Evaluating Local Community Methods in Networks

James P. Bagrow
*Department of Physics, Clarkson University, Potsdam NY 13699-5820 USA**
(Dated: October 4, 2007)

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.

PACS numbers: 89.75.Hc 87.23.Ge 89.20.Hh 89.75.-k,

I. INTRODUCTION

The study of complex networks [1, 2, 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, 8, 9, 10, 11, 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:

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

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, 16, 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. This knowledge isn't 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, 20, 21, 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.

II. LOCAL COMMUNITY DETECTION METHODS

We focus our efforts on two existing algorithms, due to Clauset [21] and Luo, Wang, and Promislow (LWP) [22], as well as a new method. Several other local methods exist, including those due to Flake, Lawrence, and Giles [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, 25, 32]) 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 [24, 25, 26, 27, 28, 29, 30, 31].

All three algorithms begin with a starting node s and divide the explored network 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. 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 Fig. 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_{border}, and focusing on incident

*Electronic address: bagrowjp@clarkson.edu

edges, Clauset defines the following local modularity:

$$R = \frac{\sum_{i,j} \beta_{ij}[i \notin B][j \notin B]}{\sum_{i,j} \beta_{ij}}, \tag{2}$$

where β_{ij} is the adjacency matrix comprising only those edges with one or more endpoints in C_{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, ΔR, which may be computed efficiently. At each step, the node with the largest ΔR 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 Fig. 2. For a network of average degree d, the cost to agglomerate $|C|$ nodes is $\mathcal{O}(|C|^2 d)$.

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

$$M_{\text{in}} = \frac{1}{2}\sum_{i,j} A_{ij}[i \in C][j \in C], \tag{3}$$

$$M_{\text{out}} = \sum_{i,j} A_{ij}[i \in C][j \in B]. \tag{4}$$

The LWP local modularity M_f is then:

$$M_f(C) = \frac{M_{\text{in}}}{M_{\text{out}}}. \tag{5}$$

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 algorithm, as an illustration of how simple an effective local method can be. Let us define the "outwardness" $\Omega_v(C)$ of node $v \in B$ from community C:

$$\Omega_v(C) = \frac{1}{k_v}\sum_{i \in n(v)} \Big(\big[i \notin C\big] - \big[i \in C\big] \Big) \tag{6}$$

$$= \frac{1}{k_v}\left(k_v^{\text{out}} - k_v^{\text{in}}\right) \tag{7}$$

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, Ω_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 Ω at each step, breaking ties at random. See Fig. 1(b).

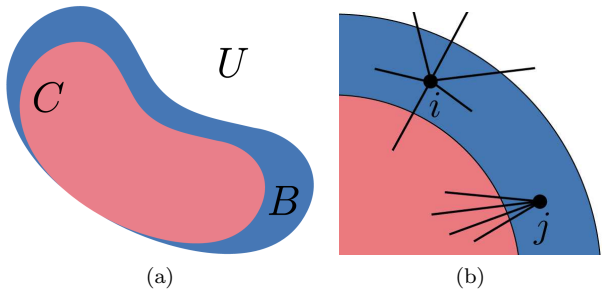

FIG. 1: (color online) (a) The community C is surrounded by a boundary of explored nodes B. This exploration implies an additional layer of nodes that are known only due to their adjacencies with B. (b) Two nodes i and j in B, with $\Omega_i = 2/3$ and $\Omega_j = -1$. Moving node j into C will give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

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 Ω_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 Ω_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 Ω'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 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 $\Delta M_{\text{out}} = 2k_v^{\text{out}} - k_v$. As shown in Fig. 3, M_{out} provides useful information about a real-world networks' community structure, in this case the amazon.com

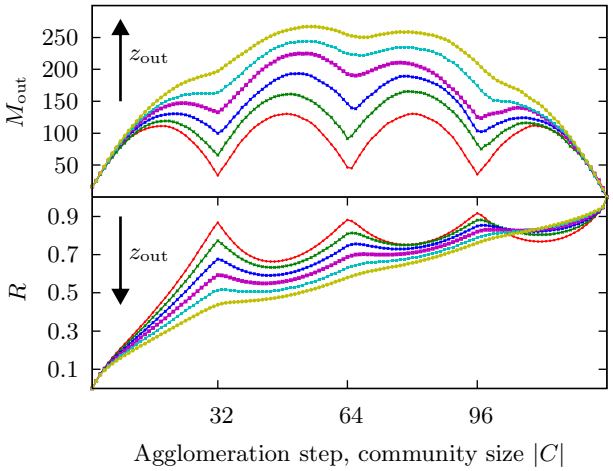

FIG. 2: (color online) Comparison between quality measures for the Clauset algorithm, R, and the method presented here, $M_{\rm out}$. Shown are the average of 500 realizations of the 128 node ad hoc networks, for $z_{\rm out} = 1, 2, \ldots, 6$.

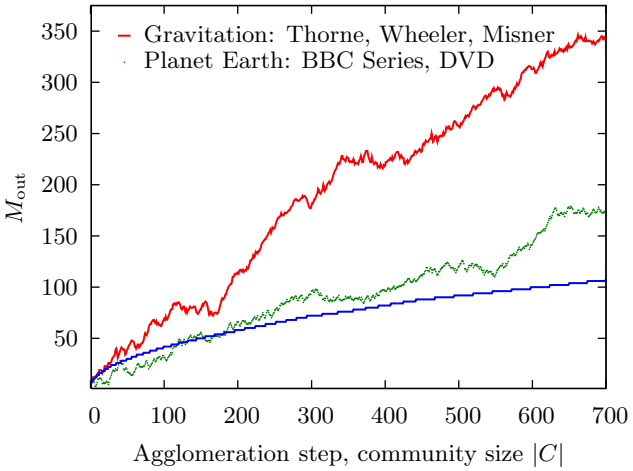

FIG. 3: (color online) Comparison of a seminal physics text and a popular DVD (#1 seller at the time of calculation) on the amazon.com co-purchasing network. Fluctuations in $M_{\rm out}$ in both items indicate the presence of non-trivial community structure. The smooth curve is for a 2D periodic lattice of 500×500 nodes.

co-purchasing network [41].

Using $M_{\rm out}$ as a measure of quality is not ideal, however: it's 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 $M_{\rm out}$ may also be trivially small when C is small. See Fig. 2 for a comparison of R and $M_{\rm out}$. We continue to use $M_{\rm out}$ for the sake of simplicity, but more involved measures may certainly lead to improved results.

III. STOPPING CRITERIA

After identifying an appropriate agglomeration scheme, a local method must also be able to appropriately *stop* adding nodes. Here we suggest two possible schemes and will use the techniques and benchmarks of Sec. IV to compare them. It is important that the stopping criteria is also local; a criteria that spreads to the entire network then finds, e.g., the largest values of $\Delta M_{\rm out}$ 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 new method from Sec. II. 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 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 $M_{\rm out} > 0$, 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.

Another stopping criterion is what we refer to as Trailing Least-Squares. Fitting a polynomial to the plot of $M_{\rm out}$ 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.

IV. BENCHMARKING

A. 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, 33], which typically possesses 128 nodes divided into four equally sized communities. Each node has (on average) degree $z = z_{\rm in} + z_{\rm out} = 16$, where $z_{\rm out}$ is the number of links a node has to nodes outside its community. A smaller $z_{\rm out}$ (and correspondingly larger $z_{\rm in}$) 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 [42]. 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 $m_0 = 8$ [43].

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) [34, 35], has been used in the past to *destroy* the presence of community structure, allowing for a null model to test for false positives [36]. 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 Eq. (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

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

Rewiring $M/4$ pairs of edges will give $Q \approx 1/2$, creating an appreciable amount of community structure in the previously randomized graph.

B. Evaluation

Any local method creates a binary partition of the network into the community itself, C, and the remaining non-communnity nodes, $\tilde{C} = V - C$. 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 $P_R = \{C_R, \tilde{C}_R\}$ is already known, while the found partition $P_F = \{C_F, \tilde{C}_F\}$ 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 [33] 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.

Following the application introduced in [13], we use Normalized Mutual Information [37, 38] to measure how well P_R and P_F correspond to each other:

$$I(P_R, P_F) = \frac{-2\sum_i \sum_j X_{ij} \log\left(\frac{X_{ij}N}{X_{i.}X_{.j}}\right)}{\sum_i X_{i.} \log\left(\frac{X_{i.}}{N}\right) + \sum_j X_{.j} \log\left(\frac{X_{.j}}{N}\right)}, \tag{9}$$

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

V. RESULTS AND DISCUSSION

The results of simulations, shown in Figs. 4–7, indicate the relative accuracies of the various algorithms and stopping criteria. As shown in Figs. 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" (Figs. 6 and 7) and trailing least-squares (Figs. 6 and 8) 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 (Fig. 8 vs. Fig. 9), suggesting that the size of the community impacts accuracy (which might be expected when fitting data).

Overall, the best of $\{p\}$-strong has the least accuracy but is also least affected by the degree of the starting

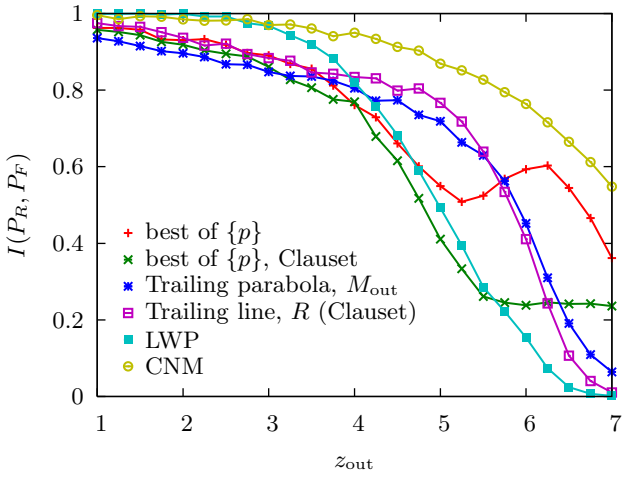

FIG. 4: (color online) An overall comparison of the various methods for the 128-node ad hoc networks, averaged over 1000 realizations. The LWP method is by far the most accurate for low z_{out}, while the trailing least-squares methods offer the best performance at higher values.

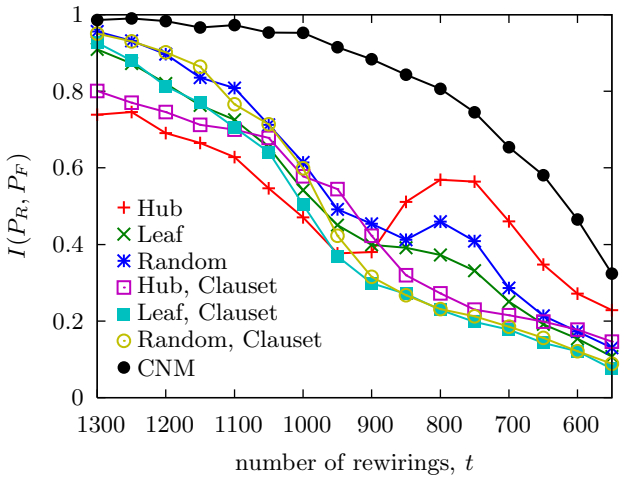

FIG. 5: (color online) Using the "best of $\{p\}$-strong" criteria on the 512-node rewired networks, for $\{p\} = 0.75, 0.76, \ldots, 1$. Each point averaged over 500 realizations. The effect of rejecting any individual p-strong results where $M_{\text{out}} = 0$ ($R = 1$) is more apparent for these networks, especially for hub nodes.

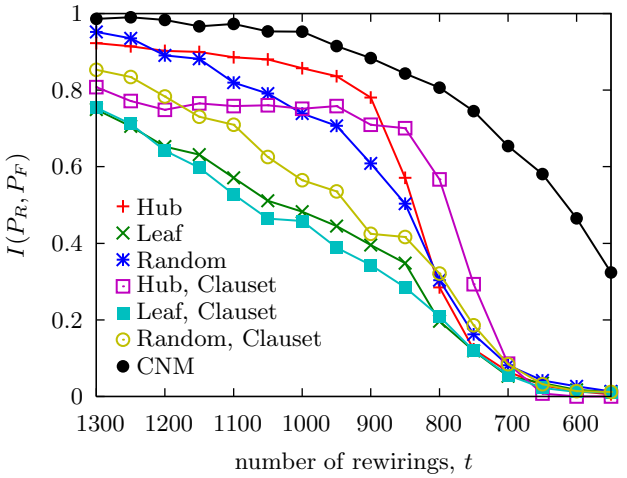

FIG. 6: (color online) A comparison of the trailing least-squares criteria for both the new algorithm and the Clauset method. Starting from a hub tends to be the most accurate, except when the communities are very well separated.

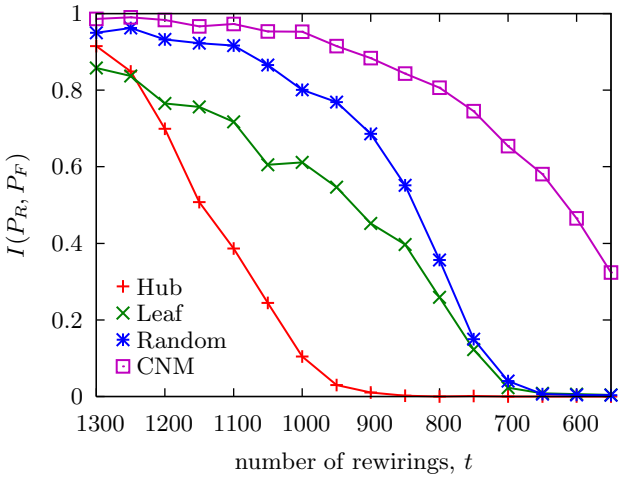

FIG. 7: (color online) The LWP algorithm used on the rewired benchmark networks. LWP performs very well for large numbers of rewirings, but becomes progressively worse as less edges are moved. Both extremes, hubs and leaves, decrease overall accuracy.

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 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 Ω 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. 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 $M_{\text{in}} > \frac{1}{2} M_{\text{out}}$. This can be generalized by introducing a parameter to control how strict the constraint is: a community is p-weak when $M_{\text{in}} > p M_{\text{out}}$. 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 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 Ω_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.

VI. 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 $\mathcal{O}(|C| \log |C|)$. 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.

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:

$$k_i^{\text{in}}(C) > k_i^{\text{out}}(C), \quad \forall i \in C. \tag{A1}$$

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 strong. If C never becomes strong, the algorithm won't terminate, indicating a possible lack of community structure in the explored region of the network.

As shown in Fig. 8, 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 Eq. (A1), despite probable membership in C.

We generalize the notion of a strong community in the following way. A community is p-**strong** if Eq. (A1) holds, not for all, but only a fraction p (or more) of the nodes:

$$\sum_{i \in C} \left[k_i^{\text{in}}(C) > k_i^{\text{out}}(C) \right] \geq p\,|C|\,. \tag{A2}$$

Equations (A1) and (A2) 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 Fig. 9.

An additional benefit of Eq. (A2) is that multiple values of p can be used simultaneously [44], since a community that is p_1-strong is also p_2-strong ($p_1 > p_2$). More specifically, for the actual fraction p_{eff},

$$p_{\text{eff}} = \frac{1}{|C|} \sum_{i \in C} \left[k_i^{\text{in}}(C) > k_i^{\text{out}}(C) \right], \tag{A3}$$

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_{out} or R, and choose the best corresponding C_i (in this case, that with the smallest M_{out} or largest R [45]). This "best

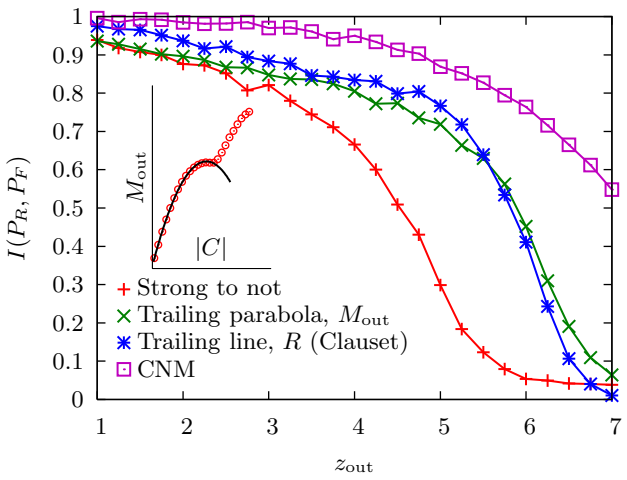

FIG. 8: (color online) The "strong to not" and trailing least-squares stopping criteria for the 128-node ad hoc networks using the Clauset method and the new algorithm presented here. Each point is averaged over 1000 realizations.

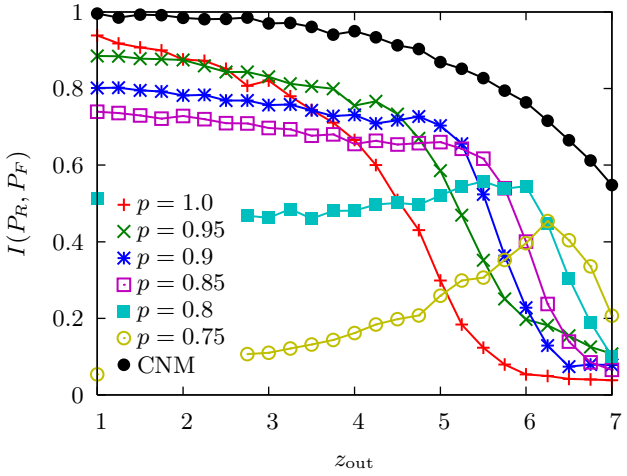

FIG. 9: (color online) Comparison of various p-strong stopping criteria for the 128 node ad hoc networks using the new algorithm shown in Sec. II.

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 Figs. 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 Sec. V for further ideas, however.

APPENDIX B: TRAILING LEAST-SQUARES

Inspired by plots of R and M_{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_{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_{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 Fig. 8's inset, when you pass the border of the community, M_{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_{out}, etc. The fitting also provides a degree of smoothing.

This criteria 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. Our results indicate that this criteria as chosen works well, but further refinement is certainly possible. We also use this criteria by fitting a line to R from the Clauset method, since Eq. (2) tends to grow linearly in the first community. Both fits have similar accuracy, as shown in Fig. 8.

ACKNOWLEDGMENTS

We thank E. Bollt, D. ben-Avraham, 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.

[1] S. H. Strogatz, Nature **410**, 268 (2001).

[2] R. Albert and A.-L. Barabási, Reviews of Modern

Physics **74** (2002).
[3] M. E. J. Newman, SIAM Review **45**, 167 (2003).
[4] D. J. Watts and S. H. Strogatz, Nature **393**, 440 (1998).
[5] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, Nature **407**, 651 (2000).
[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos, in *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication* (ACM Press, New York, 1999), vol. 29, pp. 251–262.
[7] M. Girvan and M. E. J. Newman, Proc Natl Acad Sci USA **99**, 7821 (2002).
[8] M. E. J. Newman and M. Girvan, in *Statistical Mechanics of Complex Networks*, edited by R. Pastor-Satorras, J. Rubi, and A. Diaz-Guilera (Springer, Berlin, 2003).
[9] M. E. J. Newman, The European Physical Journal B **38**, 321 (2004).
[10] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc Natl Acad Sci USA **101**, 2658 (2004).
[11] M. A. Porter, P. J. Mucha, M. E. J. Newman, and A. J. Friend, To appear in *Physica A* (2007), physics/0602033.
[12] M. E. J. Newman, Proc Natl Acad Sci USA **103**, 8577 (2006), physics/0602124.
[13] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, Journal of Statistical Mechanics: Theory and Experiment **2005**, P09008 (2005).
[14] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[15] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
[16] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[17] K. Wakita and T. Tsurumi, in *WWW '07: Proceedings of the 16th international conference on World Wide Web* (ACM Press, New York, 2007), pp. 1275–1276.
[18] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proc Natl Acad Sci USA*, 104(1): 36-41, (2007).
[19] G. Flake, S. Lawrence, and C. L. Giles, in *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Boston, MA, 2000), pp. 150–160.
[20] J. P. Bagrow and E. M. Bollt, Phys. Rev. E **72**, 046108 (2005), cond-mat/0412482.
[21] A. Clauset, Physical Review E **72**, 026132 (2005).
[22] F. Luo, J. Z. Wang, and E. Promislow, in *Web Intelligence* (IEEE Computer Society, 2006), pp. 233–239.
[23] V. Farutin, K. Robison, E. Lightcap, et. al. Edge-count probabilities for the identification of local protein communities and their organization. *Proteins: Structure, Function, and Bioinformatics*, 62(3):800 – 818, September 2005.
[24] I. Dernyi, G. Palla and T. Vicsek. Clique Percolation in Random Networks *Phys Rev. Lett.* 94, (2005) 160202
[25] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814, 2005.
[26] B. Adamcsek, G Palla, I. J. Farkas, I Derenyi and T Vicsek, CFinder: locating cliques and overlapping modules in biological networks, *BIOINFORMATICS*, 22 (2006) 1021-1023
[27] T. Vicsek. Phase transitions and overlapping modules in complex networks *Physica A* 378 (2007) 20-32
[28] G. Palla, A-L. Barabási and T. Vicsek. Quantifying social group evolution, *Nature*, 446 (2007) 664-667,
[29] M. C. Gonzalez, H. J. Herrmann, J. Kertesz and T. Vicsek, Community structure and ethnic preferences in school friendship networks, *Physica A* 379, (2007) 307-316
[30] G. Palla, I. Derenyi and T. Vicsek T, The critical point of k-clique percolation in the Erdos-Renyi graph *J. Stat. Phys.* 128 (1-2) (2007) 219-227
[31] G. Palla, A-L. Barabási and T. Vicsek. Community dynamics in social networks, *Fluct. Noise Lett.* 7 (2007) L273 - L287
[32] V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci USA*, 100(21):12123–12128, (2003).
[33] L. Danon, A. D. Guilera, and A. Arenas, Journal of Statistical Mechanics: Theory and Experiment **2006** (2006).
[34] S. Maslov, K. Sneppen, and A. Zaliznyak, Physica A **333**, 529 (2004).
[35] R. Milo, N. Kashtan, S. Itzkovitz, M. E. J. Newman, and U. Alon (2004), cond-mat/0312028.
[36] C. P. Massen and J. P. K. Doye, Physical Review E **71**, 046101 (2005), cond-mat/0412469.
[37] A. Strehl and J. Ghosh, Journal on Machine Learning Research (JMLR) **3**, 583 (2002).
[38] A. L. Fred and A. K. Jain, in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)* (2003), vol. 02, p. 128.
[39] V. Batagelj and U. Brandes, Physical Review E **71** (2005).
[40] A. Hagberg, D. Schult, and P. Swart, *Networkx. High productivity software for complex networks*, URL `https://networkx.lanl.gov/`.
[41] 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.
[42] We term the lowest degree node in the network the "leaf," which is not necessarily of degree 1.
[43] These are built quickly by relaxing the constraint on multi-edges, which are then removed [39, 40]. The total number of edges will vary slightly, and the lowest degree nodes often have less than m_0 neighbors.
[44] 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.
[45] We limit ourselves to choosing the smallest $M_{\rm out} > 0$ ($R < 1$), unless *every* C_i has $M_{\rm out} = 0$ ($R = 1$). This distinction is important for finite graphs, causing a curious (and artificial) increase in accuracy for larger values of $z_{\rm out}$ (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 $I(P_R, P_F)$.