LabelRank: A Stabilized Label Propagation Algorithm for Community Detection in Networks

Jierui Xie
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York 12180
Email: jierui.xie@gmail.com

Boleslaw K. Szymanski
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York 12180
Email: szymab@rpi.edu

Abstract—An important challenge in big data analysis nowadays is detection of cohesive groups in large-scale networks, including social networks, genetic networks, communication networks and so. In this paper, we propose *LabelRank*, an efficient algorithm detecting communities through label propagation. A set of operators is introduced to control and stabilize the propagation dynamics. These operations resolve the randomness issue in traditional label propagation algorithms (LPA), stabilizing the discovered communities in all runs of the same network. Tests on real-world networks demonstrate that LabelRank significantly improves the quality of detected communities compared to LPA, as well as other popular algorithms.

Keywords: *social network analysis, community detection, clustering, group*

I. INTRODUCTION

One type of the basic structures of sociology in general and social networks in particular are *communities* (e.g. see [2]). In sociology, community usually refers to a social unit that shares common values and both the identity of the members and their degree of cohesiveness depend on individuals' social and cognitive factors such as beliefs, preferences, or needs. The ubiquity of the Internet and social media eliminated spatial limitations on community range, resulting in online communities linking people regardless of their physical location. The newly arising *computational sociology* relies on computationally intensive methods to analyze and model social phenomena [1], including communities and their detection. Analysis of social networks has been used as a tool for linking micro and macro levels of sociological theory. The classical example of the approach is presented in [3] that elaborated the macro implications of one aspect of small-scale interaction, the strength of dyadic ties. Communities in social networks are discovered based on the observed interactions between people. With the rapid emergence of large-scale online social networks, e.g., Facebook that connected a billion users in 2012, there is a high demand for efficient community detection algorithms that will be able to handle large amount of data on a daily basis. Numerous techniques have been developed for community detection. However, most of them require a *global* view of the network. Such algorithms are not scalable enough for networks with millions of nodes.

Label propagation based community detection algorithms such as LPA [4], [5] and SLPA [7], [6] (whose source codes are publicly available at https://sites.google.com/site/communitydetectionslpa/) require only local information. They have been shown to perform well and be highly efficient. However, they come with a great shortcoming. Due to random tie breaking strategy, they produce different partitions in different runs. Such instability is highly undesirable in practice and prohibits its extension to other applications, e.g., tracking the evolution of communities in a dynamic network.

In this paper, we propose strategies to stabilize the LPA and to extend MCL [8] approach that resulted in a new algorithm called *LabelRank* that produces deterministic partitions.

II. RELATED WORK

Despite the ambiguity in the definition of community, numerous techniques have been developed including Random walks [10], [11], [12], spectral clustering [13], [14], [15], modularity maximization [16], [17], [18], [19], [20], and so on. A recent review can be found in [25]. Label propagation and random walk based algorithms are most relevant to our work.

The LPA [4] uses the network structure alone to guide its process. It starts from a configuration where each node has a distinct label. At every step, each node changes its label to the one carried by the largest number of its neighbors. Nodes with same label are grouped together after convergence. The speed of LPA is optimized in [5]. Leung [27] extends LPA by incorporating heuristics like hop attenuation score. COPRA [9] and SLPA [6] extend LPA to detection of overlapping communities by allowing multiple labels. However, none of these extensions resolves the LPA randomness issue, where different communities may be detected in different runs over the same network.

Markov Cluster Algorithm (MCL) proposed in [8] is based on simulations of flow (random walk). MCL executes repeatedly matrix multiplication followed by inflation operator. LabelRank differs from MCL in at least two aspects. First, LabelRank applies the inflation to the label distributions and not to the matrix M. Second, the update of label distributions on each node in LabelRank requires only local information. Thus it can be computed in a decentralized way. Regularized-MCL [23] also employs a local update rule of label propagation operator. Despite that, the authors observed that it still suffers from the scalability issue of the original MCL. To remedy, they

introduced Multi-level Regularized MCL, making it complex. In contrast, we address the scalability by introducing new operator, conditional update, and the novel stopping criterion, preserving the speed and simplicity of the LPA based algorithms.

III. LabelRank Algorithm

LabelRank is based on the idea of simulating the propagation of labels in the network. Here, we use node id's as labels. LabelRank stores, propagates and ranks labels in each node. During LabelRank execution, each node keeps multiple labels received from its neighbors. This eliminates the need of tie breaking in LPA[4] and COPRA [9] (e.g., multiple labels with the same maximum size or labels with the same probability). Nodes with the same highest probability label form a community. Since there is no randomness in the simulation, the output is deterministic. LabelRank relies on four operators applied to the labels: (i) propagation, (ii) inflation, (iii) cutoff, and (iv) conditional update.

Propagation: In each node, an entire distribution of labels is maintained and spread to neighbors. We define n $1 \times n$ vectors P_i (n is the number of nodes) which are separate from adjacency matrix A defining the network structure. Each element $P_i(c)$ or P_{ic} holds the current estimation of probability of node i observing label $c \in C$ taken from a finite set of alphabet C. For clarity of discussion, we assume here that $C = \{1, 2, \ldots, n\}$ (same as node id's) and $|C| = n$. In Section IV we lift this assumption to increase efficiency of execution. In LabelRank, each node broadcasts the distribution to its neighbors at each time step and computes the new distribution $P_i^{'}$ simultaneously using the following equation:

$$P_i^{'}(c) = \sum_{j \in Nb(i)} P_j(c)/k_i, \forall c \in C, \quad (1)$$

where $Nb(i)$ is a set of neighbors of node i and $k_i = |Nb(i)|$ is the number of neighbors. Note that, $P_i^{'}$ is normalized to make a probability distribution.

In matrix form this operator can be expressed as:

$$A \times P, \quad (2)$$

where A is the $n \times n$ adjacency matrix and P is the $n \times n$ *label distribution matrix*. To initialize P, each node is assigned equal probability to see each neighbor:

$$P_{ij} = 1/k_i, \forall j \text{ s.t. } A_{ij} = 1. \quad (3)$$

Since the metric space A is usually compact, P defined iteratively by Eq. 2 converges to the same stationary distribution for most networks by the Banach fixed point theorem [21]. Hence, a method is needed for trapping the process in some local optimum in the quality space (e.g., modularity Q [22]) without propagating too far.

Inflation: As in MCL [8], [23], we use the inflation operator Γ_{in} on P to contract the propagation, where in is the parameter taking on real values. Unlike MCL, we apply it to the label distribution matrix P (rather than to a stochastic matrix or adjacency matrix) to decouple it from the

network structure. After applying $\Gamma_{in} P$ (Eq. 4), each $P_i(c)$ is proportional to $P_i(c)^{in}$, i.e., $P_i(c)$ rises to the in^{th} power.

$$\Gamma_{in} P_i(c) = P_i(c)^{in} / \sum_{j \in C} P_i(j)^{in}. \quad (4)$$

This operator increases probabilities of labels that were assigned high probability during propagation at the cost of labels that in propagation received low probabilities. For example, two labels with close initial probabilities 0.6, and 0.4 after $\Gamma_{in=2}$ operator will changed probabilities to 0.6923 ad 0.3077, respectively. In our tests, this operator helps to form local subgroups. However, it alone does not provide satisfying performance in large networks. Moreover, the memory inefficiency problem implied by Eq. 2, i.e., n^2 labels stored in the networks, is not yet fully resolved by the inflation operator.

Cutoff: To alleviate the memory problem, we introduce cutoff operator Φ_r on P to remove labels that are below threshold $r \in [0, 1]$. As expected, Φ_r constrains the label propagation with help from inflation that decreases probabilities of labels to which propagation assigned low probability. More importantly, Φ_r efficiently reduces the space complexity, from quadratic to linear. For example, with $r = 0.1$, the average number of labels in each node is typically less than 3.0.

Explicit Conditional Update: As shown in Fig. 1 (green curve), the above three operations are still not enough to guarantee good performance. This is because the process detects the highest quality communities far before convergence, and after that, the quality of detected communities decreases. Hence, we propose here a novel solution based on the *conditional update operator* Θ. It updates a node only when it is significantly different from its neighbors in terms of labels. This allows us to to preserve detected communities and detect termination based on scarcity of changes to the network. At each iteration, the change is accepted only by nodes that satisfy the following update condition:

$$\sum_{j \in Nb(i)} isSubset(C_i^*, C_j^*) \le q k_i, \quad (5)$$

where C_i^* is the set of *maximum labels* which includes labels with the maximum probability at node i at the *previous* time step. Function $isSubset(s_1, s_2)$ returns 1 if $s_1 \subseteq s_2$, and 0 otherwise. k_i is the degree of node i, and q is a real number parameter chosen from the interval $[0, 1]$. Intuitively, isSubset can be viewed as a measure of *similarity* between two nodes. As shown in Fig. 1, Θ_q operator successfully traps the process in the modularity space with high quality, indicated by a long-lived plateau in the modularity curve (red curves). Equation 5 augments the stability of the label propagation.

Stop criterion: One could define the *steady* state of a node as small difference in the label distribution between consecutive iterations, and determine the overall network state built upon node states. In fact, the above conditional update allows us to derive a more efficient stop criterion (linear time). We determine whether the network reaches a relatively stable state by tracking the number of nodes that update their label distributions (i.e., implicitly tracking the number of nodes that potentially change their communities), $numChange$, at each iteration and accumulate the number of repetitions $count(numChange)$ in a hash table. The algorithm
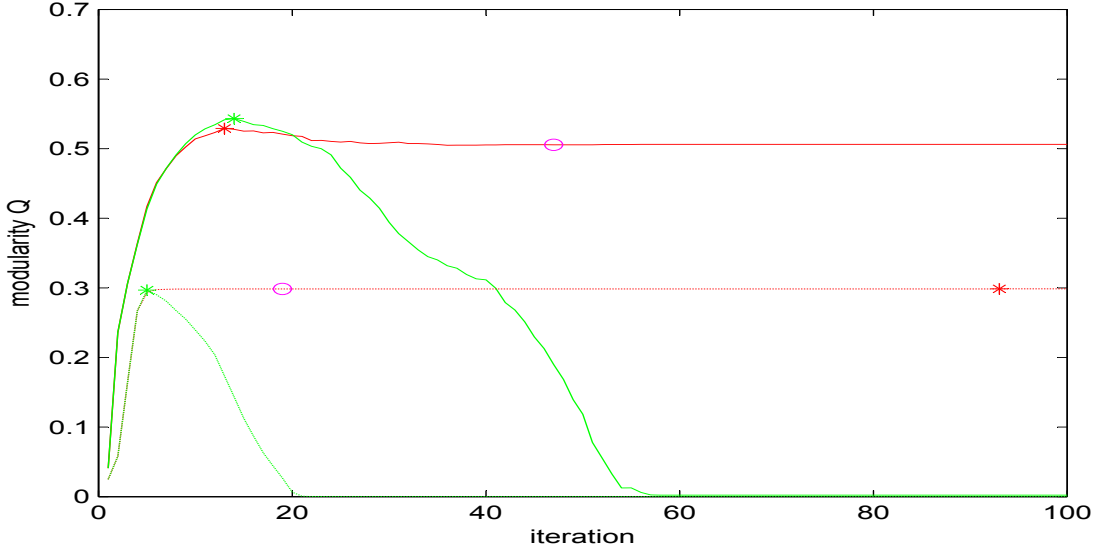
Fig. 1. The effect of conditional update operator. The plot shows the modularity Q over iterations on the email network with $n = 1,133$ (two curves on the top) and wiki network with $n = 7,066$ (two curves at the bottom). Each Q is computed explicitly for each iteration. Green curve is based on three operators Propagation+Inflation+Cutoff. Red curve is based on four operators Propagation+Inflation+Cutoff+Conditional Update. Asterisk indicates the best performance of $Q(t)$. Purple circle indicates the Q achieved when the stop criterion described in the main text is used.

stops when the *count* of any *numChange* first exceeds some predefined frequency (e.g., five in our experiments), or no change for this iteration (i.e., numChange=0).

Although such criterion does not guarantee the best performance, it almost always returns satisfying results. The difference between the found Q (purple circles) and maximum Q (red asterisks) is small as illustrated on two networks in Fig. 1. Note that, this stop criterion is also applicable when network state oscillates among a group of states.

Algorithm 1 LabelRank

1: add selfloop to adjacency matrix A
2: initialize label distribution P using Eq. 3
3: **repeat**
4:     $P' = A \times P$
5:     $P' = \Gamma_{in} P'$
6:     $P' = \Phi_r P'$
7:     $P = \Theta_q(P', P)$
8: **until** stop criterion satisfied
9: output communities

These four operators together with a post-processing that groups nodes whose highest probability labels are the same into a community form a complete algorithm (see Alg. 1). An example network as output by LabelRank is shown in Fig. 2. There are only 1.2 labels on average and at most two in each node, resulting in a sparse label distribution (Table I of which second row shows for each node the label with the highest probability identifying this node community). Three communities are identified, each sharing a common label: red community label 3, green community label 5 and blue community label 11. The resultant P also distinguishes two types of nodes, the *border* ones with high probability labels (e.g., 3, 5 and 11), and the *core* nodes with positive but not largest label probabilities (e.g., 1, 13 and 15). The latter are well connected to their communities.

In the analysis, we set the length of P_i at n, creating a $n \times n$ P matrix. In the implementation, this is not needed. Thanks to both cutoff and inflation operators, the number of labels in each node monotonically decreases and drops to a small constant in a few steps. The P matrix is replaced by n variable-length vectors (usually short) carried by each node (as illustrated in Table I). Another advantage is that the algorithm performance is not sensitive to the cutoff threshold r, so we set it to 0.1, and do not consider when tuning parameters for optimal performance.

It turns out that the preprocessing that adds a selfloop to each node (i.e., $A_{ii} = 1$) helps to improve the detection quality. The selfloop effect resembles the lazy walk in a graph that avoids the periodicity problem, but here, it smooths the propagation (update of P_i) by taking into account node's own label distribution. Thus during initialization, each node considers itself a neighbor while using Eq. 1.

Both LabelRank and MCL use matrix multiplication, $A \times P$ for LabelRank and $M \times M$ for MCL (M is the $n \times n$ stochastic matrix). For updating an element, both $P_{ij} \leftarrow A_{i.} \times P_{.j}$ and $M_{ij} \leftarrow M_{i.} \times M_{.j}$ seem to require $O(n)$ operations, where $X_{i.}$ denotes the ith row and $X_{.j}$ denotes the jth column of matrix X. However, since A represents the static network structure, no operations are needed for zero entries in A for LabelRank. Thus, the number of effective operations for each node is defined by k_i neighbors, reducing the time for computing the P_{ij} to $O(k_i)$. With x labels (typically less than 3) in each node on average, updating one row P_i requires $O(xk_i)$ operations. As a result, the time for updating the entire P in LabelRank is $O(x\overline{k}n) = O(xm) = O(m)$, where $\overline{k}$ is the average degree and m is the total number of edges. In contrast, during the expansion (before convergence), M_{ij} of M that rises to power larger than 1 is changed according to the definition of transition matrix of a random walk. After that, values in M_{ij} no longer
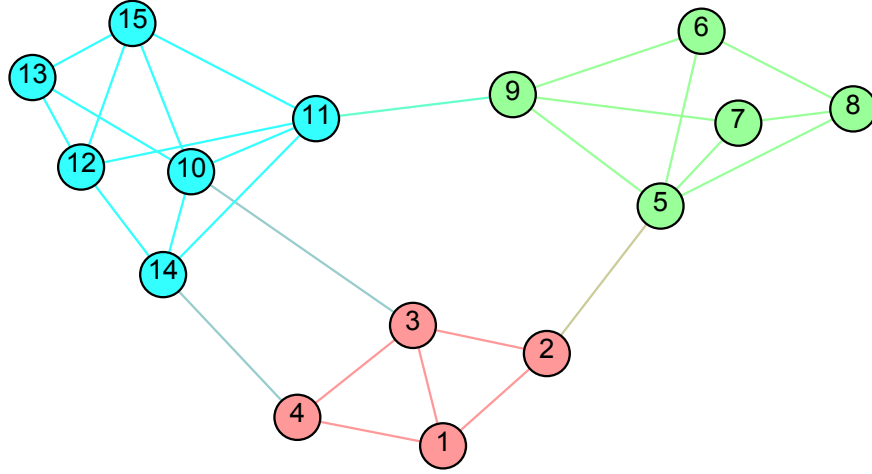
Fig. 2. The example network G(0) with $n = 15$. Colors represent communities discovered by LabelRank (see table I) with cutoff $r = 0.1$, inflation $in = 4$, and conditional update $q = 0.7$. The algorithm stopped at the 7th iteration. The average number of labels dropped from 2.933 to 1.2 during the simulation.

TABLE I.   A SPARSE REPRESENTATION OF THE RESULTANT MATRIX P ON THE EXAMPLE GRAPH G(0) THAT DEFINES PROBABILITY OF EACH LABEL FOR EACH NODE. NOTE THAT FOR THIS MATRIX WITH $N = 13$ NODES, THERE ARE AT MOST TWO LABELS WITH NON-ZERO PROBABILITY FOR EACH NODE.

Node Identifier	$Label_1$	$Probability_1$	$Label_2$	$Probability_2$
1	**3**	**0.721**	1	0.279
2	**3**	**1**	-	-
3	**3**	**1**	-	-
4	**3**	**1**	-	-
5	**5**	**1**	-	-
6	**5**	**1**	-	-
7	**5**	**1**	-	-
8	**5**	**1**	-	-
9	**5**	**1**	-	-
10	**11**	**1**	-	-
11	**11**	**1**	-	-
12	**11**	**1**	-	-
13	**11**	**0.797**	10	0.203
14	**11**	**1**	-	-
15	**11**	**0.874**	10	0.126

reflects the network connections in one hop. Therefore, the computation of M_{ij} may require nonlocal information and the time is $O(n)$, which leads to $O(nm)$ for the entire $M \times M$ operator in worst case. In conclusion, the propagation scheme in LabelRank is highly parallel and allows the computation to distribute to each individual node.

The running time of LabelRank is $O(m)$, linear with the number of edges m because adding selfloop takes $O(n)$, the initialization of P takes $O(m)$, each of the four operators takes $O(m)$ on average and the number of iterations is usually $O(1)$. Note that, although sorting the label distribution is required in conditional update, it takes effectively linear time because the size of label vectors is usually no more than 3. The execution times on a set of citation networks are shown in Fig. 3. The test ran on a single computer, but we expect further improvement on a parallel platform.

IV. EVALUATION ON REAL-WORLD NETWORKS

We first verified the quality of communities reported by our algorithm on networks for which we know the true grouping. For the classical Zachary's karate club network [28] with $n = 34$, LabelRank discovered exactly the two

existing communities centered on the teacher and manager (with $Q = 0.37$).

We also used a set of high school friendship networks [6] created by a project funded by the National Institute of Child Health and Human Development. The results on this large data set are similar and show a good agreement between the found and known partitions. An instance is shown in Fig. 4.

We also tested LabelRank on a wider range of large social networks availbale at snap.stanford.edu/data/ and compared its performance with other known algorithms including LPA with synchronous update [4], MCL that uses a similar inflation operator [8] and one of the state-of-the-art algorithms, Infomap [25]. Since the output of LPA is nondeterministic, we repeated the algorithm 10 times and reported the best performance. For MCL, the best performance from inflation in the range of $[1.5, 5]$ is shown. For LabelRank, q is 0.5 or 0.6, in is the best from the set $\{1, 1.5, 2\}$. Due to the lack of knowledge of true partitioning in most networks, we used modularity as the quality measure [22]. The detection results are shown in Table. II.

As shown, LPA works well on only two networks with relatively dense average connections ($\overline{k} \approx 10$): football and
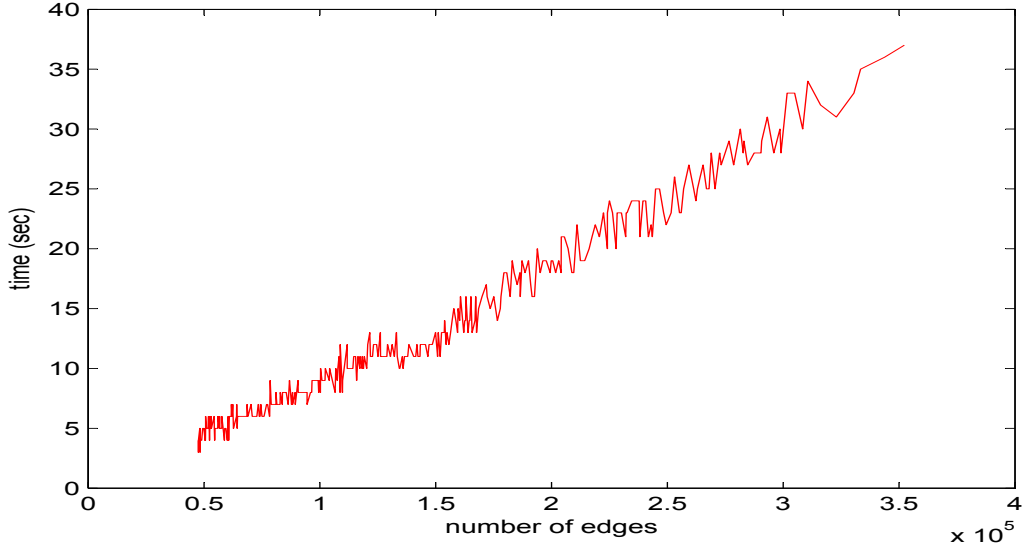
Fig. 3. The execution times on a set of arXiv high energy physics theory citation graphs [26] with n ranging from 12,917 to 27,769 and m from 47,454 to 352,285. Tested on a desktop with Intel@2.80GHz.
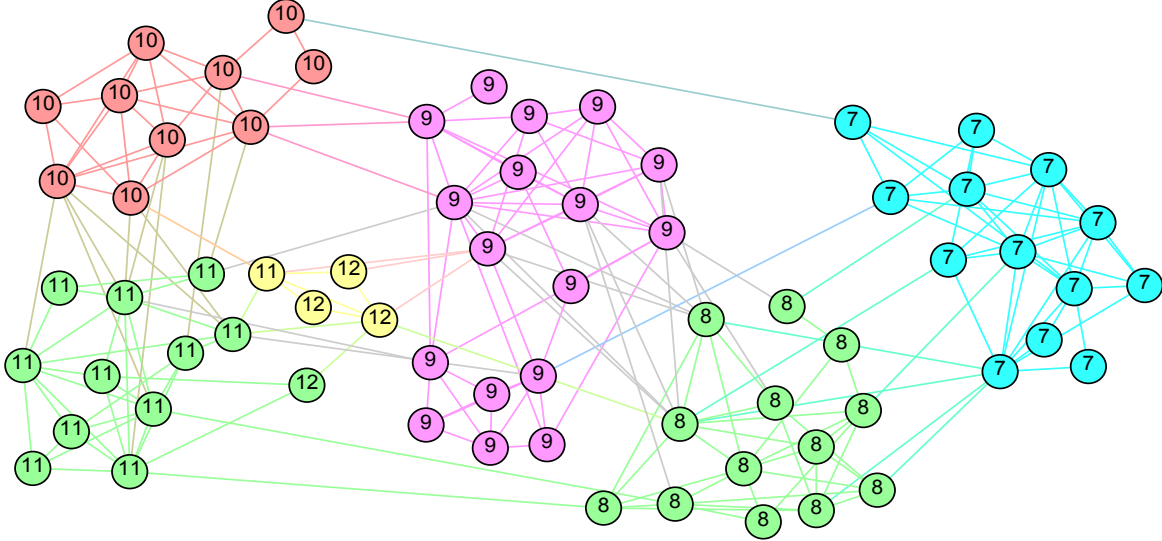

Fig. 4. Communities detected on a HighSchool friendship network ($n = 69$, $\overline{k} = 6.4$). Labels are the known grades ranging from 7 to 12. Colors represent communities discovered by LabelRank.

HighSchool networks. In general, it performs worse than the other three algorithms. However, with the stabilization strategies introduced in this paper, LabelRank, a generalized and stable version of LPA, boosts the performance significantly, e.g., with an increase of 28.57% on PGP and 87.1% on Enron Email. More importantly, LPA drawback is that it might easily lead to a trivial output (i.e., a single giant community). For instance, it completely fails on Eva and Epinions. The conditional update in LabelRank appears to provide a way to prevent such undesired output. As a result, LabelRank allows label propagation algorithms to work on a wider range of network structures, including both Eva and Epinions.

LabelRank outperforms MCL significantly on HighSchool, Epinions and Enron Email by 10%, 20.83% and 25.93%

respectively. This provides some evidence that there is an advantage of separating network structure captured in adjacency matrix A from the label probability matrix P, as done in our LabelRank algorithm. LabelRank and Infomap have close performance. LabelRank outperforms Infomap on HighSchool and Epinions by 10.34% and 9.43% respectively, while Infomap outperforms LabelRank on Epinions by 11.76%.

V. CONCLUSIONS

In this paper, we introduced operators to stabilize and boost the LPA, which avoid random output and improve the performance of community detection. We believe the stabilization is important and can provide insights to an entire family of label propagation algorithms, including SLPA and COPRA.

TABLE II.   THE MODULARITY Q'S OF DIFFERENT COMMUNITY DETECTION ALGORITHMS.

Network	n	LPA	LabelRank	MCL	Infomap
Football [29]	115	0.60	0.60	0.60	0.60
HighSchool	1,127	0.66	0.66	0.60	0.58
Eva	4,475	-	0.89	0.89	0.89
PGP [30]	10,680	0.63	0.81	0.80	0.81
Enron Email	33,696	0.31	0.58	0.48	0.53
Epinions	75,877	-	0.34	0.27	0.38
Amazon [33]	262,111	0.73	0.76	0.76	0.77

Stabilizing label propagation is our first step towards distributed dynamic network analysis. We are working on extending LabelRank for community detection for evolving networks, where new data come in as a stream. With such possible extension, we will be able to design efficient algorithms (e.g., distributed social-based message routing algorithm) for highly distributed and self-organizing applications such as ad hoc mobile networks and P2P networks. We also plan to extend LabelRank to overlapping community detection [24] in the near future. In the experiments, we explored and demonstrated the good detection quality on some real-world networks. We are parallelizing our algorithm for millions of nodes networks.

REFERENCES

[1] W. S. Bainbridge. Computational Sociology. In *Blackwell Encyclopedia of Sociology*, Blackwell Reference Online, 2007, doi:10.1111/b.9781405124331.2007.x.

[2] R. E. Park. Human communities: The city and human ecology, Free Press, 1952.

[3] M. S. Granovetter The strength of weak ties. *American journal of sociology*, 1360–1380, 1973.

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

[5] J. Xie and B. K. Szymanski. Community detection using a neighborhood strength driven label propagation algorithm. In *IEEE Network Science Workshop 2011*, pages 188-195, 2011.

[6] J. Xie and B. K. Szymanski. Towards linear time overlapping community detection in social networks. In *PAKDD*, pages 25-36, 2012.

[7] J. Xie, B. K. Szymanski and X. Liu. SLPA: Uncovering Overlapping Communities in Social Networks via A Speaker-listener Interaction Dynamic Process. In *Proc. of ICDM 2011 Workshop*, 2011.

[8] S. van Dongen. A cluster algorithm for graphs. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science, 2000.

[9] S. Gregory. Finding overlapping communities in networks by label propagation. *New J. Phys.*, 12:103018, 2010.

[10] H. Zhou and R. Lipowsky. Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. *Lect. Notes Comput. Sci.*, 3038:1062-1069, 2004.

[11] Y. Hu, M. Li, P. Zhang, Y. Fan and Z. Di. Community detection by signaling on complex networks. *Phys. Rev. E.*, 78:1, pp. 016115, 2008.

[12] Pascal Pons and Matthieu Latapy. Computing Communities in Large Networks Using Random Walks. *J. Graph Algorithms Appl.*, 10:2, pp. 191-218, 2006.

[13] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:8, pp. 888-90, 2000.

[14] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. *Proc. of SIAM International Conference on Data Mining*, pp. 76-84, 2005.

[15] A. Capocci, V.D.P. Servedio, G. Caldarelli and F. Colaiori. Detecting communities in large networks. *Physica A*, 352, pp. 669-676, 2005.

[16] V. Blondel, J. Guillaume, R. Lambiotte and E. Lefebvre. Fast Unfolding of Communities in Large Networks. *J. Stat. Mech.*, 2008.

[17] M. E. J. Newman. Finding Community Structure in Networks Using the Eigenvectors of Matrices. *Phys. Rev. E*, 74, pp. 036104, 2006.

[18] M. E. J. Newman and M. Girvan. Finding and Evaluating Community Structure in Networks. *Phys. Rev. E*, 69, pp. 026113, 2004.

[19] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks. In *WWW Conference*, pp. 1275-1276, 2007.

[20] P. Schuetz and A. Caflisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Phys. Rev. E*, 77, pp. 046112, 2008.

[21] S. Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. Math.*, 1922.

[22] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, 2004.

[23] V. Satuluri, and S. Parthasarathy. Label graph clustering using stochastic flows: applications to community discovery. In *SIGKDD*, pages 737-746, 2009.

[24] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *ACM Computing Surveys* 45(12), December 2013.

[25] S. Fortunato. Community detection in graphs. *Physics Reports*, 486:75-174, 2010.

[26] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *SIGKDD*, pages 177-187, 2005.

[27] I. Leung, P. Hui, P. Lio, and J. Crowcroft. Towards real-time community detection in large networks. *Phys. Rev. E*, 79:066107, 2009.

[28] W.W. Zachary. An Information Flow Model for Conflict and Fission in Small Groups. *Journal of Anthropological Research*, 33, pp. 452-473, 1977.

[29] M. Girvan and M. E. J. Newman. Community Structure in Social and Biological Networks. *Proc Natl Acad Sci USA*, 99, pp. 7821-7826, 2002.

[30] M. Boguna and R. Pastor-Satorras and A. Diaz-Guilera and A. Arenas. Models of social networks based on social distance attachment. *Phys. Revi. E*, 70, pp. 056122, 2004.

[31] M. Richardson and R. Agrawal and P. Domingos. Trust Management for the Semantic Web. In *Ineternational Semantic Web Conference*, 2003.

[32] J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Mathematics*, 6(1) pp. 29-123, 2009.

[33] L. Leskovec and B. Adamic. The Dynamics of Viral Marketing. *ACM Transactions on the Web*, 1(1), 2007.