Towards real-time community detection in large networks

Ian X. Y. Leung,* Pan Hui,† Pietro Liò,‡ and Jon Crowcroft§
Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, United Kingdom
(Received 2 September 2008; revised manuscript received 10 March 2009; published 16 June 2009)

The recent boom of large-scale online social networks (OSNs) both enables and necessitates the use of parallelizable and scalable computational techniques for their analysis. We examine the problem of real-time community detection and a recently proposed linear time—$O(m)$ on a network with m edges—label propagation, or "epidemic" community detection algorithm. We identify characteristics and drawbacks of the algorithm and extend it by incorporating different heuristics to facilitate reliable and multifunctional real-time community detection. With limited computational resources, we employ the algorithm on OSN data with 1×10^6 nodes and about 58×10^6 directed edges. Experiments and benchmarks reveal that the extended algorithm is not only faster but its community detection accuracy compares favorably over popular modularity-gain optimization algorithms known to suffer from their resolution limits.

DOI: 10.1103/PhysRevE.79.066107 PACS number(s): 89.75.Hc, 87.23.Ge, 89.20.Hh, 05.10.−a

I. INTRODUCTION

Recent years have seen the flourishing of numerous online social networks (OSNs). Cyber communities such as Facebook, MySpace, and Orkut, where users can keep in touch with friends on the internet, have all emerged as top ten sites globally in terms of traffic. Tools and algorithms to understand the network structures have consequently emerged as popular research topics. By their nature, OSNs contain an immense number of person nodes which are sparsely connected. Edges are often bidirectional since a mutual agreement is required before such friendship links are established. One of the most notable phenomenon in such networks is the resemblance of the so-called six degrees of separation [1] where on average every person is related to another random person via five other people in the real world. This has indeed been shown in real life communities and, much more conveniently, on online communities [2]. Networks which exhibit such small degrees of separation while being sparsely connected are famously known as small-world networks [3].

Well-established online communities often contain tens of millions of users connected by some billions of edges which enable—and necessitate—the use of parallelizable and scalable computational techniques for their analysis. In this literature, we examine the problem of network community detection. Graphically, such communities are characterized by a group of nodes which are densely connected by internal edges but less so toward the outside of the communities, as depicted by the densely connected subgraphs in Fig. 1. Understanding the community structure and dynamics of networks is vital for the design of related applications, devising business strategies and may even have direct implications on the design of the networks themselves [4].

We empirically analyze a recently proposed community detection technique by label propagation discussed in [5], which is summarized as follows. Each node in a network is first given a unique label. Every iteration, each node is updated by choosing the label which most of its neighbors have (the maximal label). If there happens to be multiple maximal labels (which is typical in the beginning), one label is picked randomly. Previous results have shown that this algorithm is extremely efficient in uncovering accurate community structure. As an example, we apply the algorithm on a set of OSN connection data crawled by Mislove *et al.* [4] of 3×10^6 nodes connected by roughly 0.2×10^9 directed links.

We give a survey of related work in the next section and look further into the characteristics of the algorithm in Sec. III. We discuss the potential implementations, improvements, and applications of the algorithm on different types of networks (Sec. IV). Section V gives detailed comparisons between the label propagation algorithm (LPA) and fast modularity-optimization algorithms. We conclude the paper with future directions of research in Sec. VI.

II. RELATED WORK

Community detection in complex networks has attracted ample attention in recent years. Apart from OSNs, researchers have engaged in community analysis in various types of networks. In the case of the internet, examples of communities are found in autonomous systems [6] and indeed web pages of similar topic [7]. In biological networks, it is widely believed that modular structure plays a crucial role in biological functions [8]. Related literatures such as [9–11] may serve as introductory reading, which also include methodological overviews and comparative studies of different algorithms.

The detection of community structure in a network is generally intended as a procedure for mapping the network into a tree [12], known as dendrogram. In this tree, the leaves are the nodes and the branches join them or (at a higher level) groups of them, thus identifying a hierarchy of communities. Nodes can either be agglomerated successively starting from single nodes (agglomerative) or the whole network can be recursively partitioned (divisive). Newman and Girvan [13] introduced a seminal divisive algorithm in which the selec-

*ian.leung@cl.cam.ac.uk
†pan.hui@cl.cam.ac.uk
‡pietro.lio@cl.cam.ac.uk
§jon.crowcroft@cl.cam.ac.uk

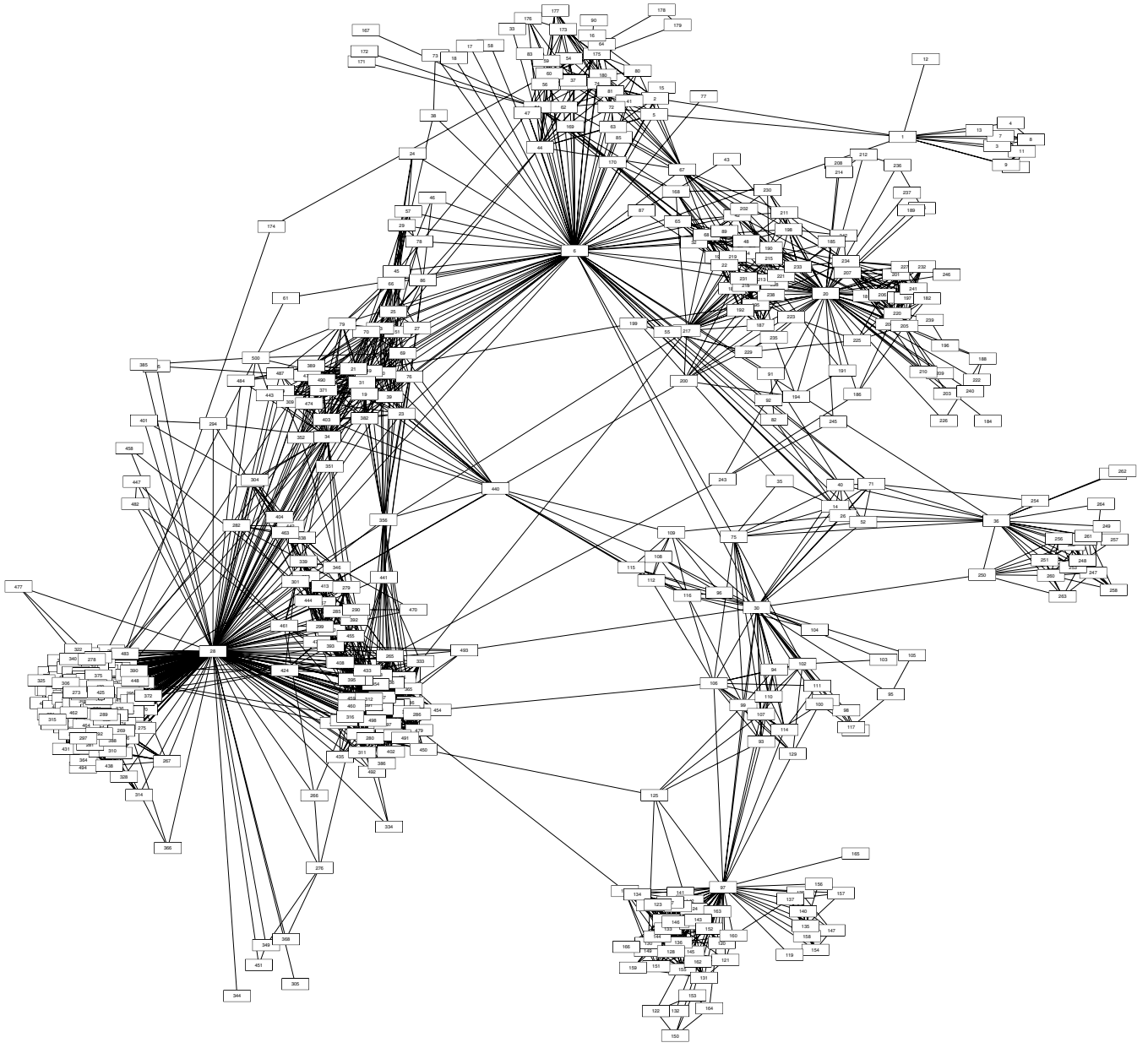

FIG. 1. Snapshot of a subgraph of an OSN (500 nodes).

tion of the edge to be cut is based on the value of its edge betweenness, the number of shortest paths between all node pairs running through it. It is clear that when a graph is made of tightly bound clusters, each loosely interconnected, all shortest paths between nodes in different clusters have to go through the few intercluster connections, which therefore have a large betweenness value. Recursively removing these large betweenness edges would partition the network into communities of different sizes.

Quantitatively, however, we need a metric to measure how well the community detection is progressing, otherwise most algorithms would either continue until every node is split into a single community or all join together into one. Newman and Girvan proposed in [13] a measure of the goodness of communities called *modularity* for the set of uncovered communities $\mathcal{C}$; the modularity is defined to be

$$Q = \sum_{c \in \mathcal{C}} \left[\frac{I_c}{E} - \left(\frac{2I_c + O_c}{2E} \right)^2 \right], \tag{1}$$

where I_c indicates the total number of internal edges that have both ends in c, O_c is the number of outgoing edges that have only one end in c, and E is the total number of edges. This measure essentially compares the number of links inside a given module with the expected value for a randomized graph of the same size and same degree sequence.

The concept of modularity has gained such popularity that it has not only been used as a measure of the community

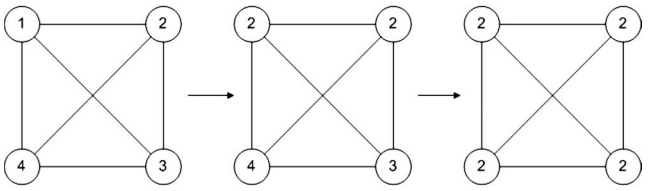

FIG. 2. Each node is looked at in a certain order and a new label is selected. The above shows how nodes in a 4-clique self-organize into one single community in one iteration.

partitioning of a network but also as a key fitness indicator in various community detection algorithms. The algorithm proposed by Clauset, Newman, and Moore (CNM) [14], which greedily combines nodes or communities to optimize modularity gain, is perhaps to date one of the most popular algorithms in detecting communities in relatively large scale networks. In the time when CNM was proposed, it was then the only algorithm capable of community detection on networks of size 500 000 in a matter of hours. Throughout the years, several variations of the CNM have been proposed [15–17]. Most of them concentrate on more efficient data structures as well as modularity gain heuristics to improve the overall performance. A latest adaptation [17] that treats newly combined communities as a single node after each iteration is able to identify community structure on a network containing 1×10^9 edges in a matter of hours.

It is vital, however, to understand that modularity is not a scale-invariant measure, and hence, by blindly relying on its maximization, detection of communities smaller than a certain size is impossible. This is famously known as the resolution limit [18] of modularity based algorithms. Since LPA does not involve modularity optimization, its community detection capability is scale independent and therefore not affected by the resolution limit as will be shown in Sec. V.

III. DISCUSSION

Here, we give a brief discussion on the characteristics of the algorithm as well as some preliminary results applying the algorithm on the OSN described above.

A. Near linear time algorithm

One can consider the label spreading as a simplified but specific case of epidemic spreading where all individuals are considered infectious with their own unique disease. Each person is infected by a disease that is prevalent in his or her neighborhood. Figure 2 depicts the labeling convergence seen in a 4-clique. The number of clusters monotonically decreases each iteration as certain labels become extinct due to domination by other labels. With certain rare and exceptional cases, the labeling self-organizes to an unsupervised equilibrium efficiently.

As suggested in [5], certain properties may prevent the equilibrium from occurring. For instance, a network with a bipartite structure might render the system to oscillate if the algorithm is run *synchronously*, i.e., all nodes are updated together only after they have selected their maximal labels.

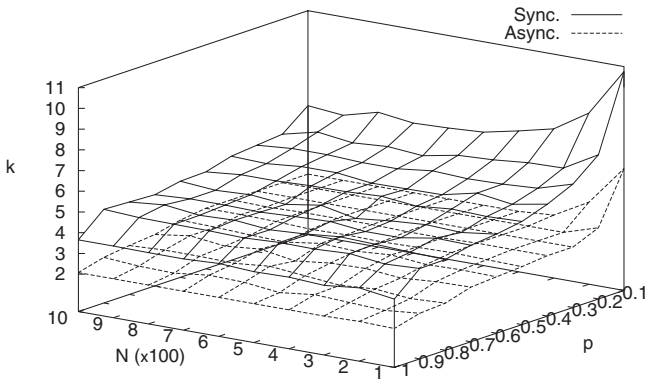

FIG. 3. The above plots show the number of iterations required before convergence for both the synchronous and asynchronous implementations on a random graph of size N with probability of pairwise connection p. All values here are averaged over 100 realizations.

Running the algorithm *asynchronously* in a randomized order every iteration, as suggested in the paper, may result in less definitive results but solves the problem. It was also suggested that a node that has two equally maximal labels to choose from may fail to converge and an extra stopping criterion to prevent the switching of label would have to be in place. It is, however, noted in our implementation that including the concerned label itself into the maximal label consideration effectively avoids all the above nonconvergent behaviors and the requirement for an extra stopping criterion.

In one iteration, each node's neighbors are examined and the maximal label is chosen. The running time of this algorithm is therefore $O(knd)$, where k is the number of iterations, n is the number of nodes, and d is the average degree of nodes. Note that nd can also be described by m, the number of edges. The number of iterations required, k, is dependent on the stopping criterion but is not very well understood. Reference [5] suggested that the number of iterations required is independent to the number of nodes and that after five iterations, 95% of their nodes are already accurately clustered.

Since labels can hardly affect nodes outside their local densely connected substructures, the convergent behavior should be dependent on these substructures rather than the whole network. This is confirmed by preliminary testing and directs us to look at substructures which can ultimately become the community. Experiments show that the average number of iterations required for the labeling to converge (no change in labels) in an N-clique for the asynchronous and synchronous implementations are 2.1 and 3.6, respectively, highly independent of N. To further investigate the average convergent behavior on a substructure, we look at Fig. 3 which summarizes the relationship between number of iterations required before convergence, k, to the pairwise connectivity, p, that controls the edge density in a random graph of size N (where $p=1$ corresponds to the N-clique).

In both implementations, we see that k remains fairly constant over both N and p until p reaches a certain threshold, which when reached we begin to see an inverse dependence between N and k. The overall averages of asynchronous and synchronous implementations in this case are 2.8 and 5.2.

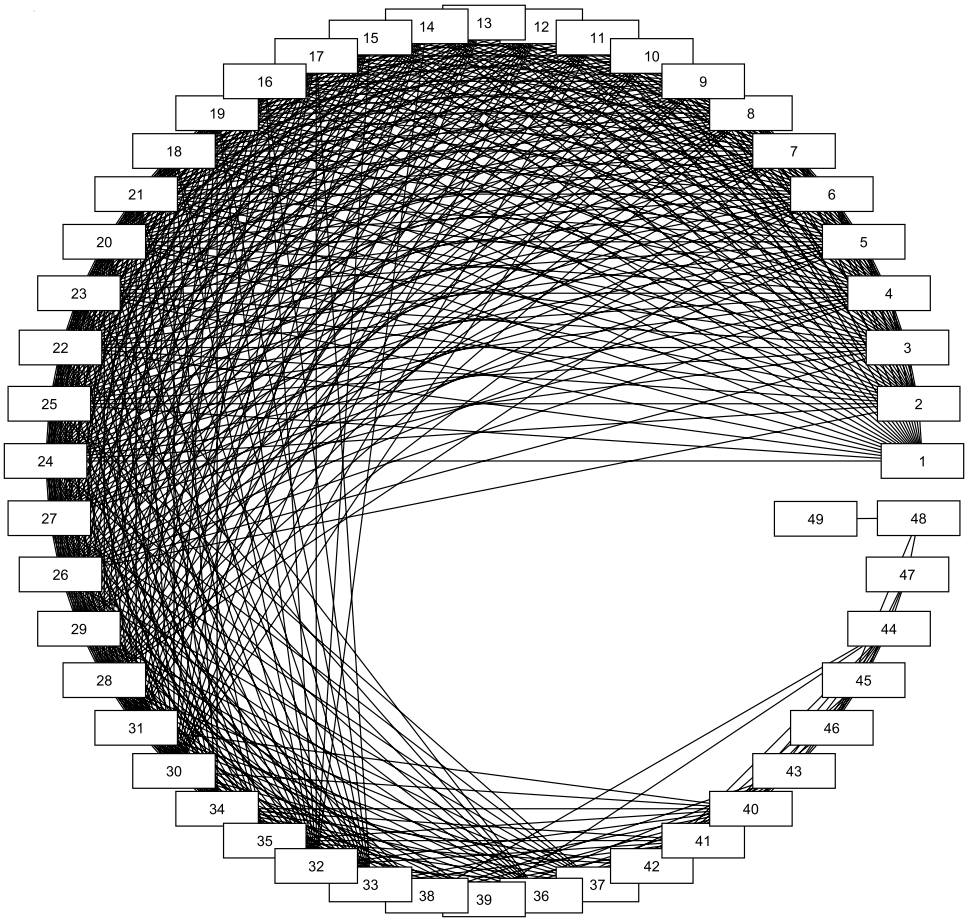

FIG. 4. This substructure is constructed on an N-clique, $N=25$, by attaching each new node, labeled l, $N<l<2N$, to existing nodes $l-1,\ldots,2(l-N)$, thus contains 49 $(2N-1)$ nodes and 600 $[N(N-1)]$ edges.

Let us, however, consider another simple but nonrandom topology. Suppose we start off with an N-clique, at each jth construction, the graph is grown by connecting the $N-j$ most recently joined nodes to the new node (cf. Fig. 4).

These structures by construction will converge into a single community by LPA. Without worrying about how abundant such patterns are in real world communities, we look at the convergent behavior shown in Fig. 5. The trend clearly reveals that k grows logarithmically with respect to N. We therefore suggest the possible worst case of k of the order of $O(\log N)$, where N is the size of the largest substructure with a topology similar to the above. Indeed, we anticipate real world social networks to contain highly heterogeneous substructures which may be intricately connected to affect each other's convergence. We thus consider the understanding of the convergent behavior in large complex networks such as OSNs as a direction for further investigation.

B. Community detection in OSN

We carry out community detection on the aforementioned OSN using a desktop PC with 4 GB ram and a 2.4 GHz quadcore processor running 32-bit JAVA VM 1.6. Due to limited memory, we restrict the number of nodes to the first million. Since the order of nodes in the original data corresponds to that of a breath-first web crawling, this way of "cutting off" the data is equivalent to extracting a snowball sample. As discussed in [4], snowball methods are known to oversample high-degree nodes, undersample low-degree ones, and overestimate the average node degree. This is seen by the higher average degree of the subgraph, 250, compared to 106 of the original graph. Nonetheless, since the purpose of this literature is to evaluate the algorithm on large-scale networks, the sampled network satisfies our requirements. The sampled subgraph contains 1 000 000 nodes and 58 793 458 directed links. Convergent behaviors of the two different implementations are shown in Fig. 6.

A crucial point is that in a complex network as large as this, the so-called "convergence" does not necessarily yield an optimal result in terms of modularity. For example, we see the asynchronous implementation merely took on average five iterations to achieve a maximum modularity but has highly volatile results in different runs as depicted by the shaded area in the figure. On the other hand, the synchronous implementation achieved maximum modularity much slower than the asynchronous version but its performance on average is much more stable (its performance range is thus omitted). The performances of these two different implementations are equally important to be understood and utilized. Further discussions on the implications of these implementations and their utilizations are given in Sec. IV.

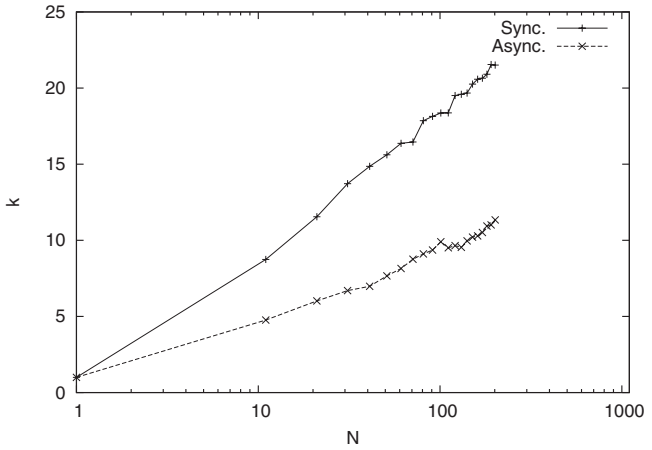

FIG. 5. The relationships between the number of iterations required before convergence, k, of both implementations to the size, N, of the aforementioned structure. All values here are averaged over 100 realizations.

Each single-threaded iteration finishes in a matter of tens of seconds, and thus, depending on the stopping criterion, it can take as little as 8–10 min up to peak performance. Extrapolating the time required with respect to the number of edges, the algorithm without any optimization should be able to detect communities on a graph with 1×10^9 edges in less than 180 min, in a magnitude similar to that in [17].

Figure 7 shows the distribution of community or cluster size collected by a specific run of the asynchronous version of the algorithm when the modularity peaked at 0.638. The size distribution of communities within the OSN follows a two-part power-law distribution in the complementary cumulative distribution function with an estimated coefficient of 1.1. The interested reader is referred to [11,19] for discussions on the characteristics of different networks.

IV. MORE RELIABLE AND EFFICIENT ALGORITHM

In this section, we discuss potential modifications to the algorithm to increase its reliability, functionality, and computational efficiency.

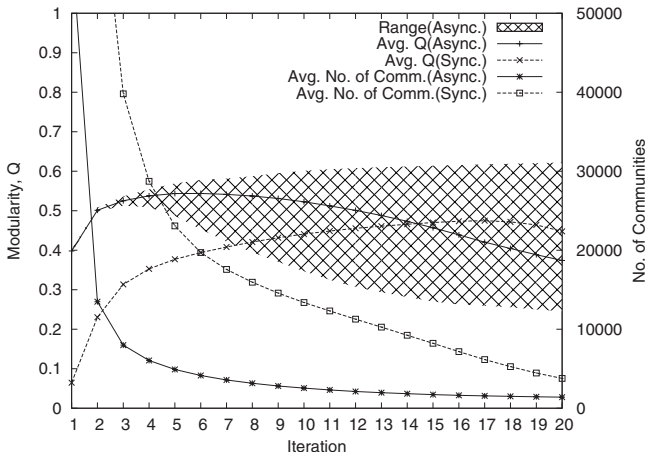

FIG. 6. Average performances of asynchronous and synchronous LPA. Values are averaged over five runs. Shaded area denotes the range of the performances of asynchronous implementation.

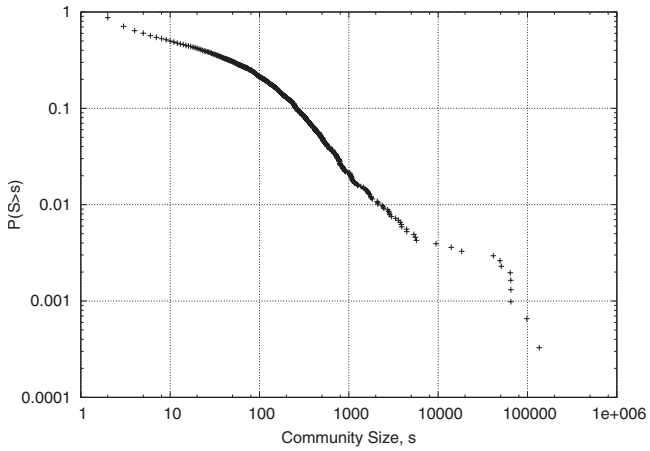

FIG. 7. The community-size distribution of communities uncovered by the algorithm, which follows a two-part power law.

A. Hop attenuation and node preference

Due to the epidemic nature of the algorithm, a major limitation of the algorithm is noted where certain "label epidemic" manages to "plague" a large amount of nodes. To be exact, in some runs a certain community of size over 500 000 (50% of the number of nodes) is formed—as opposed to the thousand other counterparts averagely sized in a magnitude of 100's—greatly contributing to modularity drop after the peak. We conjecture that this is partially due to the asynchronous nature of the algorithm and the initial formation of communities, where certain communities do not form strong enough links to prevent a foreign epidemic to sweep through. Further experiments confirm that the synchronous version of the algorithm slows down the formation of such "monster" communities but do not prevent them.

We propose an extension to this algorithm by adding a score associated with the label which decreases as it traverses from its origin. A node is initially given a score of 1.0 for its label. After a node i has collected from its neighborhood, $\mathcal{N}_i$, all the respective labels and the scores, the calculation of the new maximal label, $\mathcal{L}'$, can be generalized by

$$\mathcal{L}_i' = \underset{\mathcal{L}}{\operatorname{argmax}} \sum_{i' \in \mathcal{N}_i} s_{i'}(\mathcal{L}_{i'}) f(i')^m w_{i',i}, \tag{2}$$

where $\mathcal{L}_i$ is the label of node i, $s_i(\mathcal{L})$ is the hop score of label $\mathcal{L}$ in i, $w_{i',i}$ is the weight of the edge between i' and i (we sum the weights in both directions if the graph is directed), and $f(i)$ is any arbitrary comparable characteristic for any node i. For instance, if we define $f(i)=\mathrm{Deg}(i)$, when $m>0$, more preference is given to node with more neighbors; $m<0$, less. The final step is to assign a new attenuated score s' to the new label $\mathcal{L}'$ of i by subtracting hop attenuation δ, $0<\delta<1$:

$$s_i'(\mathcal{L}_i') = [\max_{i' \in \mathcal{N}_i(\mathcal{L}_i')} s_i(\mathcal{L}_{i'})] - \delta, \tag{3}$$

where $\mathcal{N}_i(\mathcal{L})$ is the set of neighbors of i that has label $\mathcal{L}$. The value δ governs how far a particular label can spread as a function of the geodesic distance from its origin. This addi-

tional parameter adds in extra uncertainties to the algorithm but may encourage a stronger local community to form before a large cluster start to dominate. Ideally, the selection of δ can even be adaptive to current number of iteration, the neighborhood of the node concerned and perhaps some *a priori* network parameters. We investigate the use of varying δ in the next section and assume here a constant value for δ. Note that this setting may induce a negative feedback loop, we therefore let $\delta=0$ if the selected label is equal to the current label.

As discussed, modularity has been widely used in the literature as a metric to contrast the community detection capabilities on real world networks between different algorithms. While high modularity indicates a significant modularized structure over a randomized graph of the network concerned, the correspondence between high modularity and accurately partitioned communities is not well understood due to the resolution limit of modularity. Here we attempt to contrast the behaviors of the algorithms on the OSN based on modularity but shall not draw strong conclusions on the accuracies of the community detection due to the above reason. In Sec. V, a novel benchmark proposed by Lancichinetti *et al.* [20] capable of revealing resolution limit of modularity-based algorithms is used for further comparisons.

Figure 8 depicts the average performance curves over five runs for both versions of the algorithm applying hop attenuation and preferential linkage. The results suggest that, on both implementations, a slight but not too high a preference on high-degree nodes ($m>0$) can speed up the process for achieving peak modularity on the OSN network but also gives rise to a steeper drop as shown in Fig. 8(a). We believe, however, different magnitudes of m simply restrict the choice of nodes to different subsets, some of which may contribute to a global pandemic and some may not. By simply using the degree of a node may not be a heuristic generic enough for different networks. Further study is required to understand, if at all possible, how to deduce a generic preference on neighborhood labels every iteration without resorting to a global metric, which is costly. Nonetheless, we show that giving preference to certain nodes over others when deciding between labels to accept can be beneficial in terms of number of iterations to achieve maximum modularity.

Looking at hop attenuation, we find that the application of δ indeed deters the occurrence of the "monster clusters" as expected and thereby preventing the modularity drop after certain iterations. But it was also obvious that high hop attenuation prevented the healthy growing of the communities and restricted the increase in modularity [cf. Figs. 8(b) and 8(e)]. Moreover, we conjecture that hop attenuation restrains the spread of the label from an arbitrary center and thereby the formation of circular clusters. This suppression in forming noncircular clusters may lead to the suboptimal performance in terms of modularity, as shown in the asynchronous case [Fig. 8(e)]. Finally, from Figs. 8(c) and 8(f), we see that combining both parameters, on average, benefits both versions of the algorithm in achieving a community partitioning of high modularity more efficiently and consistently.

B. Hierarchical and overlapping communities

Communities in certain networks are known to be hierarchical. For instance, students in the same classes often form some strong local communities, while these communities, say of the same school, in turn form a larger but relatively weaker community. As discussed in Sec. II, most CNM-based algorithms are inherently hierarchical since communities are agglomerated by greedy local optimization of modularity gain.

We present two simple modifications to the original method to enable the detection of hierarchical communities.

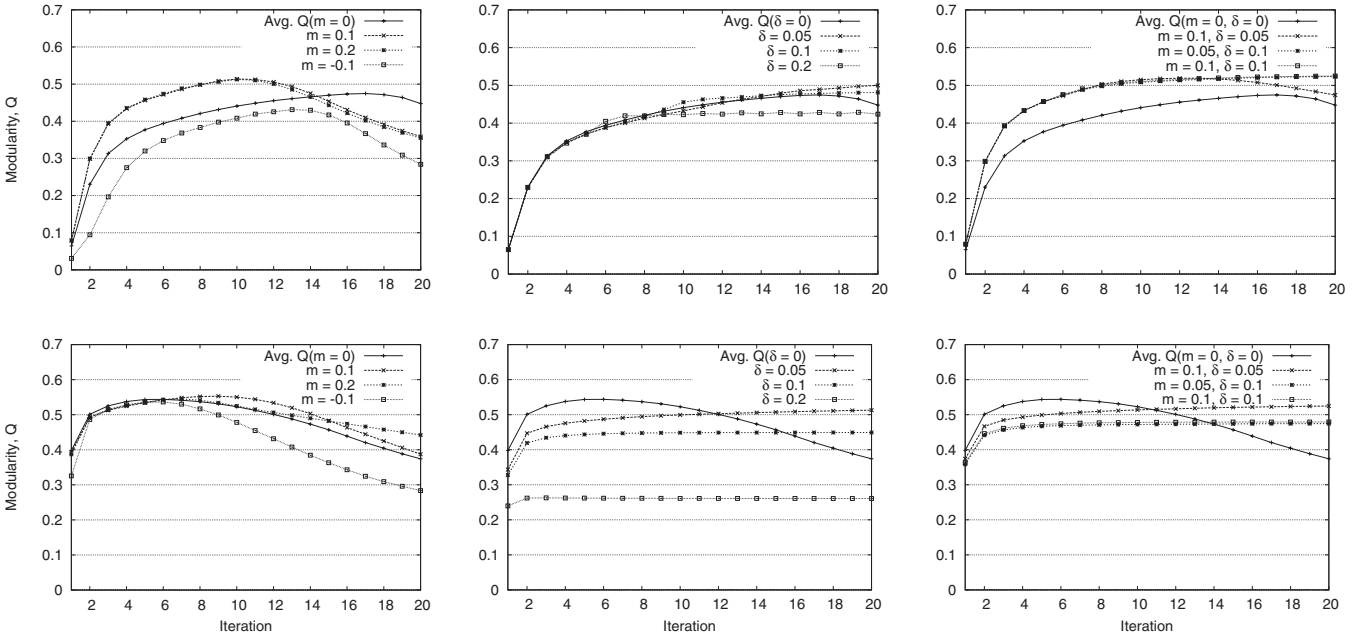

FIG. 8. Average performance comparisons of the synchronous and asynchronous implementations with varying δ and m over five runs.

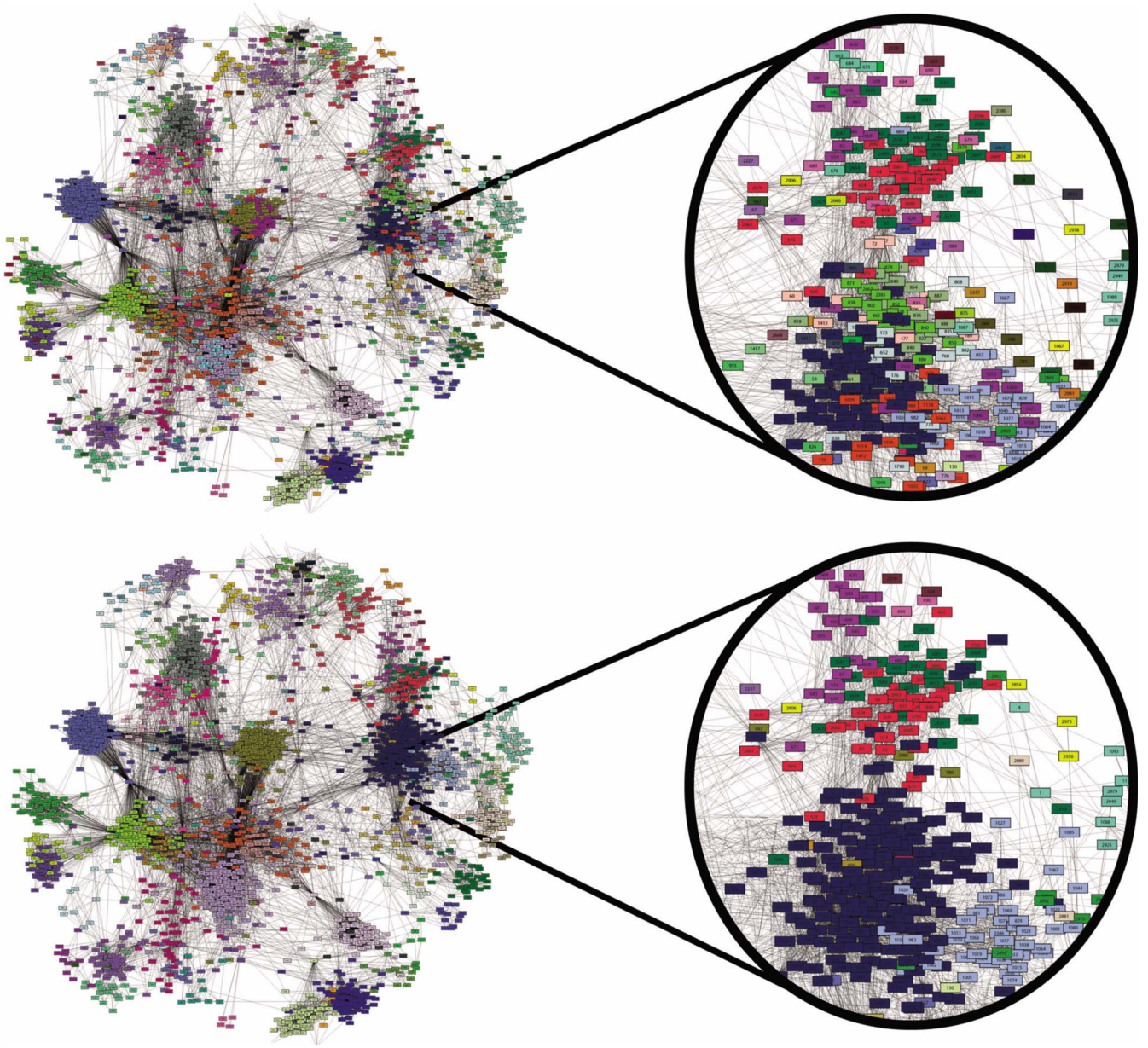

FIG. 9. (Color online) Community detection in the OSN (n=3000) by gradually decreasing hop attenuation (δ=0.5 at the *top* with Q=0.64 and δ=0 at the *bottom* with Q=0.78). Nodes with three or less neighbors are filtered to ease the visualization [21].

First, let us consider the application of hop attenuation on label propagation. Suppose we impose a very high hop attenuation at the beginning, we expect communities of small diameter to form. If we then gradually relax the attenuation value, we should expect these small communities to merge into larger ones. In order to achieve this, we modify Eq. (3) as follows:

$$s_i'(\mathcal{L}_i) = 1 - \delta[d_G(\mathcal{O}(\mathcal{L}_i), i)], \tag{4}$$

where

$$d_G[\mathcal{O}(\mathcal{L}_i), i] = 1 + \min_{i' \in \mathcal{N}_i(\mathcal{L}_i)} d_G[\mathcal{O}(\mathcal{L}_i), i']. \tag{5}$$

Essentially, instead of receiving the current hop scores from the neighborhood and carry out a subtraction, the score is now determined by the actual geodesic distance (d_G) from the label $\mathcal{L}$'s origin denoted by $\mathcal{O}(\mathcal{L})$ and the function δ. This gives greater flexibility of δ in terms of geodesic distances and can facilitate iteration-dependent hop attenuation as required here with slight extra computation cost.

Our second proposal is inspired from [17], where we can similarly treat newly combined communities as a single node, and use the number of intercommunity edges as the weight of edges between these "fresh condensed" nodes. Instead of doing this every iteration, we can apply certain amount of hop attenuation or hard limit in terms of the diameter of the community and do this after an equilibrium is reached.

Figure 9 gives an illustration of the first modification applied on a subgraph on the OSN. Note that this modification

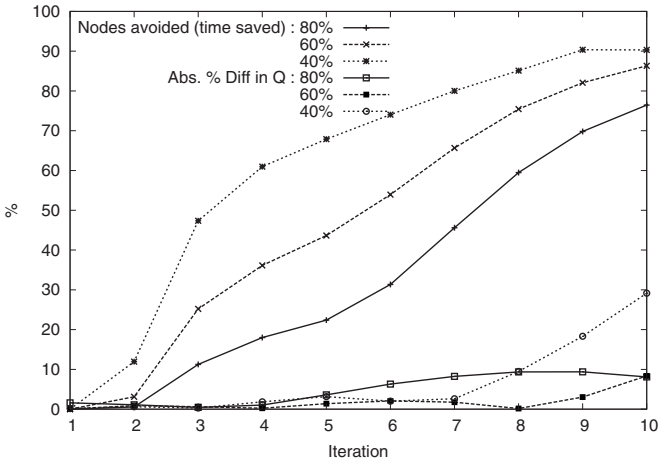

FIG. 10. The difference in percent modularity and speed of the optimized modifications with the original.

depends very much on the initial labeling of nodes because it determines the initial centers of these small communities.

Another important question which was also briefly addressed in [5] is the problem of overlapping communities [22], i.e., nodes can often be considered a member of different communities. From previous sections, we understood that different asynchronous version of the algorithm is capable of generating very different results in different runs. This is exactly how [5] suggested as a potential solution—to rerun the algorithm several times. In a parallel environment, however, the results tend to be much less fluctuating. An initial attempt was to increase the number of labels passed each time between nodes to achieve a similar effect. Preliminary experiments indicate limited success since this setting hampers the convergence process, possibly due to the potential of latent labels switching back and fro in the system. Another possibility is the exploit the fact that nodes on the border of its community have different proportions (purity) of neighbors from other communities. We can potentially use that as a measure of membership but this indeed may only be applicable to such boundary nodes.

C. Optimization

The individual inspection of every node, particularly those with many neighbors, is a crucial factor in determining the speed of the algorithm. Putting aside efficient data structures and prudent programming, an obvious optimization we can do without much compromise on the performance is to selectively update high degree nodes. The reader may have realized that, after certain iterations, it would be pointless to update certain nodes that are well inside a cluster. These nodes are surrounded by nodes with the same label, which are unlikely to change for the same reason. We employ a simple purity measure of neighbors to selectively update nodes that are on the borders of their communities. In other words, we only update nodes whose number of neighbors sharing the maximal label is less than a certain percentage. Indeed, small degree nodes are likely to be avoided in early iterations in this setting but their contributions to the overall community structure and performance are almost insignificant. We carry out the modified algorithm with thresholds set at 100% (equivalent to the unmodified algorithm), 80%, 60%, and 40% to examine the trade off between accuracy and speed.

Figure 10 reveals that after the first iteration, the extra constraint will increasingly avoid updating nodes. As more nodes settle in a more stable cluster, increasingly less amount of time will be required in an iteration. Interestingly, even with a threshold as low as 40%, the absolute difference in modularity compared to the original setting is reasonably small; and we can see the overall running time can be significantly reduced.

D. Parallel and online analysis

Clear advantages of label propagation include its ease to be parallelized and its potential online implementation in real-time networks. Since each node is required only to know information about its neighbors and updates itself according to the common rules, parallelism can be easily achieved. This brings us to another technical point that when the algorithm is completely parallelized even without explicit synchronization, it would tend to behave like the synchronous version of the algorithm. And this is the key reason why we have stressed in the literature that improving both synchronous and asynchronous versions of the algorithm are equally important.

The running time in a parallel environment effectively reduces to k if there are $\Theta(n)$ machines. This can be achieved in real world ubiquitous system such as a mobile *ad hoc* network (MANET) or potentially on OSNs themselves (if members are willing to contribute their computational power) in real time. For instance, social information such as the community structure is known to benefit routing in MANET [23]. Moreover, in such scenarios the space requirement for storing link information would become decentralized and thus insignificant.

On the same note, we see great potential in adapting the algorithm for online community detection in real-time dynamic networks where the presence of nodes and edges are constantly evolving. The microscopic movements and intermittent presence of nodes contribute to changes in terms of weights of the edges. These in turn result in five distinct macroscopic behaviors of communities, namely, growth, shrinkage, union, division, and death of communities. The challenge indeed is to detect local changes without the need for a global update given limited computational resource or time constraint. We believe label propagation is particularly suited in this paradigm and thus propose this as future work.

V. COMPARISONS

We first look at two relatively large and previously studied networks for comparisons. These networks are, respectively, the Amazon Purchasing Network analyzed in [14] and the actor collaboration network [24]. As done in [14], we assume all edges to be undirected to ease the analysis. With the added heuristics, the algorithm is able to perform within

TABLE I. The results correspond to the peak modularity achieved in ten iterations or less, with $f=\mathrm{Deg}$ and $m=0.1$ and a gradually decreasing δ as discussed in Sec. IV B.

Network	Size	Directed Links	Q (claimed)	Peak Q (sync.)	Peak Q (async.)
Amazon purchase (Mar 03)	409687	4929260	0.745 [14]	0.724	0.727
Actor collaboration	374511	30052912	0.528 [5], 0.719 [15]	0.642	0.660

5% of CNM and 10% of the adaptation by Danon, Díaz-Guilera, and Arenas (CNM-DDA) [15] in terms of modularity (c.f. Table I). LPA, however, achieves the result in a matter of minutes which is unparalleled by the above.

For a more standardized comparison, we turn to the recently proposed benchmark graphs by Lancichinetti *et al.* [20], an extension to the Girvan-Newman benchmark [13] which incorporates more realistic scale-free degree and cluster-size distributions. We follow closely the implementation of the benchmark graphs as described in [20] and compare the original LPA with the improved version on the graphs of size 1000 and 5000. To contrast label propagation with general fast modularity maximization algorithms, we also run the benchmarks on the CNM algorithm.

As shown in Fig. 11, both implementations achieve superior accuracy over CNM in terms of normalized mutual information (NMI) even up to a mixing parameter of 0.6. Interestingly, the original method shows signs of failure at $\mu=0.5$ in the $N=1000, \bar{d}=50$ benchmark graphs [cf. Fig. 11(b)]. We believe this corresponds to the formation of monster communities discussed in Sec. IV A. The number of nodes and the average degree of the benchmark graphs in effect dictate the number and sizes of the original communities generated. The results hence point out that denser and less modularized graphs are relatively prone to the formation of monster communities. However, the application of hop attenuation as exemplified in Fig. 11(b) greatly improves the overall performance of LPA in such scenarios.

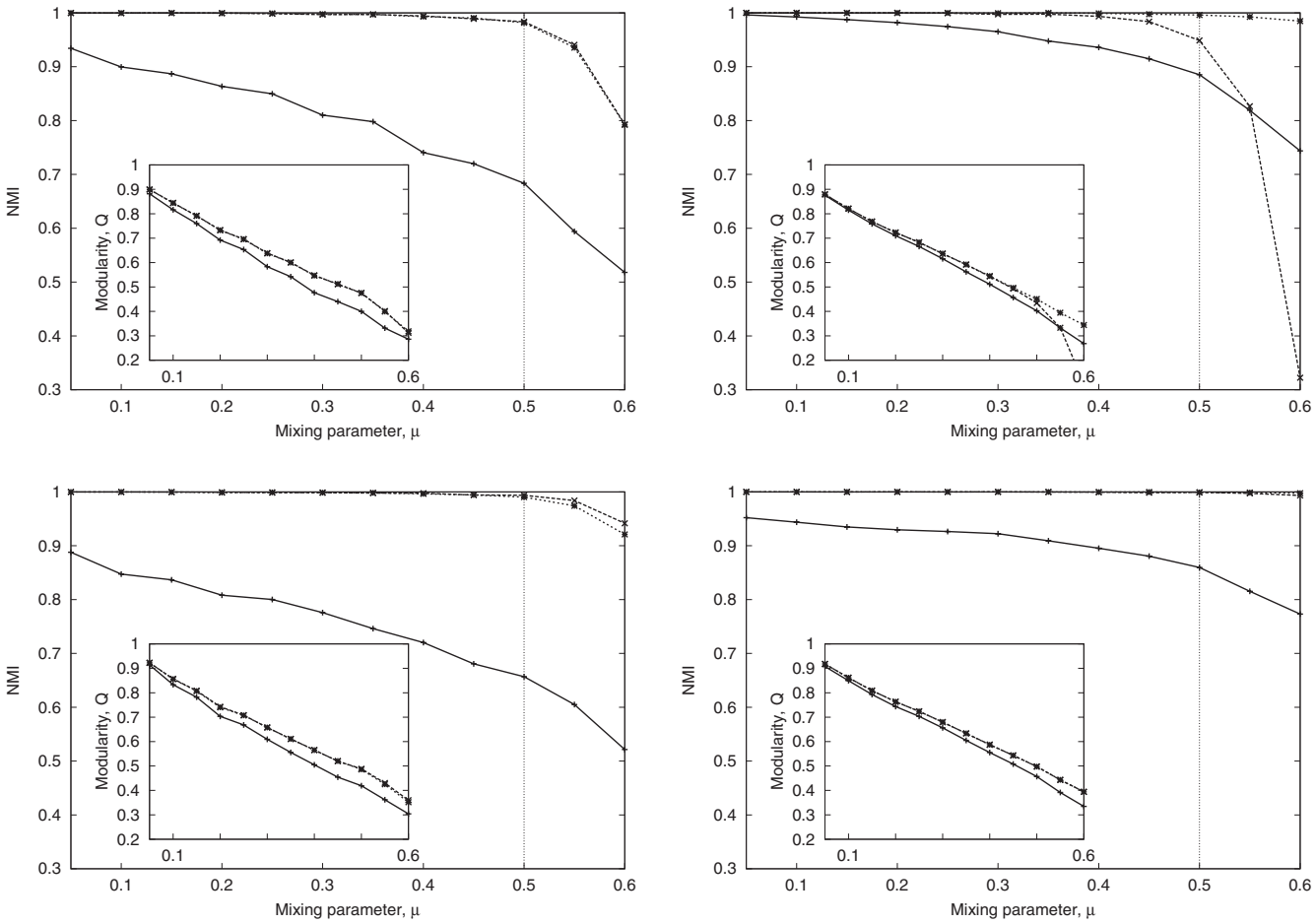

FIG. 11. Average performance comparisons between the three algorithms in the benchmark graphs with size N and average degree $\bar{d}$. Both versions of LPA here are asynchronous; LPA-δ implements a gradually decreasing δ as discussed in Sec. IV B. All benchmark graphs have power-law degree and cluster-size distributions with exponents 3 and 2. For $N=1000$, the results are the average over 100 realizations; for $N=5000$, over 10 realizations.

Importantly, as opposed to label propagation, we can see that CNM algorithm's performance does not merely depend on the mixing parameter but also the average degree of the network. Resolution limit of modularity maximization is reflected by CNM's worse performance in graphs having a smaller average degree. Although in most configurations all algorithms expectedly manage to uncover a modularity value of a similar magnitude, the real accuracy in terms of NMI does not follow. This finding corresponds to the notion in [18] that modularity maximization does not simply translate to actual communities.

VI. CONCLUSIONS

In this paper, we have empirically analyzed a scalable, efficient, and accurate community detection algorithm. We discussed the behaviors and emphasized the importance of both the synchronous and asynchronous implementations of the algorithm. We suggested potential heuristics that can be applied to improve its average detection performance and adaptability. Most importantly, we contrasted the algorithm with modularity-gain-based methods in terms of community detection accuracy and observed how it can be potentially applied online and concurrently in large-scale and real-time dynamic networks.

Understanding the dynamics of this algorithm would be the major future work of this discipline before one devises further heuristics to improve the algorithm. We believe that each notion discussed in Sec. IV is worthy of further inspection. An equally important point is to analyze mathematically or empirically on how to best adapt the algorithm to different types of networks by the added heuristics. How do different network topologies and models affect the algorithm's convergent behavior? These are all valuable questions to be investigated in future work.

In summary, we show that label propagation with the appropriate modifications is a more reliable and efficient method in detecting communities in large-scale networks than popular existing methods. We trust that with further understanding and analysis epidemic-based community detection would be of substantial value to the field.

ACKNOWLEDGMENTS

We thank Franco Bagnoli and Vito Latora for helpful comments. We are grateful to Eric Promislow for providing us with the Amazon network data. This project was supported by EC IST SOCIALNETS under Grant No. 217141.

[1] J. Kleinberg, in *STOC '00: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing* (ACM, New York, NY, 2000), pp. 163–170, ISBN: 1-58113-184-4.
[2] See the Facebook.com Six Degrees Project.
[3] D. J. Watts and S. H. Strogatz, Nature (London) **393**, 440 (1998).
[4] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, in *IMC '07: Proceedings of the Seventh ACM SIGCOMM Conference on Internet Measurement* (ACM, New York, NY, 2007), pp. 29–42.
[5] U. N. Raghavan, R. Albert, and S. Kumara, Phys. Rev. E **76**, 036106 (2007).
[6] D. Lusseau and M. E. J. Newman, Proc. R. Soc. London, Ser B **271**, S477 (2004).
[7] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee, Computer **35**, 66 (2002).
[8] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai and A. L. Barabási, Science **297**, 1551 (2002).
[9] M. E. J. Newman, Eur. Phys. J. B **38**, 321 (2004).
[10] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech.: Theory Exp. (2005) P09008.
[11] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Phys. Rep. **424**, 175 (2006).
[12] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. U.S.A. **101**, 2658 (2004).
[13] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[14] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
[15] L. Danon, A. Díaz-Guilera, and A. Arenas, J. Stat. Mech.: Theory Exp. (2006) P11010.
[16] K. Wakita and T. Tsurumi, in *WWW '07: Proceedings of the 16th International Conference on World Wide Web* (ACM, New York, NY, 2007), pp. 1275–1276.
[17] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech.: Theory Exp. (2008) P10008.
[18] S. Fortunato and M. Barthelemy, Proc. Natl. Acad. Sci. U.S.A. **104**, 36 (2007).
[19] R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).
[20] A. Lancichinetti, S. Fortunato and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[21] http://www.cytoscape.org/
[22] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature **435**, 814 (2005).
[23] P. Hui, J. Crowcroft, and E. Yoneki, in *MobiHoc '08: Proceedings of the Ninth ACM International Symposium on Mobile Ad hoc Networking and Computing* (ACM, New York, NY, 2008), pp.241–250.
[24] A.-L. Barabási and R. Albert, Science **286**, 509 (1999).