TELKOMNIKA, Vol.12, No.4, December 2014, pp. 1005~1016
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v12i4.438

A New Algorithm for Detecting Local Community Based on Random Walk

Yueping Li[1,2], Weikun Zheng[3]
[1] School of Computer Engineering, Shenzhen Polytechnic, Shenzhen 518055, P.R. China
[2] Shenzhen graduate school, Harbin Institute of Technology, Shenzhen 518055, P.R. China
[3] Information Center, Shenzhen Institute of Information Technology, Shenzhen 518172, P.R. China
e-mail: leeyueping@gmail.com[1], zhengwk@sziit.edu.cn[3]

Abstract

This paper presents one new algorithm for local community discovery. It employs a new vertex selection strategy which considers not only the boundary structure of candidate local community but also the probability which the investigated vertex will return to the candidate local community. A local random walk is adopted to compute this return probability which does not require the global information. We choose four algorithms for comparison which are the best ones existed by far. For better evaluation, the datasets include not only the computer generated graphs in standard benchmark but also the real-world networks which are classical ones in global community discovery. The experimental results show our algorithm outperforms the other ones on the computer generated graphs. The performance of our algorithm is approximately the same with the algorithm proposed by Luo, Wang and Promislow on real-world networks.

Keywords: *local community, community discovery, random walk*

1. Introduction

Extracting community structures in complex networks has gained much attention recently. Generally, networks can be modelled as graph $G=(V,E)$, where V is a set of vertices representing individuals and E is a set of edges that show the interaction between the vertices. There is no universally accepted definition of community. Conventionally, a community of a network is a group of vertices that are densely connected amongst themselves while being sparsely connected to the vertices outside the group. Usually, the vertices of one community exhibit certain common characteristics.

Many algorithms have been proposed to discover community structure in real world networks. However, these algorithms are supposed to find the entire community structure of the graph. This constraint makes these algorithms cannot handle the dynamic networks and the large scale networks. Unfortunately, networks in real world usually are larger than the scale can be settled by the fastest algorithms [1].

Recent works focus on finding local community structure, which detects the community given a start vertex. This task has many scenarios in daily applications. For example, the police might like to quantify the local communities of a suspect given his social network. Several methods have been proposed to extract local community. However, they suffer in one or more ways. For instance, proposed algorithms in [2],[3] are designed to process the graphs which has a minimal connected initial topology. The methods [4]-[6] in require some degree of global information obeys ascertain partitions. The algorithm in [7] is presented for dynamic networks. In addition, its time complexity is too high which is $O(|V|^4)$ larger than many global community discovery algorithms.

Due to above limitations, these methods are not widely used. Currently, the popular methods for finding local community are [8]-[10] which will be discussed further in Section 3. Local community can be formally defined as follows: Given an undirected graph $G=(V,E)$ and a start vertex $s \in V$. In the absence of the global knowledge, a subgraph $G_s=(V_s,E_s)$ is extracting from G containing the start vertex s, where s is densely connect with the vertices of V_s than the vertices of $V \setminus V_s$.

Received August 27, 2014; Revised October 26, 2014; Accepted November 16, 2014

The algorithms of [8]-[10] have the following problems: Clauset's algorithm [8] give hierarchical community while not output a certain local community. The algorithm proposed by [9] and [10] perform well on the graphs which contains significant local communities but work poor on the graphs without strong community structure. In addition, the correctness of these algorithms drop dramatically when process the vertices that lie on the boundary of local community. The reason is that they are designed mainly on the greedy of local community measure. The merging and removal of vertex in the temporary local community just investigate the boundary. Therefore, these methods could not explain why the output local community is relevant with the start vertex. The limitation will affect the further application of the found local community. In the following section, we will present one measure to the relevance of local community and the start vertex.

This paper proposes one new algorithm for extracting local community. The vertex selection of our algorithm considers only the boundary structure affected by the insertion of vertex but also the probability which the vertex will return to the candidate local community. This return probability is computed by a local random walk which does not demand the global structure of the graph. We compare our algorithm with four algorithms which are the best ones known by far. The datasets includes not only the computer generated ones in standard benchmark but also the ones modelled by real-world networks which are classical ones in global community discovery. The latter ones represents different type of community structure which helps to evaluate the algorithms. The experimental results show our algorithm outperforms the other ones on the graphs in the standard benchmark, and performs almost the same with the algorithm proposed by Luo, Wang and Promislow [9] on real-world datasets.

The rest of this paper is organized as follows: Section 2 presents the related works. The proposed algorithm is given in Section 3. The evaluation of the algorithm on artificial and real-world datasets is illustrated in Section 4. Section 5 discusses some furtherer improvement. Finally, Section 6 concludes the paper.

2. Related Works

This section describes three state-of-the-art algorithms for detecting local community. In addition, we provide formal definitions of related measures.

Firstly, Clauset gave the definition of local modualarity [8] as the portion of the connecting edges in the boundary edges, where connecting edge denotes the edge connects the vertex in the local community to the vertex outside the community.

Let C denotes the local community and B be the vertices comprise the boundary in which each vertex has at least one neighbor not in C. The boundary-adjacency matrix is defined by Clauset [8] as

$$B_{ij} = \begin{cases} 1 & \text{if vertices } i \text{ and } j \text{ are connected,} \\ & \text{and either vertex is in } B \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Based on this, Clauset proposed the measure "local modualarity" to be

$$R = \frac{\sum_{ij} B_{ij}\delta(i, j)}{\sum_{ij} B_{ij}} \quad (2)$$

where $\delta(i, j)$ is 1 when either $v_i \in B$ and $v_j \in C$ or vice versa, and is 0 otherwise. By means of this measure, one community is expected to have few connections from its boundary to the part outside the community, while having a greater proportion of connections from the boundary back into the community.

Clauset gave a greedy method on the local community measure "local modualarity". But the output is a hierarchical community containing given vertex which doesnot show the region of local community.

Luo, Wang and Promislow [9] proposed several algorithms based on the framework maintaining two vertex queues: adding queue and deleting queue. The merging of the vertex in adding queue and the removal of the vertex in deleting queue will both increase the local community measure. The algorithm repeats computing the these two queues and performing addition and removal operations until these two queues are empty; that is, adding any vertex into C nor removing any vertex in C will improve the measure. At that time, the community C will be output. The employed measure M is defined as follows:

$$M = \frac{\sum_{ij} B_{ij}\delta(i,j)}{\sum_{ij} B_{ij}[i \in C][j \in C]} \tag{3}$$

where $[i \in C]$ equals 1 when $i \in C$, otherwise 0. The definition of $[j \in B]$ is similar.

Luo, Wang and Promislow provided three versions of proposed algorithm [11]: greedy addition, add-all addition and K-like move. Stated by [11], the algorithm using add-all addition performs best among these three versions. We implement the versions of the add-all addition and greedy addition, denoted by LWP_{all} and LWP_{greedy}, in our experiments.

Recently, Bagrow [10] employed the greedy measure "outwardness" to select vertices merging into local community. The outwardness of vertex v with respect to community C is defined as:

$$\Omega_v(C) = \frac{1}{|\Gamma(v)|}\sum_{i \in \Gamma(v)}([i \notin C]-[i \in C]) \tag{4}$$

where $\Gamma(v)$ are the neighbors of v.

Bagrow [10] defined a p-strong community as stopping criteria. A community C is called p-strong if a fraction p of vertices in C satisfy that they have more neighbors inside C than outside. Bagrow [10] stated multiple values of p can be used simultaneously.

From the above introduction, it concludes that these three algorithms adopt greedy strategy on respective measures. Unfortunately, these measures are defined mainly on the boundary edges. They do not consider the start vertex directly. For example, LWP's algorithm has to determine whether the start vertex lies in the resulting local community since it contains removed operation. In addition, this shortcoming will be more prominent when the start vertex lies on the boundary of local community, which will be discussed further in the section presenting the experimental results.

3. Proposed method: Local Community Discovery Algorithm using Local Random Walk

Different from current methods, our algorithm dedicates to evaluate the relation of local community and the start vertex instead of investigating the boundary of local community only. To achieve this, we employ local random walk strategy to compute the visit probability of the vertices which will be used for the vertex selection. Next, we introduce the random walk strategy and our local random method.

There are various random walk strategies on graphs such as Markov chains [12], quantum random walk [13] and random walk based on vertex degree distribution [7], etc. This paper develops one new local random walk based on Marov chains. Necessary definitions are give, at first.

Let $G=(V,E)$ be a connected graph with n vertices and m edges. Consider a random walk on G: start with vertex v_0; at the t-th step, we assume the probability that move to a neighbor of v_t is $1/d(v_t)$, where $d(v_t)$ is the degree of v_t (equivalently, the number of neighbors of v_t). Then, the sequence of these random nodes $(v_t : t = 0,1,...)$ is a Markov chain. We denote the matrix of trasition probabilities of this Markov chain by $M = (p_{ij})$ where $i, j \in V$. We have

$$p_{ij} = \begin{cases} 1/d(i) & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

We denote the probability that vertex u is visited at step t-th by $\pi_t^{v_0}(u)$. The Vector $\pi_t^{v_0}$ stores the probabilities of all vertices in graph G. Then,

$$\pi_{t+1}^{v_0} = M^T \pi_t^{v_0} \quad (6)$$

This paper adopts the version of random walk with restart, because we wish to reveal the relation of the start vertex and other vertices. The formula is

$$\pi_{t+1}^{v_0} = (1-\alpha) M^T \pi_t^{v_0} + c \times e_0 \quad (7)$$

where e_0 is one vector of which the value of v_0's position is 1 and other values are 0. Then, $c \times e_0$ in Formula (7) indicates that random walk has probability c to restart with v_0 at each step.

It is necessary to mention that the computation of Formulas (6) and (7) requires the global information of graph. Therefore, we could not employ directly. We propose one approximate computation of random walk with restart when searching the candidate subgraph. In addition, we use adjacency list to store the transition probability p_{ij} instead of matrix M. Then, the probability $\pi_{t+1}^{v_0}$ can be computed by the following formula:

$$\pi_{t+1}^{v_0}(u) = \begin{cases} (1-\alpha) \sum_{v \in \Gamma(u)} \dfrac{\pi_t^{v_0}(u)}{d(v)} & u \neq v_0 \\ (1-\alpha) \sum_{v \in \Gamma(u)} \dfrac{\pi_t^{v_0}(u)}{d(v)} + c & u = v_0 \end{cases} \quad (8)$$

where $\Gamma(u)$ denotes the neighbor set of vertex u. We can search the adjacency list of vertex u to obtain $\Gamma(u)$. Then, this computation will not demand global adjacent information; that is, can be calculated locally.

Furthermore, we restrict the probability computation within the searching subgraph and the outside boundary. Formally, denote current subgraph by G_{sub} and outside boundary by OB. Then, OB can be formulated by

$$OB = \{u \mid u \in \Gamma(v) \wedge v \in G_{sub} \wedge u \notin G_{sub}\}$$

Therefore, Formula (8) is changed into:

$$\pi_{t+1}^{v_0}(u) = \begin{cases} (1-\alpha) \sum_{v \in \Gamma(u) \wedge v \in G_{sub} \bigcup OB} \dfrac{\pi_t^{v_0}(u)}{d(v)} & u \neq v_0 \\ (1-\alpha) \sum_{v \in \Gamma(u) \wedge v \in G_{sub} \bigcup OB} \dfrac{\pi_t^{v_0}(u)}{d(v)} + c & u = v_0 \end{cases} \quad (9)$$

We next introduce the vertex selection strategy of our algorithm.

We select one vertex to add into the current community at each step. Our goal is to select the vertices which lie in the same local community as the start vertex. Therefore, the visiting probability which travels from the start vertex is suitable for consideration while choosing vertices. Suppose u is the vertex which is evaluating for choosing. Beside the value of visiting probability of u at step t, we also measure the fraction of probability which u will go back into

the current candidate community at step $t+1$. We next present the measure of selection, formally.

Let G_{sub} be the candidate community at step t and OB be the outside boundary of G_{sub}. Thus, the vertices of set OB are candidates for choosing to be merged into G_{sub}. Let $\pi_t^{v_0}(u)$ be the visiting probability of vertex u while the start vertex is v_0. Then, our measure for choosing vertex u is defined as:

$$\begin{aligned}\Theta(u) &= \pi_t^{v_0}(u) \times \left(\frac{\sum_{v \in G_{sub} \wedge v \in \Gamma(u)} \pi_t^{v_0}(v)}{\pi_t^{v_0}(u)}\right)^2 \\ &= \sum_{v \in G_{sub} \wedge v \in \Gamma(u)} \pi_t^{v_0}(v) \times \frac{\sum_{v \in G_{sub} \wedge v \in \Gamma(u)} \pi_t^{v_0}(v)}{\pi_t^{v_0}(u)}\end{aligned} \tag{10}$$

The item $\left(\frac{\sum_{v \in G_{sub} \wedge v \in \Gamma(u)} \pi_t^{v_0}(v)}{\pi_t^{v_0}(u)}\right)^2$ forces that the vertices has high “return" probability obtain high priority to be selected. The reason that exponent number is set to 2 arises from the following two aspects: (i) if the number is set to 1, this measure also works but not as good as that equals 2. Because when exponent number is 1, the measure equals $\sum_{v \in G_{sub} \wedge v \in \Gamma(u)} \pi_t^{v_0}(v)$, which shows the sum of visiting probability of the neighbors of v in G_{sub}. It does not indicate the fraction of probability which the vertex will return to the candidate local community. Since one vertex has higher “return" fraction, the vertex bears closer connection with the start vertex. Therefore, we do not set the exponent number to 1; (ii) We have tested the number is larger than 2. The results of selection is almost the same.

In brief, we choose the vertex has maximum value of Θ to be merged into the candidate community G_{sub}. Then, recompute the visiting probability and the values of Θ until satisfied the stopping criteria.

The stopping criteria of existed algorithms is too simple. For example, most algorithms such as LWP_{greedy} and LWP_{all} algorithms [11] halt when there is no insertion or deletion of vertex can improved the measure. On the other hand, several algorithms such as Bagrow’s algorithm [10] stop when their measures reach the desire thresholds. Furthermore, some algorithms does not stop until searching the whole graph. One instance is the algorithm proposed by Clauset [8]. We next present the stopping criteria for our select-and-merge procedure.

Our algorithm employs the measure M defined by Formula 3 to mark "potential" local communities. Since our algorithm merges the vertices one by one, it is straightforward that measure for evaluation will increase or decrease dramatically if potential local community is found and this local community is significant. We denote these increasing or decreasing points by "jumping" points. Then, we record the potential local communities which are indicated by these increasing or decreasing points. Finally, we determine output which local community as result.

We now discuss the jumping points further by different cases. Let m_t be the value of measure M at step t.

Case 1: Increasingly jumping point. We call m_t is one increasingly jumping point if the following proposition holds

$$(m_{t-1} < m_t - \alpha_1) \wedge (m_{t-1} < m_{t+1} - \alpha_1) \wedge ..... \wedge (m_{t-1} < m_{t+n_1} - \alpha_1) \tag{11}$$

where α_1 and n_1 are parameters such that $0 < \alpha_1 < 1$ and n_1 is a positive integer. One example is illustrated on the left of Figure 1. This type of jumping point shows that one vertex which is not in the same local community as start vertex is merged into the current community. Therefore, we record the community at step $t-1$ as one potential community in this case.

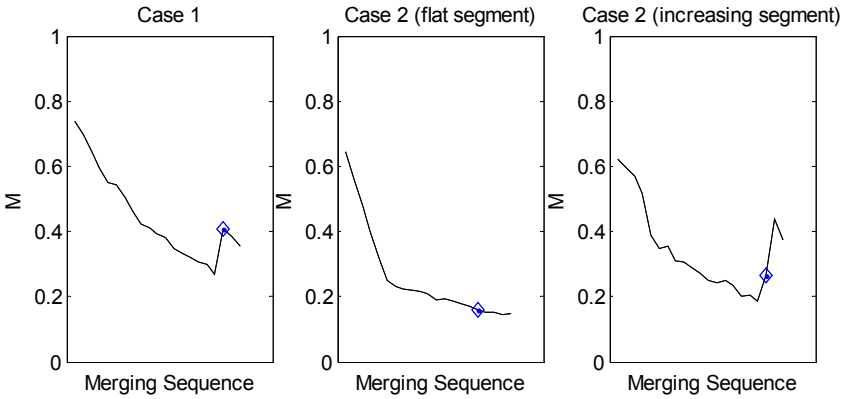

Figure 1. Different cases of jumping points

Case 2: Decreasingly jumping point. When the values of M drop dramatically, it probably finds one local community. However, it usually discovers one major part of the local community not the whole at that step. The values of M usually will still decrease in the following steps. Therefore, it requires to find the following step of which the values of M change sightly or increase.

We call m_t is one decreasingly jumping point if the following proposition holds

$$\begin{aligned}\exists m_k \forall t', \quad & (m_{k-1} > m_k + \alpha_2) \wedge (m_{k-2} > m_k + \alpha_2) \wedge ..... \wedge (m_{k-n_2} > m_k + \alpha_2) \\ & \wedge (k \le qt) \wedge (t \le t') \\ & \wedge (m_{t+1} < m_t - \beta_2) \wedge (m_{t+2} < m_t - \beta_2) \wedge ..... \wedge (m_{t+m_2} < m_t - \beta_2) \\ & \wedge (m_{t'+1} < m_{t'} - \beta_2) \wedge (m_{t'+2} < m_{t'} - \beta_2) \wedge ..... \wedge (m_{t'+m_2} < m_{t'} - \beta_2)\end{aligned} \tag{12}$$

where α_2, β_2 and n_2 are parameters such that $0 < \alpha_2, \beta_2 < 1$ and n_2 is a positive integer. By Formula 12, it is supposed that variable t must be the smallest value of which the corresponding M value indicates the flat or increasing segment. This constrain guarantees that it obtains the correct flat or increasing segment after one dramatically decreasing step.

An example of decreasingly jumping point followed by a flat segment is displayed on the middle of Figure 1. On the other hand, the case followed by an increasing segment is shown on the right of Figure 1. Since the flat segment or increasing segment begins at step t, we record the community at step t as one potential community in this case.

We next present the procedure of our algorithm as follows:

Local Community Discovery Using Local Random Walk

Input: graph G with adjacent list AL, start vertex v_0,

Parameters: α_1, n_1, α_2, β_2, n_2, m_2 and γ

Output: local community $G_{local}(v_0)$

Set $G_{sub} \leftarrow \{v_0\}$ and $\pi_0^{v_0} \leftarrow 1$;

Use vertex set OB to store the outside boundary of G_{sub}, set $OB \leftarrow \varnothing$;

For ($t \leftarrow 1$; $t \le 3$; $t++$) {

Update the boundary OB of G_{sub};

Compute $\pi_t^{v_0}$ for all vertices in $G_{sub} \cup OB$ by Formula 8;

Insert the vertex u, which $\Theta(u)$ is maximum, into G_{sub};

}

Do {
Update the boundary OB of G_{sub};
Compute $\pi_t^{v_0}$ for all vertices in $G_{sub} \cup OB$ by Formula 9;
Insert the vertex u, which $\Theta(u)$ is maximum, into G_{sub};
Compute m_t which is the measure M for G_{sub} by Formula 3;
Determine whether m_t is a jumping point at current step t;
If it is, record the jumping point;
$t \leftarrow t+1$; }
While (G_{sub} is small than G and $m_{t-1} \geq \gamma$)
Output the subgraph of the first jumping point as $G_{local}(v_0)$.

The reason that we use Formula 8 to compute $\pi_t^{v_0}$ when $1 \leq t \leq 3$ is we wish to know visiting probability of the vertices which are very closed with the start vertex v_0 (that is, the vertices of which the distance from v_0 is no larger than 3). Though this computation is not as correct as the classical random walk, it is sufficient for the vertex selection. Moreover, our algorithm just outputs the local community found by the first jumping point. It does not output the hierarchical community according to our problem statement.

We next present the time complexity of our algorithm. It needs $O((Deg_{avg})^3)$ time to compute $\pi_t^{v_0}$ for $1 \leq t \leq 3$, where Deg_{avg} is the average degree of the vertices in graph G. For the remained step $t > 3$, the running time is bounded by $O(|V(G_{local}(v_0))|^2)$. In the worst case, it is $O(|V(G)|^2)$. Usually, the result community $G_{local}(v_0)$ is much smaller than the whole graph G Thus, our algorithm runs in $O((Deg_{avg})^3 + |V(G_{local}(v_0))|^2)$ time in average.

4. Experiments for Evaluation

In this section, we employ the benchmark method proposed by Bagrow [10] to give an objective comparison with the existing algorithms for finding local community structures. Bagrow's method [10] uses computer generated networks. Besides these artificial networks, we also evaluate the algorithms on famous real-world datasets which show different structures of local community.

Computer generated networks: In Bagrow's benchmark, it creates a classical graph $G = (V, E)$ of $|V(G)| = 128$, which is then randomly partitioned into four reference communities to contain equal number of vertices (32 vertices). Each vertex has an average degree $z = z_{in} + z_{out} = 16$, where out-degree z_{out} equal the number of edges connect the vertices outside the communities. It is Clear that a small z_{out} shows a strong community structure. In order to evaluate the performance less affected by randomness, we generate 100 graphs for each z_{out} from 1 to 7.

Real-world datasets: We choose some famous datasets modeled by real-world graphs such as karate club network [1], US college football league [14] and dolphin social network [15]. These graphs has different structures of local community.

Karate club network contains 34 members as vertices and 78 edges representing friendship between members. Due to a disagreement between the club's administrator and the club's instructor, the club splits into two smaller communities. In addition, these two communities are prominent.

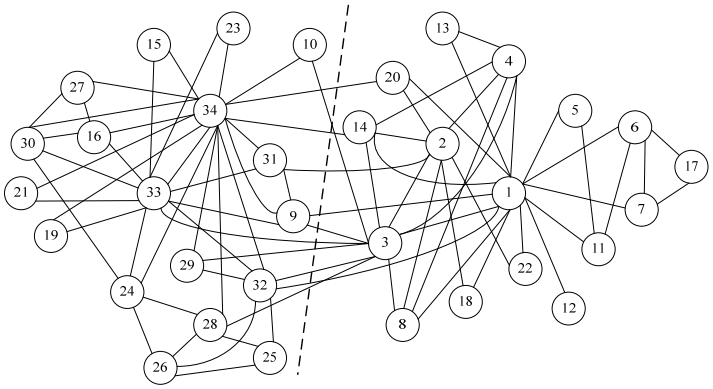

Figure 2. Karate club network

The US college football network is constructed from the game schedule of the 2000 season. The nodes in the network represent the 115 teams, while the edges represent 613 games played in year. The teams are divided into 11 conferences of 8-12 teams each and generally games are more frequent between teams of the same conference than between teams of different conferences. Therefore, the community structure is strong in each conference.

The dolphins social network studied by David and Lusseau [15] was constructed from observations of a community of 62 bottle-nose dolphins. This network is divided into two groups according to their age. But the community structure is not as significant as the karate club network. We next provide the evaluation measures.

Normalized mutual information (NMI) is an important evaluation criteria in local community discovery [16][10]. The correct partition is denoted by $P_R = \{C_R, C_R\}$, where C_R is the reference local community. Similarly, the found is denoted by $P_F = \{C_F, C_F\}$, where C_F is the result local community. A confusion matrix N is employed in NMI measure, where the rows correspond to the real communities, and the columns correspond to the found communities. The element of N, N_{ij} is the number of nodes in the real community i that appear in the found community j. Then, then NMI measure of similarity between the partitions, based on information theory, is defined below:

$$I(P_R, P_F) = \frac{-2\sum_{i=1}^{C_R}\sum_{j=1}^{C_F} N_{ij}\log(N_{ij}N/N_{i.}N_{.j})}{\sum_{i=1}^{C_R} N_{i.}\log(N_{i.}/N) + \sum_{j=1}^{C_F} N_{.j}\log(N_{.j}/N)}$$

where the sum over row i of matrix N_{ij} is denoted $N_{i.}$ and the sum over column j is denoted $N_{.j}$.

Thus, a NMI score of 1 shows that both communities are identical and a score of 0 is when these two communities are totally independent.

On the other hand, we employ the F-measure to reveal the efficiency of local community discovery algorithms. Precision is the fraction of the C_F retrieved that lies in C_R.

$$precision = \frac{|C_F \bigcap C_R|}{|C_F|}$$

The recall is the fraction of C_R which are successfully detected by C_F.

$$recall = \frac{|C_F \bigcap C_R|}{|C_R|}$$

We adopt the weighted harmonic mean of precision and recall. That is, the traditional F -measure is

$$F = \frac{2 \cdot precision \cdot recall}{(precision + recall)}$$

The parameter setting is given as follows: Formula 11 shows the conditions that one step is considered as one jumping point: α_1 represents increasing gap and n_1 requires how many consecutive points should reach the gap. Similarly, α_2 and n_2 represent the decreasing gap and the number of consecutive points obtaining the gap, respectively. In addition, β_2 stands for the threshold of one flat segment and m_2 show the number of consecutive points of which the values should not go beyond the threshold.

In our algorithm, we set $\alpha_1 = 0.05$, $\alpha_2 = 0.03$, $\beta_2 = 0.0015$ and $n_1 = n_2 = m_2 = 3$ for all the datasets.

Since the measure M will diminish to 0 when the subgraph G_{sub} covers the whole graph G, it is supposed to halt the search while G_{sub} is too large. In this case, there is probably no significant local community containing the start vertex v_0. We use parameter γ to stop the algorithm when measure M is too small. The parameter γ is set to 0.15 in our algorithm.

We compare our algorithm, denoted by LRW, with the state-of-the-art ones by far which are Bagrow's algorithm [10], LWP_n and LWP_A proposed by Luo, Wang and Promislow [11].

It is note that we choose the best p in range $\{0.75, 0.76, \ldots, 1\}$ for all the datasets according to Bagrow's algorithm. We found overall performance is best when $p = 0.9$ which coheres to the result in [10].

Table 1. Results on real-world datasets

	karate				football				dolphins			
	LRW	*Bagrow*	*LWP$_n$*	*LWP$_A$*	*LRW*	*Bagrow*	*LWP$_n$*	*LWP$_A$*	*LRW*	*Bagrow*	*LWP$_n$*	*LWP$_A$*
F$_{avg}$	**0.858**	0.7709	0.7646	0.6436	0.7922	0.7027	**0.9058**	0.1862	0.6695	0.6685	0.5021	**0.9367**
F$_{std}$	0.118	0.1535	0.2418	0.0745	0.145	0.2991	0.1591	0.0285	0.1369	0.1473	0.2945	0.1618
I$_{avg}$	**0.5372**	0.4438	0.5275	0.0332	0.6114	0.5568	**0.84**	0	0.3301	0.2603	0.2939	**0.7597**
I$_{std}$	0.205	0.2413	0.3084	0.0812	0.2095	0.311	0.2307	0	0.1616	0.1791	0.3014	0.2583

We enumerate each vertex as the start vertex and perform the algorithm to detect the local community. The experimental results are presented by Table 1, where F_{avg} and F_{std} are the average and the standard deviation of F-measure, I_{avg} and I_{std} are the average and the standard deviation of NMI measure. The best values are illustrated by bold font. The LWP_n algorithm on the dolphins dataset.

It shows that LRW performs best on F_{avg} and I_{avg} of karate networks. For the others, the results of LRW stand the second position. It appears that the overall results of LWP_n algorithm and our algorithm are better than the others. Therefore, we show the overall performance which is given by Figure 3.

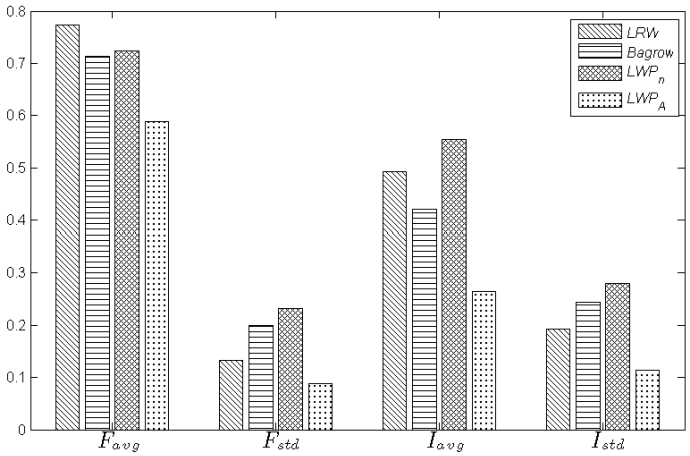

Figure 3. Overall performance of the results on real-world datasets

It appears that the overall performance of our algorithm is approximately the same as LWP_n algorithm on real-world datasets. Our algorithm works better on karate and dolphins datasets while poorer on football dataset compared with LWP_n algorithm. However, our algorithm is more stable than LWP_n. One reason is that our algorithm performs best on one dataset and stands the second on the remained two dataset. On the other hand, the F_{std} and I_{std} of our algorithm are less than the ones of LWP_n on all datasets.

Next, we present the experimental result on computer generated networks which is illustrated by Figure 4. It shows that our algorithm and LWP_n outperform the others. By the plot, it appears that our algorithm and LWP_n works approximately the same when z_{out} is small. That is, the community structure of test graphs is significant. While z_{out} goes large, the performance of LWP_n drops more dramatically than our algorithm, especially when $z_{out} = 6,7,8$.

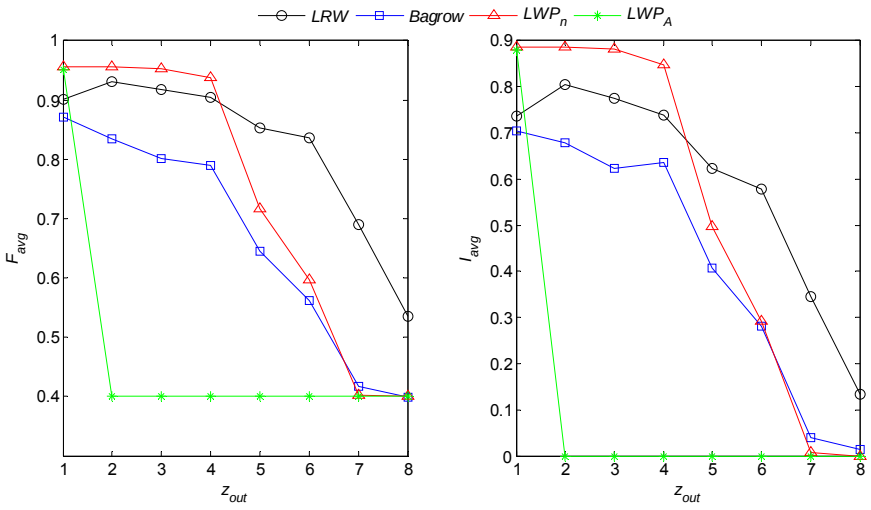

Figure 4. F_{avg} and I_{avg} results on 128-nodes network Overall

We summary the results and present theoverall performance on computer generated networks by Figure 5. It concludes that our algorithm performs better than LWP_n algorithm on all the evaluation measures.

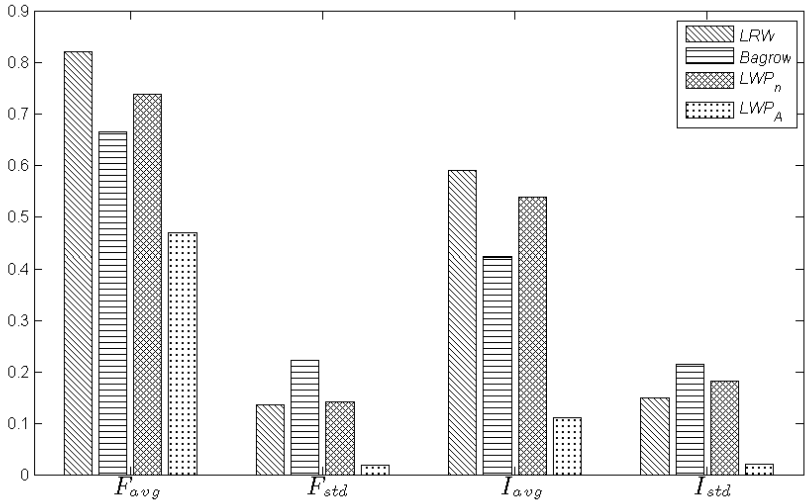

Figure 5. Overall performance of the results on 128-nodes network

5. Conclusions and future work

This paper presents one new algorithm for detecting local community containing given vertex. The vertex selection strategy of our algorithm is different from the existed ones. The strategy considers not only boundary stricture but also the "return" probability of the vertex which is computed by a local random walk. We select the algorithm which are the best ones known by far for comparison. Besides the standard benchmark of local community discovery, we also perform experiments on the datasets modeled by real-world networks which are the classical ones in global community discovery. The experimental result shows that our algorithm and LWP_n algorithm perform almost the same on real-world datasets.

In addition, these two algorithms outperform the others. On the other hand, our algorithm outperforms the other algorithms on the datasets of the standard benchmark of local community discovery. It improves the *F*-measure and normalized mutual information by about 0.05 on average. Moreover, the results show that our algorithm is more stable than the others while given different vertices.

Furthermore, our local community evaluation measure can adopt different metrics. The metrics for evaluating (local) community have been intensive studied, such as [17]. Once the quantity of the metric is normalized into [0,1], our stopping criteria is applicable. Therefore, our method can use this metric for local community discovery.

Moreover, our method can be integrated to semantic network while detecting topical community [18]. The routine is to amend the visiting probability for one vertex to its neighbor which is based on topology into the measure of topical similarity. That is, the neighbor which is more similar has high visiting probability.

Acknowledgements

This work is granted by National Science Foundation of China under no. 61100190 and by Shenzhen Strategic Emerging Industries Program under no. GJHS20120627112429515. Our work is also supported in by Science and Technology Program of Shenzhen Polytechnic under Grant no. 2212K3190005 and 2213K3190016.

Our work is also partially supported by Research Cultivation Project of Shenzhen Institute of Information Technology no. LG201416.

References

[1] Clauset A., Newman MEJ., Moore C. Finding Community Structure in very Large Networks. *Physical Review E.* 2004; 70(6): 066111.

[2] Almeida RB., Almeida VAF. Local Community Identification through User Access Patterns. *Clin. Orthopaedics Relat. Res.* 2002; cs.IR 0212045.
[3] Barbosa VC., Donangelo R., Souza SR. Emergence of Scalefree Networks from Local Connectivity and Communication Trade-offs. *Physical Review E.* 2006; 74: 016113.
[4] Farutin V., Robison K., Lightcap E., et al. Edge-count Probabilities for the Identification of Local Protein Communities and Their Organization. *Proteins*. 2006; 62(3): 800–818.
[5] Palla G., Derenyi I., Farkas I., Vicsek T. Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. *Nature*. 2005; 435: 814–818.
[6] Spirin V., Mirny LA. Protein Complexes and Functional Modules in Molecular Networks. *Proc. Natl. Acad. Sci. USA.* 2003; 100(21): 12123–12128.
[7] Thakur GS., Tiwari R., Thai MT., Chen SS., Dress AWM. Detection of Local Community Structures in Complex Dynamic Networks with Random Walks. *IET Syst. Biol*. 2009; 3(4): 266–278.
[8] Clauset A. Finding Local Community Structure in Networks. *Physical Review E*. 2005; 72: 026132.
[9] Luo F., Wang JZ., Promislow E. *Exploring Local Community Structures in Large Networks*. IEEE/WIC/ACM International Conference on Web Intelligence, Hong Kong. 2006: 233–239.
[10] Bagrow JP. Evaluating Local Community Methods in Networks. *J. STAT. MECH.* 2008: P05001.
[11] Luo F., Wang JZ., Promislow E. Exploring Local Community Structures in Large Networks. *Web Intelligence and Agent Systems: An International Journal.* 2008; 6: 387–400.
[12] Lovasz L. *Random Walk on Graphs: A Survey.* Technical Report, Department of Computer Science, Yale University, New Haven. 1994.
[13] Childs AM., Farhi E., Gutmann S. An Example of the Difference Between Quantum and Classical Random Walks. *Quantum Information Processing*. 2002; 1(1): 35–43.
[14] Girvan M., Newman MEJ. Community Structure in Social and Biological Networks. *Proc. Natl. Acad. Sci. USA.* 2002; 99: 7821–7826.
[15] Lusseau D., Schneider K., Boisseau OJ., Haase P., Slooten E., Dawson SM. The Bottlenose Dolphin Community of Doubtful Sound Features a Large Problem of Long-lasting Associations. *Behavioral Ecology and Sociobiology.* 2003; 54(4): 396–405.
[16] Danon L., Di'az-Guilera A., Duch J., Arenas A. Comparing Community Structure Identification. *J. Stat. Mech.: Theory Exp.* 2005; P09008.
[17] Cheng Jianjun, Xu Hong, Gaybullaev Mahmud, Leng Mingwei, Chen Xiaoyun. Community Detection Algorithm based on Neighbor Similarity. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(8): 4484–4490 .
[18] Liu Hongtao, Chen Hui, Lin Mao, Wu Yu. Community Detection Based on Topic Distance in Social Tagging Networks. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(5): 4038–4049.