Community Structure Analysis Using Label Propagation and Flow-Based Ensemble Learning

Yunchang He, Jian Xu and Bo Yuan
Intelligent Computing Lab, Division of Informatics
Graduate School at Shenzhen, Tsinghua University
Shenzhen 518055, P.R. China
holovewe@163.com, 365701222@qq.com, yuanb@sz.tsinghua.edu.cn

***Abstract*—Network is a powerful paradigm for representing complex relationships and finding the community structure of networks can help people better understand the real world. Infomap, which employs the minimum description length as the optimization objective, is a competent algorithm for community structure analysis. In this paper, we propose a novel algorithm combining flow-based ensemble learning and Label Propagation Algorithm (LPA). Firstly, Infomap (without recursive steps) is incorporated into Core Groups Graph Clustering (CGGC), an ensemble learning framework for community detection. Next, the output of CGGC-Infomap is used as the input of LPA, which can make LPA converge to highly stable clustering results. Experimental studies show that our algorithm can achieve better performance in terms of Normalized Mutual Information (NMI) and requires less memory than the original Infomap algorithm. Our method also features good parallelism, making it potentially more suitable for processing large scale networks.**

Keywords—community detection; ensemble learning; map equation; label propagation

I. INTRODUCTION

Network is an intuitive abstract of a wide range of sophisticated systems in our daily life. For example, at any moment, huge amount of data flows through the network nodes on the Internet, which is probably the largest man-made network. Meanwhile, the ever increasing popularity of smart devices makes it possible for people to communicate with others seamlessly without geographical restrictions, pushing the scale and the complexity of social networks to an unprecedented level.

Usually, valuable information is hidden in the massive networks and advanced techniques are required to discover the underlying patterns. Community detection is a technique of great value for uncovering communities in networks. For example, a social network typically consists of different user groups and, by adopting the user clustering results in recommendation systems, the accuracy of marketing may increase if groups are well partitioned. Furthermore, messages are propagated rapidly via social apps nowadays. Once an advertisement is received and recommended by an influential or core user in a group, much more significant impact can be expected than if the information is sent to an average user. For road networks, community detection can also help transport departments set the shuttle schedule and adjust the routes appropriately.

Modularity [1] is the most widely used metric for evaluating the performance of clustering on networks. This metric considers both the edges connecting different communities and the edges within a community and a number of community detection methods have been proposed based on this measure. Girvan and Newman [2] presented the original concept of modularity and described a greedy algorithm, referred to as the GN algorithm later on. The Louvain algorithm is a simple and efficient method, capable of finding hierarchical community structure [3]. In Core Groups Graph Clustering (CGGC) [4], the randomized greedy (RG) algorithm [5] was used as the base algorithm in the ensemble learning strategy. Other examples based on modularity include the Fast Newman algorithm [6] and Extremal Optimization (EO) [7].

Information-theoretic algorithms are another class of competent methods for community detection. Infomap [8], which is based on the flow of random walks, finds the community structure by minimizing description length per step where map equation is used as the objective function. Early Infomap is used for identify the two-level structure and updated version can uncover multilevel communities. Order statistics local optimization method (OSLOM) [9] is also an information-theoretic algorithm by statistical inference.

Other popular community detection algorithms include the RN method proposed by Ronhovde and Nussinov [10] and Label Propagation Algorithm (LPA) [11].

In this paper, we propose a novel community detection algorithm by combining Infomap, CGGC and LPA where Infomap is used as the base algorithm in CGGC. Section II describes the details of the related algorithms. Section III presents the framework and implementation details of our algorithm. Complexity analysis is given in Section IV. Section V contains experimental results and comparison with the original Infomap and this paper is concluded in Section VI with directions for future work.

II. RELATED WORK

This section gives a detailed overview of the three algorithms used in our proposed techniques, including their

strengths and weaknesses, and how they can be combined for improved performance.

A. Infomap

In random walks, the probability of movements within a community is generally higher than moving to other communities. If we use codes to represent these two types of movements with independent codebooks, compressing codes is equivalent to shortening the description length. Given a partition *M* of a network, the map equation can be defined as:

$$L(M) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} q_{\circlearrowright}^{i} H(\mathcal{P}^{i}), \tag{1}$$

where *m* is the number of communities, $q_{\curvearrowright} H(\mathcal{Q})$ is the average description length of movements between two different communities and $\sum_{i=1}^{m} q_{\circlearrowright}^{i} H(\mathcal{P}^{i})$ is the average description length of movements within community. Eq. 1 calculates the theoretical limit of description length per step according to Shannon's theorem [12].

The final form of map equation is:

$$\begin{aligned} L(M) =& (\sum_{i=1}^{m} q_{i\curvearrowright}) \log(\sum_{i=1}^{m} q_{i\curvearrowright}) - 2\sum_{i=1}^{m} q_{i\curvearrowright} \log(q_{i\curvearrowright}) \\ &+ \sum_{i=1}^{m} (q_{i\curvearrowright} + \sum_{\alpha \in i} p_{\alpha}) \log(q_{i\curvearrowright} + \sum_{\alpha \in i} p_{\alpha}) \\ &- \sum_{\alpha=1}^{n} p_{\alpha} \log p_{\alpha}, \end{aligned} \tag{2}$$

where *n* is the number of nodes, $q_{i\curvearrowright}$ is the probability of movements exiting community *i* and p_{α} is the visiting probability of node α. In directed networks, because of the possible existence of dead ends, teleportation is necessary for unique stable p_{α}.

Assuming that the teleportation probability is τ, we calculate the probability of movements exiting community *i* as follow:

$$q_{i\curvearrowright} = \tau \frac{n - n_i}{n} \sum_{\alpha \in i} p_{\alpha} + (1 - \tau) \sum_{\alpha \in i} \sum_{\beta \notin i} p_{\alpha} w_{\alpha\beta}. \tag{3}$$

Combining (2) and (3), $L(M)$ is only related to p_{α}, which can be easily calculated by PageRank [13]. $w_{\alpha\beta}$ is the relative weight of edge from α to β *s.t.* $\sum_{\beta} w_{\alpha\beta} = 1$.

Infomap utilizes the same strategy as the Louvain algorithm. Each node is classified as the same community as one of its neighbors that reaches the smallest $L(M)$ at the time. If such operations cannot obtain smaller $L(M)$, each community is regarded as a node at a new level and the above operations are repeated on new nodes. Following this procedure, new levels are built continuously until no more improvement of $L(M)$ can be achieved. Note that the nodes cannot leave from their original communities at the new level, but such operations may result in smaller $L(M)$ at later level. Therefore, two strategies can be employed to improve the hierarchical clustering.

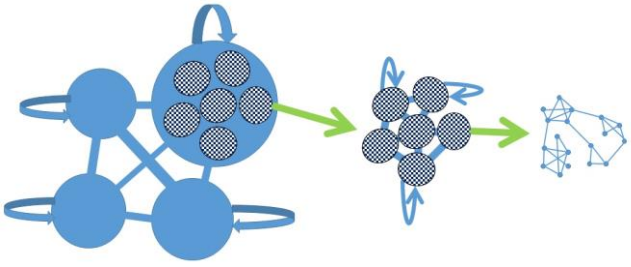

Fig. 1. Recursive execution of the core algorithm in Infomap.

The first one is to regard the communities as separated networks and apply the core algorithm to these communities. Since the core algorithm is hierarchical, there are communities at the lower level in a separated network. In this way, each community at current level is broken into several communities which can be moved freely at the higher level. This strategy can be applied recursively, which is shown in Fig. 1. The second one is to move the nodes in the initial network to their neighboring communities at the top level. In practice, these two methods are applied alternately.

Although the map equation has been extended for multi-level [14] and overlapping community structure analysis [15], Infomap still has some limits. For example, it has been proven that Infomap cannot find the communities with cut size exceeding a certain range [16]. Although the map equation for multi-level can alleviate this issue, it occurs both in synthetic networks and real networks. Meanwhile, the recursive algorithms can occupy significant amount of memory and the running time may increase rapidly. Furthermore, the recursive process has loop-carried dependence and is not suitable for parallel computing.

B. Core Groups Graph Clustering (CGGC)

The authors of the RG+ algorithm, the improved version of RG, extended the ensemble learning strategy of RG+ as CGGC. CGGC gathers the clustering results of existing community detection algorithms to generate partitions, which are called core groups, of the network. Similar to Infomap, each core group is regarded as a node and the network is rebuilt. Finally, a community detection algorithm is used to the rebuilt network. Suppose $M_1, M_2, \ldots, M_k$ are *k* clustering results, the rule of generating core groups is:

$$\begin{aligned} &\forall\, i \in \{0 < i \leq k \mid k \in N\},\ \alpha,\ \beta \in V: \\ &\qquad c_{M_i}(\alpha) = c_{M_i}(\beta) \Rightarrow c_M(\alpha) = c_M(\beta), \end{aligned} \tag{4}$$

$$\begin{aligned} &\exists\, i \in \{0 < i \leq k \mid k \in N\},\ \alpha,\ \beta \in V: \\ &\qquad c_{M_i}(\alpha) \neq c_{M_i}(\beta) \Rightarrow c_M(\alpha) \neq c_M(\beta), \end{aligned} \tag{5}$$

where *V* is the set of nodes, *M* is the partition consisting of core groups and $c_{M_i}(\alpha)$ is the community of α in M_i. The algorithms used in CGGC can be weak, but the results of CGGC tend to be better using strong base algorithms. CGGCi is the iterated version of CGGC where CGGC is used for hierarchical clustering. After obtaining the results of a certain level, communities become nodes at the new level. The last step of CGGCi is the refinement by moving a node to one of its neighbor communities if modularity increases. Algorithm 1 shows the details of CGGCi. CGGC using RG as the base algorithm is called CGGC-RG. The iterated implementation is CGGCi-RG. Both methods can reach good partition, but CGGCi-RG can reach greater modularity than CGGC-RG in most cases. Since RG is aimed at modularity maximization, it tends to merge small

communities and separate large ones as other algorithms with the same objective function [17].

Algorithm 1 iterated core groups graph clustering

input: network $G(V,\ E)$, where V for set of nodes and E for set of edges
output: best partition M_{best}
1: **initialize:** $M \leftarrow \{c_{v_i} = i \mid v_i \in V\}$
2: **repeat**
3: $M_{best} \leftarrow M$
4: $G_{best} = \text{networkRebuilt}(M_{best})$
5: **for** $i = 1 \rightarrow k$ **do**
6: $M_i = \text{baseAlgorithm}(G_{best})$
7: **end for**
8: $M = \text{coreGroups}(\{M_i \mid 0 < i \leq k,\ i \in N\})$
9: **until** M is not better than M_{best}
10: $M_{best} = \text{baseAlgorithm}(G_{best})$
11: **return** $M_{best} = \text{refinement}(M_{best})$

Since CGGC requires *k* partitions, the base algorithm needs to run *k* times. However, its actual running time can be significantly reduced using parallel computing techniques. The good parallelism potential of CGGC is the key motivation that we choose to use CGGC in combination with low time-complexity base algorithms to find a proper balance between effectiveness and efficiency.

C. Label Propagation Algorithm (LPA)

LPA uses an intuitive approach to finding the communities. Firstly, each node is a community with only one node. During the iteration, the community (label) of each node is determined by the labels of its neighbors. There are two ways of updating labels: synchronous and asynchronous. We only introduce asynchronous update due to the possible oscillation of the synchronous update. Parameter *t* is needed to distinguish whether the label of a node has been updated or not. The update function of a given node is:

$$c_\alpha(t) = f(c_{\alpha_1}(t), \ldots, c_{\alpha_r}(t), c_{\alpha_{(r+1)}}(t-1), \ldots, c_{\alpha_s}(t-1)), \quad (6)$$

where $\alpha_1,\ \ldots,\ \alpha_r,\ \alpha_{(r+1)},\ \ldots,\ \alpha_s$ are the neighbors of α and c_α is the label of α. Labels with parameter *t* means that the labels have been updated while the ones with $t-1$ means that the labels have not been updated. Function *f* returns the label with the maximum number.

An obvious disadvantage of LPA is its unstable results. The outputs are tightly related to the order of nodes to be updated and the performance of LPA is not satisfactory in general. However, the speed of LPA is remarkably fast and the memory consumption is extremely low: 95% labels are stable within 5 iterations in experiments [11]. Small extra memory is required to find the label of the maximum number among neighbors, as LPA only uses the label information. Hence, there is great room for improvement and different variations of LPA have been proposed [18].

III. Methodology

In Section II, we present three competitive communities detection algorithms with distinctive strengths and limits. In this section, we discuss how to combine these algorithms to take advantage of their benefits. In summary, we use CGGC to improve the performance of simplified Infomap, which is further refined by LPA.

A. Simplification of Infomap

Although the recursive process can help Infomap obtain good performance in terms of map equation, it leads to the increase of running time and occupied memory. Moreover, in order to avoid local optima, Infomap may be required to run several times. In this paper, we propose to run the core process in Infomap only once, removing the recursive feature of the original Infomap.

B. Core Groups Graph Clustering with Simplified Infomap

With the simplified Infomap, the running time and occupied memory can be reduced. However, its performance will fall inevitably. The good news is that the ensemble learning framework CGGC is good at utilizing different clustering results and Infomap is nondeterministic. That is, by running the simplified Infomap multiple times, it is still possible to achieve satisfactory results while making the entire process parallelable.

As mentioned earlier, modularity optimization is probably the most widely used technique for community structure analytics. In this paper, we combine the simplified Infomap into the framework of CGGC. Although CGGC-RG and CGGCi-RG work well on modularity, they cannot handle directed networks (i.e., our work focuses on directed networks). CGGC/CGGCi with simplified Infomap are named CGGC-Infomap and CGGCi-Infomap, respectively. Such a combination can exploit the information of clustering results produced by the simplified Infomap effectively and alleviate the issue of local optima.

C. Stable Results of Label Propagation Algorithm

LPA is a competitive algorithm in terms of running time and memory usage. The major weakness of LPA is its instability, which comes from the difference in node updating sequences. Furthermore, the final labels are strongly affected by the initial labels. If many nodes are already labeled correctly before applying LPA, it can produce much better results than assigning each node to an individual community.

Note that map equation also has certain limits as the objective function [16]. Although a small value of map equation tends to imply high possibility of a good partition, it should not be used as the sole standard [19]. Even when CGGC-Infomap can obtain a small value of map equation, we still apply its results to LPA for further refinement.

D. Implementation

Algorithm 2 shows the framework of CGGCi-Infomap. Note that LPA tends to merge communities when community boundaries are fuzzy. In some cases, it is possible that only a single community is reserved at the end of LPA, which is obviously meaningless. If this situation happens, we will discard the results of LPA and return the results of the ensemble learning as the final results.

Algorithm 2 LPA with CGGCi-Infomap

input: network $G(V,\ E)$ and terminating threshold ε, where V for set of nodes and E for set of edges
output: best partition M_{best}

initialize: $M \leftarrow \{c_{v_i} = i \mid v_i \in V\}$
$\{p_\alpha \mid \alpha \in V\} = \text{PageRank}(G)$
repeat
  $M_{best} \leftarrow M,\ L_{best} \leftarrow L(M_{best})$
  $G_{best} = \text{networkRebuilt}(M_{best})$
  for $i = 1 \rightarrow k$ **do**
    $M_i = \text{coreAlgorithm}(G_{best})$
  end for
  $M = \text{coreGroups}(\{M_i \mid 0 < i \leq k,\ i \in N\})$
  $L \leftarrow L(M)$
until $L_{best} - L < \varepsilon$
$M_{backup} = M_{best} = \text{coreAlgorithm}(G_{best})$
while $\exists\ c_\alpha \in M_{best},\ c_\alpha \neq f(c_{\alpha_1},\ \ldots,\ c_{\alpha_s})$ **do**
  for all $\alpha \in V$ **do**
    $c_\alpha(t) = f(c_{\alpha_{i1}}(t), \ldots, c_{\alpha_{ir}}(t), c_{\alpha_{i(r+1)}}(t-1), \ldots, c_{\alpha_{is}}(t-1))$
  end for
end while
if $\forall\ \alpha,\ \beta \in M_{best},\ c_\alpha = c_\beta$ **then**
  $M_{best} = M_{backup}$
end if
return M_{best}

core algorithm of Infomap algorithm

input: network $G(V,\ E)$ and terminating threshold ε,
output: best partition M_{best}

initialize: $M \leftarrow \{c_{v_i} = i \mid v_i \in V\},\ L \leftarrow L(M)$
repeat
  $moved \leftarrow \text{false},\ M_{best} \leftarrow M,\ L_{best} \leftarrow L(M_{best})$
  for all $\alpha \in V$ **do**
    $c_\alpha = \text{findSmallestMapEquation}(c_{\alpha_1},\ \ldots,\ c_{\alpha_s})$
    $L = L(M)$
  end for
  if $L_{best} - L < \varepsilon$ **then**
    if $M \neq M_{best}$ **then**
      $G(V,\ E) = \text{networkRebuilt}(M)$
    else
      $moved \leftarrow \text{true}$
    end if
  end if
until $moved = \text{true}$
return M_{best}

For CGGC and CGGCi, the time complexity of finding core groups is O($|V|^2$). However, the running time of this step is relatively small compared to the total running time [5]. Since only two clustering results are combined each time, in our implementation, an array is used to record whether nodes in the same community are in the same old core groups. The array is reset after scanning a community.

IV. Complexity Analysis

The complexity is O($|V|$) for initialization and O($|E|\log|E|$) for sorting edges. The complexity of PageRank is O($k_{pr}(|V|+|E|)$) where k_{pr} is the iteration times. CGGCi needs k_C times for convergence depending on the network. The number of clustering results is recommended to be $\log_2|V|$. The complexity is O($|V|+|E|$) to rebuild the network, but in our implementation it is O($|E|\log|E|$) because two arrays of edges in ascending order are required. The core algorithm of Infomap has the complexity of O($k_{ca}(|V|+|E|)$), where k_{ca} is the iteration times. As descripted in Section III, O($|V|^2$) is the complexity of finding core groups with two clustering results. Because we use $\log_2|V|$ clustering results, the complexity is O($|V|^2\log|V|$) for finding core groups. Thus, the time complexity of the iterative part of CGGCi is O(k_C(O($|E|\log|E|$)+ $k_{ca}(|V|+|E|)\log|V|$)+ O($|V|^2\log|V|$)).

As the number of communities is reduced due to the merging of communities, the number of edges in the rebuilt network decreases. Since the practical running time of LPA is typically very small, O($k_C k_{ca}(|V|+|E|)\log|V|$)) is the complexity of our algorithm in practice. For CGGC-Infomap with LPA, the complexity is O($k_{ca}(|V|+|E|)\log|V|$)).

V. Experiments and Results

We conducted the experiments in C language with Intel Xeon CPU E5-2640 v2 at 2.00Ghz on Ubuntu 15.04. Datasets used were LFR benchmark graphs [20], which are synthetic datasets widely used for the evaluation of community detection algorithms. The exponents of the degree and community size distribution were default values: -2 and -1. We put 'S' after the number of nodes in dataset names to indicate that the ranges of communities were between 10 and 50. Similarly, 'B' means that the ranges of communities were between 20 and 100. The average degree k_{ave} was 20 and the maximum degree k_{max} was 50 unless specified otherwise. The mixing parameters topology and strength were kept unchanged in our experiments. μ stands for the mixing parameter. It is more difficult to partition the network with good performance when μ is greater. All NMI values were averaged across 10 trials on each dataset.

A. Evaluation Function

Normalized Mutual Information (NMI) [21] is a measure to evaluate the performance of clustering. Community detection algorithms are usually evaluated when the real partitions are known in advance.

Suppose X and Y are two random variables, their mutual information is:

$$I(X;\ Y) = \sum_{y \in Y} \sum_{x \in X} P(x,\ y) \log_2 \frac{P(x,\ y)}{P(x)P(y)}, \quad (7)$$

where $P(x,\ y)$ is the joint distribution of X and Y, $P(x)$ and $P(y)$ are the distribution of X and Y respectively. NMI can be calculated by:

$$NMI(X;\ Y) = \frac{2I(X,\ Y)}{H(X) + H(Y)}, \quad (8)$$

where $H(X)$ is the entropy of X defined as:

$$H(X) = \sum_{i=1}^{m} P(x_i) I(x_i) = -\sum_{i=1}^{m} P(x_i) \log_2 P(x_i). \quad (9)$$

B. CGGC-Infomap and CGGCi-Infomap

With LPA, the performance of CGGC-Infomap was compared with CGGCi-Infomap. The plots in Fig. 2 show that two algorithms performed similarly on networks with different sizes, with the only exception of 10000B. Specifically, the dataset with μ = 0.8 in 10000B can be better partitioned by CGGCi-Infomap with LPA. However, its running time increases with the growth of μ. Fig. 3 shows the running time of two algorithms on 10000S and 10000B. As the community boundaries were obscurer, CGGCi-Infomap spent a lot of time on moving the nodes repeatedly and the iteration times of CGGCi became larger at the same time. Although it is difficult to predict which algorithm will perform better in practice because we do not know the value of μ in real networks, CGGC-Infomap with LPA is a good choice if we need to guarantee an acceptable running time.

C. Comparison with Infomap

Since, with LPA, the performance of CGGC-Infomap and the performance of CGGCi-Infomap are similar, we only compared the former with Infomap in the experiments below. Fig. 4 shows the performance of Infomap on the same datasets as above. The NMI of Infomap dropped down when μ = 0.7 in 1000S, compared to 0.75 using our algorithms. A similar pattern can be also found with smaller μ in 1000B. For other datasets with larger network sizes, no clear difference can be observed.

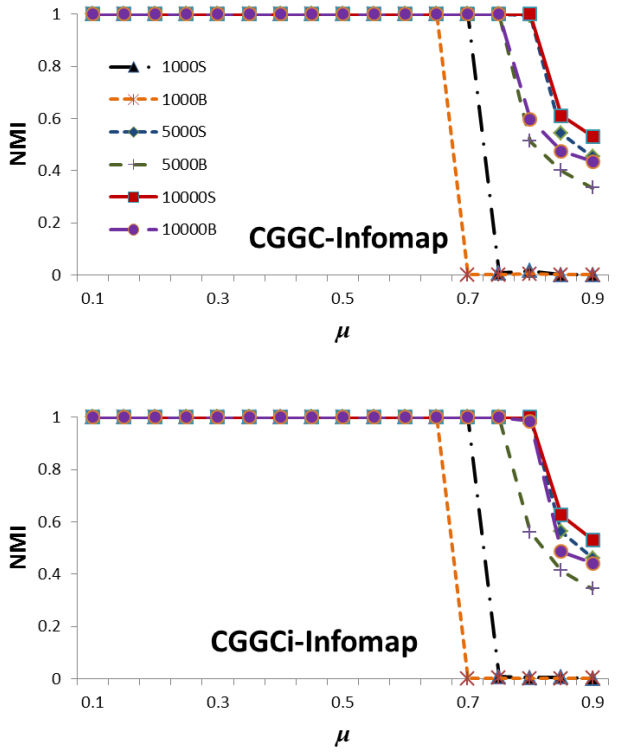

Fig. 2. NMI of CGGC-Infomap and CGGCi-Infomap with LPA.

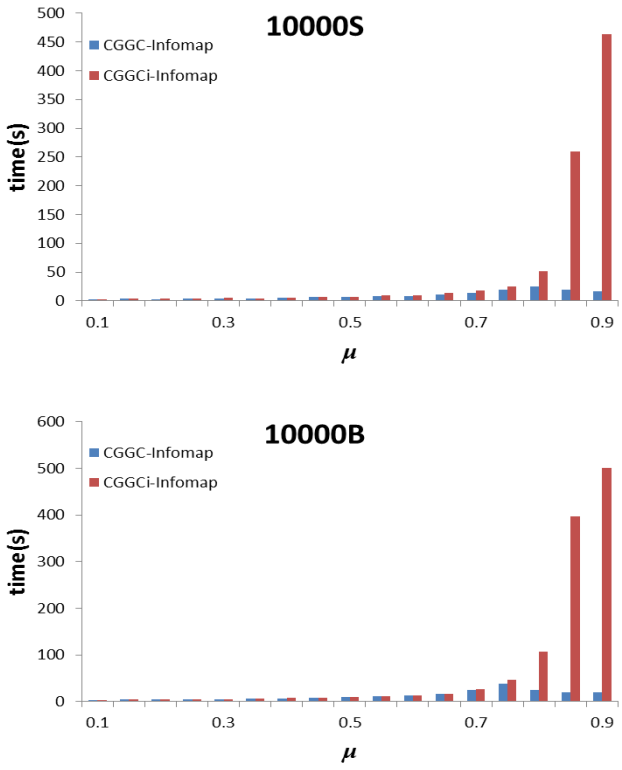

Fig. 3. Running time of CGGC-Infomap and CGGCi-Infomap with LPA.

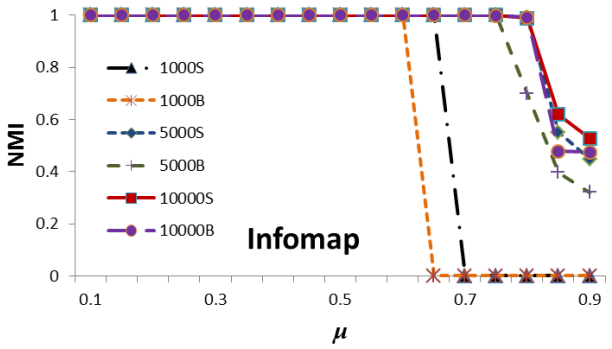

Fig. 4. NMI of Infomap.

Next, the distribution of edges was changed in different ways. In Fig. 5, k_{max} was varied while k_{ave} was unchanged with k_{ave} = 15, μ = 0.5, 10000 nodes and community sizes between 20 and 500. With the growth of k_{max}, obvious difference between Infomap and our algorithm became apparent. The increase of k_{max} means that the density of edges becomes unbalanced. It is clear that our algorithm partitioned the networks better than Infomap in this condition, as shown in Fig. 5. Similarly, in Fig. 6, the the average degree was varied with k_{max} = 50, μ = 0.5, 10000 nodes and community sizes between 20 and 500. The smaller the average degree, the less balanced the distribution of edges. In summary, Fig. 5 and Fig. 6 show that the performance of Infomap may get worse on networks where the density of edges is unbalanced while our algorithm is more robust and can still achieve high NMI values.

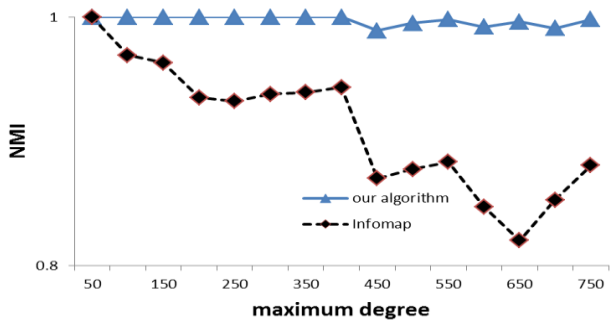

Fig. 5. NMI of Infomap and our algorithm with fixed average degree.

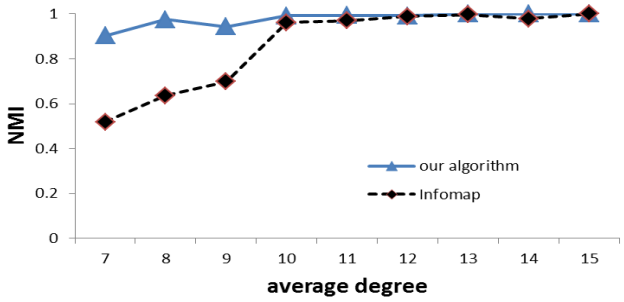

Fig. 6. NMI of Infomap and our algorithm with fixed maxminum degree.

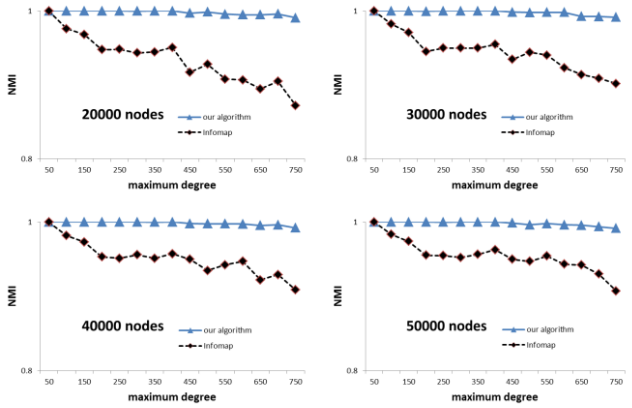

Fig. 7. NMI of Infomap and our algorithm with fixed average degree and different network sizes.

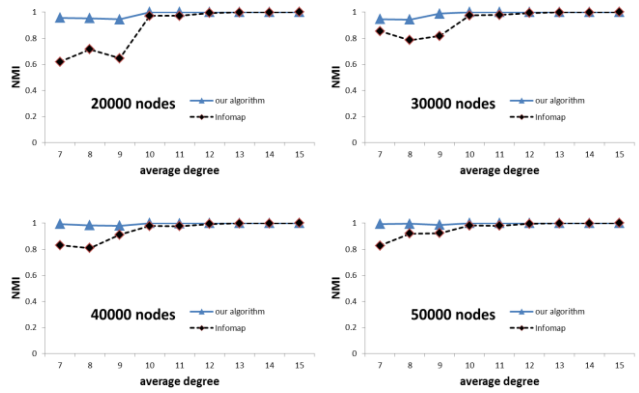

Fig. 8. NMI of Infomap and our algorithm with fixed maxminum degree and different network sizes.

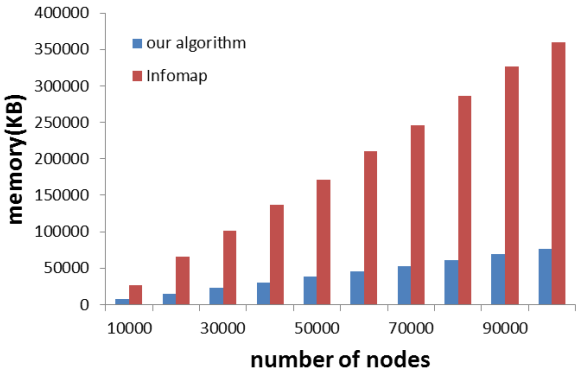

Fig. 9. Comparison on memory consumption between Infomap and our algorithm.

More experiments were conducted with different network sizes. Networks with 20000 to 50000 nodes were tested by Infomap and our algorithm. Experiment results with fixed k_{ave} and fixed k_{max} are shown in Fig. 7 and Fig. 8, respectively. Other parameters were the same as before. Although the two algorithms performed differently on these datasets, it is clear that, regardless of the network size, our algorithm was consistently superior to Infomap in terms of the NMI metric.

Fig. 9 shows the comparison of the memory consumption of Infomap and our algorithm. The parameters were: k_{ave} = 15, k_{max} = 50, μ = 0.5 and community sizes between 20 and 500. We adopted the same measuring method as in [5]. Note that the memory consumption of Infomap is dynamic as it is a recursive algorithm. Therefore, we used the maximum value in the comparison. It is clear that Infomap required several times more memory than our algorithm.

VI. Conclusion

In this paper, we proposed a novel community detection algorithm by combining three popular techniques: ensemble learning, Infomap and LPA. The recursive component of Infomap was removed to reduce its complexity, which was subsequently used as the base algorithm in CGGC and CGGCi. Meanwhile, by taking the output of the above ensemble learning as the input, the instability of LPA can be effectively reduced. Experimental results show that our algorithm achieved higher NMI values than the original Infomap on LFR benchmark graphs, especially when the distributions of edges were unbalanced. Furthermore, the memory usage was effectively reduced by our algorithm.

Since our algorithm is less demanding in terms of memory, it is expected to have better scalability in handling large networks. In the meantime, since both CGGC and CGGCi work on different clustering results, it is possible to exploit task parallelism to run several clustering algorithms simultaneously. Note that the simplified Infomap itself uses the same strategy as the Louvain algorithm, which has already been successfully parallelized [22] using data parallelism. As a result, a major direction for future work would be to fully explore the potential of parallelism of our algorithm and conduct in-depth investigation of its performance on large scale real-world networks.

REFERENCES

[1] Newman M E J, Girvan M. Finding and evaluating community structure in networks[J]. Physical Review E, 2004, 69(2): 026113.

[2] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12): 7821-7826.

[3] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(10): P10008.

[4] Ovelgönne M, Geyer-Schulz A. An ensemble learning strategy for graph clustering[J]. Graph Partitioning and Graph Clustering, 2012, 588: 187.

[5] Ovelgönne M, Geyer-Schulz A. Cluster cores and modularity maximization[C]//Data Mining Workshops (ICDMW), 2010 IEEE International Conference on. IEEE, 2010: 1204-1213.

[6] Newman M E J. Fast algorithm for detecting community structure in networks[J]. Physical Review E, 2004, 69(6): 066133.

[7] Duch J, Arenas A. Community detection in complex networks using extremal optimization[J]. Physical Review E, 2005, 72(2): 027104.

[8] Rosvall M, Bergstrom C T. Maps of random walks on complex networks reveal community structure[J]. Proceedings of the National Academy of Sciences, 2008, 105(4): 1118-1123.

[9] Lancichinetti A, Radicchi F, Ramasco J J, et al. Finding statistically significant communities in networks[J]. PloS one, 2011, 6(4): e18961.

[10] Ronhovde P, Nussinov Z. Multiresolution community detection for megascale networks by information-based replica correlations[J]. Physical Review E, 2009, 80(1): 016109.

[11] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical Review E, 2007, 76(3): 036106.

[12] Shannon C E. A mathematical theory of communication[J]. ACM SIGMOBILE Mobile Computing and Communications Review, 2001, 5(1): 3-55.

[13] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: bringing order to the Web. http://www-db.stanford.edu/~backrub/pageranksub.ps, 1998.

[14] Rosvall M, Bergstrom C T. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems[J]. PloS one, 2011, 6(4): e18209.

[15] Esquivel A V, Rosvall M. Compression of flow can reveal overlapping-module organization in networks[J]. Physical Review X, 2011, 1(2): 021025.

[16] Kawamoto T, Rosvall M. The map equation and the resolution limit in community detection[J]. arXiv preprint arXiv:1402.4385, 2014.

[17] Lancichinetti A, Fortunato S. Limits of modularity maximization in community detection[J]. Physical Review E, 2011, 84(6): 066122.

[18] Wu Z H, Lin Y F, Gregory S, et al. Balanced multi-label propagation for overlapping community detection in social networks[J]. Journal of Computer Science and Technology, 2012, 27(3): 468-479.

[19] Verma A, Butenko S. Network clustering via clique relaxations: A community based approach[J]. Graph Partitioning and Graph Clustering, 2012, 588: 129.

[20] Lancichinetti A, Fortunato S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities[J]. Physical Review E, 2009, 80(1): 016118.

[21] Strehl A, Ghosh J. Cluster ensembles---a knowledge reuse framework for combining multiple partitions[J]. The Journal of Machine Learning Research, 2003, 3: 583-617.

[22] Staudt C L, Meyerhenke H. Engineering parallel algorithms for community detection in massive networks[J]. Parallel and Distributed Systems, IEEE Transactions on, 2016, 27(1): 171-184.