2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)

A Link Strength Based Label Propagation Algorithm For Community Detection

Abdallah Lakhdari*, Aicha Chorana*, Hadda Cherroun*
*Laboratoire d'Informatique et Mathématiques (LIM)
Université Amar Telidji, Laghouat, Algeria
{a.lakhdari,a.chorana,hadda_cherroun}@mail.lagh-univ.dz

Abdelmounaam Rezgui§
§Cloud Computing and Big Data Lab (C2BD)
New Mexico Tech, USA
rezgui@cs.nmt.edu

***Abstract*—The Label Propagation Algorithm (LPA) is a fast algorithm for community detection. This algorithm has proved its efficiency and scalability even in large social networks. It is based only on updating the label of each node by the most frequent label within its neighbors. However, in case of many maximal labels, a random choice is made. This reduces the performance of the algorithm. In this paper, we propose a new version of LPA, called Link Strength-based LPA (LS-LPA), that addresses this issue. In our approach, we rank labels according to the link strength of each neighbor. The strength of links is quantified via k-neighborhood which includes common neighbors and multi-step neighbors. Experiments on different networks with different sizes show that our solution improves LPA's performance significantly.**

***Keywords*—Social networks, Community detection, LPA, $K-$neighborhood**

I. Introduction

A real world network may be modeled as a graph with a set of nodes connected with edges. A *community* can then be defined as a group of nodes with denser connections compared to the rest of the network. A community structure in a graph means the existence of these groups. It is a very common feature for many real world networks in many fields including communication, social, collaboration and biological networks [1].

Finding communities in large scale networks has many applications, e.g., discovering users with a common location, finding friends sharing similar interests or occupations in an online social networks, etc. In Biology, community detection is commonly used to discover the functional grouping in metabolic networks [5]. This has led to a great interest in community detection algorithms. By its very nature (analogous to graph partitioning), community detection is an *NP*-hard problem. Researchers in several disciplines (e.g., physics, computer science) have proposed many algorithms for community detection. They also have suggested different taxonomies to classify this body of research and identify the important works [4], [14]. Most of those taxonomies have now concurred in identifying four classes: vertex clustering [3], [25], quality optimization [13], [2], sub graph discovery [24] and model based techniques [15], [16]. The work we propose in this paper is relevant to the fourth class of techniques. Specifically, we propose a new version of the Label Propagation Algorithm (LPA) called Link Strength-based LPA (LS-LPA).

LPA was proposed by Raghavan et al. [15]. It presents some desirable qualities such as no parameters required, easy implementation, fast execution for large networks, and the ability to detect valid communities even in random graphs. However, LPA has several inherent drawbacks. In particular, in the case where multiple labels (of the neighbors of a given vertex) are equally frequent, LPA randomly selects one of those most frequent labels to assign to the given vertex. This solution leads to the possibility that two or more disconnected groups of nodes may end up having the same label, which translates into assigning the same label to different unrelated communities.

The central idea in this paper is to avoid LPA's random labeling scheme. We propose an improvement to LPA's label update rule by leveraging information from the $k-$neighborhood, where k refers to the k-step neighbors of a given node.

The rest of the paper is organized as follows: in Section II, we give a brief taxonomy of a number of LPA variants followed by some LPA related work. In Section III, we give the details of the proposed LS-LPA algorithm. The empirical study and results are discussed in Section IV.

II. Related Work

We are interested in improving the label propagation algorithm for community detection. We therefore limit this section to reviewing some LPA variants proposed as extensions to the standard LPA. LPA detects community structure using network topology [15]. It starts by assigning a distinct label to each node. At every step, each node changes its label to the one carried by the largest number of its neighbors. After convergence, the community structure is obtained by grouping together nodes with the same label.

A number of extensions to the LPA approach have been proposed in the literature. We developed a taxonomy of these extensions according to the drawbacks of LPA that they have tackled. Figure I shows our taxonomy. Most of the previous related work focused on three characteristics of LPA: stability, computational cost and overlapping communities:

Stability

Usually, the communities detected by LPA are unstable, due to the random choices during the updating rule in case of multiple maximal frequent labels among neighbors. Some variants improve this rule and avoid the random choice by different strategies:

Xie and Szymanski [21] have proposed to rank the labels assigned to each node. A set of operators is introduced to

978-1-5090-3936-4/16 $31.00 © 2016 IEEE
DOI 10.1109/BDCloud-SocialCom-SustainCom.2016.61

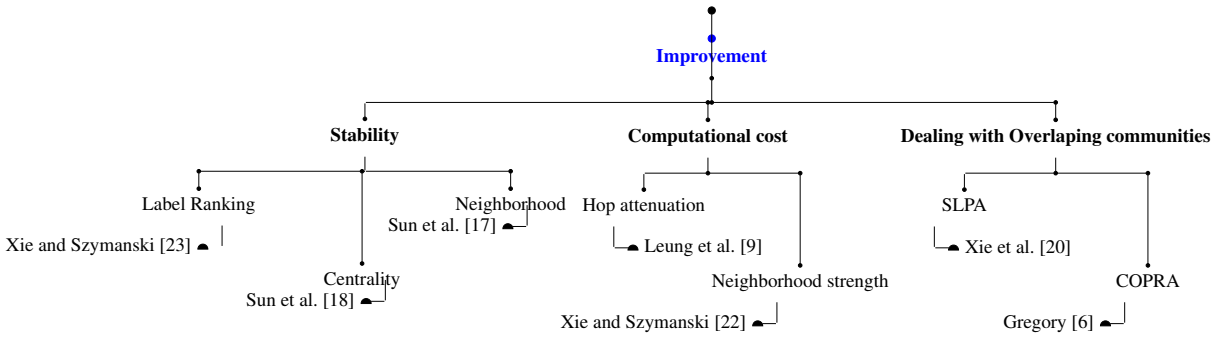


Figure 1. Taxonomy of improved variants of LPA.

control and stabilize the propagation dynamics. Briefly, these operations resolve the randomness issue in LPA by storing all assigned labels with a probabilistic score in every iteration. These scores are used to eliminate the randomness by privileging the label having the highest score.

Another algorithm, called CenLP (Centrality-based Label Propagation), was proposed by Sun et al. [18]. This algorithm replaces the random update order of LPA by arranging nodes according to a new centrality measure based on local density of neighborhood. This concept had also been exploited in a previous work by the same author [17] where the main idea consists of using the impact of a node on its neighbors in order to determine the update order of node labels.

Computational Cost

The simple principle of LPA makes propagating labels within a large network very fast. This feature results in very long iterations, which, in turn, leads to much larger communities compared to the actual ones. Some techniques were developed to reduce the number of label updates for each node. Leung et al. [9] proposed a hop attenuation technique that adds a score associated with each label which decreases as it propagates away from its origin. A node is initially given a score of 1 for its label. After it collects scored labels from its neighborhood, the node is assigned the label having the maximum score after attenuation.

Another optimization was proposed by Xie and Szymanski [22]. They add a new update rule and label propagation criterion to LPA. It is based on avoiding unnecessary updates. Most nodes are in a very diverse neighborhood. First, the fraction of attempted updates that result in changes to new labels is high. However, the competition between communities is restricted only to their boundaries after a few iterations. The authors define an interior node as a node that shares the same label with all of its neighbors. Nodes that are not interior are called boundary nodes. Moreover, the authors considered *active* and *passive* nodes depending on whether or not those nodes update their labels. Consequently, they defined a criterion that allows for active boundary nodes to be updated and avoids updating all other nodes. These changes reduce significantly the number of iterations, which consequently improves computational efficiency.

Overlapping Communities

Early community detection algorithms focused only on disjoint communities within networks. However, in many cases elements in real world networks belong to more than one group at the same time. Thus, researchers later also extended existing algorithms or developed new ones that can detect overlapping communities. LPA and its numerous extensions are able to cope with overlapping communities.

Xie et al. [20] have tackled this problem by proposing an approach that relies on the principle that one way to account for overlap in LPA is to allow each node to possess multiple labels as proposed by S. Gregory [6]. The approach detects and analyzes both individual overlapping nodes and entire communities. In fact, the authors introduced a procedure, called *Speaker/Listener*, that records label exchanges between nodes according to dynamic interaction rules similar to human communications. A node accumulates knowledge of repeatedly observed labels instead of erasing all but one of them. Moreover, the more a node observes a label, the more likely it will spread this label to other nodes (mimicking people's preference of spreading the most frequently discussed opinions).

As mentioned before, S. Gregory introduced the principle of extending LPA to detect overlapping communities. He proposed a new version of LPA called Community Overlap PRopagation Algorithm (COPRA) [6]. Gregory defined a mapping function of vertices and community identifiers to their belonging coefficient in every LPA iteration. Thus, the algorithm labels each vertex x with a set of pairs (c,b), where c is a community identifier and b is a belonging coefficient indicating the strength of x's membership in community c.

III. Link Strength Based Label Propagation Algorithm

The main idea of our community detection approach is to extend the LPA algorithm by harnessing the topological features of links. We measure their strength in order to avoid the LPA random choice problem as much as possible.

The principle of LPA is to assign a label to every node in the network and set them in a random order. For each node, LPA updates the label by choosing the most frequent label within its neighbors. If several maximal frequent labels exist, the algorithm selects one of them randomly. This process is

repeated until each node in the network gets the most frequent label from its neighbors. We maintain the simple principle of LPA, i.e., each node takes the most frequent label among its neighbors. However, we replace the random choice labeling when multiple maximal frequent labels appear by choosing the label of neighbors having the *strongest links*.

Our idea is based on the following fact. When two nodes share more common neighbors, they are more likely close to each other and susceptible of belonging to the same community. This principle has been used in several other contexts.

For example, it was used in [7] to measure the similarity between two nodes in order to define the shared nearest neighbor (SNN). If two nodes are close to a common set of neighbors, that means they are close to each other. Recently, the same principle was extended to exploit the concept of *influence* in community detection. if two nodes influence a common set of (direct and indirect) neighbors, then this suggests that they are close to each other and confirms their same-community membership [19].

To explain our contribution, we start by describing how we measure the strength of links, then we illustrate how we incorporate this measure into the LPA algorithm.

A. Link Strength

The notion of quantifying a link is commonly used in different ways by the link analysis community. For instance, it is used to predict future links among unconnected pairs in a given social network [10][11].

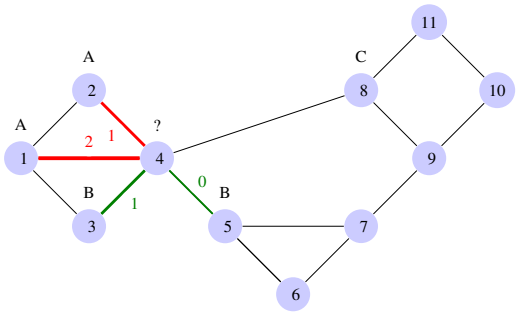


Figure 2. Illustrative Network

Figure 2 illustrates a situation of random choice. Let 4 be the node that is currently being processed by LPA. To update node 4's label, LPA has to consider two maximal frequent labels appear: *A* (two times) and *B* (two times). It is clear that the best choice is label *A* because nodes 1 and 2 belong to the same community of node 4. This can also be seen from the fact that they share more neighbors with 4 than nodes 3 and 5. So, in order to guide the LPA to the right choice, we rely on the strength of links leading to the maximal frequent labels. Thus, we quantify the strength of a link (x,y) between two nodes x and y by measuring the number of common neighbors between them noted $CN(x,y)$. If this value is null, we check the common neighbors between x and all the neighbors of y and so on.

Let us formalize this metric. First, we define what we call the k-step neighbors:

Table I. k-LPA SCORES FOR THE ILLUSTRATIVE NETWORK

Labels	link Strength: link $\rightarrow CN^k(,)$		Label Score
A	(4,1) $\rightarrow$ 2	(4,2) $\rightarrow$ 1	**3**
B	(4,3) $\rightarrow$ 1	(4,5) $\rightarrow$ 0	1

$$\Gamma^k(x) = \begin{cases} \{x\} & \text{if } k = 0 \\ \Gamma(x) & \text{if } k = 1 \\ \bigcup_{v_i \in \Gamma(x)} \Gamma^{k-1}(v_i) & \text{otherwise} \end{cases} \quad (1)$$

where $\Gamma(x)$ is the set of direct neighbors of the node x.

In the example:

$$\Gamma(4) = \{1,2,3,5,8\}$$
$$\Gamma^2(4) = \{1,2,3,5,8,11,9,7,6\}$$

Now, we define the number of the k-step common neighbors $CN^k(x,y)$ between the nodes x and y:

$$CN^k(x,y) = \begin{cases} |\Gamma(x) \cap \Gamma(y)| & \text{if } k = 1 \\ \sum_{v_i \in \Gamma^k(y)} CN^{k-1}(x,v_i) - |\Gamma^{k-1}(y)| & \text{otherwise} \end{cases} \quad (2)$$

In our approach, the strength of the link (x,y) is measured by $CN^k(x,y)$. A score is assigned to each maximal frequent label l by summing the strength of the links associated to the nodes carrying the label l. The label with the highest score is assigned to the node that is currently being processed. This label score at step k is defined as:

$$LabelScore(x,k) = \max_{l \in LabelMax} \sum_{\substack{y \in FreqMax \\ C(y)=l}} CN^k(x,y)$$

where:

- For a node x, *LabelMax* represents the set of maximal frequent labels, and
- For each $l \in LabelMax$, the set *FreqMax* contains the nodes labeled by l.

Let us return to our example of Figure 2. We illustrate the computation of the obtained label scores in Table I. For this example, the label assigned to node 4 is label *A*

B. k-LPA Algorithm

In the labeling process, when updating the label of node x, we delay the random choice until after reducing the set of neighbors having the maximal frequent label *FreqMax*. From this set, we discard all nodes that do not have common neighbors if they exist. Intuitively, those discarded neighbors are less attractive to their communities. Two possible scenarios may then occur:

- If the remaining set (*LabelMax*) is not empty, a label is chosen randomly from *LabelMax* and assigned to the node x.

- If none of them share neighbors, we apply the same process (checking attractiveness) over the neighbors of neighbors. This same process is repeated k steps.

The following algorithm implements this process:

1: Initialize the labels at all nodes in the network. For a given node x, $C_x(0) = x$.
2: Set $t = 1$.
3: Arrange the nodes in the network in a random order and set it to X.
4: **for all** $x \in X$ **do**
5: $k \leftarrow 0$
6: **while** (x Not Labeled) **do**
7: $LabelMax \leftarrow$ Compute-SetLabelMax(x) {compute their corresponding label}
8: $FreqMax \leftarrow$ Compute-SetFreqMax(x) {compute all x' neighbors that have maximal frequent label}
9: **if** $LabelMax < 2$ **then**
10: Label x with $label \in LabelMax$
11: **else**
12: **for all** ($v \in FreqMax$) **do**
13: Compute link-strength (x, v)
14: **end for**
15: Compute Labelscore(x, k)
16: **if** $Labelscore(x,k) \neq 0$ **then**
17:
18: **if** More than one label has maximal score **then**
19: Randomly assign a label for x
20: **else**
21: Assign this Maximal Label for x
22: **end if**
23: **end if**
24: **end if**
25: k++
26: **end while**
27: **end for**
28:
29: **if** Every node has a label that the maximum number of their neighbors have **then**
30: Stop the algorithm.
31: **else**
32: Set $t = t + 1$ and go to (3).
33: **end if**

Concerning the complexity of our algorithm, let us recall that LPA's complexity is near linear. Based on their experiments, the authors found that irrespective of the number of vertices n, 95% of the nodes or more are classified correctly by the end of iteration 5. Hence, generally LPA complexity is 5 times $O(m)$, where m is the number of edges in the network. Our k-LPA adds a multiplicative factor $k.D$ where D is the maximal degree of the network, in the worst case.

In spite of the seemingly significant increase in computational cost, our approach is still efficient in practice given the fact that social networks typically have low density and low average degree [1]. Further, in practice, k is bounded at most by 2. Indeed, in all of our experiments, we rarely exceeded two steps of neighborhood.

Table II. PARAMETERS OF THE NETWORKS GENERATED FOR THE EXPERIMENTS

Scenario	n	d	D	c_{min}	c_{max}
1000S	1000	10	50	10	50
1000B	1000	10	50	20	100
5000S	5000	15	50	10	50
5000B	5000	15	50	20	100
10000S	10000	20	50	10	50
10000B	10000	20	50	20	100

IV. EXPERIMENTS

To evaluate our LS-LPA community detection algorithm, we performed experiments on a set of synthetic networks generated from the LFR benchmark [8]. Several parameters were configured to generate different network scenarios: the number of vertices n, the average degree d of the network, the maximum degree D, the mixing parameter μ (each vertex shares a fraction μ of its edges with vertices in other communities), and c_{min} and c_{max}, the minimum and maximum community sizes respectively. These parameters are used to generate different networks with different community sizes and different degree distributions.

In Table II, we describe the 6 networks generated and used in our experiments. For each scenario of size n, $(1000, 5000, 10000)$, two networks were generated. The networks with a suffix "S" (resp. "B") have small (resp. big) communities.

We adopt the Normalized Mutual Information (NMI) measure from the field of information theory to evaluate k-LPA approach [12]. NMI is a de facto standard in the community detection literature. It estimates the proximity between actual results and found results in order to assess the degree of accuracy of the detected communities. We define a confusion matrix N, where the rows stand for the real communities and the columns for the communities obtained by the algorithm. N_{ij} is the number of nodes of the real community i that appear in the found community j. The number of real communities is denoted C_A and the number of found communities is denoted C_B. The sum over row i of matrix N is denoted N_i and the sum over column j is denoted N_j. NMI is defined as:

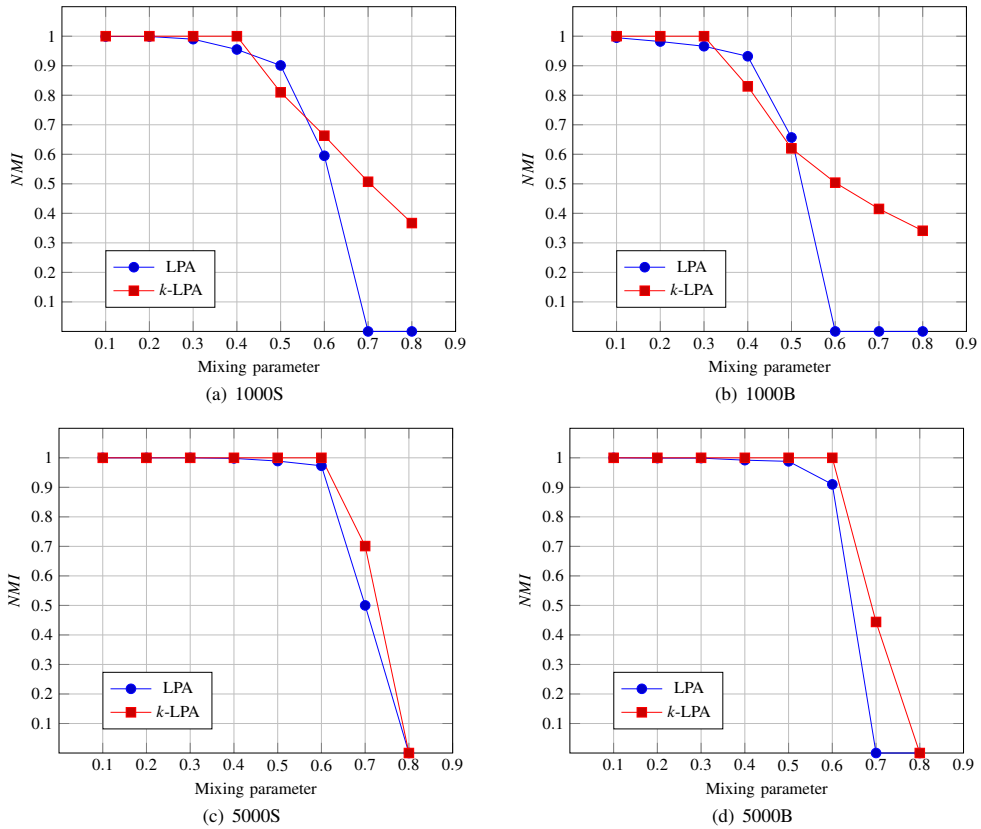
$$NMI = \frac{-2\sum_{i=1}^{C_A}\sum_{j=2}^{C_B} N_{ij} log\left(\frac{N_{ij}N}{N_i N_j}\right)}{\sum_{i=1}^{C_A} N_i log\left(\frac{N_i}{N}\right) + \sum_{j=1}^{C_B} N_j log\left(\frac{N_j}{N}\right)}$$

The value of NMI varies between 0 and 1. The closer NMI's value is to 1, the closer the detected communities are to the actual ones.

In order to evaluate the k-LPA algorithm, we ran LPA and k-LPA on all of the aforementioned network scenarios. For each one, we varied the mixing parameter μ in the interval $[0.1 \ldots 0.8]$ by a step of 0.1 and measured NMI values.

In Figure 3 and Figure 4, we report comparative results between LPA and k-LPA for the different network scenarios. In all scenarios, with small values of μ, both algorithms give

Figure 3. NMI of LPA and k-LPA on synthetic networks for $n = 1000$ and 5000.



very impressive performance. When μ increases the detecting power of both algorithms decreases with k-LPA outperforming LPA in most cases.

In the case of small networks, Figure 3.a and Figure 3.b, LPA outperforms k-LPA in the NMI interval 0.4 to 0.6. However, LPA exhibit a sharp decline after that interval, exactly, when $\mu = 0.7$ for the 1000S network (Figure 3.a) and $\mu = 0.6$ for the 1000B network (Figure 3.b). k-LPA maintains good performance even for greater values of μ. For example, $NMI = 0.362$ when $\mu = 0.8$ which means better detection ability even in a fuzzy community structure in a roughly connected network.

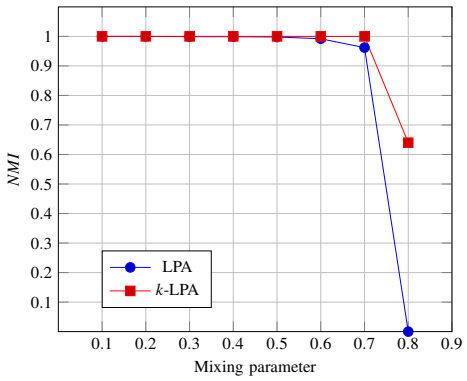
Concerning networks with 5000 nodes (Figure 3.c and Figure 3.d), both algorithms give good performance until $\mu = 0.6$. However, for greater values of μ, k-LPA gives better accuracy. In the case of larger networks, the case of $n = 10000$ nodes (Figure 4), the behavior of LPA and k-LPA remains good even for greater μ values. However, k-LPA continues to detect the community structure until $\mu = 0.75$. For all networks, with the same magnitude as LPA, k-LPA performs better in networks with large communities.

Several algorithms have been proposed to improve the original LPA. To illustrate, we compared, we compared LS-LPA with CenLP [18], a recent algorithm that tackles the same issue as our algorithm, i.e., LPA stability (as shown in Fig I in Section II.)

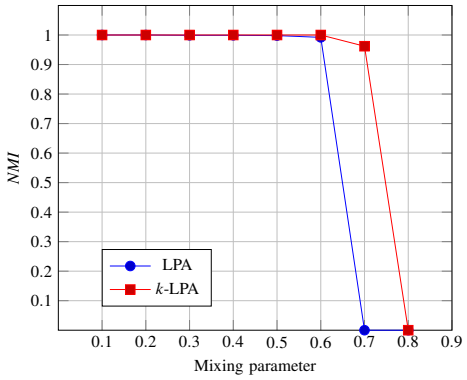
CenLP is a two-phase algorithm. The first phase gives the update order of the network nodes instead of the random order in the original LPA. The order obtained through this phase is mainly based on a new metric of centrality. The authors claim that starting by updating the border nodes then the internal nodes usually gives better performance. The second phase is the Improved LPA Label updating rule. It preserves the original principle but, in case of equal maximum number of highest frequencies, the current (to be updated) node takes the label of its most similar neighbor called preference node, a metric defined by the authors based on two other metrics *Local density*, and *Structural similarity*.

In terms of time complexity, CenLP slightly affects the near linearity of LPA in the first step to define the updating order then in the second step to define the node preference. The cost of these operations is $O(m)$ and $O(n)$ respectively. Overall, the authors claim that the complexity of their algorithm is nearly $O(n)$ in scale free networks. In contrast to our algorithm, the complexity of the original LPA is affected just by a simple factor of D, the average node degree in the network.

Figure 5 shows the performance of both algorithms in 1000 and 10000 networks respectively. The two curves have almost the same behavior. However, we should mention that, generally, in the different network scenarios, LS-LPA shows



(e) 10000S



(f) 10000B

Figure 4. NMI of LPA and k-LPA on LFR $n = 10000$.

a good performance continually even for higher values of μ. This is different from CenLP's performance, which decreases at the same values of $\mu = 0.8$.

This difference can be explained by the following fact that the further you see, the clearer the sight will be. The multistep neighborhood guarantees a wider sight from a single node, which leads to more certainty about the community structure of the network.

We also conducted experiments to measure the running time of the k-LPA . The runtime was computed in user seconds on a 2.4 GHz Intel Core i7 4500U processor (4 GB RAM) running Windows 10. Results are reported in Figure 6 and Figure 7.

We considered networks with different sizes. For each one, we measured the required time to run k-LPA and detect the communities for three values of the mixing parameter μ (0.1, 0.3, 0.6).

Figure 6 shows that the general behavior of k-LPA remains near-linear, as for the original LPA, for all scenarios, with differences between the three μ values. The higher value for the mixing parameter, the longer running time for the algorithm. This finding could be easily explained by the fact that a high value for the mixing parameter means that the majority of neighbors belong to different communities. This leads k-LPA to more investigations in the neighborhood of each node in order to find the closest neighbor that is likely to belong to the same community and propagate its labels. This gives better results than LPA for the high values of μ but requires a longer time.

For networks with large communities (Figure 7), the analysis of the runtime of k-LPA shows that its run times remain comparable to the previous case of networks with small communities.

V. Conclusion

In this paper, we presented a new algorithm for community detection called Link Strength-based Label Propagation Algorithm (LS-LPA). The proposed approaches extends the Label Propagation Algorithm (LPA). We keep the simple principle of the original algorithm but we improve its label update rule. In the case of random selection of labels among neighbors, our new version avoids the randomness by ranking the encountered labels according to the strength of links associated with neighbors holding those maximal frequent labels. Our experimental results show substantial improvements in terms of the detected communities, which proves the suitability of the approach even when we deal with large and dense networks. We plan on applying our algorithm on real world networks and on extending k-LPA to deal with overlapping communities.

References

[1] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[3] L. Donetti *et al.*, "Detecting network communities: a new systematic and efficient algorithm," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2004, no. 10, p. P10012, 2004.

[4] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75 – 174, 2010.

[5] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[6] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, p. 103018, 2010.

[7] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Transactions on Computers*, vol. C-22, no. 11, pp. 1025–1034, Nov 1973.

[8] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[9] I. X. Y. Leung, P. Hui, P. Liò, and J. Crowcroft, "Towards real-time community detection in large networks," *Phys. Rev. E*, vol. 79, p. 066107, Jun 2009.

[10] D. Liben-nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. American Society for Information Science and Technology*, 2007.

[11] L. Lu and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150 – 1170, 2011.

[12] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[13] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, p. 066133, Jun 2004.

[14] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos, "Community detection in social media," *Data Mining and Knowledge Discovery*, vol. 24, no. 3, pp. 515–554, 2011.

Figure 5. NMI of k-LPA and CenLP on synthetic networks for $n = 1000$ and 10000.

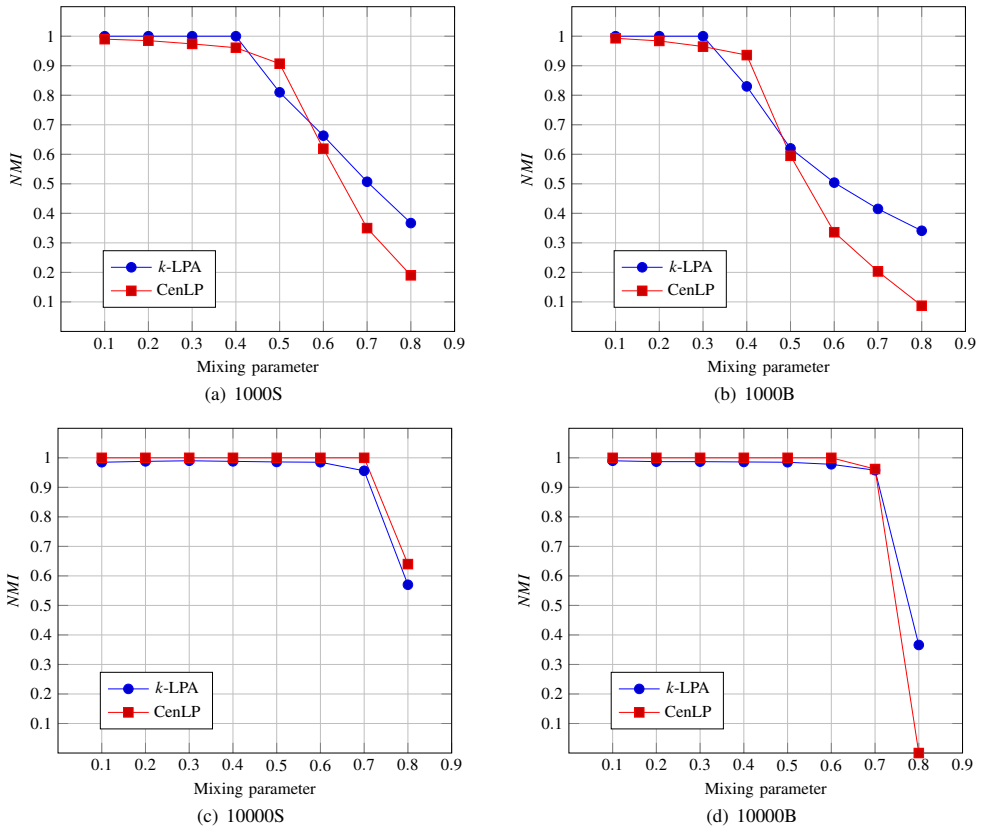


Figure 6. Running time for different values of μ: case of Small Communities

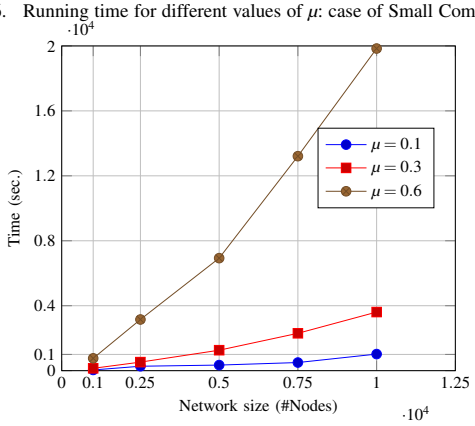
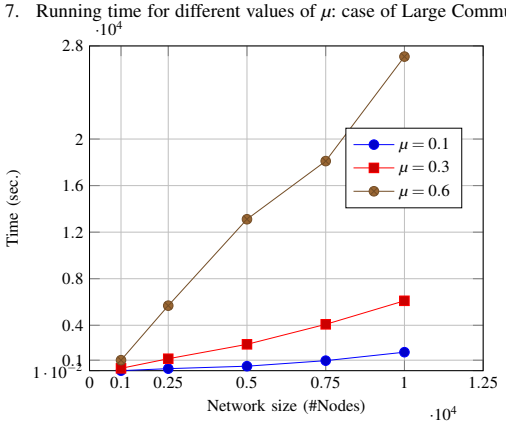


Figure 7. Running time for different values of μ: case of Large Communities



[15] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, p. 036106, Sep 2007.

[16] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[17] H. Sun, J. Huang, X. Zhong, K. Liu, J. Zou, and Q. Song, "Label propagation with alpha-degree neighborhood impact for network community detection," *Intell. Neuroscience*, vol. 2014, Jan. 2014.

[18] H. Sun, J. Liu, J. Huang, G. Wang, Z. Yang, Q. Song, and X. Jia, "Cenlp: A centrality-based label propagation algorithm for community detection in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 436, pp. 767 – 780, 2015.

[19] W. Wang and W. N. Street, "A novel algorithm for community detection and influence ranking in social networks," in *ASUNAM*, 2014.

[20] J. Xie, B. K. Szymanski, and X. Liu, "Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *2011 IEEE 11th International Conference on Data*

Mining Workshops, Dec 2011, pp. 344–349.

[21] J. Xie, M. Chen, and B. K. Szymanski, "LabelRankT: incremental community detection in dynamic networks via label propagation," in *Proceedings of the Workshop on Dynamic Networks Management and Mining, DyNetMM 2013, New York, New York, USA, June 22-27, 2013*, 2013, pp. 25–32.

[22] J. Xie and B. K. Szymanski, "Community detection using a neighborhood strength driven label propagation algorithm," in *Proceedings of the 2011 IEEE Network Science Workshop*, ser. NSW '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 188–195.

[23] ——, "Labelrank: A stabilized label propagation algorithm for community detection in networks," in *Proceedings of the 2nd IEEE Network Science Workshop, NSW 2013, April 29 - May 1, 2013, Thayer Hotel, West Point, NY, USA*, 2013, pp. 138–143.

[24] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "Scan: a structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 824–833.

[25] X. Zhang and M. E. J. Newman, "Multiway spectral community detection in networks," *Phys. Rev. E*, vol. 92, p. 052808, Nov 2015.