Detecting Communities Using Link and Content Triangles

Qiuling Yan[1], Baoli Li[2], Dongqing Yang[1]

[1] Department of Computer Science, Peking University, Beijing, China
[2] Department of Computer Science, Henan University of Technology, Zhengzhou, China

yqlpku@gmail.com,dqyang@pku.edu.cn, csblli@gmail.com

Abstract. Community detection for uncovering the hidden community structure in large networks is an important task in analyzing complex networks. Most of the existing methods only consider link structure in networks, where the link information is usually sparse and noisy, which may result in a poor partition of a network. Fortunately, besides link structure, nodes, especially in social networks, are often associated with certain symbolic or textual attributes, which we refer to as content. Content, therefore, is expected to serve as a reasonable complement for finding a good partition. In this work, we propose an algorithm LICT to detect communities with link and content triangles. It works in three steps: 1) network expansion with content similarity; 2) community detection in weighted network; and 3) refinement by weighted triangle modularity. Experimental results on several real data sets demonstrate that the proposed algorithm is effective for community detection and robust in the presence of link noise.

Keywords: community detection; social network analysis; link and content triangles; weighted triangle modularity; spectral optimization

1 Introduction

Real networks are often organized in local clusters called communities, which can be considered as relatively independent modules. Nodes in the same community are more densely connected to each other than that of nodes in different communities. Communities can occur in many networked systems. For example, in social networks, a community is a group of friends that communicate with each other much frequently. In citation networks, a community is a set of papers that have citation relationship and focus on the same topic. In protein-protein interaction networks, communities are a group of proteins having the same specific function within the cell. Thus Detecting communities is crucial to understand the structural properties of networks [1] and helpful to improve other tasks such as link prediction [2].

Many existing methods only use network structure to detect communities. However, there exists noise in networks, representing as incorrect links and missing links, which weaken clustering quality. To reduce the impact of noise, content is a good complement. The similarities and differences in the content of nodes can affect the patterns of

linking. Thus, it is sensible to combine links and content together to detect communities. There exist some solutions aiming at this problem, which can be categorized into two classes. One is generative probabilistic modeling [3] [4] [5] [6] [7] [8] [9]. Although these solutions can model links and content simultaneously, they are either too complex to be applied or only able to handle relatively small networks. Another type of approaches is heuristic[10] [11] [12] [13]. They either embed content information into edges or store link structure into a distance function between nodes. However, these methods either limit content to attributes of nodes or lose the ability to discriminate different nodes when too many features of content are involved.

In this work, we propose a simple but effective algorithm to detect communities using link and content triangles. It works in three steps. First, given a network, we add new edges into the network according to content similarity. Then we compute weights of edges using both structural information and content similarity. At the second step, we use *k*-way spectral method to partition the weighted network. Thirdly, the partition is refined according to weighted triangle modularity. We apply the method to several real networks. Experimental results show that it is effective for community detection and robust in the presence of link noise.

The paper is organized as follows: section 2 presents related work; section 3 explains the proposed algorithm; section 4 shows the experimental results, and section 5 concludes the paper.

2 Related Work

A lot of algorithms have been proposed in the past years to detect communities in a complex network. Fortunato provides an excellent survey[14]. Here, we focus on related work in two specific directions, as they are highly relevant to our proposed algorithm. One is to combine links and content to detect communities, and the other is the usage of triangles in network analysis.

Community detection using both links and content: There are various approaches to utilize both sources, which can be categorized into two classes. One class is generative probabilistic modeling [3] [4] [5] [6] [7] [8] . In these works, it was assumed that either community generates links and content or communities and content generate link structure. For example, Liu et al. [3] argued that network structure is dependent on both communities and content. The authors in [4] merged the idea of topic model and stochastic model, with the assumption that links and content share the same topic space. Sun et al. [5] proposed a probabilistic model that clusters the objects of different types into a common hidden space. Nallapati et al. [8] used LDA and PLSA to model citing documents and cited documents respectively and introduced a method called Link-PLSA-LDA to jointly model content and links. Similar to [3] [4] [8], topic model based approaches are also proposed in [6] [7] [9].

Another popular category to combine links and content is the hybrid approaches [10] [11] [12] [13], most of which computes pairwise distances by fusing similarities of links and content. Akoglu et al. [10] proposed a method that compresses adjcent matrix and

feature matrix simultaneously to disclose community blocks. Ruan et al. [11] constructed content edges and fused them into original network to get an extended network with the same vertices. Then he sampled the network to obtain a sparse one and applied some existing methods to partition the sampled network. Zhou et al. [12] introduced a method named as SA-Cluster which inserts attribute nodes to a network to get an augmented network. Then they used the neighborhood random walk model to estimate the vertex closeness on the new network. Moser et al. [13] integrated the concepts of dense subnetworks and of subspace clusters in a feature space. Then they find out subsets of nodes that are close in the feature space. Our work is inspired by the work [11]. The difference is that we convert pairwise similarity into edge weights and use triangles modularity to improve partition quality.

Community detection based on triangles: Since many metrics in network analysis can be obtained by graph triangulation, it provides insight into social network analysis[15] [16] [17]. Coefficient and transitivity are representative, which are two important metrics quantifying density of sub-networks. Consequently, we can use triangles to improve community results. For example, Klymko et al. [18] applies triangles information to detect community in directed networks. Prat-Pérez et al. [19] assumes that well-defined communities are dense in terms of triangles. Accordingly, he proposed a metric called WCC to measure the quality of community results. Serrour et al. [20] extends the modularity metric with triangles. The most prominent difference between our work and the works above is that we utilize content information as well as structure.

3 Community Detection Using Link and Content Triangles

Let $G(V, E, T)$ be an undirected network. V is the set of vertices $(v_1, v_2, \ldots, v_n)$. E is the edge set without weights. Each node v_i in V corresponds to a content vector t_i in T. Our goal is to cluster vertices according to both network structure and content similarity, with the assumption that the density of triangles in a cluster is larger than that outside the cluster. In this work, we propose a method called LICT[1], which consists of three steps. First, we add edges and weights to the original network, according to link structure and content similarity. Then any state-of-art method that aims at weighted networks can be applied on the new network. At the last step, we refine the partition according to weighted triangle modularity, a metric that accords with our intuition that triangles are building blocks for community.

Now we proceed with more details.

3.1 Combining Links and Content

To combine links and content, our idea is to compute pairwise affinity of vertices utilizing both link structure and content similarity. Then we add weights and some

[1] Detecting communities using **LI**nks and **C**ontent based on **T**riangles.

edges to the original network to get a weighted one. Algorithm 1 demonstrates our idea in detail. For each node pair (v_i, v_j), we compute the cosine similarity between vector t_i and t_j (Line 2-4). Elements in each vector t_i can be a binary value, or TF-IDF value, or number of word occurrence. After computing pairwise content similarity, for each node *v* we choose top *K* vertices and add edges between *v* and those vertices into the original network (Line 5-7) to get a new network *G*'. To avoid over-expanding network, we do not consider those similarities smaller than a threshold *T*. Consequently, there exists the case that the number of new neighbors for a node is smaller than *K*. In addition, if there already exists an edge between node *v* and one of its top *K* vertices, we do not need to add a new one. In line 6, Neighbors(*v*) is the neighbor set of node *v*. To decide the value of *K*, the scale of a network is an indispensable factor to take into account.

Algorithm 1: Converting a network to a weighted one
Input: a network $G(V, E, T)$ without weights Output: a weighted network $G'(V, E')$
1. $G' \leftarrow G$ 2. For each pair (v_i, v_j) in G' 3. Compute content similarity; 4. End For 5. For each v in V 6. Add top K vertices to Neighbors(v) according to content similarity and add edges to G' accordingly; 7. End For 8. For each edge (u, v) in G' 9. Weight of edge(u, v)$\leftarrow \partial \times simt_{uv} + (1 - \partial) \times simc_{uv}$ 10. End For

Fig. 1. Combining links and content

At line 9, we combine link structure and content similarity to compute weights for edges in *G*'. $simt_{uv}$ represents structural affinity for node *u* and node *v* in the original network *G*, which is computed as $simt_{uv} = \frac{1}{l_{sp}}$, where l_{sp} is the length of the shortest path between *u* and *v* in network *G*. Bidirectional search algorithm is used to compute $simt_{uv}$. For a node *v*, since l_{sp} is known as 1 between *v* and one of its original neighbors in *G*, we only need to compute length of the shortest path between *v* and its new neighbors. $simc_{uv}$ is the value of content similarity between *u* and *v*, which is normalized using *zero-one* in globe scope. ∂ is a balancing coefficient between $simt_{uv}$ and $simc_{uv}$.

It is important to note that our method can be also extended to weighted networks, by merging the new weights with the original ones into the networks.

By now, we convert a network $G(V, E, T)$ without weights to a weighted network $G'(V, E')$. Then any state-of-art method that aims at weighted networks can be applied on the network $G'(V,E')$ to get a partition of *V*. In this work, we use *k*-way spectral

clustering method [21]. After clustering the vertices, we need to refine the partition results. Our refinement method is described in next section.

3.2 Refining Clustering Results

In a network $G(V, E)$, a triangle is a complete subnetwork that consists three nodes $(u, v, w) \in V$ and three edges $\{(u, v), (v, w), (u, w)\} \in E$. Triangles play an important role in network analysis. Many metrics of networks can be computed directly by counting triangles, such as cluster coefficient [22], neighborhood density [15]. Triangles are also useful to improve clustering quality [18] [19] [20] [23]. In our work, we assume that density of triangles inside a community is larger than that across different communities. Then we propose a metric called weighted triangle modularity and use it to refine the initial partition obtained in previous section.

Weighted triangle modularity is an extension of modularity[24], which is a widely used metric. The modularity metric is based on the assumption that there are more dense edges in a community than in a random network with the same degree distribution. Given a partition $P=\{C_1, C_2, \dots, C_k\}$ of a network G, the generalized definition of modularity is as follows.

$$Q(P) = \frac{1}{2w}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(w_{ij} - \frac{w_i^{out}w_j^{in}}{2w}\right)\delta(C_i, C_j)\ . \quad (1)$$

Where w_{ij} is the weight of an edge (v_i, v_j). If there is no edge between v_i and v_j, w_{ij} is zero. $w_i^{out}(=\sum_j w_{ij})$ is the degree going from the node v_i, while $w_j^{in}(=\sum_i w_{ij})$ is the strength of links coming to the node v_j. C_i is the index of a community to which node v_i belongs. Finally, $\delta(C_i, C_j)$ is the Kronecker function assigning to 1 if node v_i and node v_j belong to the same community, 0 otherwise. The larger the Q value is, the better the community partition is.

Through comparing density of triangles rather than density of edges, we extend modularity to get weighted triangle modularity. The formula of this metric is as follows.

$$Q(P) = \sum_{i,j,k,} B_{ijk}\delta(C_i, C_j)\delta(C_j, C_k)\delta(C_k, C_i)\ . \quad (2)$$

With the conditions that $(i,j),(j,k),(k,i) \in E$ and triangle inequality holds among w_{ij}, w_{jk} and w_{ki}.

In equation 2, B_{ijk} is the mathematical object that evaluates difference of the triangle density between a sub-network and a corresponding random network. We compute B_{ijk} according to the following formula.

$$B_{ijk} = \frac{1}{T_G}w_{ij}w_{jk}w_{ki} - \frac{1}{T_R}(w_i w_j)(w_j w_k)(w_k w_i)\ . \quad (3)$$

T_G is the total number of triads of nodes that form triangles in the network G. The formula of T_G is:

$$T_G = \sum_i \sum_j \sum_k w_{ij}w_{jk}w_{ki}\ . \quad (4)$$

T_R is the counterpart of T_G in null case. The formula for T_R is as follows.

$$T_R = \sum_i \sum_j \sum_k (w_i w_j)(w_j w_k)(w_k w_i) \ . \quad (5)$$

In equations 3-5, w_i is the sum of weights for edges that node v_i intervenes.

Although Serrour [20] also extended the modularity using triangles, he did not control conditions. We believe that those conditions are important to detect more cohesive communities.

With weighted triangle modularity, we further refine the initial partition obtained in previous section. The heuristic idea is to move vertices among communities to increase the value of weighted triangle modularity. This process is repeated until Q does not increase any more. We demonstrate the details in algorithm 2. To be noted, we use the network obtained using algorithm 1 rather than original network in this phase.

When moving node *v* from a community to another, it is sensible to choose the communities to which node *v* connect densely, rather than trying each of other communities (Line 4). Triangle number that node *v* involves in a community can be set as the choosing criterion.

In algorithm 2, the computation for the Q value costs mostly. Since the computation part of Q that does not relate to node *v* stay unchanged, we only need to focus on the relative change of Q, which relates to two communities at most: the source community C_s and the destiny one C_d. Let us consider the simplest case firstly that the source community C_s only contain node *v*. when moving node *v* from C_s to C_d, the relative change of Q is computed as follows.

$$\Delta Q_I = \sum_{i,j,k \in C'} B_{ijk} - \sum_{i,j,k \in C_d} B_{ijk}, \quad (6)$$

where $C' = C_d \cup \{v\}$.

Since the relative change ΔQ_I results from participation of node *v* into community C_d, we only need to consider triads involving node *v* in community C_d. Thus we can rewrite equation 6 as follows.

$$\Delta Q_I = \sum_{i,j \in C_d} \left(\frac{1}{T_G} w_{vj} w_{ij} w_{jv} - \frac{1}{T_R} (w_v w_i)\left(w_i w_j\right)\left(w_j w_v\right)\right) \ . \quad (7)$$

With the conditions that $(v,j),(j,v),(i,j) \in E$ and triangle inequality holds among w_{vj}, w_{jv} and w_{ij}.

Now suppose that the source community C_s for node *v* contains other nodes. We have the following theorem.

Theorem 1. Let $P = \{C_1, C_2, \dots, C_s, C_d\}$ and $P' = \{C_1, C_2, \dots, C_s', C_d'\}$ be two partition for network *G*(*V*, *E*), where $C_s' = C_s \backslash \{v\}$, $C_d' = C_d \cup \{v\}$. Then, when moving node *v* from C_s to C_d, the change of Q is computed as follows.

$Q(P') - Q(P) = -\Delta Q_{Is} + \Delta Q_{Id}$. (8)Where ΔQ_{Is} is the change for node *v* from the community $\{v\}$ to be inserted into community C_s, which can be obtained using equation 7. ΔQ_{Id} is the same case as ΔQ_{Is} except that the destination is community C_d.

Proof. Moving node *v* from C_s to C_d can be considered as two steps, each of which leads to a new partition of $G(V, E)$. For each step, we use Q_{old} and Q_{new} to represent Q for the old partition and the new one, respectively.

In the first step, node *v* is removed from C_s to get a community consisting only of node *v*. In the second step, node *v* is inserted from the community $\{v\}$ to community C_d. As for the first step, suppose that ΔQ_R is the change of Q, the following formula holds obviously.

$$Q_{old} = Q_{new} + \Delta Q_{Is},$$

Then $\Delta Q_R = Q_{new} - Q_{old} = -\Delta Q_{Is}$. Since the change of Q in the second step is ΔQ_{Id}, equation 8 holds by combining the two steps.

Algorithm 2: Refinement

Input: partition P of a weighted network $G'(V, E')$
Output: refinement partition P'

```
1. P' ← P;
2. Repeat
3.     For each v in V
4.         candidates ← candidateComm(v, P');
5.         For each C in candidates
6.             Compute ΔQ(v, SourceC, C) according to equation 8;
7.         End For
8.         Choose the candidate C* that maximizes ΔQ value;
9.         If ΔQ > 0, move v from SourceC to C*;
10.    End For
11. Until Q value does not increase
```

Fig. 2. The process of refinement

3.3 Complexity of Algorithms

Given a network $G(V, E, T)$, let *n* be the number of nodes and *m* the number of edges. We assume that the average degree in *G* is $d=2m/n$.

In algorithm 1, the costly part is to compute pairwise similarity of content among vertices (Lines 2-4). The complexity for this part is $O(tn^2)$, where *t* is the length of content vector. The loop in lines 5-7 is $O(nK)$, where *K* is a constant given by users to choose top *K* new neighbors. To compute structural affinity in line 9, the complexity is $O(b^{l/2})$ for a pair of nodes (u, v) in G', where *b* is a branching factor and *l* is the length of the shortest path. Thus, for the loop in lines 8-10, the complexity is $O(nKb^{l/2})$. Then the complexity of algorithm 1 is $O(tn^2 + nK + nKb^{l/2})$. For a network that $Kb^{l/2} \ll n$, the complexity is $O(tn^2)$. It means that computing pairwise similarity of content takes the most time in algorithm 1.

Remember that we adopt *k*-way spectral optimization to get an initial partition, the main cost results from *k*-means algorithm. Since the cost of other part can be negligible, the complexity for this phase is $O(nka)$, where a is the number of iteration for *k*-means.

Now we turn to algorithm 2, the refinement part. To move a node *v* from a community to another one, we need to compute the change of Q value. The corresponding cost

is $O(c^2)$, where c is the average scale of communities in a partition. In fact, to choose candidate communities, we can adopt some heuristic schemes, rather than examining every community. For example, voting is a feasible choice, which can reduce the cost to O(1). Then the computation for loops in line 3-10 is $O(nd^2)=O(m^2/n)$. Let r be the number of iteration to find the best partition, the total complexity for algorithm 2 is $O(rm^2/n)$.

To sum up, the total cost for our method is $O(tn^2+ nka+ rm^2/n)= O(tn^2)$. In another word, computing content similarity costs mostly in this work.

4 Experiments

4.1 Datasets

In our experiments, we use three real datasets with different domains ranging from citation networks to social networks, all of which are treated as undirected. Each dataset is described below.

- CORA[2][25]. This is a citation network, in which each paper is considered as a node. We randomly choose a seed and use breadth-first search to get a small network with 2527 papers and 8427 edges. The small network is used as our first dataset. We extract title and abstract as content for each paper, which is represented as a vector of word occurrence. Our dictionary contains 5688 words. Each paper is labeled with a category. There are 10 classes for the chosen papers, which is defined as ground-truth communities.
- Flickr. We use the dataset used in the work [11] as our second dataset, which was gathered from the Flickr site. This is a user-user contacting network, which contains 16710 users and 716,063 edges among users. Tags adopted by users for photos are used as content information. The elements of content vectors are binary. There are 184421 user groups and a user can join in several groups. We use these groups as ground-truth communities.
- Facebook[3]. The dataset includes several ego-networks, consisting of 4039 nodes and 88234 edges. User profiles are used as content information, including locations, education information and so on. The social circles are labeled by the owners of ego-networks. We use those social circles as ground-truth communities.

4.2 Experimental Settings

In algorithm 1, we need to decide the parameter K for choosing top K content neighbors for each node. Since we extend original networks by adding edges according to content similarity, we assume that the number of new edges is no more than original edges. Specifically, we set K as 10, 50 for Cora and Flickr, respectively. We set K as 5 for Facebook because that the average size for each ego-network is small. Besides, to filter

[2] https://people.cs.umass.edu/~mccallum/data.html
[3] http://snap.stanford.edu/

content similarities, we set the threshold T as the average of content similarities in each network. In algorithm 2, parameter ∂ balances the contributions of structural information and content similarity. Since we are interested in how different ∂ influences performance of the whole method, we would like to examine different values for ∂. When using k-way spectral method to get an initial partition for a network, we set the community number k as the same with that of ground truth, except the Flickr dataset. Different from CORA and Facebook, a node in the Flickr dataset can belong to several communities. For simplicity, we set k as 50 for the Flickr dataset.

We choose two other methods as the baselines. One is the method proposed in this work, without regard to content. It enables us to examine to which extent content information contributes to community detection. We call this method LIT[4] for the sake of convenience. In the process of refinement for LIT, we set the weight as 1 for an edge. The other baseline method is CODICIL[11], which detects communities using links and content. Comparing our method LICT with CODICIL can help us to investigate the role of triangles in community detection. We set parameters of CODICIL in the same way as LICT, and also use k-way spectral method for CODICIL. Among state of art methods that combine links and content, SA-Cluster[12] and Link-PLSA-LDA are typical. The former is heuristic and the latter is a generative probabilistic model. Since CODICIL has been shown to outperform the two methods, we do not compare LICT with them.

Given a predicted partition P and the ground truth P' for a network G, we compute average F1-score used in [26] to measure the clustering quality. Specifically, both the predicted communities and ground-truth communities are considered as reference. After matching predicted communities with those in ground-truth sets, we also match ground-truth communities with predicted ones. Then the performance is measured by the average of F1-scores, which is calculated as follows:

$$F1(P,P') = \frac{1}{2|P|}\sum_{C_i \in P} F1(C_i,P') + \frac{1}{2|P'|}\sum_{C_i \in P'} F1(C_i,P)$$

$$F1(C,P) = \operatorname{argmax} F1(C,S_i), S_i \in P = \{S_1, \dots, S_n\}$$

$$F1(C,S) = \frac{2\times p\times r}{p+r}, p = \frac{|C\cap S|}{|C|}, r = \frac{|C\cap S|}{|S|}.$$

4.3 Experimental Results

First of all, we examine the clustering quality for the proposed method compared with baselines. For the method LICT and CODICIL, we set parameter ∂ as 0.6, unless noted otherwise. We show the result in Figure 3.

Compared with CODICIL, the proposed method LICT performs better on all of three datasets. Although both LICT and CODICIL leverage links and content, LICT uses triangles in community detection. The results verify our intuition that triangles play a role for the improvement of clustering quality. On the other hand, LICT outperforms LIT in all cases, which shows that content information is valuable for clustering. Besides, LIT performs better than CODICIL on Facebook. The reason is that the Facebook dataset is ego-networks, which contains more triangles than other networks. This enables LIT to work well, in spite of ignoring content. No matter what domain of a network is, LICT can be applied more widely than both LIT and CODICIL.

[4] Detecting communities using **LI**nk **T**riangles.

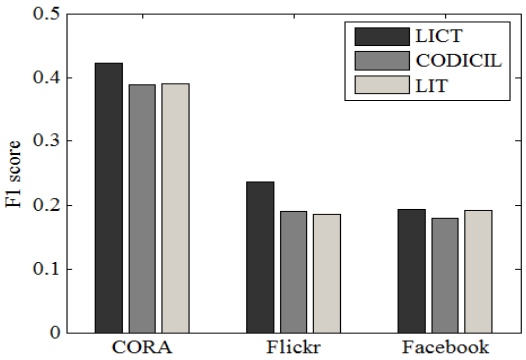

Fig. 3. Performance comparison of LICT, CODICIL and LIT in term of F1 score

To investigate the impact of parameter ∂, we apply LICT on the three datasets with ∂ valued from 0.1 to 0.9, stepping by 0.1. We show the results in Figure 4. For CORA and Flickr, we get the highest F1 scores with $\partial = 0.6$, so we set $\partial = 0.6$ in the rest of experiments.

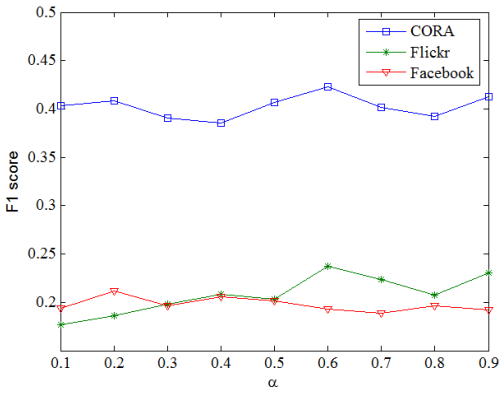

Fig. 4. Performance of LICT with ∂ valued from 0.1 to 0.9

To further investigate the role of content information, we remove some fraction of edges randomly and apply both LICT and LIT to the obtained networks. Figure 5 shows the relative improvement of LICT compared to LIT. For all of three datasets, when we remove more edges, relative improvements of LICT increase. Especially for the document network CORA, the improvement is much more obvious. Thus, when the network becomes unreliable or contains link noise, we can use content information to improve clustering quality.

5 Conclusion

In this work, we propose a method that combines links and content to detect communities. The method consists of three steps. First, we add edges to a network according to

content similarity and convert the network to a weighted one. Then we apply *k*-way spectral algorithm to get an initial partition for the weighted network. In the third step, we refine the partition further according to weighted triangle modularity. Experimental results on several real datasets show that the proposed method is effective for detecting communities and robust in the presence of network noise.

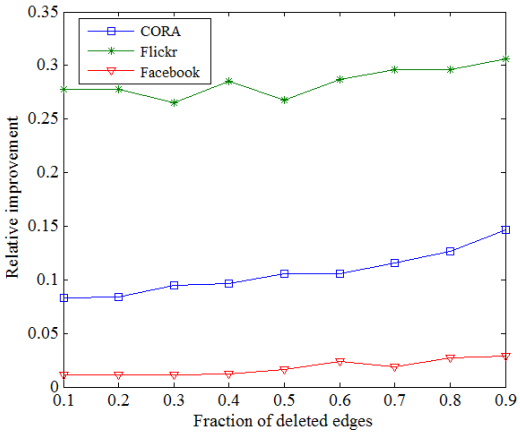

Fig. 5. Relative improvement of LICT against LIT when deleting edges

In the future, we plan to improve the work from two directions. Firstly, as the two procedures of choosing the top *K* nodes and refining partition with weighted triangle modularity are time-consuming, we would like to explore how to speed up these two parts. Secondly, we also consider how to make the proposed method applicable for much larger networks.

Acknowledgements

Baoli Li was partly supported by the Henan Provincial Research Program on Fundamental and Cutting-Edge Technologies (No. 112300410007), and the High-level Talent Foundation of Henan University of Technology (No. 2012BS027).

References

1. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.-U.: Complex networks: Structure and dynamics. Physics reports 424, 175-308 (2006)
2. Chang, J., Blei, D.M.: Relational topic models for document networks. In: International Conference on Artificial Intelligence and Statistics, pp. 81-88. (2009)
3. Liu, Y., Niculescu-Mizil, A., Gryc, W.: Topic-link LDA: joint models of topic and author community. In: proceedings of the 26th annual international conference on machine learning, pp. 665-672. ACM, (2009)
4. Balasubramanyan, R., Cohen, W.W.: Block-LDA: Jointly modeling entity-annotated text and entity-entity links. In: SDM, pp. 450-461. SIAM, (2011)

5. Sun, Y., Aggarwal, C.C., Han, J.: Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. Proceedings of the VLDB Endowment 5, 394-405 (2012)
6. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: A model-based approach to attributed graph clustering. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 505-516. ACM, (2012)
7. Zhu, Y., Yan, X., Getoor, L., Moore, C.: Scalable text and link analysis with mixed-topic link models. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 473-481. ACM, (2013)
8. Nallapati, R.M., Ahmed, A., Xing, E.P., Cohen, W.W.: Joint latent topic models for text and citations. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 542-550. ACM, (2008)
9. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: Data Mining (ICDM), 2013 IEEE 13th International Conference on, pp. 1151-1156. IEEE, (2013)
10. Akoglu, L., Tong, H., Meeder, B., Faloutsos, C.: PICS: Parameter-free Identification of Cohesive Subgroups in Large Attributed Graphs. In: SDM, pp. 439-450. Citeseer, (2012)
11. Ruan, Y., Fuhry, D., Parthasarathy, S.: Efficient community detection in large networks using content and links. In: Proceedings of the 22nd international conference on World Wide Web, pp. 1089-1098. International World Wide Web Conferences Steering Committee, (2013)
12. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proceedings of the VLDB Endowment 2, 718-729 (2009)
13. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining Cohesive Patterns from Graphs with Feature Vectors. In: SDM, pp. 593-604. SIAM, (2009)
14. Fortunato, S.: Community detection in graphs. Physics Reports 486, 75-174 (2010)
15. Schank, T.: Algorithmic aspects of triangle-based network analysis. vol. PhD. Universit¨at Karlsruhe (2007)
16. Chu, S., Cheng, J.: Triangle listing in massive networks and its applications. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. (2011)
17. Wang, N., Zhang, J., Tan, K.-L., Tung, A.K.H.: On triangulation-based dense neighborhood graph discovery. Proceedings of the VLDB Endowment (2010)
18. Klymko, C., Gleich, D., Kolda, T.G.: Using Triangles to Improve Community Detection in Directed Networks. arXiv preprint arXiv:1404.5874 (2014)
19. Prat-Pérez, A., Dominguez-Sal, D., Brunat, J.M., Larriba-Pey, J.-L.: Shaping communities out of triangles. In: Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 1677-1681. ACM, (2012)
20. Serrour, B., Arenas, A., Gómez, S.: Detecting communities of triangles in complex networks using spectral optimization. Computer Communications 34, 629-634 (2011)
21. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. NIPS, pp. 849-856 (2001)
22. Holme, P., Kim, B.J.: Growing scale-free networks with tunable clustering. Physical review E 65, 026107 (2002)
23. Prat-Pérez, A., Dominguez-Sal, D., Larriba-Pey, J.-L.: High quality, scalable and parallel community detection for large real graphs. In: Proceedings of the 23rd international conference on World wide web. (2014)
24. Newman, M.E.: Analysis of weighted networks. Physical Review E 70, 056131 (2004)
25. McCallum, A., Nigam, K., Rennie, J., Seymore, K.: Automating the Construction of Internet Portals with Machine Learning. Information Retrieval Journal 3, 127--163 (2000)
26. Yang, J., Leskovec, J.: Overlapping community detection at scale: a nonnegative matrix factorization approach. In: Proceedings of the sixth ACM international conference on Web search and data mining, pp. 587-596. ACM, (2013)