2016 IEEE 28th International Conference on Tools with Artificial Intelligence

# Node-Grained Incremental Community Detection for Streaming Networks

Siwen Yin[1], Shizhan Chen[1], Zhiyong Feng[2], Keman Huang[1], Dongxiao He[1*], Peng Zhao[1], Michael Ying Yang[3]

[1]:School of Computer Science and Technology, Tianjin University, Tianjin, China

[2]: School of Computer Software, Tianjin University, Tianjin, China

[3]: Department of Earth Observation Science, University of Twente, Enschede, the Netherlands

Email: siwenyin@tju.edu.cn; shizhan@tju.edu.cn; zyfeng@tju.edu.cn; keman.huang@tju.edu.cn; hedongxiao@tju.edu.cn; 2013216106@tju.edu.cn; michael.yang@utwente.nl

***Abstract*—Community detection has been one of the key research topics in the analysis of networked data, which is a powerful tool for understanding organizational structures of complex networks. One major challenge in community detection is to analyze community structures for streaming networks in real-time in which changes arrive sequentially and frequently. The existing incremental algorithms are often designed for edge-grained sequential changes, which are sensitive to the processing sequence of edges. However, there exist many real-world networks that changes occur on node-grained, i.e., node with its connecting edges is added into network simultaneously and all edges arrive at the same time. In this paper, we propose a novel incremental community detection method based on modularity optimization for node-grained streaming networks. This method takes one vertex and its connecting edges as a processing unit, and equally treats edges involved by same node. Our algorithm is evaluated on a set of real-world networks, and is compared with several representative incremental and non-incremental algorithms. The experimental results show that our method is highly effective for discovering communities in an incremental way. In addition, our algorithm even got better results than Louvain method (the famous modularity optimization algorithm using global information) in some test networks, e.g., citation networks, which are more likely to be node-grained. This may further indicate the significance of the node-grained incremental algorithms.**

***Keywords*—*Community Detection; Complex Network; Incremental Algorithm; Modularity.***

## I. INTRODUCTION

Complex systems, widely studied in many scientific fields, e.g., biology, social science and engineering, can be represented as networks, where elementary units of a system and mutual interactions between them are represented as nodes and edges respectively[1, 2]. Real-world networks which represent complex systems have some special properties, one of which is local inhomogeneity of edge distribution, i.e., high density of edges within some groups of nodes while low density of edges between different groups. This is so-called community structure [3]. So far, there is no widely accepted accurate definition for community structure. Generally, communities are considered as groups of nodes in which there are edges connecting nodes, while between which there are few edges [4].

Community structure indicates that individuals within a community have common or similar properties/functions, or play similar roles, while individuals in different communities usually have significant dissimilarity [5]. Thus it is helpful for discovering organizational structures and underlying features of networks. Furthermore, it provides rich information for studying existing networks and powerful help for mining some uncovered parts of the networks such as link prediction. Community detection has been successfully used in many applications, e.g., event detection, topic detection, protein interaction analysis and terrorist organization recognition.

Most real networks are not static but often change frequently over time. These frequent changes are usually represented as streaming networks, which require *real-time* analytical methods which can process the network incrementally and update the community structure in time. The existing incremental methods [6,7,8,9,10] usually take the evolution of large scale networks as a sequence of additions of edges, i.e., modeling a streaming network as edge-grained sequential changes which is very sensitive to the processing sequence of edges. However, there is another type of changes which happens on the level of nodes: a single node with its connecting edges is added into the network simultaneously, and all the edges are added at the same time. There is no reason to assign them an assumed adding sequence. For example, in citation networks in which node represents article and edge represents citation between two articles, a network was expanded when an article with some citations was published. These citations happened at the same time, and hence there is no reason assuming one happened after another. Webpage networks take web-pages as nodes and hyperlinks as edges. The node-grained change took place when a brand new webpage was deployed, i.e., a new node (representing webpage) and all its connecting edges (representing the hyperlinks of this webpage) were added into the network at the same time. To the best of our knowledge, there is no special node-grained incremental community detection methods designed for streaming networks.

In this paper, a **N**ode-**G**rained **I**ncremental community detection algorithm, namely NGI, is proposed to handle the frequent node-grained addition which involves the simultaneous additions of edges. This algorithm is based on the optimization of modularity. It equally treats the edges arrived at the same time, and updates community structures in *real-*

*Corresponding Author: Dongxiao He, hedongxiao@tju.edu.cn

2375-0197/16 $31.00 © 2016 IEEE
DOI 10.1109/ICTAI.2016.92

*time* when node-grained changes arrive. The experimental results showed the applicability of our method for processing networks changing frequently, and showed its superior performance over competing incremental and non-incremental methods in terms of both community quality and efficiency. Besides, our nearly almost one-pass algorithm got better results than the well-known static algorithm Louvain [2] (using global information and iterating multi-times) on many citation and Web networks. This indicates specific characteristics of node-grained streaming networks.

## II. Related Work

Modularity was proposed by Newman and Girvan [11] which is built on the comparison between network with community structure and a random graph (also called null model) without this property [3].

Let $G$ be a given graph $G=\{V,E\}$, whose vertex set is denoted by $V$ and edge set is denoted by $E$. The modularity of a community partition $P=\{C_1, C_2,..., C_K\}$, where $C_k$ denote one of community $k$, $k$=1,...,$K$ and $\forall C_\lambda$, $C_\mu \in P$, $C_\lambda \cap C_\mu = \varnothing$, $V=\cup_1^K C_k$, is defined as :

$$Q(P)=\sum_{C_k\in P}\left(\frac{\mathrm{edg}(C_k)}{m}-\left(\frac{\deg(C_k)}{2m}\right)^2\right) \quad (1)$$

where edg($C_k$) denotes the number of intra-community edges in community $C_k$ , deg($C_k$) denotes the sum of degree of every vertex in community $C_k$, and $m$ is the total number of edges in $C$. The modularity of an identified community partition is a scalar value between -1 and 1 that measures the density of edges within communities compared to the density of edges between different communities [3]. Hence, a partition with higher modularity on a given graph is thought to be better.

Finding a partition corresponding to the maximum value of modularity on a given graph is a NP-hard problem due to the space of possible partitions grows quite fast [12]. Many heuristic methods for maximizing modularity have been proposed, including fast greedy method by Newman [11], CNM algorithm [14], Eig algorithm [15], method by Xie[16], algorithm by Wakita and Tsurumi [14], and Louvain algorithm [2] which is considered as state-of-the-art. However, all of these methods were designed for static networks, and need to use whole network topology to find community structure. When changes happened on a network, these methods have to be re-performed on changed network topology so as to discover community structure for the changed network [18,19]. Hence, the time cost will grow rapidly and it become intractable as network changes frequently.

In order to analyze networks that change frequently in *real time*, many incremental community detection methods were proposed. They updated community structure by using local information around the change and the priori community information, which do not need to re-perform a community detection algorithm like for processing a brand new network and can avoid unnecessary computing. There is no doubt that incremental method is more suitable for discovering community structures in streaming networks with frequent changes.

Here we summarized some recent modularity-based incremental community detection algorithm. Nguyen's QCA method [11] handles four types of changes including node addition/deletion and edge addition/deletion, but it transfers node addition/deletion into sequence of edge changes which is very sensitive to the processing order of edges. Shang's GreMod method [7] uses similar processing technique in certain simple scenarios, i.e., intra/inter edge addition, while leaves node addition unhandled. A disadvantage of Shang's method is that it needs an initial community partition produced by non-incremental community detection methods as its input, and this also leads to another drawback, i.e., its results are influenced by the initial partition and the order in which edges are processed. Pan's OLTM [8] and Zhang's OLEM [10] are designed for processing an edge-grained network stream by optimizing modularity and expected modularity respectively. These traditional modularity-based incremental algorithms have a common drawback, i.e., being very sensitive to the processing order of edges, which may lead to unstable results. More importantly, they cannot handle node-grained changes well because they impose an assumed order for edges involved by a node addition which violates the fact these edges are added simultaneously, thus poor performance may occur when they are applied on networks growing in node-grained.

## III. Methods

In this paper, we aim to propose an algorithm for the node-grained streaming network which has two characteristics: one is network expands node-by-node, the other is multiple edges is added simultaneously and should be treated equally. Assuming a node stream containing $N$ elementary units $\varphi_1$, $\varphi_2$,..., $\varphi_N$ where $\varphi_n$, $n$=1,..., $N$ is defined as an elementary incremental unit including a single vertex with its connecting edges,

$$\varphi_n=\{v_n, Links=\{e_{nz}\,|\,z<n,\ e_{nz}\in E\}\}$$

In order to analyze this type of streaming network in *real-time*, we will update community structure as each node is added, by analyzing the local topology of newly added nodes $v_n$ and current community partition so as to maximize modularity.

### A. The Proposed Method

#### 1) Formalization

Assuming a network denoted as $G_T=\{V_T, E_T\}$, $V_T$ and $E_T$ denotes its vertex set and edge set at time $T$ respectively, the corresponding modularity is,

$$Q(P_T)=\sum_{C_{k,T}\in P_T}\left(\frac{\mathrm{edg}(C_{k,T})}{m}-\left(\frac{\deg(C_{k,T})}{2m}\right)^2\right) \quad (2)$$

where $C_{k,T}$ and $P_T$ denotes the community $C_k$ and community partition at time $T$ and $m$ denotes the total number of edges of network at time $T$.

At time $T$+1, i.e., after $v_{T+1}$'s arriving, the modularity is,

$$Q(P_{T+1}) = \sum_{C_{k,T+1} \in P_{T+1}} \left( \frac{\mathrm{edg}(C_{k,T+1})}{m'} - \left( \frac{\mathrm{deg}(C_{k,T+1})}{2m'} \right)^2 \right) \quad (3)$$

where $m'$ denote the total number of edges in the new network. Let $h=|\varphi_{T+1}.Links|$, then $m'=m+h$.

At first, modularity gain from time $T$ to time $T$+1 is,

$$\Delta Q = Q_{T+1}(p_{T+1}) - Q_T(p_T)$$

$$= \frac{1}{m+h} \left\{ \sum_{C_{k,T+1} \in P_{T+1}} \mathrm{edg}(C_{k,T+1}) - \sum_{C_{k,T} \in P_T} \mathrm{edg}(C_{k,T}) \right\}$$

$$- \frac{1}{4(m+h)^2} \left\{ \sum_{C_{k,T+1} \in P_{T+1}} \mathrm{deg}(C_{k,T+1})^2 - \sum_{C_{k,T} \in P_T} \mathrm{deg}(C_{k,T})^2 \right\}$$

$$+ \left( \frac{1}{m+h} - \frac{1}{m} \right) \left\{ \sum_{C_{k,T} \in P_T} \mathrm{edg}(C_{k,T}) \right\}$$

$$+ \left( \frac{1}{4m^2} - \frac{1}{4(m+h)^2} \right) \left\{ \sum_{C_{k,T} \in P_T} \mathrm{deg}(C_{k,T})^2 \right\} \quad (4)$$

let

$$\Theta_1 = \left( \frac{1}{m+h} - \frac{1}{m} \right) \left\{ \sum_{C_{k,T} \in P_T} \mathrm{edg}(C_{k,T}) \right\} \quad (5)$$

$$\Theta_2 = \left( \frac{1}{4m^2} - \frac{1}{4(m+h)^2} \right) \left\{ \sum_{C_{k,T} \in P_T} \mathrm{deg}(C_{k,T})^2 \right\} \quad (6)$$

$$\Theta_3 = \sum_{C_{k,T+1} \in P_{T+1}} \mathrm{edg}(C_{k,T+1}) - \sum_{C_{k,T} \in P_T} \mathrm{edg}(C_{k,T}) \quad (7)$$

$$\Theta_4 = \sum_{C_{k,T+1} \in P_{T+1}} \mathrm{deg}(C_{k,T+1})^2 - \sum_{C_{k,T} \in P_T} \mathrm{deg}(C_{k,T})^2 \quad (8)$$

then a concise expression for modularity gain was attained,

$$\Delta Q = \Theta_1 + \Theta_2 + \frac{1}{m+h}\Theta_3 - \frac{1}{4(m+h)^2}\Theta_4 \quad (9)$$

Note that in $\Theta_1$(5) and $\Theta_2$ in (6) only depend on the snapshot of time $T$. Therefore, they can be calculated in constant time if edg($C_k$) and deg($C_k$) are stored and updated as network changes, and the calculations of $\Theta_3$ in (7) and $\Theta_4$ in (8) are also fast and simple by using local topology.

*2) Local Topology Analysis*

In this section, we will introduce different strategies for updating community structure when node $v_{T+1}$ arrives, so as to maximize $\Delta Q$ .

According to the local topological characteristic of newly arrived nodes, we adopt different strategies. Here we classify the local topology of new node into 3 cases as shown in TABLE I., where topology-specific designed operators for each case are also shown.

TABLE I. TOPOLOGICAL CLASSIFICATION AND OPERATORS

| | Topological Characteristic | Operator |
|---|---|---|
| **Case 1** | Without any edge | ISOLATE |
| **Case 2** | All neighbors are isolating singleton community | AGGREGATION |
| **Case 3** | Having adjacent community with multiple vertices | INSERT,MERGE |

**Case 1**: When new node joined the network without any connected edge, i.e., $\varphi_{T+1}.Links=\emptyset$. It is obvious that it is not proper to let the new coming node join any community, and the only reasonable way based on current information is isolating it as a new community. Operator ISOLATE is adopted.

**Case 2**: For the case that all of the adjacent communities of new node are singletons (i.e., only containing one node), we aggregate all of them with the newly arrived node into a community, considering synchronicity of all adjacent single vertices and a basic regulation that a partition with maximum modularity has no community including a single node with degree one [10].

**Case 3**: For new node without above characteristics, we employ two operators (INSERT and MERGE) to update community partition. At first, according to a widely accepted generative mechanism, i.e., networks expanded continuously through the addition of new nodes and new nodes attached preferentially to communities which are already well connected [20], we define INSERT operator by letting the new node join one of its adjacent communities. Moreover, considering that community structures in real-world networks are always hierarchical, it is reasonable to provide MERGE operator which merges two adjacent communities with new coming node together when two communities are similar enough.

*3) Operators*

In this section, we will show how does each operator update community partition and how does the corresponding modularity gain $\Delta Q$ for each operator be calculated according to (9). As mentioned above, $\Theta_1$ and $\Theta_2$ in (9) only depend on the snapshot of time $T$ and can be calculated easily, we will mainly introduce how to calculate $\Theta_3$ and $\Theta_4$ for each operator.

| **Definition 1** | ISOLATE Operator **ISO**($\varphi_n$) |
|---|---|
| | $V = V \cup \varphi_n.v$<br>$C_{new} = \{\varphi_n.v\}$<br>$P = P \cup C_{new}$ |

Since $h=0$ in case 1, thus we got $\Theta_1=0$, $\Theta_2=0$ by using (5) and (6). And according to (7) and (8), we got:

$\Theta_3 = \text{edg}(C_{new}) = 0$ and $\Theta_4 = \deg(C_{new})^2 = 0$.
Thus, according to (9), the modularity gain $\Delta Q = 0$.

| **Definition 2** | AGGREGATION Operator **AGG**($\varphi_n$) |
|---|---|
| | **For each** $e_{zn}$ in $\varphi_n.Links$ and $v_z$ is isolated<br>$C_{new} = C_{new} \cup \{v_z\}$<br>$P = P/\{v_z\}$<br>**End for**<br>$P = P \cup C_{new}$ |

When $C_{new}$ is created, $\text{edg}(C_{new})=h$ and $\deg(C_{new})=2h$, thus according to (7) and (8), we got:$\Theta_3 = h$ and $\Theta_4 = 4h^2$.

In the following, we use $y_s$ denoted the number of links between the new node and community $Cx_s$, $s$=1,...,$S$ and each $x_s$ denotes one of community number among $S$ adjacent communities of new node respectively.

| **Definition 3** | INSERT Operator **IST**($\varphi_n$, $r$) |
|---|---|
| | $C_r = C_r \cup \{\varphi_n.v\}$<br>**for each** neighbor $v_{nb}$ of $\varphi_n.v$<br>**if** $C_r$ has most neighbour of $v_{nb}$ **then**<br>$C_r = C_r \cup \{v_{nb}\}$<br>**end for** |

Supposing community $C_r$ is the community joined by new node. The number of inner-community edge of community $C_r$ is increased by $y_r$, and the total degree of community $C_r$, i.e., $\deg(C_r)$, is added by both the degree of new node and the increased degree of other vertices in $C_r$ which is also equal to $y_r$. And for other adjacent communities $C_{k'}$, $k'=x_1,...,x_s$, $k' \neq r$ the degree gain is $y_{k'}$ while there is no inner-community edge being added. Thus, according to (7) and (8), we got:

$$\Theta_3 = y_r \text{ and}$$

$$\Theta_4 = 2\sum_S \deg(C_s) \cdot y_s + \sum_S y_s^2 + 2 \cdot h \cdot \deg(C_r) + 2 \cdot h \cdot y_r + h^2$$

In addition, considering the local topology of neighbors of the new node may be modified as well, we did a fast, effective adjustment for neighbors of the new node, i.e., letting each neighbor in one hop join the community which has most members among its own neighbors.

| **Definition 4** | MERGE Operator **MRG**($\varphi_n$,$p$,$q$) |
|---|---|
| | $C_p = C_p \cup C_q \cup \{\varphi_n.v\}$<br>$P = P/C_q$ |

Supposing merging $C_q$ into $C_p$ and letting the new node join $C_p$, the total gain number of inner-community edges of $C_p$ after operation of MERGE contains four parts: (a) the number of inner-community edges of $C_q$, (b) the number of connections between new node and $C_p$, (c) the number of connections between new node and $C_q$, (d) the number of connections between $C_p$ and $C_q$, denoted by $M_{pq}$. And for other communities there is no change in inner-community edges. The degree of $C_p$ is increased by the sum of degree of $C_q$ and degree of new node. According to (7) and (8), we have,

$$\Theta_3 = y_p + y_q + M_{pq}$$

$$\Theta_4 = \sum_{s \neq p, s \neq q}^{S} \left( (\deg(C_s) + y_s)^2 - \deg(C_s)^2 \right)$$

$$+ \left( \deg(C_p) + \deg(C_q) + y_p + y_q + h \right)^2$$

$$- \deg(C_p)^2 - \deg(C_q)^2$$

$$= 2y_s \cdot \sum_S \deg(C_s) + \sum_S y_s^2 + h^2$$

$$+ 2\{ (y_p \cdot y_q) + \deg(C_p)\deg(C_q) + (y_p + y_q) \cdot h$$

$$+ \left( \deg(C_p) + \deg(C_a) \right) \cdot h + \deg(C_p) y_a + \deg(C_a) y_p \}$$

### *B. Algorithm Framework*

Here we gave the pseudo code of our algorithm for node-grained streaming networks.

**Algorithm NGI: N**ode-**G**rained **I**ncremental community detection

**Inputs:** A queue of vertex with/without edges $\varphi_1, \varphi_2, ..., \varphi_N$
**Output:** Community partition $P = \{C_1, C_2, ..., C_K\}$

```
1   BEGIN
2   P = ∅
3   For each φn do
4     If φn.Links = ∅ then
5       ISO(φn)
6     else
7       if φn.v has adjacent isolated vertex then
8                 AGG(φn)
9       else
10        Find r that maxmize ΔQinsert(r)
11        if    max ΔQinsert(r) >0 then
12              IST(φn, r)
13        else
14              Find p and q that maxmize ΔQmerge(p,q)
15              if      maxΔQmerge(p,q) < ΔQinsert(r)
16                      IST(φn, r)
17              else
18                      MRG(φn, p, q)
19        end if
20      end if
21    end if
22  end while
23  End
```

### *C. Complexity Analysis*

In the following, we will give time complexity analysis of our proposed NGI. Let $N$ denote the total vertex number of network, $H$ is average degree of all vertices and $c$ is average number of adjacent communities. In our algorithm, when a new node arrives, we sweep its neighbors with time cost O($H$) and analyzed the local topology as follows. For **Case 1** (or **Case** 2), i.e., ISOLATE (or AGGREGATION) operator is adopted, the time cost is O(1) (or O($H$)). For

**Case 3**, the worst situation is that no positive modularity gain is achieved by INSERT operator and any combination of two adjacent communities are searched for MERGE operator, which leads to $O(c^2)$ time cost. Thus, the total time cost of processing one node is $O(H)+\max\{O(1), O(H), O(c^2)\}$. Note that $c$ is the average number of adjacent communities, not the total number of communities in whole network, which is usually smaller than $H$. Thus, the time complexity to process one vertex is smaller than $O(H^2)$ and time complexity on whole network is $O(N*H^2)$, equal to $O(m*H)$. As usual, the average degree of all vertices $H$ can be considered as a constant, compared with the total number of edges $m$. Thus, the time complexity of our algorithm is near linear.

## IV. EXPERIMENTS

In order to validate the effectiveness and efficiency of our proposed algorithm, we tested our algorithm on a set of large-scale real-world networks, and compared our algorithm with several competing incremental community detection algorithms and several representative non-incremental algorithms which use global information to attain a community structure.

We implemented our algorithm in JAVA. All experiments were done on Lenovo Server (Intel (R) CPU i5@2.70 GHz and 8GB RAM)

### *A. Effectiveness*

#### *1) Comparison with Incremental Methods*

In this section, we compared our algorithm with a number of incremental algorithms: OLTM [8], GreMod [7] and QCA [6]. We compared our NGI with OLTM, since both OLTM and our algorithm accomplished to detect community structure incrementally from the very beginning (i.e., the first node of network arriving). Furthermore, we also did some comparison with GreMod and QCA, which begin their incremental process from a basic network with certain scale and require an initial community structure obtained by static algorithm as their input.

Our comparisons were on 11 real-world networks which were collected from Stanford Large Network Dataset Collection, and these networks are listed in TABLE II, of which the number of nodes varies from thousands to near a million and the number of edges range from tens of thousands to several millions. Among these datasets, citation networks (4, 5, 11) and webpage networks (7-10) have the property that changes happens in node-grained. More importantly, for dataset 4 **Cit-HepPh**, most of vertices have timestamps which denote the time a node was added into the network, and we extracted a sub-network containing all of nodes with timestamps and defined an adding sequence of nodes according to the time information. We denoted this sub-network as **Cit-HepPh_REAL** which is listed as network 11 in TABLE II. For other networks, due to the lack of real timestamps, all nodes were added in random sequence and each edge was added along with its later adjacent node.

TABLE II. SUMMARY OF DATASETS.

| NO. | Data Name | Nodes | Edges |
|---|---|---|---|
| **1** | Ca-AstroPh | 18772 | 198110 |
| **2** | Ca-HepPh | 12008 | 118521 |
| **3** | Ca-HepTh | 9877 | 25998 |
| **4** | Cit-HepPh | 34546 | 421578 |
| **5** | Cit-HepTh | 17770 | 352807 |
| **6** | Ca-CondMat | 23133 | 93497 |
| **7** | Web-NotreDame | 325729 | 1497134 |
| **8** | Web-Stanford | 281903 | 2312497 |
| **9** | Web-BerkStan | 685230 | 7600595 |
| **10** | Web-Google | 875713 | 5105039 |
| **11** | Cit-HepPh_REAL | 30638 | 346703 |

At first, we compared our algorithm with OLTM. To make the comparison more fully, we modified OLTM into an algorithm for processing node-grained streaming network by assigning connecting edges of a node a random assumed addition order and renamed it as 'OLTM+'. We applied our algorithm, OLTM and OLTM+ on 11 datasets. For each datasets, we simulated a growing process of the network, and set 10 observation points during this expansion to compare results obtained by these 3 different algorithms. In order to ensure the comparisons for 3 algorithms on a same network topology at every observation point, our algorithm and OLTM+ which are node-grained algorithms adopted a same node adding sequence. On the contrary, OLTM is edge-grained algorithm, which is fed with an edge adding sequence that maintains a same set of edges between any two observation points but ranks these edges in a random order.

As the optimization objectives of all of these 3 algorithms are modularity functions, we took modularity as a quality criterion to evaluate their performance of these algorithms. Note that all results in Fig.1. are the average results over 10 runs. As shown in Fig.1, our proposed algorithm outperforms both OLTM and OLTM+ on all of these real-world networks, and the superiority on most of them is significant. Moreover, OLTM+ significantly outperformed OLTM on most of citation networks and webpage networks (as shown in Fig.1. (4)(5)(7)(8)(9)(10)) which are considered to be networks expanded on node-grained. This fact also indicates the significance of developing node-grained incremental algorithm to cope with this type of networks.

Moreover, we compared our NGI with GreMod and QCA. As GreMod and QCA algorithm need a basic network and require to be fed with an initial community structure obtained by static algorithm, in the following experiments GreMod and QCA took a sub-network with 10 percent of nodes as basic network and adopted community structure obtained by Louvain algorithm for the basic network as their initial community structure. Note that our NGI didn't used initial community structure obtained by Louvain which is an excellent static algorithm, and just began its incremental process from network with a single node.

The experimental results of the 11 data sets are shown in Fig. 2. It shows our incremental algorithm outperforms QCA and GreMod on all of networks as network grows. This is because GreMod and QCA used initial community structure obtained by Louvain algorithm (one of the best

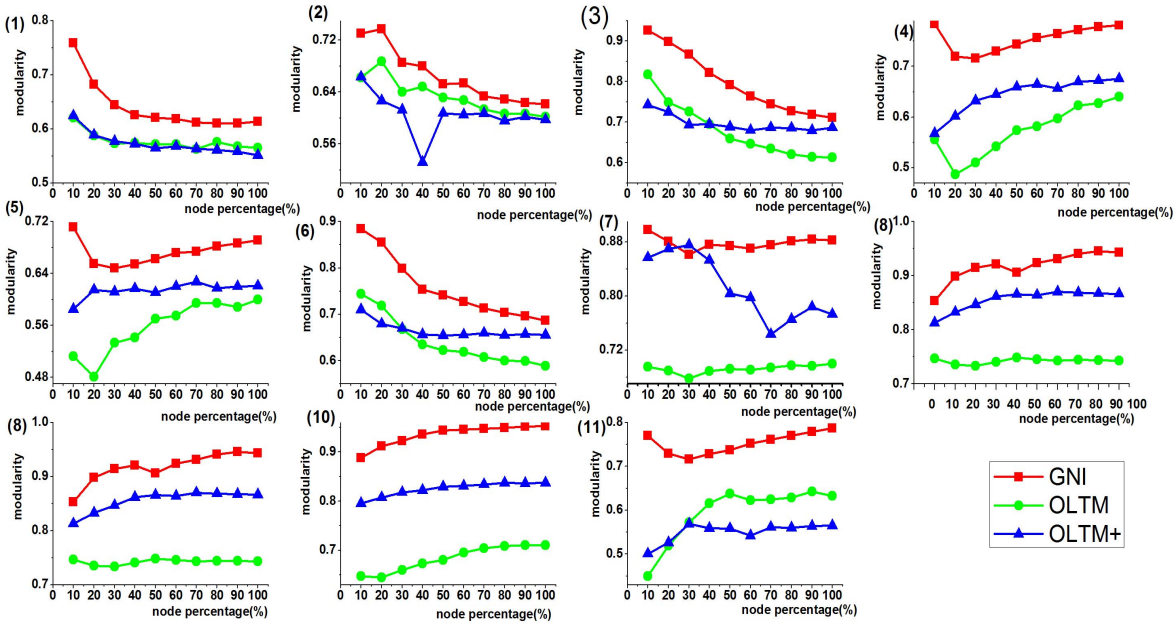


**Fig. 1.** The change of modularity as network grow

static algorithm using global information) while our algorithm did not. Therefore, at first, GreMod and QCA obtained higher modularity than our NGI. But, as it shows in Fig. 2, the modularity values obtained by GreMod and QCA decrease dramatically as nodes were added and the superiority of our algorithm became more and more obvious. It is noteworthy that, for network 10, although our NGI got a much smaller modularity value than GreMod and QCA at the first observation point, it outperforms GreMod and QCA significantly in the end. This demonstrates the strong capability of our algorithm on processing streaming network incrementally.

### 2) *Comparison with Static Methods*

In this section, we further compared our algorithm with some representative static algorithms which need to use the whole network topology to identify community structure. Here we chose CNM algorithm [14], Eig algorithm [15], Louvain algorithm [2] and FAMBGC [21]. Experiments in this section were on datasets of TABLE II. Note that our algorithm almost processed network stream in a one-pass way and only utilized the local information of newly added node in every updating, while these static methods used global information and may iterate many times on a network. TABLE III listed the modularity values obtained by 4 representative static algorithms (CNM, Eig, Louvain, FAMBGC) and our NGI on these real-world networks. As one can see, our algorithm outperforms CNM significantly and has comparable or even better modularity values with Louvain, state-of-the-art, and FAMBGC. Furthermore, our algorithm got higher modularity values than Louvain on most of citation networks and webpage networks, especially on the dataset 11 with real timestamps. One reason why our algorithm outperforms these static methods may be that the way we update community structure on node-grained and treat every edges of newly added node equally is more close to the way network evolves in reality

Although Louvain get better results on some networks, it needs much more time for processing streaming networks which was demonstrated *in Sec. IV.B,* as it is a static algorithm. Furthermore, our algorithm has the flexibility to update community structure in real time when a new node arrives.

TABLE III. COMPARISON OF MODULARITY

| Data set | NGI | Louvain | Eig | CNM | FAMBGC |
|---|---|---|---|---|---|
| 1 | 0.613 | **0.628** | 0.487 | 0.535 | 0.582 |
| 2 | 0.621 | **0.658** | 0.582 | 0.581 | 0.643 |
| 3 | 0.710 | **0.768** | 0.18 | - | 0.743 |
| 4 | **0.781** | 0.724 | 0.494 | 0.523 | 0.665 |
| 5 | **0.692** | 0.655 | 0.596 | 0.613 | 0.586 |
| 6 | 0.689 | **0.731** | 0.573 | 0.646 | 0.706 |
| 7 | 0.882 | **0.935** | - | - | 0.933 |
| 8 | **0.945** | 0.930 | 0.419 | 0.894 | 0.911 |
| 9 | **0.965** | 0.935 | 0.339 | 0.904 | 0.923 |
| 10 | 0.953 | **0.976** | - | - | 0.973 |
| 11 | **0.788** | 0.728 | 0.567 | 0.564 | 0.661 |

### 3) *Comparisons on Networks with Ground-truth*

In this section, to further illustrate the effectiveness of our algorithm we did our comparisons on networks with ground-truth by using a widely used accuracy metric, i.e., Normalized Mutual Information (NMI)[1], which measures the similarity between the results and truth community structures. Here we chose OLTM and Louvain as comparing algorithms, as OLTM is a representative incremental algorithm and can begin its incremental process from the

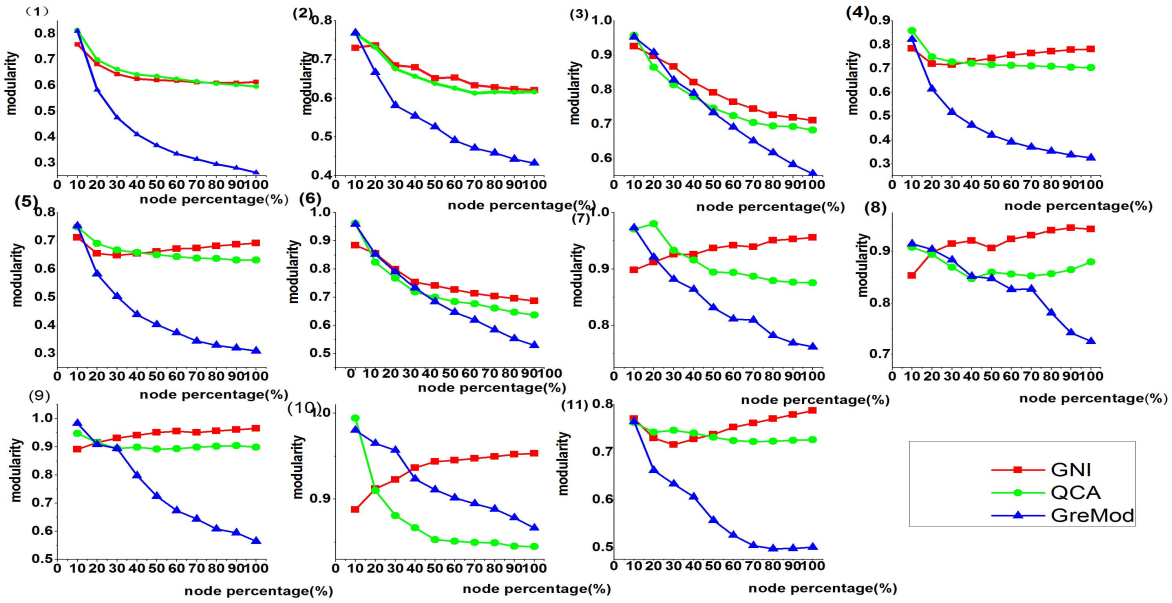


**Fig. 2.** The change of modularity as network grows

very beginning, i.e., a single edge, and Louvain is one of the best static algorithms using global information. All contrast experiments are on 4 public large datasets with ground-truth community structure, which are from Stanford Large Network Dataset Collection. Here for each data set, we use the sub-network with top 5,000 communities processed by data provider for community detection[22]. **com-Amazon** is a product co-purchasing network, where nodes represent products and there will be existed an edge between two products if the two products were brought together. It has 16716 nodes and 48739 edges. **com-DBLP** is a collaboration network, where a node denotes an author and two authors are connected if they co-published at least one paper together. It has 93432 nodes and 335520 edges. **com-Youtube** is a video sharing web, where nodes represent users and edges will be generated between two users if the two users share a video. It has 39841nodes 224235 edges. **com-LiveJournal** is an on-line blogging network where nodes represent blog users and an edge will connect two users if they declared to be friends with each other. It has 84438 nodes and 1521988 edges.

TABLE IV. NMI BENCHMARK BY FOUR COMMUNITY DETECTION ALGORITHMS

| Data set | NGI | OLTM | Louvain |
|---|---|---|---|
| com-Amazon | 0.913 | 0.879 | 0.916 |
| com-DBLP | 0.526 | 0.692 | 0.606 |
| com-DBLP | 0.526 | 0.692 | 0.606 |
| com-Youtube | 0.591 | 0.494 | 0.574 |
| com-LiveJournal | 0.824 | 0.862 | 0.866 |

The NMI value of 1 means that two assignments are identical and 0 implies the opposite. As we can see from TABLE IV., for **com-Amazon** and **com-Youtube** network, the NMI scores of our NGI are obvious higher than OLTM's. For **com-DBLP** network, nodes denote computer science researchers and an edge will be added between two researchers if they co-publish a paper. Thus, it is more likely the network expanded in edge-grained, not in node-grained. Maybe that is why OLTM obtained better results than ours in **com-DBLP**. Furthermore, compared with Louvain, our algorithm still achieves competitive results. Especially in **com-Amazon** network, both the result of ours and Louvain are considerably high and close to 1, indicating that both the results of our NGI and Louvain have high quality. Note that Louvain makes use of the whole network information to identify communities, and it wastes lots of time for processing network changes frequently, which is shown in detail in the following section.

### *B. Efficiency*

In this section, we compared the time cost of our algorithm with that of Louvain and OLTM.

For every dataset, sub-graphs were stored when every tenth of total vertices were added and then 10 sub-graphs were tested for each network. Fig.3(a) shows the time cost of Louvain on total 110 sub-graphs of 11 networks and the regression curve indicated the relationship between time cost and number of vertex. Fig.3(b) shows the cost of accumulative time which grows as the number of vertex increases. For the static algorithm Louvain, the cost of accumulative time was approximately calculated by integrating the regression expression because it must be re-performed the whole algorithm on new network topology after every node addition. On the contrary, the total time cost of incremental methods is the total running time of our algorithm (OLTM) for processing a sequence of node additions.

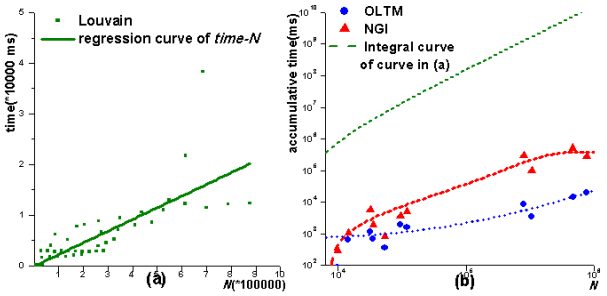


**Fig. 3.** Consuming time on networks of different scale

## V. CONCLUSION

While most of traditional incremental techniques are designed for processing edge-grained changes, there exist some real-world networks which expanded on node-grained. We propose a node-grained incremental community detection algorithm which took each vertex and its connecting edges as a processing unit and equally treat edges involved by the same node. The main idea of this algorithm is to update the community structure by analyzing the local topology of a new node and utilizing some prior information as nodes are added. To be specific, we defined 4 operators (ISOLATE, AGGARAGATION, INSERT and MERGE) which are adopted according to different local topological characteristics of newly arrived node, so as to attain the largest modularity gain. Experimental results demonstrated that our algorithm achieves better results than the competing edge-grained algorithms, and outperforms the representative static algorithms which use global information. Besides, it even obtains better results than Louvain method, one of the best methods for modularity optimization, in networks which are more likely to evolve on node grained. This also indicates the significance of node-grained incremental algorithms.

## ACKNOWLEDGMENT

This work was supported by Natural Science Foundation of China (61373035, 61502334, 61572350, 61502333) and the Tianjin Research Program of Application Foundation and Advanced Technology (14JCYBJCI5600).

## REFERENCES

[1] A.Lancichinetti, S. Fortunato, 2009. Community detection algorithms: A comparative analysis. Physical review E 80(50):056117.

[2] V.D. Blondel, J. Guillaume, R. Lambiotte, E. Lefebvre, 2008. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008(10): P10008.

[3] S. Fortunato, 2010. Community detection in graphs. Physics Reports 486(3-5): p. 75-174.

[4] M. Coscia, F. Giannotti, D. Pedreschi, 2011. A classification for community discovery methods in complex networks. Statistical Analysis and Data Mining: The ASA Data Science Journal4(5): p. 512--546.

[5] J. Duch, A. Arenas, 2005. Community detection in complex networks using extremal optimization. Physical review E 72, 027104 (2005)

[6] N.P. Nguyen, T.N. Dinh, Y. Xuan, M.T. Thai, 2011.Adaptive algorithms for detecting community structure in dynamic social networks. INFOCOM, 2011 Proceedings IEEE. IEEE, 2011: 2282-2290.

[7] J. Shang, L. Liu, F. Xie, Z. Chen, J. Miao, X. Fang, C. Wu, 2014. A Real-Time Detecting Algorithm for Tracking Community Structure of Dynamic Networks. In6th SNA-KDD Workshop, 2012.

[8] G. Pan, W. Zhang, Z. Wu, S. Li, 2014. Online Community Detection for Large Complex Networks. PLoS ONE 9(7): e102799. doi:10.1371

[9] J. Xie, M. Chen, B.K. Szymanski, 2013. LabelRankT: Incremental Community Detection in Dynamic Networks via Label Propagation. Proceedings of the Workshop on Dynamic Networks Management and Mining. ACM, 2013: 25-32.

[10] W. Zhang, G. Pan, Z. Wu, S. Li, 2013. Online community detection for large complex networks. Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press2013: 1903-1909. 2013.

[11] M.E.J. Newman, M. Girvan 2004. Finding and evaluating community structure in networks. Physical review E 69(2): 026113.

[12] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, D. Wagner, 2006. On modularity-np-completeness and beyond. Univ. Fak. für Informatik, Bibliothek, 2006

[13] M.E.J. Newman, 2004. Fast algorithm for detecting community structure in networks. Physical review E 69(6): 066133

[14] A. Clauset, M.E.J. Newman, C. Moore, 2004. Finding community structure in very large networks. Physical review E 70(6): 066111.

[15] M.E.J. Newman, 2006. From the Cover: Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103(23): 8577-8582.

[16] J. Xie, B.K. Szymanski, 2013. LabelRank: A Stabilized Label Propagation Algorithm for Community Detection in Networks. Network Science Workshop (NSW), 2013 IEEE 2nd. IEEE, 2013: 138-143.

[17] K. Wakita, T. Tsurumi. 2007. Finding community structure in mega-scale social networks. Proceedings of the 16th international conference on World Wide Web.ACM, 2007: 1275-1276.

[18] W.H. Chong, L.N. Teow, 2013. An incremental batch technique for community detection. Information Fusion (FUSION), 2013 16th International Conference on. IEEE, 2013: 750-757.

[19] C.E. Tsourakakis, C. Gkantsidis, B.R.M. Vojnovic, 2014. FENNEL: streaming graph partitioning for massive scale graphs. Proceedings of the 7th ACM international conference on Web search and data mining. 2014.

[20] A.L. Barabási, R. Albert 1999. Emergence of scaling in random networks. science 286(5439): p. 509-512.

[21] H. Shiokawa, Y. Fujiwara, M. Onizuka, 2013. Fast Algorithm for Modularity-Based Graph Clustering. AAAI. 2013.

[22] J. Yang, J. Leskovec (2012) Defining and evaluating network communities based on ground-truth. In: Proceedings of the Twelfth International Conference on Data Mining. pp. 745–754.