Evaluating local community methods in networks

James P Bagrow
Department of Physics, Clarkson University, Potsdam, NY 13699-5820, USA
E-mail: bagrowjp@clarkson.edu

Received 15 November 2007
Accepted 14 April 2008
Published 7 May 2008

Online at stacks.iop.org/JSTAT/2008/P05001
doi:10.1088/1742-5468/2008/05/P05001

Abstract. We present a new benchmarking procedure that is unambiguous and specific to local community finding methods, allowing one to compare the accuracy of various methods. We apply this to new and existing algorithms. A simple class of synthetic benchmark networks is also developed, capable of testing properties specific to these local methods.

Keywords: analysis of algorithms, heuristics, network dynamics, random graphs, networks

ArXiv ePrint: 0706.3880
Evaluating local community methods in networks

Contents

1. Introduction 2
2. Local community detection methods 3
3. Stopping criteria 6
4. Benchmarking 7
   4.1. Test graphs ................................... 7
   4.2. Evaluation .................................... 8
5. Results and discussion 9
6. Conclusions 12
   Acknowledgments 12
   Appendix A. Strong communities 12
   Appendix B. Trailing least-squares 14
   References 15

1. Introduction

The study of complex networks [1]–[3] has recently arisen as a powerful tool for understanding a variety of systems, such as biological and social interactions [4,5], technology communications and interdependencies [1,6], and many others. The problem of detecting communities, subsets of network nodes that are densely connected amongst themselves while being sparsely connected to other nodes, has attracted a great deal of interest due to a variety of applications [7]–[12]. Many techniques have been developed to find these subsets, with a broad array of costs and associated accuracies [13].

Many community finding algorithms hinge upon maximizing a quantity known as modularity [14,15], often defined as

$$Q = \frac{1}{2M} \sum_{v,w} \left( A_{vw} - \frac{k_v k_w}{2M} \right) \delta(c_v, c_w),$$

where $A$ is the adjacency matrix, $M$ is the total number of edges, $k_i$ is the degree of vertex $i$, and $\delta(c_v, c_w) = 1$ if nodes $v$ and $w$ are in the same community and zero otherwise. Thus $Q$ is the fraction of edges found to be within communities, minus the expected fraction if edges were randomly placed, irrespective of an underlying community structure but respecting degree. The second term then acts as a null model, and large values of $Q$ indicate deviations away from a random network structure.

Very efficient algorithms have been created utilizing greedy optimization of $Q$ [15]–[17], but any algorithm using $Q$ must necessarily be a global method, requiring complete knowledge of the entire network. Meanwhile, it has been shown [18] that $Q$ is not ideal and a variety of other techniques exist [13], but these too generally require global knowledge.

doi:10.1088/1742-5468/2008/05/P05001
This knowledge is not available for certain types of networks, such as the WWW, which is simply too large and evolves too quickly to have a fully known structure. In these circumstances, one must rely on a local method capable of finding a particular community within a network, without knowledge of the structure outside of the discovered community. Several local methods exist, all of which attempt to find the community containing a particular starting node [19]–[22].

In this work we present a new technique for quantifying the accuracy of a local method, so that one can determine how various algorithms perform relative to each other. Due to the unique dependence a local method has upon its starting node, we also develop a simple set of ad hoc benchmark networks, with a generalized degree distribution, allowing one to test accuracy when the starting node is a hub, for example. We also present a new local method, as well as several types of stopping criteria indicating when an algorithm has best found the enclosing community. These will serve as a setting in which the new benchmarking procedure will guide improvements in accuracy. We will also show that the most important aspect of the problem is developing new stopping criteria (appropriate agglomeration schemes perform comparably), and that this remains an avenue for further research.

2. Local community detection methods

We focus our new benchmarking procedure on two existing algorithms, due to Clauset [21] and Luo, Wang and Promislow (LWP) [22], as well as a simple new method. Several other local methods exist, including those due to Flake et al [19] and Bagrow and Bollt [20], but these are either reliant on a priori assumptions of network properties (limiting applicability to specific types of networks, such as the WWW), or tend to be accurate only when used as part of a more global method. Other methods (for example, [23, 24, 26, 34]) concern themselves with local community structure, but either require global knowledge to first determine this structure, or are defined locally but do not provide a definitive partition necessary for evaluation [25]–[32]. Some of these methods may work locally with simple estimates of global values such as the total number of nodes but we neglect these as well, mainly for brevity. Some works (e.g. [33]) use similar terminology but are not concerned with local methods in the sense discussed here.

All three chosen algorithms begin exploring the network from a starting node \( s \) and divide the explored portion into two regions: the community \( C \) and the set of nodes adjacent to the community, \( B \) (each has at least one neighbor in \( C \)). At each step, one or more nodes from \( B \) are chosen and agglomerated into \( C \), then \( B \) is updated to include any newly discovered nodes so that all neighbors of nodes in \( C \) are known. This continues until an appropriate stopping criteria has been satisfied. When the algorithms begin, \( C = \{s\} \) and \( B \) contains the neighbors of \( s \): \( B = \{n(s)\} \). See figure 1(a).

The Clauset algorithm focuses on nodes inside \( C \) that form a "border" with \( B \): each has at least one neighbor in \( B \). Denoting this set \( C_{\text{border}} \), and focusing on incident edges, Clauset defines the following local modularity:

\[
R = \frac{\sum_{i,j} \beta_{ij}[i \notin B][j \notin B]}{\sum_{i,j} \beta_{ij}},
\]

where \( \beta_{ij} \) is the number of edges between nodes \( i \) and \( j \).

2.1. The Clauset algorithm

The Clauset algorithm begins with \( C = \{s\} \) and \( B = \{n(s)\} \). At each step, one or more nodes from \( B \) that are in a "border" with \( C \) are chosen and agglomerated into \( C \). The nodes added to \( C \) are those that maximize the local modularity, defined as:

\[
R = \frac{\sum_{i,j} \beta_{ij}[i \notin B][j \notin B]}{\sum_{i,j} \beta_{ij}},
\]

where \( \beta_{ij} \) is the number of edges between nodes \( i \) and \( j \).

2.2. The Luo, Wang and Promislow (LWP) algorithm

The Luo, Wang and Promislow (LWP) algorithm begins with \( C = \{s\} \) and \( B = \{n(s)\} \). At each step, one or more nodes from \( B \) that are in a "border" with \( C \) are chosen and agglomerated into \( C \). The nodes added to \( C \) are those that maximize the local modularity, defined as:

\[
R = \frac{\sum_{i,j} \beta_{ij}[i \notin B][j \notin B]}{\sum_{i,j} \beta_{ij}}.
\]

2.3. Our new local method

We develop a new technique for quantifying the accuracy of a local method, so that one can determine how various algorithms perform relative to each other. Due to the unique dependence a local method has upon its starting node, we also develop a simple set of ad hoc benchmark networks, with a generalized degree distribution, allowing one to test accuracy when the starting node is a hub, for example. We also present a new local method, as well as several types of stopping criteria indicating when an algorithm has best found the enclosing community. These will serve as a setting in which the new benchmarking procedure will guide improvements in accuracy. We will also show that the most important aspect of the problem is developing new stopping criteria (appropriate agglomeration schemes perform comparably), and that this remains an avenue for further research.

2.4. Stopping criteria

Several types of stopping criteria are used in the algorithms:

- **Maximum modularity**: The algorithm stops when the increase in modularity becomes negligible.
- **Maximum perimeter**: The algorithm stops when the perimeter of the community becomes too large.
- **Maximum number of edges**: The algorithm stops when the number of edges in the community becomes too large.

2.5. Ad hoc benchmark networks

We develop a simple set of ad hoc benchmark networks, with a generalized degree distribution, allowing one to test accuracy when the starting node is a hub, for example. These will serve as a setting in which the new benchmarking procedure will guide improvements in accuracy.
Evaluating local community methods in networks

Figure 1. (a) The community $C$ is surrounded by a boundary of explored nodes $B$. This exploration implies an additional layer of nodes that are known only due to their adjacencies with $B$. (b) Two nodes $i$ and $j$ in $B$, with $\Omega_i = 2/3$ and $\Omega_j = -1$. Moving node $j$ into $C$ will give improved community structure, compared to moving $i$.

Figure 2. Comparison between quality measures for the Clauset algorithm, $R$, and the method presented here, $M_{\text{out}}$. Shown are the average of 500 realizations of the 128-node ad hoc networks, for $z_{\text{out}} = 1, 2, \ldots, 6$.

where $\beta_{ij}$ is the adjacency matrix comprising only those edges with one or more endpoints in $C_{\text{border}}$ and $[P] = 1$ if proposition $P$ is true, and zero otherwise. Each node in $B$ that can be agglomerated into $C$ will cause a change in $R$, $\Delta R$, which may be computed efficiently. At each step, the node with the largest $\Delta R$ is agglomerated. This modularity $R$ lies on the interval $0 \leq R \leq 1$ (defining $R = 1$ when $|C_{\text{border}}| = 0$) and local maxima indicate good community separation, as shown in figure 2. For a network of average degree $d$, the cost to agglomerate $|C|$ nodes is $O(|C|^2d)$.

The LWP algorithm defines a different local modularity, which is closely related to the idea of a \textit{weak} community [10]. Define the number of edges internal and external to $C$ as $M_{\text{in}}$ and $M_{\text{out}}$, respectively:

$$M_{\text{in}} = \frac{1}{2} \sum_{i,j} A_{ij}[i \in C][j \in C],$$

$$M_{\text{out}} = \sum_{i,j} A_{ij}[i \in C][j \in B].$$
The LWP local modularity \( M_f \) is then
\[
M_f(C) = \frac{M_{in}}{M_{out}}.
\]
(5)

When \( M_f > 1/2 \), \( C \) is a weak community, according to [10]. The algorithm consists of agglomerating every node in \( B \) that would cause an increase in \( M_f \), \( \Delta M_f > 0 \), then removing every node from \( C \) that would also lead to \( \Delta M_f > 0 \) so long as the node’s removal does not disconnect the subgraph induced by \( C \). (Removed nodes are not returned to \( B \); they are never re-agglomerated.) Finally, \( B \) is updated and the process repeats until a step where the net number of agglomerations is zero. The algorithm returns a community if \( M_f > 1 \) and \( s \in C \). Similar to the Clauset method, the cost of agglomerating \(|C|\) nodes is \( \mathcal{O}(|C|^2 d) \).

Finally, we present a new (almost toy model) algorithm, as an illustration of how simple a local method can be and as a new test setting for our benchmarking procedure. Let us define the ‘outwardness’ \( \Omega_v(C) \) of node \( v \in B \) from community \( C \):
\[
\Omega_v(C) = \frac{1}{k_v} \sum_{i \in n(v)} \left( [i \notin C] - [i \in C] \right)
\]
(6)
\[
= \frac{1}{k_v} \left( k_v^{out} - k_v^{in} \right),
\]
(7)
where \( n(v) \) are the neighbors of \( v \). In other words, the outwardness of a node is the number of neighbors outside the community minus the number inside, normalized by the degree. Thus, \( \Omega_v \) has a minimum value of \(-1\) if all neighbors of \( v \) are inside \( C \), and a maximum value of \( 1 - 2/k_v \), since any \( v \in B \) must have at least one neighbor in \( C \). Since finding a community corresponds to maximizing its internal edges while minimizing external ones, we agglomerate the node with the smallest \( \Omega \) at each step, breaking ties at random. See figure 1(b).

This method is efficient for the following reasons. When a node \( v \in B \) is moved into \( C \), only the neighbors of \( v \) will have their outwardness altered. For a node \( i \in n(v) \), the change in \( \Omega_i \) is just \( \Delta \Omega_i = -2/k_i \) since only a single link can exist between \( v \) and \( i \). If node \( i \) was not previously in \( B \), it will now have a single edge to \( C \) and \( \Omega_i = 1 - 2/k_i \). Calculating \( \Omega_i \) at each step thus requires knowing only \( k_i \), which may be expensive (for example, on the WWW), but needs only be calculated upon the initial discovery of \( i \).

For efficiency, one can maintain a min-heap of the outwardness of all nodes in \( B \) then, at each step, extract the minimum with cost \( \mathcal{O}(\log |B|) \), and update or insert the neighboring \( \Omega \) s. For a network with average degree \( d \), the cost of this updating is \( \mathcal{O}(d^2 \log |B|) \). This is often an overestimate, depending on the community structure, since a node’s degree need only be calculated once. Then, the cost of agglomerating \(|C|\) nodes is \( \mathcal{O}(|C|d^2 \log |B|) \). The relative sizes of \( C \) and \( B \) are highly dependent on the particular network and the current state of the algorithm, but \( |B| \sim |C| \) seems reasonable. A sparse network with rich community structure would give a cost of \( \mathcal{O}(|C| \log |C|) \).

While seeking to agglomerate the least outward nodes at each step seems natural, it lacks a nicely defined measure of the quality of the community, analogous to \( R \) in the Clauset agglomeration. To overcome this we simply track \( M_{out} \) during agglomeration. The smaller this is, the better the community separation, so we expect local minima in

doi:10.1088/1742-5468/2008/05/P05001
M_{out} when a community has been fully agglomerated. In addition, M_{out} can be easily computed alongside agglomeration. After agglomerating node v, the change in M_{out} is just ΔM_{out} = 2k_{out}^{v} − k_v. As shown in figure 3, M_{out} provides useful information about a real-world networks’ community structure, in this case the amazon.com co-purchasing network\(^1\).

Using M_{out} as a measure of quality is not ideal, however: it is not normalized, and (like the Clauset modularity) obtains a trivial value when the entire network has been agglomerated. The latter is less of an issue for local methods. More worrisome is the fact that M_{out} may also be trivially small when C is small. See figure 2 for a comparison of R and M_{out}. We continue to use M_{out} for the sake of simplicity, but more involved measures may certainly lead to improved results.

3. Stopping criteria

After identifying an appropriate agglomeration scheme, a local method must also be able to appropriately stop adding nodes. This point is often neglected and, as will be shown, is a critical component of the accuracy of a local algorithm. Here we suggest two possible schemes and will use the techniques and benchmarks of section 4 to compare them. It is important that the stopping criteria is also local; a criteria that spreads to the entire network then finds, for example, the largest values of ΔM_{out} is no longer a local algorithm.

These stopping criteria are essentially divorced from the agglomeration schemes of most local algorithms, allowing one to mix and match to find more accurate methods. We show this with the Clauset and the new method from section 2. The LWP algorithm already contains a stopping criteria and we use it unaltered.

A subgraph C \subset G is a strong community when every node in C has more neighbors inside C than outside [10,19]. This may be used as a local stopping criterion in the

\(^1\) This data was generated by crawling the actual links on each Amazon product page that point to co-purchased products. This network evolves over time and results are necessarily altered.
following way: agglomerate nodes until \( C \) becomes, and then ceases to be, strong. Unfortunately, this can be too strict, since a single node can terminate the algorithm. Define a \( p \)-strong community as one where this is true for only a fraction \( p \) of nodes in \( C \). Then, one can relax the condition by lowering \( p \). Multiple values of \( p \) can be used simultaneously, at little cost, and the ‘best’ result (smallest \( M_{\text{out}} > 0 \), largest \( R < 1 \)) can be retained as \( C \). We do this for \( \{p\} = \{0.75, 0.76, \ldots, 1\} \). For specific details, see appendix A.

It should be kept in mind that these strong communities can be satisfied by random networks \([24, 35]\), so perhaps this is not the best starting point for a local stopping criteria. Our benchmarking procedure will also show (section 5) that there is room for improvement, especially when the communities are less clearly separated.

Motivated by the appearance of \( M_{\text{out}} \) in figure 2, and the room for improvements over the ‘best of \( \{p\}\)-strong’ criteria, we introduce another stopping criterion that we refer to as Trailing Least-Squares. Fitting a polynomial to the plot of \( M_{\text{out}} \) during agglomeration, one can identify the cusp or inflection point that indicates a community border. This method is somewhat involved but our benchmarking procedure shows that it works quite well. See appendix B. Moreover, the benchmarking procedure was actually used to tune some aspects of this criteria, to maximize accuracy.

4. Benchmarking

We now reach the main focus of this work, a specific method for testing the accuracy of a local algorithm. We will show that our new method provides insight into how and why a local method performs well or poorly. It is also useful for designing new algorithms as well as comparing existing ones, as was done with the trailing least-squares stopping criteria.

4.1. Test graphs

It has become standard practice to test community algorithms with synthetic networks that possess a given community structure and a parameter to control how well separated the communities are. The traditional example is the so-called ‘ad hoc’ networks \([14, 36]\), which typically possess 128 nodes divided into four equally sized communities. Each node has (on average) degree \( z = z_{\text{in}} + z_{\text{out}} = 16 \), where \( z_{\text{out}} \) is the number of links a node has to nodes outside its community. A smaller \( z_{\text{out}} \) (and correspondingly larger \( z_{\text{in}} \)) leads to communities that are easier to detect.

These ad hoc networks have a sharply peaked degree distribution. Since local algorithms are dependent on a particular starting node, their accuracy might be affected if the starting node is a hub or a leaf\(^2\). So one would also like more realistic synthetic networks which possess a wider degree distribution, such as a power law. To do this, we propose the following:

(1) Build a graph \( G \) of \( N \) nodes and \( M \) edges, perhaps using the configuration model and a given degree distribution. Throughout this work, we use Barabási–Albert graphs of \( N = 512 \) and \( m_0 = 8 \).\(^3\)

\(^2\) We term the lowest degree node in the network the ‘leaf’, which is not necessarily of degree 1.

\(^3\) These are built quickly by relaxing the constraint on multi-edges, which are then removed \([42, 43]\). The total number of edges will vary slightly, and the lowest degree nodes often have less than \( m_0 \) neighbors.
(2) Randomly partition the nodes of $G$ into two or more groups. These will serve as the ‘actual’ communities. We limit ourselves to four equally sized partitions.

(3) Choose random pairs of edges that are between the same two groups and rewire them to be within the groups, in such a way that the degree distribution is unaltered.

This rewiring (or switching) technique, replacing edges $(i, j)$ and $(k, l)$ with edges $(i, k)$ and $(j, l)$ [37,38], has been used in the past to destroy the presence of community structure, allowing for a null model to test for false positives [39]. Here we do the opposite, and communities become more sharply separated as the number of rewirings increases.

Since the partition is random, the initial modularity $Q_0$ will be very small. As edges are moved within communities, the first sum in equation (1) will grow but the second term will remain unchanged, since the degree distribution is unaffected. Therefore, the modularity of the actual partition $Q(t)$ after $t$ pairs of edges have been moved is

$$Q(t) = Q_0 + \frac{2}{M} t. \quad (8)$$

Rewiring $M/4$ pairs of edges will give $Q \approx 1/2$.

It has been shown that even random networks can possess large values of $Q$ [24,35]. This is due to the sparsity of such networks when, for example, $\langle k \rangle \leq 2$. The benchmark networks used here possess much higher $\langle k \rangle$.

4.2. Evaluation

Any local method creates a binary partition of the network into the community itself, $C$, and the remaining non-community nodes, $\bar{C} = V - C$. In a realistic setting $V$ is unknown, but synthetic benchmarks allow one to know the full division. In addition, for a synthetic benchmark, the true partition $P_R = \{C_R, \bar{C}_R\}$ is already known, while the found partition $P_F = \{C_F, \bar{C}_F\}$ may differ.

Traditionally, the accuracy of the found communities is quantified by the fraction of correctly identified nodes. This has been shown to have drawbacks [36] and the binary partitioning of a local algorithm poses further problems. For example, if the algorithm fails to stop in time, it has still identified every node in the community correctly; there are just additional nodes incorrectly attributed to that community. Should each incorrect node give a penalty? If the algorithm incorrectly finds one community of $N$ nodes, when there were actually $K$ communities of $N/K$ nodes each, one could assign a $+1/N$ for each correct node and $-1/N$ for each incorrect node, giving a composite score of $2/K - 1$. This means that synthetic networks with different $K$ s cannot be directly compared. While scores could be subsequently re-normalized to lie between 0 and 1, we propose an alternative that avoids these problems and is unambiguous. (Section IX of [24] provides another alternative.)

Following the application introduced in [13], we use normalized mutual information [40,41] to measure how well $P_R$ and $P_F$ correspond to each other:

$$I(P_R, P_F) = \frac{-2 \sum_i \sum_j X_{ij} \log (X_{ij}N/(X_i X_j))}{\sum_i X_i \log (X_i N) + \sum_j X_j \log (X_j N)}, \quad (9)$$

where $X$ is a $2 \times 2$ matrix with $X_{ij}$ being the number of nodes from real group $i$ that were placed in found group $j$, $X_{ij} = X_{1j} + X_{2j}$, and $X_i = X_{i1} + X_{i2}$. In a sense, $I(P_R, P_F)$ is a
Evaluating local community methods in networks

Figure 4. An overall comparison of the various methods for the 128-node ad hoc networks, averaged over 1000 realizations. The LWP method is by far the most accurate for low $z_{\text{out}}$, while the trailing least-squares methods offer the best performance at higher values. (The artificial behavior of both ‘best of $\{p\}$’ criteria for large $z_{\text{out}}$ is discussed in appendix A.)

measure of how much is known about partition $P_R$ by knowing partition $P_F$, with $I = 1$ corresponding to perfect knowledge and $I = 0$ to no knowledge at all.

In general, the confusion matrix $X$ is $N_R \times N_F$ where $N_R$ and $N_F$ are the number of real and found communities, respectively. The application of equation (9) is a limiting case corresponding to the binary partitioning inherent to local algorithms.

In most figures, we have included a ‘faked’ global method, the Clauset–Newman–Moore (CNM) algorithm [15,16], for comparison. This was done by running CNM to find the partitioning with the highest modularity, one random community was designated $C$ and the other communities were grouped together in $\tilde{C}$. A local algorithm is unlikely to match the accuracy of a global method, as shown. More accurate algorithms than CNM exist, meaning the gap between local and global methods is often worse than illustrated.

5. Results and discussion

The results of simulations, shown in figures 4–7, indicate the relative accuracies of the various algorithms and stopping criteria. These figures show how performance degrades as the communities become less separated (larger $z_{\text{out}}$ or smaller number of rewirings). Error bars representing the variance have been omitted for clarity, but note that they are comparable across all algorithms, increase as the communities become more difficult to find and are larger than for the global method.

As shown in figures 4 and 7, the LWP method performs extremely well for clearly separated communities, with a rapid decrease in accuracy as the separation blurs. The ‘best of $\{p\}$-strong’ (figures 4 and 5) and trailing least-squares (figures 4 and 6) stopping criteria first perform at comparable accuracy for both algorithms for the 128-node ad hoc networks, but the trailing least-squares tends to perform better as community distinction blurs. Trailing least-squares outperforms $\{p\}$-strong in the 512-node networks (figure 5 versus figure 6), suggesting that the size of the community impacts accuracy (which might be expected when fitting data).

doi:10.1088/1742-5468/2008/05/P05001
**Figure 5.** Using the ‘best of \{p\}-strong’ criteria on the 512-node rewired scale-free networks, for \{p\} = 0.75, 0.76, \ldots, 1. Each point is the average of 500 realizations. The effect of rejecting any individual \(p\)-strong results where \(M_{out} = 0\) (\(R = 1\)) (see appendix A) is more apparent for these networks, especially for hub nodes.

**Figure 6.** A comparison of the trailing least-squares criteria for both the new algorithm and the Clauset method, using the rewired scale-free networks. Starting from a hub tends to be the most accurate, except when the communities are very well separated.

Overall, the best of \{\(p\)\}-strong has the least accuracy but is also least affected by the degree of the starting node. Meanwhile, trailing least-squares performs better overall but is more dependent on the starting node. The LWP algorithm is also quite accurate overall, though trailing least-squares can outperform it when the community separation is less clear.

The ‘take-home message’ from figures 4–7 is this: the performance of a local algorithm is far more dependent on the stopping criteria than the agglomeration scheme. Both the new algorithm and the Clauset algorithm have nearly identical accuracy when using the same stopping criterion. Additionally, there is no clear winner among the algorithms, and they do not perform nearly as well as global methods. The benchmarking procedure shows that these local methods can benefit from improvements.
Evaluating local community methods in networks

Figure 7. The LWP algorithm used on the rewired scale-free networks. LWP performs very well for large numbers of rewirings, but becomes progressively worse as less edges are moved. Both extremes, hubs and leaves, decrease overall accuracy.

The agglomeration schemes presented share many similarities and a certain amount of ‘cross-pollination’ is possible. For example, accuracy may improve if one maintains the outwardness of nodes after agglomeration and, as per LWP, remove every node from $C$ with positive outwardness. Another possibility is simply agglomerating all nodes with the minimum $\Omega$ together, instead of breaking ties. This is not necessarily a trivial difference: the agglomeration histories may diverge since the sequence of nodes exposed to $B$ can differ.

There is much room open to develop accurate stopping criteria, and this should be a primary focus of further research. For example, the notion of a weak community can also be generalized to provide a (perhaps improved) stopping criteria. As defined, a community is weak when $M_{in} > \frac{1}{2}M_{out}$. This can be generalized by introducing a parameter to control how strict the constraint is: a community is $p$-weak when $M_{in} > pM_{out}$. Thus, a weak community corresponds to $\frac{1}{2}$-weak and the LWP stopping criteria is 1-weak. While the introduction of a further parameter is not ideal, and the lack of performance of the $p$-strong criteria versus the trailing least-squares is not promising, it may still be worth pursuing this and other, similar stopping criteria. Furthermore, stopping criteria using LS-sets and $k$-cores, as mentioned in [10], may also be worth investigation.

In addition to finding a single community, any local method could be easily adapted to find more community structure, simply by running the local algorithm multiple times (possibly without repeated agglomeration of nodes or similar modifications). These quasi-local methods may not have the same level of accuracy as a global method—agglomerating communities sequentially may lead to compounding errors—but it may still be worth pursuing, even if only as an initialization step for a different algorithm.

There is an implicit assumption, in all these methods, that the underlying network is truly undirected. Of course, this is not generally true. In the WWW it is easy to know what pages an explored web page links to, but it is impossible to know how many other pages may link to the explored page. These back links are simply disregarded by the local methods, and it seems a difficult problem to overcome, especially when applying
Evaluating local community methods in networks

a quasi-local method and back links continue to be discovered as more communities are found. One possible way to overcome this is to maintain $\Omega_v$ after agglomeration, then go through all the found communities, remove nodes with, say, $\Omega > 0$, then re-agglomerate them into the community with the smallest outwardness. Another idea, suggested in [19] is to use a global index, such as a search engine, to list all the back links. It seems that, in a different context, such as a partially explored social network, one has no choice but to ignore these back links until they are discovered, then adjust the results accordingly.

6. Conclusions

Much recent work has been applied to the problem of finding communities in complex networks. In this paper, we have focused on the idea of finding a particular community inside of a network without relying on global knowledge of the entire network’s structure, knowledge that is unavailable in a variety of areas. We have introduced a new and very simple local method, with a running time of $O(|C| \log |C|)$. Several types of stopping criteria have been introduced, which can be used in conjunction with different agglomeration schemes.

Using normalized mutual information, we have introduced a simple and unambiguous means of quantifying the accuracy of a local algorithm when applied to a synthetic network with pre-defined community structure. Synthetic networks with generalized degree distributions have been used to allow one to test the impact of the starting node’s degree, something not possible with existing ad hoc networks.

These techniques have been applied to compare the accuracy of a variety of agglomeration schemes and stopping criteria and we feel they will be of great use when testing newly designed local algorithms. The fact that multiple stopping criteria and algorithms can perform with comparable accuracy shows that the community problem is ill-posed to the point of requiring heuristic methods, and thus it is worth using an evaluation scheme to compare and contrast alternative methods. Developing improved stopping criteria should be a goal for future work in this area.

Acknowledgments

We thank E Boltt, D ben-Avraham, L S Schulman, and especially H Rozenfeld for useful discussions; A Clauset for discussions and shared source code; and A Harkin, W Basener, and the RIT Math Department for their hospitality and feedback. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

Appendix A. Strong communities

As per [10, 19], a subgraph $C \subset G$ is a strong community (denoted ‘ideal’ in [19]) when every node in $C$ has more neighbors inside $C$ than outside:

$$k_{in}^i(C) > k_{out}^i(C), \quad \forall i \in C. \quad (A.1)$$

This local quantity allows for a very simple, natural stopping criteria: agglomerate nodes until the community becomes strong then, at each agglomeration step, check $k_{in}$ and $k_{out}$ for the newly chosen node and stop agglomerating if the community would cease to be

doi:10.1088/1742-5468/2008/05/P05001
strong. If $C$ never becomes strong, the algorithm will not terminate, indicating a possible lack of community structure in the explored region of the network.

As shown in figure A.1, this ‘strong to not’ criteria works well for sharply separated communities, but tends to fail as the contrast decreases. In a sense, a strong community is too strong of a requirement: as the distinction between communities blurs, some nodes must fail equation (A.1), despite probable membership in $C$.

We generalize the notion of a strong community in the following way. A community is $p$-strong if equation (A.1) holds, not for all, but only a fraction $p$ (or more) of the nodes:

$$\sum_{i \in C} \left[ k_i^{\text{in}}(C) > k_i^{\text{out}}(C) \right] \geq p |C|.$$  \hfill (A.2)

Equations (A.1) and (A.2) are equivalent when $p = 1$, while the requirement becomes increasingly lenient as $p$ decreases. This allows one to tune the sensitivity by varying $p$. See figure A.2.
An additional benefit of equation (A.2) is that multiple values of \( p \) can be used simultaneously\(^4\), since a community that is \( p_1 \)-strong is also \( p_2 \)-strong (\( p_1 > p_2 \)). More specifically, for the actual fraction \( p_{\text{eff}} \)

\[
p_{\text{eff}} = \frac{1}{|C|} \sum_{i \in C} [k_i^{\text{in}}(C) > k_i^{\text{out}}(C)],
\]

(A.3)

\( C \) is \( p \)-strong for all \( p \leq p_{\text{eff}} \), and not \( p \)-strong for all \( p > p_{\text{eff}} \).

To use, simply choose a set of appropriate parameters, \( \{p_1, p_2, \ldots \} \), perform the local algorithm and maintain the state of \( C \) as each \( p_i \)-stopping criteria is satisfied. One can further use a quality value, such as \( M_{\text{out}} \) or \( R \), and choose the best corresponding \( C_i \) (in this case, that with the smallest \( M_{\text{out}} \) or largest\(^5 \) \( R \)). This ‘best of \{p\}’ stopping criterion does not entirely negate the introduction of a new parameter; choosing \( p \) too small (e.g. \( p = 0.1 \)) can lead to stopping very early. For this work, we use \( \{p\} = \{0.75, 0.76, \ldots, 1.0\} \), but this may be worth further exploration. See figures 4 and 5.

In addition to strong communities, weak communities have been defined [10]. A community is weak when

\[
M_{\text{in}} > \frac{1}{2} M_{\text{out}}.
\]

We have found the usage of a ‘weak-to-not’ stopping criteria to be problematic. The impact of a single agglomeration is so small that the community will blissfully continue to grow, far past the appropriate stopping point. Just as the strong stopping criteria is too strong, a weak stopping criteria is too weak. See section 5 for further ideas, however.

**Appendix B. Trailing least-squares**

Inspired by plots of \( R \) and \( M_{\text{out}} \), and in an effort to increase accuracy when community structure is less favorable, we propose another stopping criteria, based on fitting a polynomial to \( M_{\text{out}} \) (or \( R \)) to find local minima/maxima. Suppose \( n \) nodes have been agglomerated, fit \( y = ax^2 + bx + c \) to the first \( n-3 \) values of \( M_{\text{out}} \). Then extrapolate \( y \) to points \( n-2, n-1, n \) and test the following:

1. parabola opens downward, \( a < 0 \) and,
2. \( M_{\text{out}}(i) > y(i), \ i = n, n-1, n-2 \) and,
3. \( n-3 > -b/2a \) and,
4. \( M_{\text{out}}(n) \geq M_{\text{out}}(n-1) \geq M_{\text{out}}(n-2) \).

If all are satisfied, stop agglomerating (and remove the final three nodes).

As shown in figure A.1’s inset, when you pass the border of the community, \( M_{\text{out}} \) will start to increase, while the parabola, unaware of the next three values, continues downward. This works whether the minima is a cusp or just an inflection point, so one need not resort to testing first versus second differences in \( M_{\text{out}} \), etc. The fitting also provides a degree of smoothing.

\(^4\) Indeed, since stopping criteria are often divorced from agglomeration, all manner of criteria may be used simultaneously, to the point where testing to stop can be more expensive than agglomerating.

\(^5\) We limit ourselves to choosing the smallest \( M_{\text{out}} > 0 \) \( (R < 1) \), unless every \( C_i \) has \( M_{\text{out}} = 0 \) \( (R = 1) \). This distinction is important for finite graphs, causing a curious (and artificial) increase in accuracy for larger values of \( \omega_{\text{out}} \) (smaller numbers of rewirings). This is because inaccurate results that previously spread to most of the network now spread to the entire network and are subsequently being ignored, raising the average value of \( I(P_H, P_F) \).
This criterion is somewhat involved and has several semi-arbitrary factors: one could extrapolate to a different number of points, relax some of the constraints, fit a different order polynomial, continue fitting until the criteria ceases to be satisfied, etc. These choices (especially criteria 2 and 4) were actually made by running the benchmarking procedure over multiple possibilities and choosing the best one, showing that one can use the benchmarks and equation (9) to actually design new algorithms. Our results indicate that these criteria as chosen work well, but further refinement is certainly possible. We also use this criteria by fitting a line to $R$ from the Clauset method, since equation (2) tends to grow linearly in the first community. Both fits have similar accuracy, as shown in figure A.1.

References

[1] Strogatz S H, 2001 Nature 410 268
[2] Albert R and Barabási A-L, 2002 Rev. Mod. Phys. 74
[3] Newman M E J, 2003 SIAM Rev. 45 167
[4] Watts D J and Strogatz S H, 1998 Nature 393 440
[5] Jeong H, Tombor B, Albert R, Oltvai Z N and Barabási A-L., 2000 Nature 407 651
[6] Faloutsos M, Faloutsos P and Faloutsos C, 1999 SIGCOMM ’99: Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication vol 29 (New York: ACM Press) pp 251–62
[7] Girvan M and Newman M E J, 2002 Proc. Nat. Acad. Sci. 99 7821
[8] Newman M E J and Girvan M, 2003 Statistical Mechanics of Complex Networks ed R Pastor-Satorras, J Ruhi and A Díaz-Guilera (Berlin: Springer)
[9] Newman M E J, 2004 Eur. Phys. J. B 38 321
[10] Radicchi F, Castellano C, Cecconi F, Loreto V and Parisi D, 2004 Proc. Nat. Acad. Sci. 101 2658
[11] Porter M A, Mucha P J, Newman M E J and Friend A J, Community structure in the United States House of Representatives, 2007 Physica A 386 414 [physics/0602033]
[12] Newman M E J, 2006 Proc. Nat. Acad. Sci. 103 8577 [physics/0602124]
[13] Danon L, Díaz-Guilera A, Duch J and Arenas A, 2005 J. Stat. Mech. P09008
[14] Newman M E J and Girvan M, 2004 Phys. Rev. E 69 026113
[15] Clauset A, Newman M E J and Moore C, 2004 Phys. Rev. E 70 066111
[16] Newman M E J, 2004 Phys. Rev. E 69 066133
[17] Wakita K and Tsurumi T, 2007 WWW ‘07: Proc. 16th Int. Conf. on World Wide Web (New York: ACM Press) pp 1275–6
[18] Fortunato S and Barthélemy M, Resolution limit in community detection, 2007 Proc. Nat. Acad. Sci. 104 36
[19] Flake G, Lawrence S and Giles C L, 2000 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (Boston, MA) pp 150–60
[20] Bagrow J P and Bollt E M, 2005 Phys. Rev. E 72 046108 [cond-mat/0412482]
[21] Clauset A, 2005 Phys. Rev. E 72 026132
[22] Luo F, Wang J Z and Promislow E, 2006 Web Intelligence (Piscataway, NJ: IEEE Computer Society) pp 233–9
[23] Farutin V et al, Edge-count probabilities for the identification of local protein communities and their organization, 2005 Proteins Struct. Funct. Bioinformatics 62 800–18
[24] Reichardt J and Bornholdt S, 2006 Phys. Rev. E 74 016110
[25] Dernyi I, Palla G and Vícek T, Clique Percolation in Random Networks, 2005 Phys. Rev. Lett. 94 160202
[26] Palla G, Derenyi I, Farkas I and Vícek T, Uncovering the overlapping community structure of complex networks in nature and society, 2005 Nature 435 814
[27] Adámek F, Palla G, Farkas I J, Derenyi I and Vícek T, CFinder: locating cliques and overlapping modules in biological networks, 2006 Bioinformatics 22 1021
[28] Vícek T, Phase transitions and overlapping modules in complex networks, 2007 Physica A 378 20
[29] Palla G, Barabási A-L and Vícek T, Quantifying social group evolution, 2007 Nature 446 664
[30] Gonzalez M C, Herrmann H J, Kertész J and Vícek T, Community structure and ethnic preferences in school friendship networks, 2007 Physica A 379 307
[31] Palla G, Derenyi I and Vícek T, The critical point of k-clique percolation in the Erdős-Rényi graph, 2007 J. Stat. Phys. 128 219
Evaluating local community methods in networks

[32] Palla G, Barabási A-L and Vicsek T, Community dynamics in social networks, 2007 Fluct. Noise Lett. 7 L273
[33] Muff S, Rao F and Callisch A, 2006 Phys. Rev. E 72 056107
[34] Spirin V and Mirny L A, Protein complexes and functional modules in molecular networks, 2003 Proc. Nat. Acad. Sci. 100 12123
[35] Guimerà R, Sales-Pardo M and Amaral L A N, 2004 Phys. Rev. E 70 025101
[36] Danon L, Guiller A D and Arenas A, 2006 J. Stat. Mech. P11010
[37] Maskov S, Steppen K and Zaliznyak A, 2004 Physica A 333 529
[38] Milo R, Kashtan N, Itzkovitz S, Newman M E J and Alon U, 2004 Preprint cond-mat/0312028
[39] Massen C P and Doye J P K, 2005 Phys. Rev. E 71 046101 [cond-mat/0412469]
[40] Strehl A and Ghosh J, 2002 J. Mach. Learning Res. (JMLR) 3 583
[41] Fred A L and Jain A K, 2003 CVPR ’03: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition vol 02, p 128
[42] Batagelj V and Brandes U, 2005 Phys. Rev. E 71 036113
[43] Hagberg A, Schult D and Swart P, Networkx. High productivity software for complex networks, https://networkx.lanl.gov/

doi:10.1088/1742-5468/2008/05/P05001