applied sciences

MDPI

Article

Modularity-Based Incremental Label Propagation Algorithm for Community Detection

Yunlong Ma [1], Yukai Zhao [1], Jingwei Wang [1], Min Liu [1], Weiming Shen [2] and Yumin Ma [1,*]

[1] School of Electronic and Information Engineering, Tongji University, Shanghai 201804, China; evanma@tongji.edu.cn (Y.M.); zhaoyukaijake@tongji.edu.cn (Y.Z.); jwwang@tongji.edu.cn (J.W.); lmin@tongji.edu.cn (M.L.)

[2] School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China; wshen@ieee.org

* Correspondence: ymma@tongji.edu.cn

Received: 26 April 2020; Accepted: 9 June 2020; Published: 12 June 2020

Abstract: Label Propagation Algorithm (LPA) is a fast community detection algorithm. However, since each node is randomly assigned a different label at first, there is serious randomness in the label updating process of LPA, resulting in great instability of detection results. This paper proposes a modularity-based incremental LPA (MILPA) to address this problem. Unlike LPA, MILPA first assigns all nodes the same label, and then repeatedly uses divide strategy to split locally dense connected nodes into a community and give them a new label. After that, MILPA uses modularity gain as the optimization function to fine-tune the label of nodes so as to obtain an optimal partition. The proposed MILPA has been compared with LPA and other known methods. Experimental results show that MILPA has the best and most stable performance in LFR benchmark networks and is comparable to the best algorithm in many real networks.

Keywords: label propagation; community detection; modularity

1. Introduction

Community structure is an important feature of complex networks [1,2]. It can help us understand the nature or function of networks [3]. For instance, communities are likely to group proteins having the same specific function within the cell and they play a particularly important role in our understanding of how specific biological functions are encoded in cellular networks [4–6]. In most cases, however, the community structure of a network is not known in advance and needs to be detected by algorithms. Therefore, developing algorithms to detect community structure, i.e., community detection, has been one of the most widely studied topics in network science. From a classical view, community detection is a clustering process to detect communities in a large network, because the edge density of nodes inside each community is greater than that between communities. However, the current view focuses on the probability that edges may be generated between nodes. It indicates that community is a preferential linking pattern [7,8].

In the last two decades, many scholars have performed extensive studies on how to identify the community structure of a complex network. A number of powerful methods have been put forward to address this problem, which can be generally divided into three categories: divisive, modularity optimization based, and agglomerative algorithms.

Divisive algorithms aim to detect inter-community edges and then remove these links from the network, including the GN algorithm proposed by Newman [9]. However, GN has a high computational complexity, $O(n^3)$, where n is the number of nodes in a network. To improve the efficiency of GN, Radicchi et al. [10] proposed a fast splitting algorithm to find the edge set with the

minimum clustering coefficient (C) by calculating the C of each edge in a network, and then removing these edges to uncover the community structure.

The second category of algorithms are based on modularity optimization [11–14]. Modularity was originally introduced by Newman and Girvan to define a stopping metric for the GN algorithm, and then Newman built on it to design a fast greedy algorithm (Fastgreedy) [15], which improves the execution speed of GN. Yet, in a later study [16], Clauset et al. pointed out that the update of the adjacency matrix in Newman's algorithm involves a great number of useless operations. Thus, they utilized complex data structure, such as balanced binary trees and the largest heap, to store data, drastically reducing the complexity of the Fastgreedy algorithm. Another well-known and fast approach is the Louvain algorithm proposed by Blondel et al. [17], which involves two steps. In the first step of modularity optimization, each node is put into the community of the neighbor that yields the largest increase of modularity as long as it is positive. In the second step of community aggregation, communities resulted from the first step are aggregated into new nodes to reconstruct a new network. The two steps of this algorithm are then repeated until the communities do not change.

Examples of agglomerative algorithms include LPA proposed by Raghavan et al. [18]. The advantages of LPA are simple and easy to realize, and it is unnecessary to predefine the number and the size of communities. It only needs to use the similarity of neighbor nodes, that is, whether they are closely connected in this community, to determine whether they belong to the same community. In addition, LPA has linear time complexity so it can identify the community structure of large networks. However, as LPA always randomly selects a node to start spreading labels, which makes some nodes that are neither tightly connected nor sparsely allocated to different communities during each update process, resulting in some instability of detection results. To solve this problem, Barber et al. [19] proposed a modularity-specialized LPA (LPAm) which uses modularity as an objective function for optimization, so that the results of community detection are always in the direction of increasing modularity. However, one obvious defect of this algorithm is that it is easy to fall into a local optimal and it may misclassify the community of nodes. Based on LPA and LPAm, Li et al. [20] proposed another improved LPA, called LPAMP, which first optimizes the modularity to obtain a coarse clustering of nodes and then performs label propagation. LPAMP can reduce the randomness of the LPA and improve the stability and accuracy of the community detection results, but it is slow and cannot be used for large networks.

To solve the instability problem of LPA mentioned above, a modularity-based incremental LPA (MILPA), is proposed in this paper. First, the node degree and the node membership are introduced to determine initial community of nodes. After all nodes have initial labels, we define an objective function based on modularity gain to guide the label updating of nodes. Experiments on both synthetic networks and real-world networks show that the proposed MILPA greatly reduces the randomness of the label updating process and has stable and great performance in community detection.

2. Methods and Materials

2.1. LPA

Here, we first introduce the basic steps of LPA [21]. Nodes in a network are initially given unique labels. All nodes, in a random sequential order, perform this operation where each node takes the label shared by the majority of its neighbors. If there is no unique majority, one of the majority labels is picked randomly. In this way, labels propagate across the network: most labels will disappear, and others will dominate. The process reaches a convergence when each node has the majority label of its neighbors. Communities are defined as groups of nodes having identical labels at convergence. By construction, each node has more neighbors in its community than in any other communities. The algorithm does not deliver a unique solution. Due to the random initialization of labels and many ties encountered along the process of label propagating, it is possible to derive completely different partitions every time running the LPA on the same network, resulting in an instability problem, as mentioned above.

2.2. Evaluation Metrics

2.2.1. Normalized Mutual Information

For a network with a known community structure, normalized mutual information (NMI) [22] can be used to evaluate the coincidence degree between the ground truth and the result detected by one algorithm, so as to measure the quality of this community detection algorithm. NMI is defined as

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

where C_A represents the ground truth, C_B denotes the result detected by one algorithm N is a mixing matrix where its row number is the number of real communities and its column number is the number of communities detected by the algorithm. N_{ij} represents the number of nodes in real community i in community j obtained by the algorithm, N_i and N_j denote the sum of i row and the sum of j column, respectively. Following can be known by analyzing Equation (1):

1. $NMI = 1$ when the community partition generated by one algorithm is consistent with the real community structure.
2. $NMI = 0$ when the community partition generated by one algorithm is the opposite of the real community structure.
3. $NMI \in [0,1]$ when the community partition generated by one algorithm is partly similar to the real community structure. The closer to 1 the NMI value, the closer to the real community structure the community detection result, and the better the performance of the algorithm.

2.2.2. Modularity

When the real community structure of a network is unknown, modularity [23] proposed by Newman is the most popular quality function to assess the community detection result of one algorithm. According to the general definition of community structure, nodes within the same community are densely connected, and nodes in different communities are sparsely connected. Modularity is used to measure whether a network has such a community structure, defined as

$$Q = \sum_{c=1}^{n_c}\left[\frac{l_c}{m} - \left(\frac{d_c}{2m}\right)^2\right] \quad (2)$$

where m is the total number of edges in a graph, n_c is the number of communities, l_c is the number of edges joining nodes of community l_c and d_c is the sum of degrees of the nodes of C. In Equation (2), the first term of each summand is the part of edges of the network within the community, whereas the second term represents the expected part of edges that would be there if the network was a random graph with the same expected degree for each node. High values of modularity indicate good community structure.

2.3. MILPA

To solve the instability problem mentioned in the introduction, this paper proposes the MILPA to improve the performance of LPA. There are two steps in the label propagation process of MILPA. The first step is incremental label propagation, and the other step is modularity-based label updating. At the starting point, N nodes are given N unique labels in LPA. Unlike LPA, MILPA uses the opposite strategy that all nodes are assigned the same label so that there is only one label at first. Then the whole graph is partitioned into two groups and a new label is emerged. As this procedure goes on, the number of labels increases gradually, so it is called incremental label propagation. After the end of this process, our algorithm uses the modularity gain as the optimization function to fine-tune the label

of nodes, i.e., modularity-based label updating, which greatly reduces the randomness of the label updating process. The entire label propagation process of the algorithm is shown in Figure 1 and the flow chart of the proposed algorithm is shown in Figure 2.

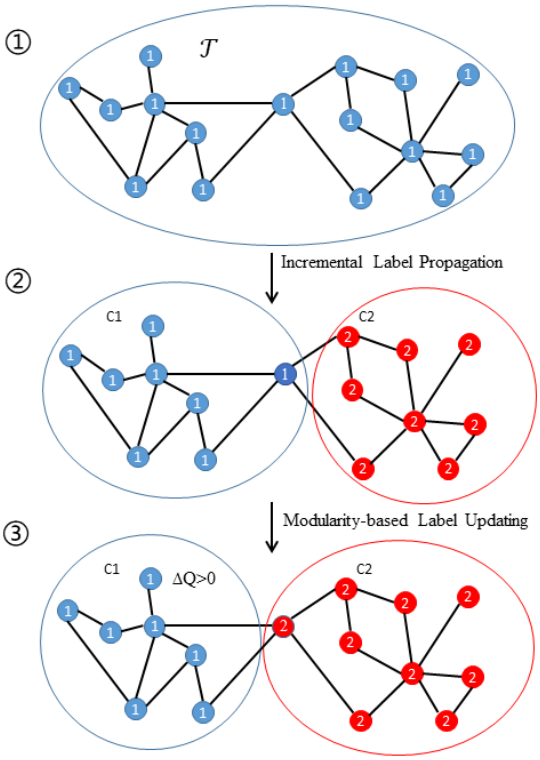

Figure 1. The process of MILPA. Steps **1**–**2** show the process of incremental label propagation and steps **2**–**3** show the process of modularity-based label updating.

Given an undirected network G with n nodes and m edges, the relationship between nodes u and v is denoted by

$$W_{uv} = \left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} \tag{3}$$

where $W_{uv} = 1$ indicates that there is a connection between nodes u and v, while $W_{uv} = 0$ indicates that there is no connection between them. The goal of the algorithm is to find a good partition with k communities, $C_1, C_2, \ldots, C_k$, to maximize the modularity. Note that there is no overlap between any two communities C_i and C_j, that is, each node belongs to only one community.

There are two important concepts in MILPA, intensity and membership degree. The intensity of node u is defined as

$$I_u = \sum_{v \in \mathcal{F}(u)} w_{uv} \tag{4}$$

where $\mathcal{F}(u)$ denotes the neighbors of u. In an undirected network, I_u is the degree of u, i.e., k_u. The membership degree of a node indicates the degree to which it belongs to a community C, defined as

$$M(u, C) = \frac{\sum_{v \in \mathcal{F}(u) \text{ and } v \in C} w_{uv}}{I_u}. \tag{5}$$

In fact, $M(u, C)$ is equivalent to the ratio between the number of the neighbors of node u in community C and the degree of u. A high value of membership degree indicates that the node is more likely to belong to the community.

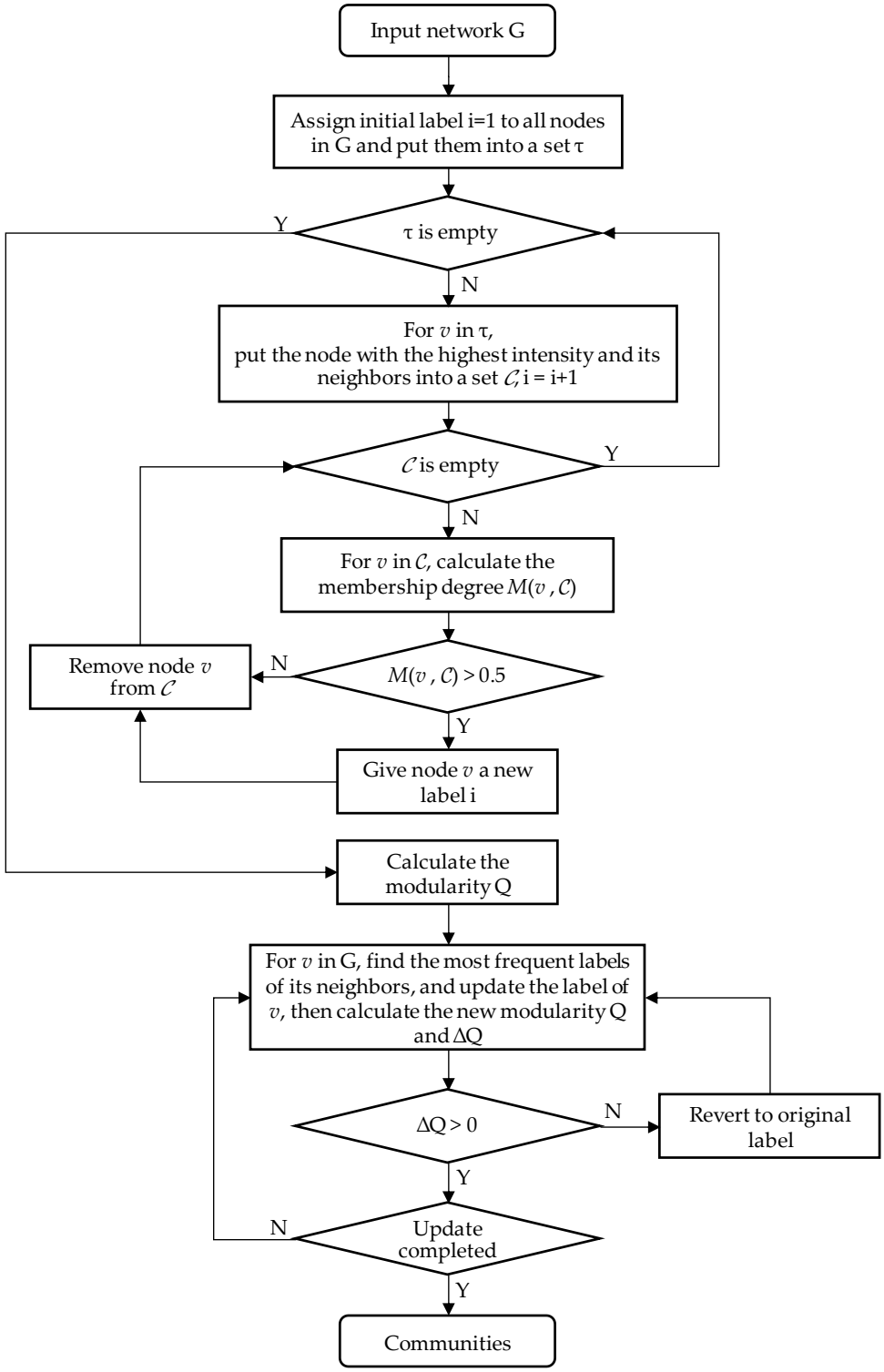

Figure 2. The flow chart of MILPA.

Incremental label propagation. The detailed steps of this process are as follows:

1. All nodes in the network are assigned into a set τ and given the same label $i = 1$.
2. The intensity of all nodes in set τ is calculated according to Equation (4).
3. Among the nodes in τ, the node with the highest intensity and its neighbors are put into a new set, denoted by C.

4. For any node $v \in C$ if the membership degree of v to C is less than ε, it indicates that node v is not closely connected with other nodes in C, and then node v is deleted from C. When all nodes in C satisfy $M(u, C) \geq \varepsilon$, they are marked F.
5. A new label, $i = i + 1$, is assigned to these nodes in C, and then the set C is cleared.
6. Steps (2) to (5) are repeated to create new densely connected subgraphs until no new label is generated, resulting in the initial community partition. The threshold of membership degree $\varepsilon = 0.5$ in this paper since it is proved to be the best for most networks.

Modularity-based label updating. Incremental label propagation can greatly reduce the randomness of LPA to obtain more stable community partition. However, when there are two or more nodes having the highest intensity in step (3), one has to randomly select one from them so that the solution derived from incremental label propagation may not be unique. Moreover, the solution may not be the best partition. To improve the performance of our method, MILPA uses the modularity gain as an optimization function to fine-tune the labels of nodes based on the initial community partition. In particular, one calculates the modularity values before and after a node is moved to the community of its neighbors from its original community. This can be seen in Figure 1, before and after node i in community C_1 is moved to community C_2, the modularity respectively is

$$Q_{before} = \frac{l_{c_1}}{m} - \left(\frac{d_{c_1}}{2m}\right)^2 + \frac{l_{c_2}}{m} - \left(\frac{d_{c_2}}{2m}\right)^2, \tag{6}$$

$$Q_{after} = \frac{l_{c_1} + k_{i,c_1}}{m} - \left(\frac{d_{c_1} + k_i}{2m}\right)^2 + \frac{l_{c_2} - k_{i,c_2}}{m} - \left(\frac{d_{c_2} + k_i}{2m}\right)^2. \tag{7}$$

where k_{i,c_1} denotes the number of edges connected to node i in community C_1 and k_{i,c_2} denotes the number of edges connecting node i with nodes in community C_2.

Therefore, one has the modularity gain

$$\Delta Q = Q_{after} - Q_{before} = \frac{1}{m}\left[k_{i,c_1} - k_{i,c_2} - \frac{k_i(d_{c_1} - d_{c_2} + k_i)}{2m}\right]. \tag{8}$$

If the number of the label of a node's neighbors is more than two, there will be multiple communities for the node to move in. One can compute the ΔQ generated by each move and select the community with the largest ΔQ (if $\Delta Q > 0$). If there are two or more communities with the optimal ΔQ, then one moves this node to one of them randomly. If the maximum of ΔQ is not positive, then this node remains in the original community. Besides, when a node is moved to a new community, its label is also updated. Until the labels of all nodes no longer change or a specified number of iterations is reached, our MILPA method yields the final community partition.

2.4. Experimental Datasets

To verify the performance of the proposed MILPA, experiments are implemented on both LFR benchmark datasets [24] and eight real network datasets, as shown in Tables 1 and 2. The performance of various community detection algorithms, MILPA, LPA, LPAMP, Fastgreedy, GN and Louvain were compared in this paper.

Table 1. LFR benchmark network parameters.

Networks	N	*Mink*	*Maxk*	*Minc*	*Maxc*	*Mu*
Group A	1000	20	50	20	100	0.1~0.5
Group B	5000	50	100	20	100	0.1~0.5

Table 2. Real-world network datasets.

Networks	Node Number	Edge Number
Karate	34	78
Dolphins	61	159
Polbooks	105	441
Football	115	616
Email	1133	5451
Hamsterster	2426	16631
DM-CX	4040	76717
Facebook	4039	88234

2.4.1. LFR Networks

LFR benchmark is a widely used program to build artificial networks to test the performance of community detection algorithms. It can flexibly generate high quality synthetic networks which are close to the real networks. The LFR program provides a series of configuration parameters for users to define, including the size of a network N, i.e., the number of nodes in a network; the maximum and minimum of nodes' degree, *maxk* and *mink* respectively; the maximum and minimum of nodes in a community, *maxc* and *minc* respectively; the ratio of the number of links between communities and the total number of edges in the network, *mu*, which indicates the significance of community structure in the network. The smaller the value of *mu* is, the more obvious the community structure of a network is. In this paper, the LFR program is used to generate two types of benchmark datasets, i.e., Group A and Group B. Each network in Group A has 1000 nodes and different *mu* values ranging from 0.1 to 0.5. Similarly, Group B has five networks with 5000 nodes. The two sets of data simulate small networks and large networks, respectively.

2.4.2. Real-World Networks

The real network is not random but has some characteristics. For example, many nodes have a small degree and also several nodes have a large degree. The nodes with a large degree play a very important role in the entire network. In addition, the distribution of edges is not all uniform, but is distributed more within the community, but less between the community. The network has a community structure, which means that the nodes in the community are likely to have some common attributes or play a similar role in the network. Community structures exist in many real networks, such as social networks, biological networks, engineering networks, and political networks. These real-world networks that are often used in community detection problems are applied to test MILPA, which are Karate [25], Dolphins [26], Polbooks [27], Email [28], Football [29], Hamsterster [30], DM-CX [31] and Facebook [32], as shown in Table 2.

3. Results

All the experiments were implemented in pycharm with python3.6 on a PC with 16 GB memory, Intel core i7 processor and Win10 system. In the process of writing the algorithm, the *networkx* library was mainly used.

3.1. NMI

In this experiment, LFR networks are applied to test the performance of MILPA, LPA, LPAMP, Fastgreedy and Louvain. GN has a worst-case complexity y(n^3) on a sparse graph that is infeasible in large networks, so it is not taken into consideration. The NMI of each method in different networks of each group is shown in Figure 3. The *mu* value in Every group is sampled at intervals of 0.1 at [0.1, 0.5].

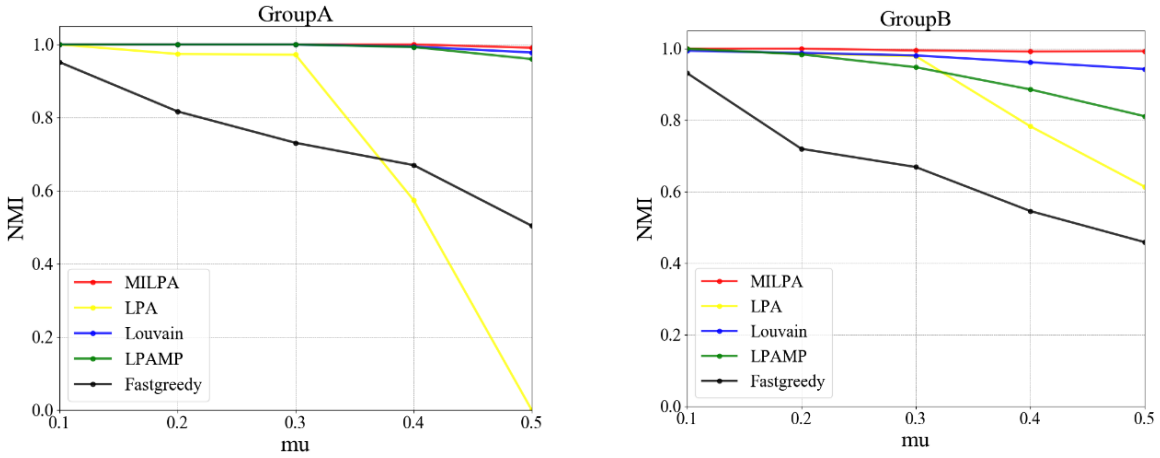

Figure 3. NMI of five methods in LFR networks of Group **A** and **B**.

From the results of Figure 3, the proposed MILPA is proved to be the best, followed by Louvain and LPAMP. Especially in those networks with 5000 nodes, the NMI values of other methods show a significant downward trend, whereas that of MILPA are always close to 1 as the *mu* varies. LPA and Fastgreedy have rather worse performance that the NMI values of the two methods significantly decrease as *mu* grows in both groups. Besides, LPA is unstable due to its large performance gap between the two groups. In those networks with 1000 nodes, when $mu \geq 0.3$, LPA has a sharp decline of NMI and finally cannot detect communities at all.

MILPA, LPAMP and Louvain, as modularity-based methods, all have relatively stable performance in the LFR networks. MILPA and LPAMP which are based on LPA show their improvements comparing to LPA. As a greedy algorithm, Fastgreedy is easy to fall into local optimal solutions, resulting in the worst performance.

3.2. Modularity

To test the ability of MILPA to detect the community structure of real networks, eight real-world datasets are applied, including four small networks and four large networks. To better show the effect of community division, three small networks are selected and their results of community detection are shown in Figure 4. The nodes in a community are marked with the same color and form a circle. It can be seen that most edges are inside the community and a few of edges are between the communities.

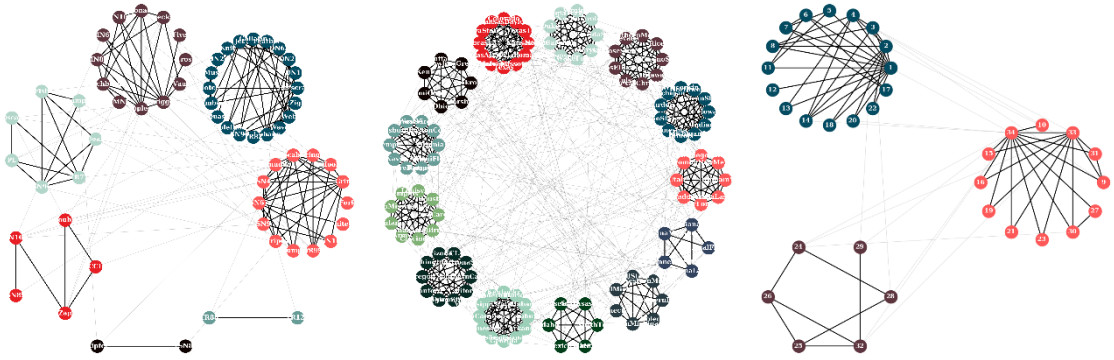

Figure 4. The results of community detection for three networks. Left: Dolphins. Middle: Football. Right: Karate.

Then, to show the stability of the proposed MILPA, twenty independent tests are performed on these networks with LPA, LPAMP, and MILPA, and the results of three small networks are shown in Figure 5. Firstly, it can be clearly seen from this figure that the average modularity of MILPA is much higher than that of LPA and LPAMP. Secondly, the modularity distribution of LPA is relatively scattered, and the difference between the maximum and minimum of its modularity in each network is much larger than MILPA. Besides, while the modularity of LPAMP may sometimes be greater than that of MILPA, the overall stability of MILPA is better than that of LPAMP. The same observations can be obtained from other networks.

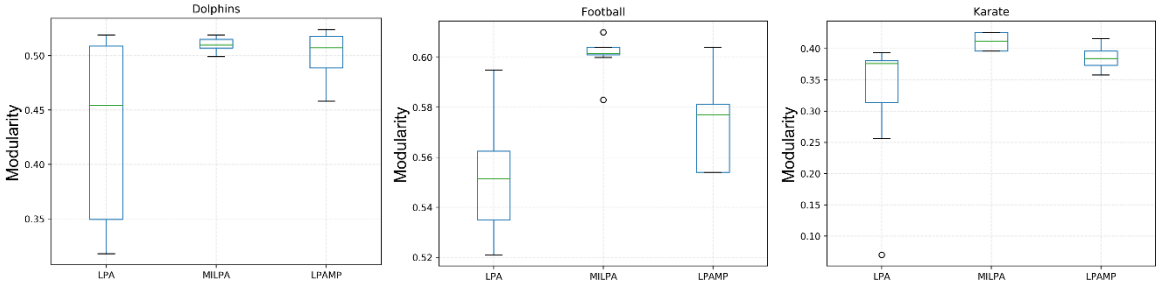

Figure 5. The modularity distribution of twenty community partitions of three networks by LPA, MILPA and LPAMP.

For a further comparison, three other methods are also used for community detection, i.e., Fastgreedy, GN, and Louvain. To ensure the effectiveness of the results, every experiment is performed ten times independently and the average modularity of each method in each network is shown in Figure 6. These methods are sorted according to their modularity in descending order in each chart of this figure. Note that in four large networks, GN and LPAMP take a lot of time, so they are not considered. It is evident that, among these networks, MILPA shows outstanding performance compared to LPA, LPAMP, GN, and Fastgreedy, and it is only second to the best method Louvain. Moreover, in four small networks, MILPA is almost as good as Louvain and always better than other methods. Another significant phenomenon is that with the rise of network size, the performance gap between methods becomes larger. For instance, the performance of MILPA is improved by no more than 14% in four small networks compared with LPA, while it achieves 16.6%, 23.9%, 32.3% and 58.1% improvement in four large networks, Email, Facebook, DM-CX, and Hamsterster, respectively. There is no doubt that LPA is the worst of these methods since its modularity is the lowest in six of eight networks.

It can be observed from above results that, no matter in a small network or a large network, MILPA can achieve a high modularity value and greatly improve the stability of the standard LPA in community detection.

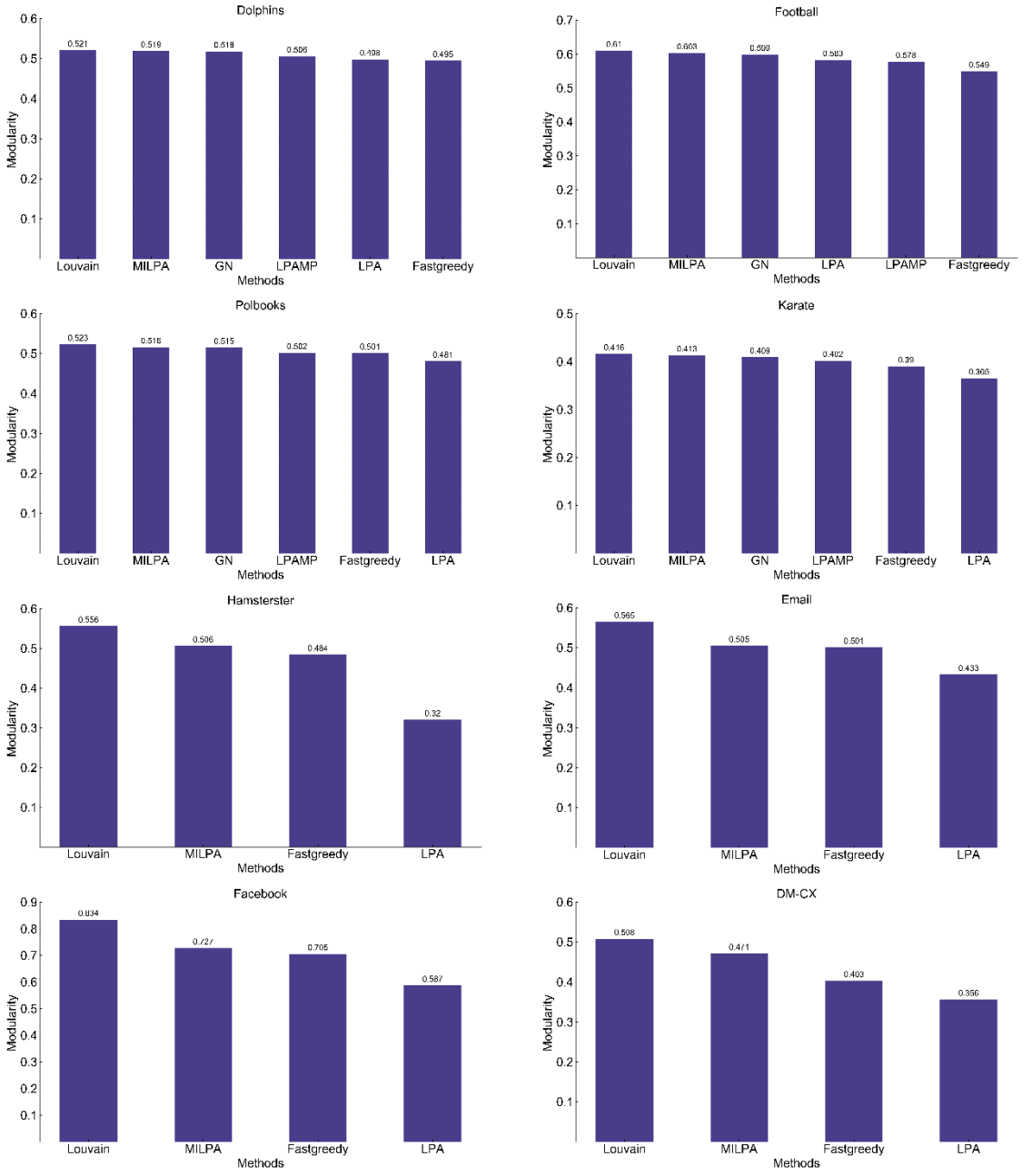

Figure 6. The modularity of different methods in eight real-world networks.

4. Conclusions and Future Work

This paper proposed a new approach, namely MILPA, to detect community structures in networks. While it is an improved version of LPA, MILPA has several unique features. The major difference is that it takes the incremental strategy to carry out label propagation which is the opposite of LPA. In the incremental strategy, MILPA always centers on the node with the highest intensity and starts to find other neighbors that are closely connected, so it greatly reduces the randomness of the algorithm. Experiment results have shown that this strategy can speed up the convergence of label propagation and greatly reduce the randomness of the process. Another feature of MILPA is the introduction of modularity optimization, which enables some nodes to be moved into more appropriate communities to produce a better partition. This is a key factor of the proposed method to have better performance

than LPA. Due to this operation, however, MILPA is slower than LPA which has linear time complexity. Complexity analysis and improvements of MILPA will be important tasks in our future research.

One point that cannot be ignored is the execution order of the algorithm, that is, incremental label propagation first and modularity-based label updating later. This combination, i.e., staged operation, makes MILPA outperform GN, LPAMP, and Fastgreedy which are also based on modularity optimization. The three existing methods do not take this operation but find the best community by modularity optimization every time they move nodes, so it is easy for them to fall into local optimal solutions. Interestingly, Louvain, the best algorithm which is based on modularity optimization, takes the staged operation too. Thus, this phenomenon sheds some light on the design of community detection algorithms.

There is also one important advantage of MILPA. Experiments on LFR benchmark networks demonstrated that it can find community structure of a network even it is not obvious. As the community structure of synthetic networks becomes more and more difficult to distinguish, the performance of other algorithms (including Louvain) declines significantly, but the proposed algorithm maintains a high detection performance. MILPA therefore may be viewed a high-quality competitor when new methods are tested in LFR networks.

In summary, the proposed MILPA has a number of major improvements over traditional LPA and it can be considered as an important supplement of community detection techniques for its performance superiority to many modularity-based methods. In the future work, we will focus on the time complexity of the algorithm to further improve the performance of the algorithm. We will also consider developing our algorithm to weighted networks and directed networks and generalizing it to larger networks.

Author Contributions: The authors designed the experiment together. Y.M. (Yunlong Ma), Y.Z. and J.W. performed the experiment. M.L. and Y.M. (Yumin Ma) curried out the analyses. Y.M. (Yunlong Ma), Y.Z., J.W. and W.S. wrote the manuscript. All authors reviewed the manuscript. Y.M. (Yunlong Ma) and Y.Z. contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key R&D Program of China under Grant 2019YFB1704700; in part by the National Natural Science Foundation of China under Grants 61573257, 71690234, 61873191, 61973237 and 71690234; in part by the Science and Technology Commission of Shanghai Municipality under Grant 19JG0500700.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

LPA	Label propagation algorithm
MILPA	Modularity-based label propagation
Q	Modularity
v	Node
C	Community(set)
I	Intensity
M	Membership degree

References

1. Pianka, E.R. *Analyses of the Ecological Niche and Community Structure*; Princeton University Press: Princeton, NJ, USA, 2017.
2. Su, Y.; Liu, C.; Niu, Y.; Cheng, F.; Zhang, X. A Community Structure Enhancement-Based Community Detection Algorithm for Complex Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *99*, 1–14. [CrossRef]
3. Girdhar, N.; Bharadwaj, K.K. Community Detection in Signed Social Networks Using Multiobjective Genetic Algorithm. *J. Assoc. Inf. Sci. Technol.* **2019**, *70*, 788–804. [CrossRef]
4. Zheng, X. Privacy-preserved community discovery in online social networks. *Future Gener. Comput. Syst.* **2019**, *93*, 1002–1009. [CrossRef]

5. Cunchao, T. A Unified Framework for Community Detection and Network Representation Learning. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 1051–1065.
6. Guishan, W.; Xuezao, R.; Xueying, L. Research on Community Center-metric and Community Detection Algorithm for Complex Networks. In Proceedings of the 2019 International Conference on Applied Mathematics, Modeling, Simulation and Optimization, Gui Lin, China, 21–22 April 2019.
7. Nerurkar, P.; Chandane, M.; Bhirud, S. A Comparative Analysis of Community Detection Algorithms on Social Networks. *Comput. Intell. Theor. Appl. Future Dir.* **2019**, *1*, 287–298.
8. Fortunato, S.; Hric, D. Community detection in networks: A user guide. *Phys. Rep.* **2016**, *659*, 1–44. [CrossRef]
9. Newman, M.E.J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2004**, *69*, 026113. [CrossRef]
10. Filippo, R.; Claudio, C.; Federico, C.; Vittorio, L.; Domenico, P. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA* **2004**, *101*, 2658–2663.
11. Brandes, U. On Modularity Clustering. *IEEE Trans. Knowl. Data Eng.* **2007**, *20*, 172–188. [CrossRef]
12. Biswas, A.; Biswas, B. Analyzing evolutionary optimization and community detection algorithms using regression line dominance. *Inf. Sci.* **2017**, *396*, 185–201. [CrossRef]
13. Murata, T.; Afzal, N. Modularity Optimization as a Training Criterion for Graph Neural Networks. In Proceedings of the 2018 International Conference on Complex Networks (ComplexNet), Boston, MA, USA, 5–8 March 2018.
14. Qin, J.; Yang, I.; Rajagopal, R. Submodularity of Storage Placement Optimization in Power Networks. *IEEE Trans. Autom. Control.* **2018**, *99*, 3268–3283. [CrossRef]
15. Newman, M.E.J. Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **2003**, *69*, 066133. [CrossRef] [PubMed]
16. Clauset, A.; Newman, M.E.J.; Moore, C. Finding community structure in very large networks. *Phys. Rev. E* **2004**, *70*, 066111. [CrossRef] [PubMed]
17. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech.* **2008**, *10*, 155–168. [CrossRef]
18. Raghavan, U.N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **2007**, *76*, 036106. [CrossRef]
19. Barber, M.J.; Clark, J.W. Detecting network communities by propagating labels under constraints. *Phys. Rev. E* **2009**, *80*, 026129. [CrossRef]
20. Li, L.; Ni, L. Community detection algorithm for label propagation based on modularity optimization. *Comput. Syst. Appl.* **2016**, *25*, 212–215.
21. Zongwen, L.; Jianping, L.; Fan, Y.; Petropulu, A. Detecting community structure using label propagation with consensus weight in complex network. *Chin. Phys. B* **2014**, *23*, 594–601.
22. Funk, M.J.; Combs, M. Strangers on a theoretical train. *J. Stud.* **2015**, *20*, 1–21. [CrossRef]
23. Yan, C.; Yan, J.; Yu, Y.; Chen, J. Uncovering the community structure in signed social networks based on greedy optimization. *Mod. Phys. Lett. B* **2017**, *31*, 1750158.
24. Andrea, L.; Santo, F.; Filippo, R. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **2008**, *78*, 046110.
25. Zachary, W.W. An Information Flow Model for Conflict and Fission in Small Groups1. *J. Anthr. Res.* **1976**, *33*, 452–473.
26. Lusseau, D.; Newman, M.E.J. Identifying the role that animals play in their social networks. *Proc. R. Soc. B Biol. Sci.* **2004**, *271*, 477–481. [CrossRef] [PubMed]
27. Krebs, V. Books about US Politics Network Dataset. Available online: http://www.orgnet.com (accessed on 10 June 2020).
28. Roger, G. Self-similar community structure in a network of human interactions. *Phys. Rev. E* **2003**, *68*, 065103.
29. Girvan, M.; Newman, M.E.J. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [CrossRef]
30. KONECT. Hamsterster Full Network Dataset. Available online: http://konect.uni-koblenz.de/networks/petster-hamster (accessed on 10 June 2020).

31. Ryan, A.; Nesreen, K. The Network Data Repository with Interactive Graph Analytics and Visualization. Available online: http://networkrepository.com/bio-DM-CX.php (accessed on 10 June 2020).
32. Jim, M.; Jure, L. Learning to Discover Social Circles in Ego Networks. *NIPS* **2012**, *1*, 539–547.