# Advanced modularity-specialized label propagation algorithm for detecting communities in networks

X. Liu *, T. Murata

*Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro, Tokyo 152-8552, Japan*

## A R T I C L E   I N F O

## A B S T R A C T

A modularity-specialized label propagation algorithm (LPAm) for detecting network communities was recently proposed. This promising algorithm offers some desirable qualities. However, LPAm favors community divisions where all communities are similar in total degree and thus it is prone to get stuck in poor local maxima in the modularity space. To escape local maxima, we employ a multistep greedy agglomerative algorithm (MSG) that can merge multiple pairs of communities at a time. Combining LPAm and MSG, we propose an advanced modularity-specialized label propagation algorithm (LPAm+). Experiments show that LPAm+ successfully detects communities with higher modularity values than ever reported in two commonly used real-world networks. Moreover, LPAm+ offers a fair compromise between accuracy and speed.

© 2009 Elsevier B.V. All rights reserved.

## 1. Introduction

Detecting communities in networks has attracted a great deal of interest recently. Informally, a community is a densely connected subnetwork that is only sparsely linked to the remaining network. It is said that constructing algorithms for detecting communities is of great importance as it provides insight into the structures of real-world systems [1].

Modularity [2] is a scalar value that measures the quality of a particular division of a network into communities. Among various kinds of methods for detecting network communities, one that is widely used is modularity optimization [3]. The modularity optimization method detects communities by searching over possible divisions of a network for one that have particularly high modularity. Since finding the "best" community division with the highest modularity value is proven to be NP-hard [4], exhaustive search over all possible divisions is in general intractable. Therefore, all of the modularity optimization algorithms are based on approximate optimization.

Recently Raghavan, Albert et al. proposed a label propagation algorithm (LPA) [5] for detecting network communities. This innovative and promising algorithm uses only the network structure as a guide, and can detect communities at very high speed. Barber and Clark extend LPA by relating it to modularity, and introduce a modularity-specialized LPA (LPAm) [6]. However, it is found that LPAm is prone to get stuck in poor local maxima in the modularity space [6]. To detect communities with high modularity values, we improve LPAm by driving it out of local maxima and devise a new algorithm called LPAm+ in this paper. Experiments show that LPAm+ successfully detects communities with the highest modularity values in several commonly used real-world networks.

The structure of the paper is as follows: in the next section, the definition of modularity is reviewed. In Section 3, we give a survey of LPA and LPAm. Our algorithm is proposed in Section 4. Experiments are shown in Section 5, followed by a conclusion and discussion in the last section.

---

* Corresponding author.
   *E-mail address:* tsinllew@ai.cs.titech.ac.jp (X. Liu).

## 2. Modularity

To evaluate the goodness of a particular division of a network into communities, Newman introduces a measure called modularity [2]. Consider a (undirected and unweighted) network with $n$ nodes and $m$ edges represented by an adjacency matrix $A$, whose element $A_{uv}$ is equal to 1 if there is an edge between nodes $u$ and $v$, and 0 otherwise. The degree of a node $u$ is denoted by $k_u$. Suppose a particular division of the network into $N_c$ communities, such that each node $u$ is assigned to a community $l_u$. Modularity essentially measures the actual fraction of intra-community edges minus its expected value in a null model, where the community division is the same but connections are made randomly between nodes. Formally, modularity is defined as:

$$Q = \frac{1}{2m} \sum_{u,v=1}^{n} (A_{uv} - P_{uv}) \delta(l_u, l_v), \tag{1}$$

where $P_{uv} = k_u k_v / 2m$ is probability in the null model that an edge exists between nodes $u$ and $v$, and $\delta(i, j)$ is Kronecker's delta. Further, a modularity matrix $B$ is defined with elements $B_{uv} = A_{uv} - P_{uv}$. Hence, modularity is expressed as:

$$Q = \frac{1}{2m} \sum_{u,v=1}^{n} B_{uv} \delta(l_u, l_v). \tag{2}$$

We can also reformulate modularity as the addition of contributions over all communities:

$$Q = \sum_{t=1}^{N_c} \left( \frac{I_t}{m} - \left( \frac{D_t}{2m} \right)^2 \right), \tag{3}$$

where $I_t$ is the number of intra-community edges that have both ends in community $t$, $D_t$ the sum over all degrees of nodes in community $t$.

## 3. LPA

In this section, we give a survey of LPA and LPAm, which are the bases of the following discussion.

### 3.1. LPA

The idea of LPA is simple [5]: initially each node in the network is assigned with a unique label, indicating the community it belongs to. At every label propagation step, each node sequentially updates its label to a new one which is the most frequent label among its neighbors. Formally, the label updating rule for node $x$ is:

$$l_x^{\text{new}} = \arg\max_l \left( \sum_{u=1}^{n} A_{ux} \delta(l_u, l) \right), \tag{4}$$

where $l_x^{\text{new}}$ indicates new label for node $x$. If more than one label are the most frequent ones, the new label is chosen randomly from them. The label propagation step is performed iteratively until each node has a label that is (one of) the most frequent label(s) of its neighbors. Finally communities are identified as groups of nodes bearing the same labels.

The most striking feature of LPA is its less expensive computation than what is possible so far (near linear time complexity) [5]. The weakness is that LPA is not stable: the algorithm is sensitive to the order in which node labels are updated in each step. Thus the solutions (and their corresponding modularity values) can be quite different in different runs [7]. Sometimes LPA may even end up with a trivial solution—all nodes are identified in the same community [6].

### 3.2. LPAm

Barber and Clark extend LPA by modifying the label updating rule so that modularity can be maximized, and propose a new algorithm called LPAm [6]. We can rewrite (2) by separating elements regarding the label of node $x$ from others, yielding:

$$Q = \frac{1}{2m} \left( \sum_{u \neq x} \sum_{v \neq x} B_{uv} \delta(l_u, l_v) - B_{xx} \right) + \frac{1}{m} \left( \sum_{u=1}^{n} B_{ux} \delta(l_u, l_x) \right). \tag{5}$$

When updating the label for $x$, by selecting a new label that maximizes the second term on the right-hand side of (5), we actually maximize $Q$. Hence, to consider updating the label for node $x$, the updating rule of LPAm is:

$$l_x^{\text{new}} = \arg\max_l \left( \sum_{u=1}^{n} B_{ux} \delta(l_u, l) \right). \tag{6}$$

Implementing LPAm would bring about a monotone increase in modularity, hindering the trivial solution being formed. Besides, LPAm preserves the merit of high speed of LPA. But LPAm is prone to get stuck in poor local maxima in modularity
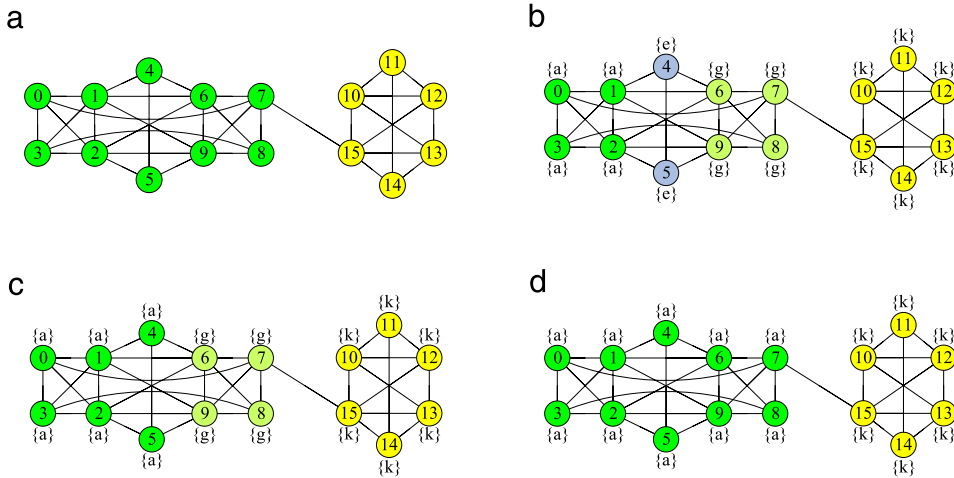
**Fig. 1.** A toy network. (a) The network is intuitively divided into two communities. (b) LPAm gets stuck in the poor local maximum where the network is divided into 4 communities and the modularity is 0.399. (c) We escape the local maximum descried in (b) by merging the community labeled 'a' and 'e', with modularity increased by 0.008. (d) After carrying out LPAm again, we climb onto another local maximum, which is also the global maximum, with modularity increasing from 0.407 to 0.413.

space, with a similar total degree of nodes in different communities [6]. Moreover, LPAm still suffers from the weakness of instability.

## 4. LPAm+

In this section, we first give an example in which LPAm gets stuck in a local maximum, then we introduce how to escape the local maximum and propose our improved algorithm LPAm+.

Take the toy network shown in Fig. 1a as an example. This network is intuitively divided into two communities (painted in yellow and green colors respectively), with its modularity equal to 0.413. Feeding LPAm with this network, we obtain a division into four communities (Fig. 1b) and its modularity is 0.399. Evidently this division corresponds to a local maximum in the modularity space. Under the label updating rule (6), LPAm favors community divisions where all communities are similar in total degree, which immediately leads to the separation of communities {0–3}, {4, 5} and {6–9}.

To escape the local maximum, we have to get rid of the current constraint. Note that (6) is a modularity maximization rule based on local structure of the network. Viewing broadly, we can adopt the greedy rule for merging communities that maximizes modularity: when LPAm gets stuck in a local maximum (no modularity gain can be achieved from further label propagation), we calculate the modularity changes for merging pairs of communities, and merge those pairs that improve modularity most. In real operation, we employ the technique used in the multistep greedy agglomerative algorithm (MSG) [8] that promotes simultaneously merging of multiple pairs of communities at a time, under the following criteria: suppose $t_1$ and $t_2$ is a pair of communities to be merged; neither $t_1$ nor $t_2$ is present in another pair inducing a higher modularity changes (see Appendix for additional details).

After merging communities, we escape the local maximum. Then we should carry out another round of LPAm. This is analogous to climbing onto another local maximum. However, it is not guaranteed that the new local maximum we arrived at is good enough (although it is better than the previous local maximum). Hence we should repeat the above process (escaping the local maximum and climbing onto another local maximum) for many times, until no improvement of modularity can be reached. Fig. 1b–d give an illustration for LPAm+ working in the toy network. The pseudo-code of LPAm+ is presented in Algorithm 1. It is clear that LPAm+ brings a monotone increase in modularity. Since the time complexity estimation of LPAm+ is a little complex, we left it to the next section.

## 5. Experiments

We test LPAm+ in several real-world networks that are commonly used by other researchers for evaluating modularity optimization algorithms. These networks include: the karate club network (Karate Club) [9], the dolphin association network (Dolphins) [10], the network of co-purchased political books (Political Books) [11], the network of games between college football teams (College Football) [12], the network of collaborations between jazz musicians (Jazz) [13], the network of metabolic reactions in *Caenorhabditis elegans* (*C. elegans*) [14], the network of email contacts at a university (E-mail) [15], the Pretty Good Privacy web of trust social network (PGP) [16], and the network of co-authorships for e-print papers posted to the condensed matter archive (Condmat2003) [17]. As most of the researchers did, we uniformly treat all networks as undirected and unweighted, and exclude all self-loop edges. Table 1 lists the numbers of nodes and edges after preprocessing the data.

**Algorithm 1** LPAm+.

1: Each node is assigned with a unique label
2: maximize modularity by LPAm
3: **while** $\exists$ community pair $(t_1, t_2)$ with $\Delta Q_{t_1 t_2} > 0$ **do**
4:    **for** every community pair $(t_1, t_2)$: $(\Delta Q_{t_1 t_2} > 0) \wedge [!\exists t : (\Delta Q_{t t_1} > \Delta Q_{t_1 t_2}) \vee (\Delta Q_{t t_2} > \Delta Q_{t_1 t_2})]$ **do**
5:       merge communities $t_1$ and $t_2$;
6:    **end for**
7:    maximize modularity by LPAm
8: **end while**

$*$   $\Delta Q_{t_1 t_2}$ denotes the modularity change for merging community $t_1$ and $t_2$.

**Table 1**
The numbers of nodes and edges of the networks in our experiment.

| Network | # of nodes | # of edges |
|---|---|---|
| Karate club | 34 | 78 |
| Dolphins | 62 | 159 |
| Political books | 105 | 441 |
| College football | 115 | 613 |
| Jazz | 198 | 2,742 |
| *C. elegans* | 453 | 2,025 |
| E-mail | 1,133 | 5,451 |
| PGP | 10,680 | 24,316 |
| Condmat2003 | 27,519 | 116,181 |

**Table 2**
Comparisons between LPAm and LPAm+. Values are collected from one hundred runs for each network. $Q_{max}$ denotes the maximal modularity value, $Q_{avg}$ the average modularity value, $\sigma$ the standard deviation of the modularity value, and $t$ the average execution time (in seconds, on a PC with Intel Core 2 Duo CPU @ 2.53 GHz).

| Network | LPAm | | | | LPAm+ | | | |
|---|---|---|---|---|---|---|---|---|
| | $Q_{max}$ | $Q_{avg}$ | $\sigma$ | $t$ | $Q_{max}$ | $Q_{avg}$ | $\sigma$ | $t$ |
| Karate club | 0.399 | 0.352 | 0.0277 | 0.009 | 0.420 | 0.418 | 0.0061 | 0.014 |
| Dolphins | 0.516 | 0.495 | 0.0076 | 0.019 | 0.529 | 0.523 | 0.0023 | 0.034 |
| Political books | 0.522 | 0.493 | 0.0199 | 0.048 | 0.527 | 0.527 | 0.0011 | 0.088 |
| College football | 0.604 | 0.579 | 0.0182 | 0.049 | 0.605 | 0.604 | 0.0018 | 0.080 |
| Jazz | 0.445 | 0.436 | 0.0092 | 0.229 | 0.445 | 0.444 | 0.0013 | 0.368 |
| *C. elegans* | 0.409 | 0.379 | 0.0138 | 0.354 | 0.452 | 0.441 | 0.0045 | 1.247 |
| E-mail | 0.537 | 0.496 | 0.0155 | 1.097 | 0.582 | 0.576 | 0.0028 | 3.589 |
| PGP | 0.726 | 0.705 | 0.0085 | 5.396 | 0.884 | 0.882 | 0.0009 | 114.221 |
| Condmat2003 | 0.582 | 0.568 | 0.0036 | 31.952 | 0.755 | 0.751 | 0.0012 | 461.599 |

We apply LPAm and LPAm+ one hundred times to each of the networks. Table 2 shows the maximal modularity, the average modularity, the standard deviation of modularity, and the average execution time collected from samples. We can see that both the maximal modularity and the average modularity obtained by LPAm+ are markedly higher than those by LPAm, consistently in all of the networks. This implies the success of our trick for escaping the local maxima. For the index of the standard deviation of modularity, we can find that LPAm+ value is significantly smaller than that of LPAm. As a matter of fact, normally the difference of modularity values between solutions of LPAm+ in different runs is within 1%. Even in extreme cases, the difference between the worst and the best modularity values is no more than 5%. Therefore, LPAm+ is much more stable than LPAm.

Fig. 2 portrays the running time of LPA and LPAm+ in networks of different sizes. In the following, we give a time complexity analysis for LPAm+. On the one hand, one step of label propagation in LPAm costs O($m$) time [6], so the time complexity of LPAm is O($rm$), where $r$ is the number of label propagation steps required to reach a local maximum in modularity space. On the other hand, one round of merging pairs of communities that corresponds to the for-loop in Algorithm 1 requires a time of O($m \log n$) [8]. Let $h$ denote the number of needed iterations for the while-loop in Algorithm 1. The overall time of LPAm+ can be written as O($rm$) + $h$(O($m \log n$) + O($rm$)).

An exact estimation of $h$ is not possible, as it depends on the quality of the intermediate solution obtained by LPAm. Suppose $d$ is defined as the depth of the dendrogram describing the community structure. The number of merging rounds for a single MSG algorithm (the step width is set to be $+\infty$) would be $d$. The number of merging rounds in our algorithm, namely $h$, seems a little obscure, since LPAm is performed after each merging rounds. However, note that only two cases can happen during the label propagation process in LPAm: some communities disappear and the remaining communities
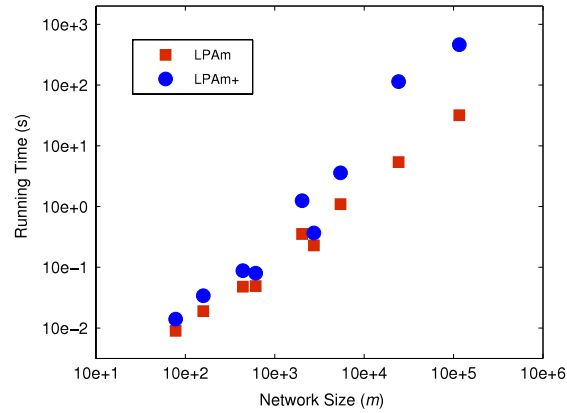
**Fig. 2.** Comparison of running time for LPAm and LPAm+ in networks of different sizes (on a PC with Intel Core 2 Duo CPU 2.53 GHz).

**Table 3**
The average number of label propagation steps required for the embedded LPAm to converge, denoted by $r$, and the number of iterations for the while-loop, denoted by $h$, when LPAm+ is applied to real-world networks. Values are averaged over one hundred runs in each of the networks. The uncertainty of the final digit, calculated as the standard error of the mean, is shown parenthetically.

| Network | $r$ | $h$ |
|---|---|---|
| Karate club | 6.13(6) | 1.02(1) |
| Dolphins | 5.66(4) | 1.71(4) |
| Political books | 6.52(6) | 1.98(2) |
| College football | 5.45(4) | 1.02(1) |
| Jazz | 6.93(9) | 1.54(4) |
| *C. elegans* | 6.17(5) | 6.95(9) |
| E-mail | 7.11(6) | 6.56(8) |
| PGP | 4.61(1) | 73.8(2) |
| Condmat2003 | 5.59(2) | 55.9(7) |

**Table 4**
The (maximal) modularities obtained by LPAm+ and many previously published methods.

| Network | LPAm+ | MSG-VM | SS-ML | Greedy | VP/LP | EO | SA | SO |
|---|---|---|---|---|---|---|---|---|
| Karate club | 0.420 | 0.398 | 0.420 | 0.381 | 0.420 | 0.419 | 0.420 | 0.419 |
| Dolphins | 0.529 | – | 0.528 | – | 0.529 | – | 0.528 | 0.489 |
| Political books | 0.527 | – | 0.527 | – | 0.527 | – | 0.527 | 0.399 |
| College football | 0.605 | 0.603 | 0.600 | 0.556 | 0.605 | – | 0.605 | 0.602 |
| Jazz | 0.445 | 0.445 | 0.445 | 0.439 | 0.445 | 0.445 | 0.445 | 0.442 |
| *C. elegans* | 0.452 | 0.450 | 0.446 | 0.412 | 0.450 | 0.434 | 0.450 | 0.435 |
| E–mail | 0.582 | 0.575 | 0.577 | 0.503 | 0.579 | 0.574 | 0.579 | 0.572 |
| PGP | 0.884 | 0.878 | 0.884 | 0.849 | – | 0.846 | – | 0.855 |
| Condmat2003 | 0.755 | 0.748 | 0.814 | 0.661 | – | 0.679 | – | 0.723 |

exchange parts of their nodes with each other.[1] Hence we can safely arrive at: $h \sim d$. In Table 3, we list the true values of $h$ when LPAm+ is applied to the various networks mentioned above.

As for $r$, it is still not very well understood. In Ref. [5], the authors suggest that the number of label propagation steps required for LPA algorithm to converge is independent to the number of nodes, and after 5 steps 95% of the nodes can get their "right" labels. We show the actual values of $r$ obtained from running LPAm+ in real-world networks in Table 3. It seems that $r$ is bounded by a small constant. Therefore, $r = o(\log n)$.

Taken all together, in a hierarchical network where $d \sim \log n$, LPAm+ requires an overall time of $O(m \log^2 n)$. This scaling is the same as MSG [8] and the classical greedy agglomerative algorithm [18].

To compare the performance of LPAm+ with other algorithms, in Table 4 we include the (maximal) modularity values obtained by LPAm+ and by many previously published methods in these networks. These methods are, in order, the hybrid algorithm of MSG algorithm in combination with node moving refinement algorithm proposed by Schuetz and Caflisch (MSG-VM) [8], the hybrid algorithm of single-step greedy agglomerative algorithm by significance in combination

---

[1] In theory, there are three cases during the label propagation process in LPAm: (1) existent communities disappear (this situation happens when all nodes of a community select the labels of other communities as their new labels); (2) communities exchange part of their nodes with each other (this situation happens when a part of nodes of one community select the labels of other communities as their new labels); (3) new communities appear (this situation happens when some nodes select unused labels as their new labels). But case (3) never happens in practice.

**Table 5**
Comparison between the solution with maximal value of modularity obtained by LPAm+ and the one with the highest modularity ever reported. $N_c$ is the number of detected communities. $Q$ denotes the modularity value. Sources indicate the referenced papers where we collected the data.

| Network | LPAm+ | | Published Algorithms | | |
|---|---|---|---|---|---|
| | $N_c$ | $Q$ | $N_c$ | $Q$ | Sources |
| Karate club | 4 | 0.420 | 4 | 0.420 | [19,20,24,25] |
| Dolphins | 5 | 0.529 | 5 | 0.529 | [20,26] |
| Political books | 5 | 0.527 | 5 | 0.527 | [19,20] |
| College football | 10 | 0.605 | 10 | 0.605 | [20,25] |
| Jazz | 4 | 0.445 | 4(5) | 0.445 | [8,19–21] |
| *C. elegans* | 9 | 0.452 | 11 | 0.450 | [8,20] |
| E-mail | 10 | 0.582 | 11 | 0.579 | [20] |
| PGP | 99 | 0.884 | 93 | 0.884 | [19] |
| Condmat2003 | 72 | 0.755 | 76 | 0.814 | [19] |

with multilevel node moving refinement algorithm advanced by Noack and Rotta (SS-ML) [19], the greedy agglomerative algorithm put forward by Clauset, Newman and Moore (Greedy) [18], the mathematical programming approach proposed by Agarwal and Kempe (VP/LP) [20], the extremal optimization algorithm introduced by Duch and Arenas (EO) [21], the simulated annealing implementation proposed by Guimerà and Amaral (SA) [22], and the spectral optimization method suggested by Newman (SO) [23].

To make it clearer, in Table 5, we summarize the best solutions obtained by LPAm+ and the ones with the highest modularity values ever reported in these networks. It is found that, for eight of the nine networks considered here (Karate Club, Dolphins, Political Books, College Football, Jazz, *C. elegans*, E-mail and PGP), LPAm+ finds the highest modularity values. Especially, for two networks (*C. elegans* and E-mail), LPAm+ finds modularity values higher than previously published. Only in the Condmat2003 network, LPAm+ is outperformed by SS-ML algorithm. It is interesting to note that SS-ML, which employs the single-step greedy agglomerative algorithm followed by the multilevel node moving refinement algorithm[2], achieves much higher modularity value than other algorithms in this network. In Ref. [19], the devisers of SS-ML argue that MSG algorithm is generally less effective than the single-step greedy agglomerative algorithm. Perhaps the reason that LPAm+ does not work well in the Condmat2003 network is that its component MSG, with a too aggressive strategy, diverts the algorithm to a suboptimal portion of the solution space. It is also worthwhile to note that the VP/LP [20] and SA [22] algorithms can as well find the highest modularity values in some of the networks. But they are computationally much expensive and do not scale to larger networks like PGP and Condmat2003. Therefore, though not as fast as LPAm which is noted for its speed, LPAm+ offers a fair compromise between accuracy and speed.

## 6. Conclusion and discussion

In this paper, we introduce a new community detection algorithm LPAm+ based on the previously proposed algorithm LPAm. The main idea is that we try to drive LPAm out of local maxima and hereby employ MSG to merge pairs of communities which are similar in total degree. Experiments show that LPAm+ improves LPAm in terms of modularity of the detected communities, with extra computational time. Besides, LPAm+ is more stable than LPAm. Compared with other algorithms, LPAm+ distinguishes itself by its accuracy (measured by modularity) while preserving relatively high speed. The fact that LPAm+ detects the highest modularity values in almost all of the test networks is impressive.

It should be noted that the speed of LPAm+ can still be substantially improved. First, when updating the label for a node in LPAm, candidates of the new label can be safely confined to the labels of the neighbors of that node and an unused label [6] (further, we find through experiments that unused label is never selected as a new label). In light of this, we can only update the labels of nodes whose neighbors had a label change. This means only a few labels need to be updated after most of the other labels are fixed. Hence the speed of LPAm can be dramatically increased. Second, it is possible to introduce a threshold and then stop LPAm as soon as the modularity gain from the latest label propagation step does not exceed this threshold. Although these two heuristics have little influence on the final modularity value, the computational time can be reduced to a great extent (the time complexity of the algorithm remains the same, since the order of the number of iterations for the while-loop is unchanged). For example, if we apply these two heuristics (the threshold is set to be 0.00001) in the Condmat2003 network, the running time is considerably reduced from 461.599 s to 96.1 s, with modularity dropped by only 0.013 (based on an average value).

It is also interesting to note that MSG is not the only means to drive LPAm out of local maxima. After this work is done, we are informed that Blondel et al. use a reduction method (communities are reduced into nodes) [27] to escape the local maxima involved in another algorithm different from LPAm, and propose a two-phase community detection algorithm [28]. It seems that such a reduction method can also be used to drive LPAm out of local maxima.

Another important issue is that the solutions of LPAm+ in different runs, though give similar high modularity values, are not distinct in their compositions. This phenomenon is more obvious in large-scale networks. A very recent paper [29] discusses the origin of this problem. How to make the algorithm more deterministic is left for our future work.

---

[2] This algorithm is designed to improve modularity by "adjusting" misplaced nodes.

Overall, the presented LPAm+ algorithm is a suitable choice for analyzing community structures in networks.

## Acknowledgements

## Appendix. The label updating rule for driving LPAm out of local maxima

The label updating rule of LPAm (6) can be rewritten as:

$$l_x^{\text{new}} = \arg\max_l \left( \sum_{u=1}^n (A_{ux} - P_{ux})\, \delta(l_u, l) \right) \tag{7}$$

$$= \arg\max_l \left( \sum_{u=1}^n A_{ux}\delta(l_u, l) - \frac{k_x}{2m} \sum_{u=1}^n k_u \delta(l_u, l) \right) \tag{8}$$

$$= \arg\max_l \left( \sum_{u=1}^n A_{ux}\delta(l_u, l) - \frac{k_x D_l}{2m} \right). \tag{9}$$

The first term in (9) is equal to the number of $x$'s neighbors labeled $l$, and the second term is the product of $k_x/2m$ ($k_x$ denotes the degree of $x$) and the sum of degrees of nodes labeled $l$ ($D_l$).

As shown in Section 4, the toy network is divided into four communities by LPAm, with communities {0–3}, {4,5} and {6–9} being separated (Fig. 1b). This division corresponds to a local maximum. But what is the reason? In further analysis, we find that node 4 has one neighbor labeled 'a', 'e', and 'g' respectively, and the sum of degrees of nodes labeled 'a' or 'g' is large while the sum of degrees of nodes labeled 'e' is small. Consider that node 4 is being updated. For the choice of the candidate new label $l$ as 'a', 'e' or 'g', the value of $\sum_{u=1}^n A_{u4}\delta(l_u, l)$ that is the first term of (9) would all amount to 1. Yet the value of the second term of (9) $k_4 d_l/2m$ would be smaller for the choice of $l$ as 'e' than as 'a' or 'g'. According to the updating rule (9), node 4 would keep its label unchanged and still select 'e' as the new label. Similar case is applied to node 5: it would also stick to label 'e' when updated. Consequently, under the current updating rule, neither node 4 nor 5 is willing to give in first. Suppose we disregard the current updating rule, and change the label of node 4 from 'e' to 'a' forcedly. Though this will bring about a temporary decrease in modularity, it is reasonable to expect a greater reward from the subsequent label updation for node 5 and other nodes. Conceptually, we call nodes like 4 and 5 that block the system from further progressing as stubborn nodes. It is these stubborn nodes that results in LPAm getting stuck in local maxima.

To escape the local maxima, we should attempt to let one or more of the stubborn nodes make a compromise to break down the blocked situation. Suppose $i_1, \ldots, i_k$ is a set of stubborn nodes labeled $l_{(i_1,\ldots,i_k)}$. Our trick is to keep $i_1, \ldots, i_k$ holding the same label and update it (let $i_1, \ldots, i_k$ make a compromise at the same time). Treating the labels of $i_1, \ldots, i_k$ separately and rewriting (2), we have:

$$Q = \frac{1}{2m} \left( \sum_{u \notin \{i_1,\ldots,i_k\}} \sum_{v \notin \{i_1,\ldots,i_k\}} B_{uv}\delta(l_u, l_v) - \sum_{u \in \{i_1,\ldots,i_k\}} \sum_{v \in \{i_1,\ldots,i_k\}} B_{uv} \right) + \frac{1}{m} \left( \sum_{u=1}^n \sum_{v \in \{i_1,\ldots,i_k\}} B_{uv}\delta(l_u, l_{(i_1,\ldots,i_k)}) \right). \tag{10}$$

The first term on the right-hand side of (10) are independent of the label of $i_1, \ldots, i_k$. Hence the label updating rule to jump out of the local maxima is:

$$l_{(i_1,\ldots,i_k)}^{\text{new}} = \arg\max_l \left( \sum_{u=1}^n \sum_{v \in \{i_1,\ldots,i_k\}} B_{uv}\delta(l_u, l) \right). \tag{11}$$

If $l_{(i_1,\ldots,i_k)}^{\text{new}} \neq l_{(i_1,\ldots,i_k)}$, the change of label for this set of stubborn nodes from $l_{(i_1,\ldots,i_k)}$ to $l_{(i_1,\ldots,i_k)}^{\text{new}}$ is in effect equivalent to merging the community pairs labeled $l_{(i_1,\ldots,i_k)}$ and $l_{(i_1,\ldots,i_k)}^{\text{new}}$. In real operation, instead of identifying the stubborn nodes and then updating their label according to (11), we directly merge a pair of communities, choosing the one that results in the greatest increase in modularity.

It is often the case that there are several sets of stubborn nodes that block the system from progressing. Of course, we can merge them pair after pair. To enhance the efficiency, we adopt the technique used in MSG [8] that promotes simultaneously merging of multiple pairs of communities at a time. The implementation details are also discussed in Ref. [8].

## References

[1] M.A. Porter, J.P. Onnela, P.J. Mucha, Communities in networks, Notices. Amer. Math. Soc. 56 (2009) 1082–1097.
[2] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026113.
[3] S. Fortunato, C. Castellano, Community Structure in Graphs (2007) arXiv:0712.2716.

[4] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, IEEE Trans. Knowl. Data Eng. 20 (2008) 172–188.
[5] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (2007) 036106.
[6] M.J. Barber, J.W. Clark, Detecting network communities by propagating labels under constraints, Phys. Rev. E 80 (2009) 026129.
[7] I.X.Y. Leung, P. Hui, P. Liò, J. Crowcroft, Towards real-time community detection in large networks, Phys. Rev. E 79 (2009) 066107.
[8] P. Schuetz, A. Caflisch, Efficient modularity optimization by multistep greedy algorithm and vertex refinement, Phys. Rev. E 77 (2008) 046112.
[9] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (1977) 452–473.
[10] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations, Behav. Ecol. Sociobiol. 54 (2003) 396–405.
[11] V. Krebs, A network of co-purchased books about us politics sold by the online bookseller amazon.com., 2008 URL: http://www.orgnet.com/.
[12] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 (2002) 7821–7826.
[13] P. Gleiser, L. Danon, Community structure in jazz, Adv. in Complex Systems 6 (2003) 565.
[14] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, A.L. Barabási, The large-scale organization of metabolic networks, Nature 407 (2000) 651–654.
[15] R. Guimerà, L. Danon, D.A. Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (2003) 065103.
[16] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, A. Arenas, Models of social networks based on social distance attachment, Phys. Rev. E 70 (2004) 056122.
[17] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (2004) 066133.
[18] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111.
[19] A. Noack, R. Rotta, Multi-level algorithms for modularity clustering, Lecture Notes in Comput. Sci. 5526 (2009) 257–268.
[20] G. Agarwal, D. Kempe, Modularity-maximizing graph communities via mathematical programming, Eur. Phys. J. B 66 (2008) 409–418.
[21] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Phys. Rev. E 72 (2005) 027104.
[22] R. Guimerà, L.A.N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (2005) 895–900.
[23] M.E.J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. USA 103 (2006) 8577–8582.
[24] A. Medus, G. Acuna, C.O. Dorso, Detection of community structures in networks via global optimization, Physica A 358 (2005) 593–604.
[25] Z.Q. Ye, S.N. Hu, J. Yu, Adaptive clustering algorithm for community detection in complex networks, Phys. Rev. E 78 (2008) 046115.
[26] G. Xu, S. Tsoka, L.G. Papageorgiou, Finding community structures in complex networks using mixed integer optimization, Eur. Phys. J. B 60 (2007) 231–239.
[27] A. Arenas, J. Duch, A. Fernández, S. Gómez, Size reduction of complex networks preserving modularity, New J. Phys. 9 (2007) 176.
[28] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. (2008) P10008.
[29] B.H. Good, Y.-A. de Montjoye, A. Clauset, The performance of modularity maximization in practical contexts, (2009) arXiv:0910.0165.