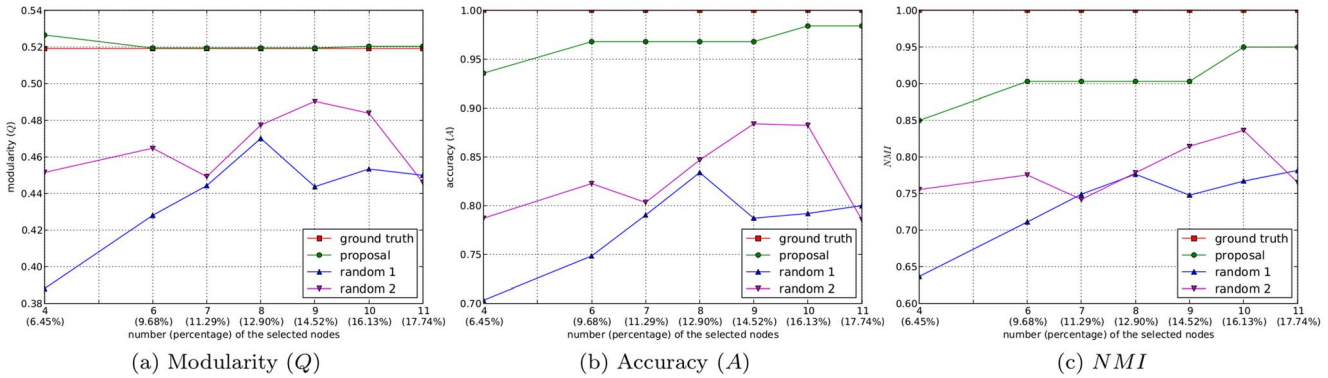
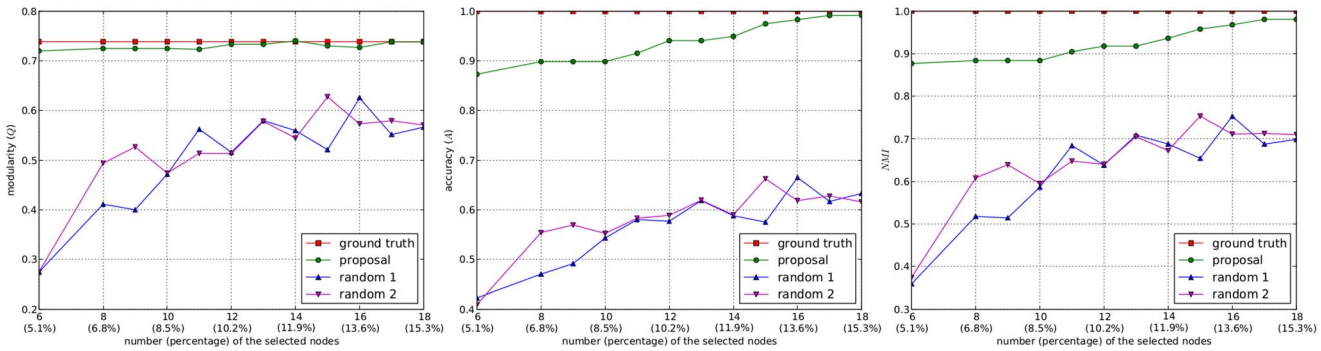


Figure 6. The evolutions of the three metrics on the dolphin social network.
doi:10.1371/journal.pone.0110088.g006



(a) Modularity (Q) (b) Accuracy (A) (c) NMI

Figure 7. The evolutions of the three metrics on the scientist collaboration network.
doi:10.1371/journal.pone.0110088.g007

of the selected nodes from the dolphin social network, and 6 of the selected nodes from the scientist collaboration network, to generate the semi-supervised components.

However, no minimum limit is needed in this type of experiments, and thus we loosened the termination criteria in Algorithm 2 step by step, such that the number of nodes selected to generate the semi-supervised components increased one by one. Each time the semi-supervised components were generated, we incorporated them in Algorithm 1 to obtain the resulting community structure. This process ended when the values of A and NMI no longer increased. For the two random methods, the selected nodes were non-deterministic. To eliminate the occasionality, we repeated the two random methods 10 times each for each number of selected nodes, and took the average values of Q, A, and NMI as the resulting values of the three metrics. In this way, for dolphin social network, we carried out 6 groups of experiments starting from 6 selected nodes, and increased one selected node each time. The evolutions of the values of Q, A, and NMI corresponding to the community structures extracted by the proposed method and the two random methods are shown in Figure 6. For the scientist collaboration network, starting from 8 selected nodes, we carried out 11 groups of experiments by adding one node into the selected node set each time. The evolutions of the values of Q, A, and NMI of the extracted community structures from this network are illustrated in Figure 7. To maintain the completeness of the experiments, the values of the three metrics corresponding to the scenarios that the minimum number limit is met are also plotted in Figures 6 and 7.

In both Figures 6 and 7, all values of Q, A, and NMI of the community structures extracted by the proposed algorithm are significantly larger than the counterparts of the two random methods. For the proposed algorithm, along with the increase in the number of selected nodes, the values of Q approach those of the ground truth community structures, and the values of A and NMI increase steadily. When the number of selected nodes is increased to 10 in the dolphin social network (about $10 \div 62 \approx 16.13\%$ of the total nodes in the network) and to 17 in the scientist collaboration network (about $17 \div 118 \approx 14.4\%$ of the total nodes in the network), the values of A and NMI reach their peaks, and the extracted community structures are almost identical with the ground truth community structures (only 1 node was misclassified in both of the two networks). However, for the two random methods, the values of all three evaluation metrics fluctuate along with the increase in the number of the selected nodes, and even when A and NMI get their peak values, more than 12% of the nodes in the networks still cannot be assigned to the correct communities. These comparisons show that the proposed active learning algorithm can generate the semi-supervised components with the maximum utility from the networks.

Conclusions

In this paper, we introduced active learning into the problem of community detection, and presented a community detection method, which is a combination of a semi-supervised community detection algorithm and a *must-link* and *cannot-link* constraint generation strategy based on active learning. In the semi-supervised community detection algorithm, the skeleton of the initial community structure is constructed from the nodes involved in the *must-link* and *cannot-link* constraints first. The (*community, unclassified node*) pair with the largest similarity value is then identified, and that *unclassified node* and all of its *must-link* and transitive *must-link* partners are inserted into the *community* repeatedly, until all nodes in the network are assigned to the corresponding community. In this way, we obtain the final community structure. To acquire the high-quality *must-link* and *cannot-link* constraints, a semi-supervised component generation algorithm was proposed. We first calculate a *score* value for every node in the network, and the nodes whose *score* values are larger than a given threshold, ξ, are then extracted into a node set from the network. Next, this node set is quickly partitioned into some clusters, and the nodes with the maximal degrees in each cluster, along with the boundary nodes of each cluster, are selected step by step, and the *must-link* and *cannot-link* constraints are finally generated by accessing a noiseless oracle. We also performed extensive experiments on 4 real-world networks, the experimental results illustrate the effectiveness and significant ability of our proposed method.

Author Contributions

Conceived and designed the experiments: JC XC. Performed the experiments: LL HZ. Analyzed the data: ML. Wrote the paper: JC XC.

References

1. Kleinberg J, Lawrence S (2001) The structure of the web. Science 294: 1849–1850.
2. Flake GW, Lawrence S, Giles CL, Coetzee FM (2002) Self-organization and identification of web communities. Computer 35: 66–71.

3. Pan Y, Li DH, Liu JG, Liang JZ (2010) Detecting community structure in complex networks via node similarity. Physica A: Statistical Mechanics and its Applications 389: 2849–2857.
4. Bommarito MJ, Katz DM, Zelner JL, Fowler JH (2010) Distance measures for dynamic citation networks. Physica A: Statistical Mechanics and its Applications 389: 4201–4208.
5. Chen P, Redner S (2010) Community structure of the physical review citation network. Journal of Informetrics 4: 278–290.
6. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99: 7821–7826.
7. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69: 026113.
8. Newman MEJ (2004) Fast algorithm for detecting community structure in networks. Phys Rev E 69: 066133.
9. Qin H, Liu T, Ma Y (2012) Mining user's real social circle in microblog. In: Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on. pp.348–352.
10. Newman MEJ (2006) Modularity and community structure in networks. Proceedings of the National Academy of Science 103: 8577–8582.
11. Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74: 036104.
12. Guimera R, Nunes Amaral LA (2005) Functional cartography of complex metabolic networks. Nature 433: 895–900.
13. Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435: 814–818.
14. Ravasz E, Somera AL, Mongru DA, Oltvai ZN, Barabási AL (2002) Hierarchical organization of modularity in metabolic networks. Science 297: 1551–1555.
15. Lewis A, Jones N, Porter M, Deane C (2010) The function of communities in protein interaction networks at multiple scales. BMC Systems Biology 4: 100.
16. Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70: 066111.
17. Duch J, Arenas A (2005) Community detection in complex networks using extremal optimization. Phys Rev E 72: 027104.
18. Zhang S, Wang RS, Zhang XS (2007) Uncovering fuzzy community structure in complex networks. Phys Rev E 76: 046103.
19. Zhang S, Wang RS, Zhang XS (2007) Identification of overlapping community structure in complex networks using fuzzy c-means clustering. Physica A: Statistical Mechanics and its Applications 374: 483–490.
20. Chauhan S, Girvan M, Ott E (2009) Spectral properties of networks with community structure. Phys Rev E 80: 056114.
21. Arenas A, Díaz-Guilera A, Pérez-Vicente CJ (2006) Synchronization reveals topological scales in complex networks. Phys Rev Lett 96: 114102.
22. Donetti L, Muñoz MA (2004) Detecting network communities: a new systematic and efficient algorithm. Journal of Statistical Mechanics: Theory and Experiment 10: 12.
23. Cheng XQ, Shen HW (2010) Uncovering the community structure associated with the diffusion dynamics on networks. Journal of Statistical Mechanics: Theory and Experiment 2010: P04024.
24. Shen HW, Cheng XQ, Fang BX (2010) Covariance, correlation matrix, and the multiscale community structure of networks. Phys Rev E 82: 016114.
25. van Gennip Y, Hunter B, Ahn R, Elliott P, Luh K, et al. (2012) Community detection using spectral clustering on sparse geosocial data. CoRR abs/1206.4969.
26. Shen HW, Cheng XQ (2010) Spectral methods for the detection of network community structure: a comparative analysis. Journal of Statistical Mechanics: Theory and Experiment 2010: P10020.
27. Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76: 036106.
28. Barber MJ, Clark JW (2009) Detecting network communities by propagating labels under constraints. Phys Rev E 80: 026129.
29. Liu X, Murata T (2010) Advanced modularity-specialized label propagation algorithm for detecting communities in networks. Physica A: Statistical Mechanics and its Applications 389: 1493–1500.
30. Xie J, Szymanski BK (2011) Community detection using a neighborhood strength driven label propagation algorithm. In: Proceedings of the 2011 IEEE Network Science Workshop. Washington, DC, USA: IEEE Computer Society, NSW '11, pp.188–195.
31. Pons P, Latapy M (2006) Computing communities in large networks using random walks. J Graph Algorithms Appl 10: 191–218.
32. Zhou H, Lipowsky R (2004) Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In: Bubak M, van Albada GD, Sloot PMA, Dongarra J, editors, International Conference on Computational Science. Springer, volume 3038 of *Lecture Notes in Computer Science*, pp.1062–1069.
33. Hu Y, Li M, Zhang P, Fan Y, Di Z (2008) Community detection by signaling on complex networks. Phys Rev E 78: 016115.
34. van Dongen S (2000) Graph Clustering by Flow Simulation. Ph.D. thesis, University of Utrecht.
35. Steinhaeuser K, Chawla NV (2010) Identifying and evaluating community structure in complex networks. Pattern Recognition Letters 31: 413–421.
36. Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences 105: 1118–1123.
37. Rosvall M, Bergstrom CT (2011) Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. PLoS ONE 6: e18209.
38. Tabrizi SA, Shakery A, Asadpour M, Abbasi M, Tavallaie MA (2013) Personalized pagerank clustering: A graph clustering algorithm based on random walks. Physica A: Statistical Mechanics and its Applications 392: 5772–5785.
39. Cohn DA, Ghahramani Z, Jordan MI (1996) Active learning with statistical models. Journal of Artificial Intelligence Research 4: 129–145.
40. Settles B, Craven M (2008) An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA, USA: Association for Computational Linguistics, EMNLP '08, pp.1070–1079.
41. Settles B (2010) Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.
42. Dasgupta S, Hsu D, Monteleoni C (2008) A general agnostic active learning algorithm. In: Platt J, Koller D, Singer Y, Roweis S, editors, Advances in Neural Information Processing Systems. MIT Press, Cambridge, MA: MIT Press, volume 20, p. 353–360.
43. Balcan MF, Beygelzimer A, Langford J (2009) Agnostic active learning. Journal of Computer and System Sciences 75: 78–89.
44. Grira N, Crucianu M, Boujemaa N (2008) Active semi-supervised fuzzy clustering. Pattern Recogn 41: 1851–1861.
45. Vu VV, Labroche N, Bouchon-Meunier B (2010) Active learning for semi-supervised k-means clustering. In: 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI). volume 1, pp.12–15.
46. Zhao W, He Q, Ma H, Shi Z (2012) Effective semi-supervised document clustering via active learning with instance-level constraints. Knowledge and Information Systems 30: 569–587.
47. He X (2010) Laplacian regularized d-optimal design for active learning and its application to image retrieval. IEEE Transactions on Image Processing 19: 254–263.
48. Wang X, Davidson I (2010) Active spectral clustering. In: Proceedings of the 2010 IEEE International Conference on Data Mining. Washington, DC, USA: IEEE Computer Society, ICDM '10, pp.561–568.
49. Bilgic M, Mihalkova L, Getoor L (2010) Active learning for networked data. In: Frnkranz J, Joachims T, editors, ICML. Omnipress, pp.79–86.
50. Ji M, Han J (2012) A variance minimization criterion to active learning on graphs. Journal of Machine Learning Research - Proceedings Track 22: 556–564.
51. Ana L, Jain A (2003) Robust data clustering. In: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. volume 2, pp. II-128-II-133 vol.2.
52. Fortunato S, Barthélemy M (2007) Resolution limit in community detection. Proceedings of the National Academy of Sciences 104: 36.
53. Zachary W (1977) An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33: 452–473.
54. Lusseau D, Schneider K, Boisseau O, Haase P, Slooten E, et al. (2003) The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. Behavioral Ecology and Sociobiology 54: 396–405.
55. The code can be downloaded from: http://cs.unm.edu/~aaron/research/fastmodularity.htm. There is a hyperlink on Newman's Web site (http://http://www-personal.umich.edu/~mejn/) connected to that page.
56. http://www.igraph.org/