Online Community Detection for Large Complex Networks

Gang Pan*, Wangsheng Zhang, Zhaohui Wu, Shijian Li

Department of Computer Science, Zhejiang University, Hangzhou, Zhejiang, China

Abstract

Complex networks describe a wide range of systems in nature and society. To understand complex networks, it is crucial to investigate their community structure. In this paper, we develop an online community detection algorithm with linear time complexity for large complex networks. Our algorithm processes a network edge by edge in the order that the network is fed to the algorithm. If a new edge is added, it just updates the existing community structure in constant time, and does not need to re-compute the whole network. Therefore, it can efficiently process large networks in real time. Our algorithm optimizes expected modularity instead of modularity at each step to avoid poor performance. The experiments are carried out using 11 public data sets, and are measured by two criteria, modularity and NMI (Normalized Mutual Information). The results show that our algorithm’s running time is less than the commonly used Louvain algorithm while it gives competitive performance.

Introduction

Complex networks describe a wide range of systems in nature and society [1–3]. Frequently cited examples include the Internet in which routers and computers are connected by physical links, and collaboration networks in which researchers are linked by coauthoring. To understand the formation, evolution, and function of complex networks, it is crucial to investigate their community structure, not only for uncovering the relations between internal structure and functions, but also for practical applications in many disciplines such as biology and sociology [4–6].

Intuitively, a community of a complex network consists of a cohesive group of nodes that are relatively densely connected to each other but sparsely connected to other dense groups in the network [7]. Community detection aims to identify the communities by only using the information encoded in the network topology [8]. It is one of the critical issues in the study of complex networks. A wide variety of community detection methods have been developed to serve different scientific needs [8,9].

Modularity is a commonly used criterion for community detection. It was first proposed in Newman et al. [10]. Good et al. [11] describe the performance of modularity maximization in practical contexts and present a broad characterization of its performance in such situations. A wide variety of algorithms for solving the modularity optimization problem have been developed [12]. For example, Clauset et al. [13] present a hierarchical agglomeration algorithm for detecting communities. Newman et al. [14] show that the modularity can be expressed in terms of the eigenvectors of a characteristic matrix for the network. This expression leads to a spectral algorithm for community detection. Modularity can be generalized in a principled fashion to incorporate the edge information such as direction and weight. Leicht et al. [15] consider the problem of finding communities in directed networks. Newman et al. [16] point out that weighted networks can, in many cases, be analyzed using a simple mapping from a weighted network to an unweighted multigraph. Lancichinetti et al. [9] generate directed and weighted networks with built-in community structure and show how modularity optimization performs on their benchmark. However, Fortunato et al. [17] find that modularity optimization may fail to identify communities smaller than a scale which depends on the total size of the network and on the degree of interconnectedness of the communities, which is called a resolution problem. To mitigate the resolution issue, Reichardt et al. [18] show how community detection can be interpreted as finding the ground state of an infinite range spin glass. Ruan et al. [19] propose a recursive algorithm HQCUT to solve the resolution limit problem. Arenas et al. [20] propose a method that allows for multiple resolution screening of modular structures. Aldecoa et al. [21] introduce a criteria called “Surprise” to resolve the resolution problem.

In some kinds of complex networks, new edges continually appear while old edges do not disappear, resulting in a large network. For example, citation networks are growing as new papers cite existing papers. To efficiently process these kinds of networks, we desire a community detection algorithm that will be able to process a network (1) without recomputing whole network after every new edge/node and (2) without the need of whole network structure available at each update. Although many community detection algorithms have been proposed, to our best knowledge, there is no algorithm that can meet these two
requirements. Many existing algorithms need to start from the beginning when the network is expanded, even when only one node or edge is added.

Many efforts have been made to meet the two requirements. Leung et al. [22] identified novel characteristics and drawbacks of label propagation algorithm, and extended it by incorporating different heuristics to facilitate reliable and multi-functional real time community detection. Huang et al. [23] introduced a new quality function of local community, and presented a fast local expansion algorithm for uncovering communities in large-scale networks. Kawadia et al. [24] presented a new measure of partition distance called estrangement, and showed that constraining estrangement enables it to find meaningful temporal communities in diverse real-world data sets. However, both Leung’s algorithm and Huang’s algorithm cannot handle growing networks, since they must recompute the whole network after every new edge/node. Kawadia’s algorithm requires the whole network structure to be available at each update.

In this paper, we develop a community detection algorithm to meet the two requirements. Our algorithm is an online algorithm, i.e. it can process a network edge by edge in the order that the network is fed to the algorithm, without having the whole network available from the start. Our algorithm updates existing community structure in constant time once a new edge is added. The update avoids re-processing the whole network structure, since it only needs knowledge about a network’s local structure related to the new edge, thus our algorithm can efficiently process large networks in real time. Our algorithm has $O(M)$ time complexity and $O(NK)$ space complexity, where $M$ is number of edges, $N$ is number of nodes, and $K$ is number of communities.

This paper is an extension of our previous work [25] published in IJCAI’13 (downloaded for free in http://ijcai.org/papers13/Papers/ijcai13-281.pdf). The main differences are three-fold: (1) This paper proposes a generative model for complex network based on preferential attachment mechanism, which helps us to infer network’s future structure by its current structure and gives a solid theoretical motivation to the algorithm; (2) This paper develops a deterministic online community detection algorithm, which uses expected modularity to make an informed choice. The conference paper’s non-deterministic algorithm may need many runs; (3) This paper uses additional datasets and extensive experiments for more convincing results.

Method

To achieve the online community detection, we first propose a generative model for complex networks based on the preferential attachment mechanism [26,27], which helps us to predict a network’s future structure based on its current structure. We then develop an online community detection algorithm, which processes a network edge by edge. It optimizes expected modularity instead of modularity to avoid poor performance in some specific cases. Expected modularity can be calculated based on our generative model.

Preliminaries

A network $G=(V,E)$ is a set of $N$ nodes $V=\{v_1, \ldots, v_N\}$ connected by a set of $M$ edges $E=\{e_i=\{v_i, v_j\}\}$. The network considered here is undirected, unweighted, and without self-loops or isolated node. Let $P=\{C_1, \ldots, C_K\}$ denote a partition of $V$. It is a division of $V$ into $K$ non-overlapping and non-empty communities $C_k$ that cover all of $V$. As a performance measure for the partition quality, modularity was first proposed by Newman et al. [28]. It can be expressed as

$$q(P) = \sum_{C_k \in P} \frac{|e_{ij}| - \left(\frac{\deg(C_k)}{2|E|}\right)^2}{|E|}$$

where $|e_{ij}|=|\{v_i, v_j\}\}$ is the number of intra-community edges within community $C_k$, $|E|$ is the number of edges within network $G$, and $\deg(C_k)$ is the degree of community $C_k$, defined as $\deg(C_k)=\sum_{v_i \in C_k} \deg(v_i)$, where $\deg(v_i)$ is the degree of node $v_i$. Hence community detection can be formulated as a modularity optimization problem

$$\max q(P)$$

and Brandes et al. [29] prove the conjectured hardness of this problem both in the general case and in the case with restriction to number of partitions $K$. This result makes heuristic techniques the only viable option for modularity optimization problem. However, heuristic techniques cannot guarantee that the partition is good enough. It may result in a poor partition for some networks. In other words, the algorithms fail to achieve an acceptable modularity. We say an algorithm encounters failure if all nodes are assigned to the same community.

Generative Model for Complex Network

Complex networks have non-trivial topological features that do not occur in some simple networks but often occur in real networks. An important feature of many complex networks is that their degree distributions follow a particular mathematical function called the power law [27,30,31], although it does not always hold [32]. The power law implies that the degree distribution of the network has no characteristic scale.

It is widely recognized as a seminal work presenting a model for the observed stationary scale-free distributions of complex networks by Price et al. [26]. Barabasi et al. [27] conclude that this feature is a consequence of two generic mechanisms: (1) networks expand continuously by the addition of new nodes; (2) new nodes attach preferentially to communities that are already well connected. Barabasi’s model is recognized by academia [33,34]. Specifically, a new node $v_i$ will attach to an existing node $v_j$ with probability $p(v_i)$ in proportion to the degree of node $v_i$

$$p(v_i) \propto \deg(v_i).$$

The above model only considers the case that a new edge links a new node to an existing node. However, a new edge may link two existing nodes or two new nodes. In fact, estimating the likelihood of the appearance of a new edge between two existing nodes, called link prediction, is one of the fundamental problems in network analysis. A variant of preferential attachment mechanism can be used to do link prediction [35]. Specifically, a new edge will link two existing nodes $v_i$ and $v_j$ with probability $p(v_i, v_j)$ in proportion to the product of the degree of node $v_i$ and the degree of node $v_j$

$$p(v_i, v_j) \propto \deg(v_i)\deg(v_j).$$
For a complete review of the statistical mechanics of network topology and dynamics of complex networks, one can refer to Boccaletti et al. [34] or Albert et al. [36]. Mitzenmacher et al. [37] briefly surveyed some other generative models that lead to scale-free distributions. For a summary of recent progress about link prediction algorithms, one can refer to Lu et al. [38].

To facilitate subsequent work, we generalize a preferential attachment mechanism from node to community. A new node will attach to an existing community \( C_k \) with probability \( p(C_k) \) in proportion to the degree of community \( C_k \)

\[
p(C_k) \propto \text{deg}(C_k)
\]

and a new edge will link two existing communities \( C_{k_1} \) and \( C_{k_2} \) with probability \( p(C_{k_1}, C_{k_2}) \) in proportion to the product of the degree of community \( C_{k_1} \) and the degree of community \( C_{k_2} \)

\[
p(C_{k_1}, C_{k_2}) \propto \text{deg}(C_{k_1}) \text{deg}(C_{k_2}).
\]

Here we propose a generative model for complex networks. Our model generates a network \( G \) with \( M \) edges by addition of new edges. It is starting from an empty network \( G_0 = \emptyset \). For \( m = 0, \ldots, M - 1 \), there are three cases for a new edge \( e_{m+1} = \{v_i, v_j\} \) to be added in network \( G_m = \{V_m, E_m\} \), namely:

**Case (a):** link a new node to an existing node, \( \{v_i, v_j\} \cap V_m = \{v_i\} \) or \( \{v_i, v_j\} \cap V_m = \{v_j\} \), with probability \( p_a \);

**Case (b):** link two existing nodes, \( \{v_i, v_j\} \subseteq V_m \), with probability \( p_b \);

**Case (c):** link two new nodes, \( \{v_i, v_j\} \cap V_m = \emptyset \), with probability \( p_c \).

For case (a) and (b), the addition of the new edge follows preferential attachment mechanism mentioned above (See Fig. 1).

Notice that \( p_a + p_b + p_c = 1 \). When \( p_a = 1 \), our model is the same as Barabasi’s model for growing networks.

**Online Community Detection Algorithm**

A straightforward way to do online community detection is to take a sequence of edges as input, and optimize modularity \( q(P_{m+1}) \) at each step for current network \( G_{m+1} \) based on previous partition \( P_m \). However, this greedy algorithm may have poor performance. Considering Barabasi’s model that every new edge links a new node to an existing node, Brandes et al. [39] prove that a partition with maximum modularity has no community that consists of a single node with degree one, and a new node should be assigned to an existing community, however this operation makes all nodes in a same community and results in zero modularity.

To avoid poor performance, our algorithm optimizes expected modularity \( E[q(P_M)] \) for final network \( G_M \), instead of modularity \( q(P_{m+1}) \) for current network \( G_{m+1} \) at each step. We calculate \( E[q(P_M)] \) based on our generative model and the partition as follows: for existing nodes, we keep them in their current communities; for new nodes, we assign them to the corresponding existing communities to keep the degree of every existing community (defined as sum of degree of nodes which belong to that community) increasing and the expected increment of the degree of community is proportional to the degree of community. Such partition can make subsequent deriving of expected modularity simple.

First we calculate \( q(P_{m+1}) \). Notice that \( \sum_{k,m} C_k \cdot \text{deg}(C_k,m) \) can be expressed as

\[
\sum_{k,m} \frac{\text{deg}(C_k,m)}{|E_m|} = q(P_m) + \sum_{k,m} \left( \frac{\text{deg}(C_k,m)}{2|E_m|} \right)^2
\]

Figure 1. Three cases for a new edge to be added in an existing network. (a) linking a new node to an existing node; (b) linking two existing nodes; (c) linking two new nodes.

doi:10.1371/journal.pone.0102799.g001
where $C_k$, $m$ is community $C_k$ at step $m$ and $|E_m|$ is the number of edges within network $G_m$. $|E_m|$ is always equal to $m$ as our algorithm processes one edge at one step. Hence $q(P_{m+1})$ can be expressed as

$$q(P_{m+1}) = \sum_{C_{k,m+1} \in P_{m+1}} \left[ \frac{deg(C_{k,m+1})}{|E_{m+1}|} \right] - \left( \frac{deg(C_{k,m+1})}{|E_{m+1}|} \right)^2$$

$$= \frac{m}{m+1} q(P_m) + \frac{1}{4m(m+1)} \sum_{C_{k,m} \in P_m} deg(C_{k,m})^2$$

$$+ \frac{1}{m+1} \sum_{C_{k,m} \in P_m} deg(C_{k,m}) - \sum_{C_{k,m} \in P_m} deg(C_{k,m})$$

$$- \frac{1}{4(m+1)^2} \sum_{C_{k,m} \in P_m} deg(C_{k,m})^2 - \sum_{C_{k,m} \in P_m} deg(C_{k,m})^2.$$  

Then we calculate $E(q(P_{m+1}))$ under three cases separately as follows:

**Case (a):** link a new node to an existing node. Without loss of generality, we assume $v_i$ is the existing node and $v_j$ is the new node. We assign the new node to the same community as the existing node and have

$$E[q(P_{m+1})] = \sum_{C_{k(i),m} \in P_m} q(P_{m+1}) p(C_{k(i),m})$$

$$= \sum_{C_{k(i),m} \in P_m} q(P_{m+1}) \frac{deg(C_{k(i),m})}{2m}$$

$$= \frac{m+1}{m} q(P_m) - \frac{1}{4m(m+1)} \sum_{C_{k,m} \in P_m} deg(C_{k,m})^2$$

$$+ \frac{m}{m+1}.$$  

where $C_{k(i)}$ is the community which node $v_i$ belongs to.

**Case (b):** link two existing nodes. We do not change the partition and have

$$E[q(P_{m+1})] = \frac{m}{m+1} q(P_m)$$

$$+ \frac{1}{8m^2(m+1)^2} \sum_{C_{k,m} \in P_m} deg(C_{k,m})^2 - \frac{1}{2(m+1)^2}.$$  

**Case (c):** link two new nodes. We assign two new nodes to an existing community with probability in proportion to the degree of the existing community. Case (c)'s $q(P_{m+1})$ and $E[q(P_{m+1})]$ are the same as case (a)'s.

---

Figure 2. Two operations to process a new edge linking a new node to an existing node. (a) A new node attaches to an existing node with degree two, it joins the same community as the existing node; (b) Another new node attaches to the previous new node with degree one, it splits as a new community.

doi:10.1371/journal.pone.0102799.g002
Finally we calculate $E[q(P_M)]$ by combining $E[q(P_{m+1})]$ under three cases together and applying it iteratively

$$E[q(P_M)] = \frac{m+1}{M} q(P_{m+1})$$

$$- \frac{p_a}{4M} \sum_{m'< m < M} \frac{1}{m'(m'+1)} E \left( \sum_{z \in P_{m'}} \deg(C_{k,m'})^2 \right)$$

$$+ \frac{p_b}{8M} \sum_{m'< m < M} \frac{1}{m'(m'+1)} E \left( \sum_{z \in P_{m'}} \deg(C_{k,m'})^2 \right)^2$$

$$- \frac{p_c}{4M} \sum_{m'< m < M} \frac{1}{m'(m'+1)} E \left( \sum_{z \in P_{m'}} \deg(C_{k,m'})^2 \right)^2$$

$$+ f(M,m)$$

where $f(M,m)$ only depends on $M$ and $m$.

As our partition keeps the degree of every existing community increasing, we have

$$\deg(C_{k,m+1}) \leq \deg(C_{k,m}) \leq \deg(C_{k,m+1}) + 2(m'- (m+1))$$

and the expected increment of the degree of community is proportional to the degree of community, thus the expected degree of community $C_k$ at step $m'$ can be expressed as

$$E[\deg(C_{k,m'})] = \frac{m'}{m+1} \deg(C_{k,m+1}).$$

According to the Popoviciu inequality on variance, the variance of $\deg(C_{k,m'})$ has a loose upper bound

$$Var[\deg(C_{k,m'})] \leq [m' - (m+1)]^2.$$
### Table 1. Summary of network data sets.

| Data set    | Number of nodes | Number of edges |
|-------------|-----------------|-----------------|
| ca-CondMat  | 23,133          | 93,439          |
| ca-HepPh    | 12,008          | 118,489         |
| email-Enron | 36,692          | 183,831         |
| ca-AstroPh  | 18,772          | 198,050         |
| cit-HepTh   | 27,770          | 352,285         |
| cit-HepPh   | 34,546          | 420,877         |
| com-Amazon  | 334,863         | 925,872         |
| com-DBLP    | 317,080         | 1,049,866       |
| web-Stanford| 281,903         | 1,992,636       |
| Amazon0601  | 403,394         | 2,443,408       |
| WikiTalk    | 2,394,385       | 4,659,565       |

doi:10.1371/journal.pone.0102799.t001

**Figure 4.** The evolution of temporal modularity over time by OLEM and OLTM.
doi:10.1371/journal.pone.0102799.g004
same community as the existing node (See Fig. 2). Without loss of
generality, we assume \( v_i \) is the existing node and \( v_j \) is the new
node.

For the Split operation, we have

\[
q_{\text{Split}}(P_{m+1}) = \frac{m}{m+1} q(P_m) + \frac{1}{4m(m+1)^2} \sum_{C_k,m=P_m} \text{deg}(C_k,m)^2 - \frac{\text{deg}(C_{k(i),m}+1)}{2(m+1)^2}.
\]

The existing community \( C_{k(i)} \) has degree \( \text{deg}(C_{k(i),m+1}) = \text{deg}(C_{k(i),m}) + 1 \) and the new community \( C_{K+1} \) has degree \( \text{deg}(C_{K+1,m+1}) = 1 \) at step \( m+1 \).

For the Join operation, we have

\[
q_{\text{Join}}(P_{m+1}) = \frac{m}{m+1} q(P_m) + \frac{1}{4m(m+1)^2} \sum_{C_k,m=P_m} \text{deg}(C_k,m)^2 - \frac{\text{deg}(C_{k,i},m)-m}{(m+1)^2}.
\]

We estimate \( p_b \) by observed frequency of case (b). Taking
together and omitting the error term, we can obtain \( \Delta E(q(P_M)) \),
and take the Split operation if it is positive or the Join operation
otherwise.

**Case (b):** link two existing nodes, two existing nodes may or
may not belong to the same community (See Fig. 3). If both nodes
belong to the same community, it is hard to propose a suitable
candidate operation. So, we take the Dense operation where we
keep current partition unchanged. Otherwise we consider two

| Data set       | OLEM     | OLTM     | Louvain  |
|---------------|----------|----------|----------|
| ca-CondMat    | 0.6446   | 0.6585   | 0.7288   |
| ca-HepPh      | 0.5734   | 0.6052   | 0.6549   |
| email-Enron   | 0.5447   | 0.0464   | 0.5876   |
| ca-AstroPh    | 0.5418   | 0.5523   | 0.6149   |
| cit-HepTh     | 0.5885   | 0.6146   | 0.6571   |
| cit-HepPh     | 0.6278   | 0.6771   | 0.7228   |
| com-Amazon    | 0.7050   | 0.7057   | 0.9256   |
| com-DBLP      | 0.7252   | 0.7335   | 0.8091   |
| web-Stanford  | 0.8377   | 0.8702   | 0.9256   |
| Amazon0601    | 0.7785   | 0.4533   | 0.8670   |
| WikiTalk      | 0.5344   | 0.0897   | 0.5831   |

| Data set       | OLEM     | OLTM     | Louvain  |
|---------------|----------|----------|----------|
| ca-CondMat    | 0.3120   | 0.2683   | 0.5029   |
| ca-HepPh      | 0.3344   | 0.2387   | 0.3872   |
| email-Enron   | 0.6396   | 0.4040   | 0.7004   |
| ca-AstroPh    | 0.6334   | 0.5257   | 0.5806   |
| cit-HepTh     | 1.0972   | 0.8502   | 1.0353   |
| cit-HepPh     | 1.3736   | 1.1576   | 1.1707   |
| com-Amazon    | 5.0592   | 4.7429   | 6.6453   |
| com-DBLP      | 4.6914   | 4.3096   | 6.9377   |
| web-Stanford  | 5.8879   | 5.1023   | 32.0137  |
| Amazon0601    | 9.8601   | 8.1857   | 12.7132  |
| WikiTalk      | 25.0043  | 17.6910  | 27.0956  |

Table 2. Modularity by three community detection algorithms.

Table 3. Average running time (in seconds) over 10 runs by three community detection algorithms.
Online Community Detection for Complex Networks

### Table 4. Number of communities by three community detection algorithms and Yang’s labeled community structure.

| Data set     | OLEM | OLTM | Louvain | Labeled |
|--------------|------|------|---------|---------|
| Amazon       | 1,988| 1,979| 217     | 5,000   |
| DBLP         | 4,122| 3,854| 301     | 5,000   |

### Table 5. NMI Benchmark by three community detection algorithms comparing with Yang’s labeled community structure.

| Data set     | OLEM | OLTM | Louvain |
|--------------|------|------|---------|
| Amazon       | 0.7261| 0.7273| 0.3118  |
| DBLP         | 0.2355| 0.2376| 0.1958  |

operations: (1) the Move operation where we move one node from its community to another node’s community; (2) the Keep operation where we keep the current partition unchanged. Without loss of generality, we assume $v_i$ is the moving node and have

\[
\Delta q(P_{m+1}) = q_{Move}(P_{m+1}) - q_{Keep}(P_{m+1})
\]

\[
= \deg(v_i, C_{k(m+1)}) - \deg(v_i, C_{k(m)}) + 1
\]

\[
- \frac{(\deg(v_i) + 1)(\deg(C_{k(m+1)}) - \deg(C_{k(m)}) + \deg(v_i) + 1)}{2(m+1)^2}
\]

where $\deg(v_i, C_{k(m)})$ is number of edges from the node $v_i$ to the community $C_k$ at step $m$ and

\[
\Delta \sum_{C_{k(m+1)} \in P_{m+1}} \deg(C_{k(m)}^2) = 4\deg(v_i)(\deg(C_{k(m)})
\]

\[
- \deg(C_{k(m)}) + \deg(v_i)).
\]

Therefore, we obtain $\Delta E[q(P_M)]$ and determine the operation in the same way as we do in case (a).

**Case (c):** link two new nodes, we consider two operations: the New operation where we assign two new nodes to a new community and the Merge operation where we assign them to an existing community. We have

\[
\Delta q(P_{m+1}) = q_{New}(P_{m+1}) - q_{Merge}(P_{m+1}) = \frac{\deg(C_{k(m+1)})}{(m+1)^2}
\]

where $C_{k(m)}$ is the existing community and

\[
\Delta \sum_{C_{k(m+1)} \in P_{m+1}} \deg(C_{k(m)}^2) = -4\deg(C_{k(m)})
\]

Notice that $\Delta E[q(P_M)]$ is almost always positive for large complex networks. So we take the New operation for case (c) to reduce complexity.

In summary, our algorithm takes a sequence of edges as input and optimizes expected modularity at each step. We assign node to community according to the maximum expected modularity gain principle. If only one node of the current edge belongs to the existing network, we split another node to a new community if this operation can maximize expected modularity gain, otherwise we let it join the same community as the existing node; if both nodes of current edge belong to the existing network but they belong to different communities, we move one node according to the same principle; if neither node of current edge belongs to the existing network, we just assign them to a new community. Obviously, our algorithm has $O(M)$ time complexity. The space complexity is $O(NK)$ because we need to store $\deg(v_i, C_k)$ for calculating expected modularity gain in constant time. Our algorithm has two major advantages: (1) the update only uses knowledge about network’s local structure related to the new edge; (2) the update can be done in constant time. Thus it can efficiently process large networks in real time.

### Results

In this section, we present the experimental results of our online community detection algorithm and compare it with a state-of-the-art algorithm, Louvain algorithm, proposed by Blondel et al. [39]. For simplicity, we use OLEM to refer to our algorithm, OLTM to refer to a simplified version of our algorithm which greedily optimizes temporal modularity $q(P_{m+1})$ (See Eq.(4)) instead of expected modularity $E[q(P_M)]$ (See Eq.(6)), and Louvain to refer to the Louvain algorithm.

The experiments use 11 public real-world large network data sets from Stanford Large Network Dataset Collection (http://snap.stanford.edu/data/), which are commonly used by researchers. Their number of nodes varies from 12,008 to 2,394,383 and their number of edges varies from 93,439 to 4,659,565 (See Table 1). These data sets are

- **ca-CondMat**: Collaboration network of Arxiv Condensed Matter [40];
- **ca-HepPh**: Collaboration network of Arxiv High Energy Physics [40];
- **email-Enron**: Email communication network from Enron [41];
- **ca-AstroPh**: Collaboration network of Arxiv Astro Physics [40];
- **cit-HepTh**: Arxiv High Energy Physics paper citation network [42];
- **cit-HepPh**: Arxiv High Energy Physics paper citation network [40];
com-Amazon: Amazon product network with labeled community structure [43];
com-DBLP: DBLP collaboration network with labeled community structure [43];
web-Stanford: Web graph of Stanford.edu [41];
Amazon0601: Amazon product co-purchasing network from June 1 2003 [44];
WikiTalk: Wikipedia talk (communication) network [45].

The edges should be processed in the same order as expanding procedure of the networks. However, those data sets do not have timestamps on the edges. In the experiments, we process the edges in order of their appearance in the raw files.

We use C# to implement our algorithms (Our C# implementation can be downloaded from http://www.cs.zju.edu.cn/~gpan/code/pone2013.zip). For comparison, we employ the C implementation of the Louvain algorithm provided by the authors (https://sites.google.com/site/findcommunities/). We carry out experiments on a Windows based Genuine Intel (R) CPU i7 @ 2.70 GHz machine with 4.00 GB memory.

Modularity and average running time (in seconds) over 10 runs by OLEM, OLTM, and Louvain are reported in Table 2 and Table 3. The evolution of temporal modularity over time by OLEM and OLTM is shown in Fig. 4.

We can see that OLTM is faster than Louvain in all data sets and OLEM is faster than Louvain in many data sets except ca-AstroPh, cit-HepTh and cit-HepPh. With the modularity measure, OLEM and OLTM cannot achieve similar performance to Louvain. This is due to our algorithms being online one-pass algorithms while Louvain is an offline multi-pass algorithm. Our algorithms’ running times are linear in number of edges as we expected while Louvain is not. This is due to the number of passes of Louvain is not fixed. Most of all, Louvain needs to start from the beginning when a new edge is added while our algorithms do not.

OLTM is faster than OLEM because \( \Delta q(P_{m+1}) \) is simpler than \( \Delta E(P_M) \). In fact, we calculate \( 2(m+1)^2 \Delta q(P_{m+1}) \) instead of \( \Delta q(P_{m+1}) \) in our implementation as the former only involves integer arithmetic which is faster than float-point arithmetic. OLEM keeps relatively stable performance in all data sets while OLTM has exceptionally poor performance in the email-Enron and WikiTalk data sets. We will further investigate the underlying cause for OLTM later. OLTM often performs slightly better than OLEM in the other data sets. It may be due to our approximation of expected modularity by a lower bound in OLEM.

As we mentioned in the Introduction Section, the modularity optimization based approach may fail to identify communities smaller than a scale, which is called a resolution limit problem [17]. To investigate this problem, we compare results of OLEM, OLTM and Louvain in the com-Amazon and com-DBLP data sets. We choose the two data sets because Yang et al. [43] released a labeled community structure for either of the data sets (http://snap.stanford.edu/data/com-Amazon.html, http://snap.stanford.edu/data/com-DBLP.html). For com-Amazon data set, Yang et al. labeled products from the same category as a community and

![Figure 5. The percentage of different operations of OLTM over time.](https://example.com/figure5.png)
nodes (products) that belong to a common community share a common function or purpose. For com-DBLP data set, they labeled authors who published to a certain journal or conference as a community and nodes (authors) that belong to a common community share a common research interest. For each data set, we use the top 5,000 subset, same as [43], for comparison.

We find that, although both our method and the Louvain method optimize the modularity function, the number of communities in Louvain’s result is less than that in our results (See Table 4). It is due to our method and the Louvain method achieving optimization in different ways. The Louvain method optimizes the modularity function by merging pair of communities in each pass, while our method optimizes the modularity function.

Table 6. The statistics of modularity on 10 reordered data sets as well as modularity on original data set by our algorithm.

| Data set       | q(Original) | AVG(q) | MAX(q) | MIN(q) |
|----------------|-------------|--------|--------|--------|
| ca-CondMat     | 0.6446      | 0.5344 | 0.5375 | 0.5298 |
| ca-HepPh       | 0.5734      | 0.5844 | 0.5872 | 0.5823 |
| email-Enron    | 0.5447      | 0.4730 | 0.4872 | 0.4541 |
| ca-AstroPh     | 0.5418      | 0.5468 | 0.5536 | 0.5427 |
| cit-HepTh      | 0.5885      | 0.5777 | 0.5873 | 0.5588 |
| cit-HepPh      | 0.6278      | 0.6376 | 0.6524 | 0.6288 |
| com-Amazon     | 0.7050      | 0.5903 | 0.5916 | 0.5895 |
| com-DBLP       | 0.7252      | 0.5706 | 0.5715 | 0.5694 |
| web-Stanford   | 0.8377      | 0.7431 | 0.7501 | 0.7385 |
| Amazon0601     | 0.7785      | 0.5682 | 0.5708 | 0.5647 |
| WikiTalk       | 0.5344      | 0.5102 | 0.5104 | 0.5101 |

doi:10.1371/journal.pone.0102799.t006

Figure 6. The percentage of different operations of OLEM over time. The height of each color segment represents the percentage of an operation at a certain progress. “Op” is the abbreviation for “Operation” of OLEM at each step.
doi:10.1371/journal.pone.0102799.g006
by moving nodes of the new edge at each step in order to satisfy the real-time processing. Generally speaking, merging communities may obtain higher modularity gain than moving nodes, so the Louvain method is better than our method to optimize the modularity. However, merging communities in each pass will reduce the number of communities in final result as each merging operation will eliminate one community. It causes that the Louvain method will miss small communities.

Further, the similarity between the results and labeled community structures can be measured by NMI (Normalized Mutual Information) criterion [46]. We find that, measured in NMI, our results are more similar to labeled community structure than Louvain’s result (See Table 5). The main reason may be that our methods can find more communities of small scale, which the Louvain method may be hard to identify.

The reason for OLTM’s poor performance in the email-Enron and WikiTalk data sets is that OLTM has no Split operation for case (b) edge. As OLM is a greedy approach, it only takes the Join operation for case (b) edge to maximize temporal modularity. Hence the only way for OLTM to create new community is its New operation for case (c) edge. If a data set has few case (c) edges at its beginning, OLTM cannot create enough communities in the early stage and obtains a poor final partition. In the worst situation, the data set has no case (c) edge and OLTM fails. In fact, email-Enron and WikiTalk data sets have very few case (c) edges at their beginning, comparing with the other data sets.

In contrast, with the help of expected modularity, OLEM can take the Split operation for case (b) edge. Hence it can create enough communities in the early stage and obtains an acceptable final partition in email-Enron and WikiTalk data sets.

To compare OLTM and OLEM’s operations, we plot the percentage of different operations of OLTM and OLEM over time in Fig. 5 and 6. We can see that OLTM generally only takes the Join and Dense operations until very later stage while OLEM takes many Split operations in the early stage in the email-Enron and WikiTalk data sets. Therefore, OLEM’s temporal modularity increases steadily over time while OLTM’s temporal modularity remains zero until very later stages in email-Enron and WikiTalk data sets (See Fig. 4). In fact, OLEM can obtain an acceptable modularity even in early stage for the email-Enron and WikiTalk data sets.

As a statistical analysis, we created 10 copies of each original data set with the edges randomly reordered and ran our algorithm on those reordered data sets. The statistics of modularity in those reordered data sets as well as modularity in original data set are reported in Table 6. Modularity in original data set is significantly better than those in reordered data sets for ca-CondMat, email-Enron, com-Amazon, com-DBLP, web-Stanford, and Amazon0601. We guess, for these six data sets, the storing order of edges may be close to the order of their expanding. For the other data sets, modularity difference between the original and reordered is slight. We think the edges may be not stored by their creation time in those data sets.

Conclusions

In this paper we have examined the problem of online community detection for large complex networks in which new edges continually appear while old edges do not disappear. We have formulated it as a modularity optimization problem. We have proposed a generative model for complex networks and developed an online algorithm with linear time complexity. Our algorithm processes a network edge by edge in the order that the network is fed to the algorithm. It does not optimize modularity but expected modularity to avoid poor performance. The two major advantages of our algorithm are (1) the update only uses knowledge about network’s local structure related to the new edge; (2) the update can be done in constant time. Our algorithm can efficiently process large networks in real time. The algorithm has been applied to 11 public real-world large network data sets and our experiments give very encouraging results. Not only is the proposed algorithm scalable in terms of both time and space complexity, but it also gives comparable performance. Our future research will consider (1) combining OLTM and OLEM into a better one; (2) improving the generative model to allow edge to appear and disappear in general probability distribution; (3) exploring how to apply our method to other objective functions.

Author Contributions

Conceived and designed the experiments: GP WZ. Performed the experiments: WZ. Analyzed the data: GP WZ. Contributed reagents/materials/analysis tools: SL ZW WZ. Wrote the paper: GP WZ.

References

1. Rubinov M, Sporns O (2010) Complex network measures of brain connectivity: Uses and interpretations. NeuroImage 52: 1059–1069.
2. Barrat A, Barthelemy M, Pastor-Satorras R, Vespignani A (2004) The architecture of complex weighted networks. Proceedings of the National Academy of Sciences 101: 3747–3752.
3. Pithakkitnukoon S, Calabrese F, Smoreda Z, Ratti C (2011) Out of sight out of mind–how our mobile social network changes during migration. In: Proceedings of the Third International Conference on Social Computing. pp. 515–520.
4. Fan X, Wong KC, Kui T, Racase T, Gao X (2012) Second: A novel hash seed and community detection based-approach for genome-scale protein domain identification. PLoS ONE 7: e39475.
5. Ratti C, Sobolevsky S, Calabrese F, Andris C, Reades J, et al. (2010) Redrawing the map of great britain from a network of human interactions. PLoS One 5: e14248.
6. Tang C, Li X, Cao L, Zhan J (2012) The law of evolutionary dynamics in community-structured population. Journal of Theoretical Biology 306: 1–6.
7. Porter MA, Onnela JP, Mucha PJ (2009) Communities in networks. Notices of the American Mathematical Society 56: 1082–1097.
8. Fortunato S (2010) Community detection in graphs. Physics Reports 496: 75–174.
9. Lancichinetti A, Fortunato S (2009) Community detection algorithms: a comparative analysis. Physical Review E 80: 056117.
10. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Physical Review E 69: 026113.
11. Good BH, de Montjoye YA, Clauset A (2010) Performance of modularity maximization in practical contexts. Physical Review E 81: 046106.
24. Kawadia V, Sreenivasan S (2012) Sequential detection of temporal communities by estrangement confinement. Scientific Reports 2.
25. Zhang W, Pan G, Wu Z, Li S (2013) Online community detection for large complex networks. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. pp. 1903–1909.
26. Price DDS (1976) A general theory of bibliometric and other cumulative advantage processes. Journal of the American Society for Information Science 27: 292–306.
27. Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286: 569–571.
28. Newman MEJ (2004) Detecting community structure in networks. The European Physical Journal B-Condensed Matter and Complex Systems 38: 321–330.
29. Brandes U, Delling D, Gaertler M, Goerke R, Hoefer M, et al. (2008) On modularity clustering. IEEE Transactions on Knowledge and Data Engineering 20: 172–188.
30. Clauset A, Shalizi C, Newman M (2009) Power-law distributions in empirical data. SIAM Review 51: 661–703.
31. Faloutsos M, Faloutsos P, Faloutsos C (1999) On power-law relationships of the internet topology. SIGCOMM Computer Communication Review 29: 251–262.
32. Stumpf MPH, Porter MA (2012) Critical truths about power laws. Science 335: 665–666.
33. Newman MEJ (2003) The structure and function of complex networks. SIAM Review 45: 167–256.
34. Boccaletti S, Latora V, Moreno Y, Chavez M, Huco DIU (2006) Complex networks: structure and dynamics. Physics Reports 424: 175–308.
35. Xie YB, Zhou T, Wang BH (2008) Scale-free networks without growth. Physica A: Statistical Mechanics and its Applications 387: 1683–1688.
36. Albert R, Barabási AL (2002) Statistical mechanics of complex networks. Reviews of Modern Physics 74: 47.
37. Mitzenmacher M (2004) A brief history of generative models for power law and lognormal distributions. Internet Mathematics 1: 226–251.
38. Lu L, Zhou T (2011) Link prediction in complex networks: A survey. Physica A: Statistical Mechanics and its Applications 390: 1150–1170.
39. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008: P10008.
40. Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. ACM Transactions on Knowledge Discovery from Data 1: 2.
41. Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2009) Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics 6: 29–123.
42. Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the Eleventh International Conference on Knowledge Discovery in Data Mining. pp. 177–187.
43. Yang J, Leskovec J (2012) Defining and evaluating network communities based on ground-truth. In: Proceedings of the Twelfth International Conference on Data Mining. pp. 745–754.
44. Leskovec J, Adamic LA, Huberman BA (2007) The dynamics of viral marketing. ACM Transactions on the Web 1: 5.
45. Leskovec J, Huttenlocher D, Kleinberg J (2010) Predicting positive and negative links in online social networks. In: Proceedings of the Nineteenth International Conference on World Wide Web. pp. 641–650.
46. Lancichinetti A, Fortunato S, Kertész J (2009) Detecting the overlapping and hierarchical community structure in complex networks. New Journal of Physics 11: 033015.