Detecting highly overlapping communities with Model-based Overlapping Seed Expansion

Aaron McDaid and Neil Hurley
School of Computer Science and Informatics, University College Dublin, Ireland
aaronmcdaid@gmail.com neil.hurley@ucd.ie

***Abstract*—As research into community finding in social networks progresses, there is a need for algorithms capable of detecting overlapping community structure. Many algorithms have been proposed in recent years that are capable of assigning each node to more than a single community. The performance of these algorithms tends to degrade when the ground-truth contains a more highly overlapping community structure, with nodes assigned to more than two communities. Such highly overlapping structure is likely to exist in many social networks, such as Facebook friendship networks. In this paper we present a scalable algorithm, MOSES, based on a statistical model of community structure, which is capable of detecting highly overlapping community structure, especially when there is variance in the number of communities each node is in. In evaluation on synthetic data MOSES is found to be superior to existing algorithms, especially at high levels of overlap. We demonstrate MOSES on real social network data by analyzing the networks of friendship links between students of five US universities.**

I. Introduction

In this paper we introduce MOSES, a Model-based Overlapping Seed ExpanSion[1] algorithm, for finding overlapping communities in a graph. The algorithm is designed to work well in applications, such as social network analysis, in which the graph is expected to have a complex, highly-overlapping community structure.

Many of algorithms for finding communities in graphs have been limited to *partitions*, where each node is assigned to exactly one community. While there are still very many open questions about the basic structure of empirical graphs, it is difficult to accept that a partition is an appropriate description of the complete community structure in a graph. In recent years, many algorithms have been proposed to detect overlapping communities. We repeat experiments similar to those carried out in [1], which show that many such algorithms are only capable of detecting weakly overlapping community structure, where a typical node is in just two communities. If we are to be able to make reasonable inferences about the community structure in empirical graphs, we need algorithms capable of detecting highly overlapping communities, if only so that we can credibly rule out highly overlapping community structure for a given graph.

The method presented here is similar in spirit to many existing algorithms, in that a global objective function is defined to assign a score to each proposed community assignment. The algorithm proceeds by using simple heuristics to search for communities in the graph, maximizing our objective function. This allows for scalability as the MOSES objective function can be efficiently updated throughout.

[1]Our C++ implementation of MOSES is available at http://sites.google.com/site/aaronmcdaid/moses.

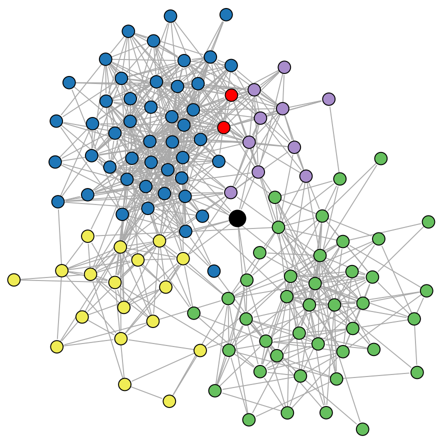


Fig. 1. Four communities of a single user (the node in black) of Facebook, as determined by MOSES. Two other users (red) have been assigned to both the blue and purple communities. A typical user, like many of those in this diagram, will be a member of seven communities, which we have not attempted to visualize here.

A. Structure of this paper

We first briefly consider related work in the field on overlapping community finding. Then, in sections III and IV we introduce our objective function and describe the algorithm. Next, there is an analysis of the algorithm with two types of synthetic benchmark data, the LFR benchmarks proposed in [2] and a second model that allows for greater variance in the community overlap structure. We conclude with an analysis of a Facebook friendship network from five US universities.

a) Notation: In this paper we consider the community assignment problem on an unweighted, undirected graph G, with vertices V and edges E and no self-loops. Boldcase letters, such as $\mathbf{Z}$, $\mathbf{z}$ denote column vectors with the uppercase $\mathbf{Z}$ referring to a random vector variable and the lowercase $\mathbf{z}$ referring to a particular realization of $\mathbf{Z}$. We use capital Roman letters, such as Z to denote random matrices and their realizations. The components of random matrices are denoted by the corresponding uppercase letter e.g. Z_{ij}, while the components of matrix realizations are denoted by the corresponding lowercase letter e.g. z_{ij}.

II. Related Work

While there is no single generally accepted definition of a community within a social network, most definitions try to encapsulate the concept as a sub-graph that has few external connections to nodes outside the sub-graph, relative to its number of internal connections. We find the following distinctions useful in characterizing commonly-used community definitions:

1) *Structural communities*: A deterministic set of properties or constraints that a sub-graph must satisfy in order to be considered a community is given and thus a decision can be made on whether any particular sub-graph is, or is not, a community, e.g. we may consider all maximal cliques to be communities. Thus finding such communities is a process of searching the graph for all sub-graph instances that satisfy the defining properties.
2) *Evaluated communities*: Every sub-graph is considered to be a community to a certain extent, given by the value of a *community fitness function*. The fitness function may be local or global in nature and sometimes is associated with the entire community decomposition rather than with each single community.
3) *Algorithmic communities*: As pointed out in [3], often there is no explicit definition of a community, other than as the sub-graphs that result from some community extraction algorithm. A good example of this is the edge-betweenness algorithm of [4].

The last decade has seen a lot of publications on the topic of community detection in networks. For a good review, see [3]. Much work has concentrated on *modularity* maximization algorithms, that produce partitions of graphs in which each node is assigned to a single community. Modularity defines evaluated communities, where the community fitness is related to its number of internal edges relative to its expected number in a particular 'null model'. While modularity maximization results in a decomposition of the entire network into a partition of communities, in fact, a more general view of community-finding is from the node perspective as *community assignment*, i.e. the task is to assign each node in the graph to the communities (if any) it belongs to and we may describe algorithms for community-finding as *community assignment algorithms* (CAAs).

A number of CAAs that allow overlapping communities have emerged since 2005 [1, 5–13]. For example GCE [1], LFM [11] and Iterative Scan (IS) [12] find evaluated communities. Each uses various local iterative methods to expand (or shrink) proposed communities such that some function of the density of the communities is maximized, but the decision on whether a proposed community is accepted or not depends on somewhat arbitrary criteria. At the other end of the spectrum, the Clique Percolation Method (CPM) of [5] has proved very influential and is essentially a structural community-finding algorithm, where communities are defined as sub-graphs consisting of a set of connected k-cliques.

With the recent release of the LFR synthetic benchmark graphs [14], it has become possible to more thoroughly explore the performance of these different approaches. Studies on this benchmark data have illustrated that performance of the algorithms generally degrades as nodes are shared between larger numbers of communities [1]. It is our contention that real-world social communities do in fact contain rich overlapping structures like those of the LFR benchmarks and that it is necessary to develop CAAs that perform well when on average each node is assigned to multiple communities. There is need for further extensions of these synthetic benchmarks as, for example, the current LFR model places each overlapping node into exactly the same number of communities.

Model-based CAAs have the advantage of being based on a model which can explain the rationale of the communities found, thus avoiding the often arbitrary criteria which are used in many overlapping CAAs. We develop a scalable, model-based CAA that performs well on highly overlapping community structures. In the next section, we review the model-based network algorithms that are most relevant to our approach.

A. Model-Based Community-Finding

In model-based community-finding, the graph G is considered to be a realization of a statistical model. Assuming unweighted, undirected graphs, with no self-loops, the graph edges are represented by a random symmetric adjacency matrix X such that $x_{ij} = x_{ji} = 1$ if an edge connecting nodes i and j exists and zero otherwise. Statistical network models are reviewed in [15]. Of particular interest in the context of the work presented here is the *stochastic blockmodel* introduced in [16] which is also referred to as the *Erdos-Renyi Mixture Model for Graphs* (ERMG) in [17].

The ERMG assumes a partitioning of the graph into communities, so that community assignments can be described by the vector $\mathbf{Z} = (Z_1, \ldots, Z_N)^T$, where $Z_i = q$ if node i is assigned to community q. The graph edges are assumed independent given the node assignments $\mathbf{Z}$, and drawn from a Bernoulli distribution with connection probability dependent on the community assignments of the end-points:

$$P(X_{ij} = 1|\mathbf{Z} = \mathbf{z}) = \pi(z_i, z_j) \equiv \pi_{z_i z_j} \, .$$

Assuming that $\pi_{qr} = \pi_{rq}$, this leads to the conditional probability for X given $\mathbf{Z}$,

$$P(\mathrm{X}|\mathbf{Z}, \boldsymbol{\theta}, \Pi) = \prod_{i=1}^{N} \prod_{j=i+1}^{N} \pi_{z_i z_j}^{x_{ij}} (1 - \pi_{z_i z_j})^{(1-x_{ij})} \, , \qquad (1)$$

where Π is the $Q \times Q$ matrix of inter-community connection probabilities $\{\pi_{qr}\}$. Ultimately, the goal is to predict the unobserved community assignments $\mathbf{z}$. As discussed in [16] parameter estimation is difficult as the observed likelihood:

$$P(\mathrm{X}|\boldsymbol{\theta}, \Pi) = \sum_{\mathbf{z} \in \{1, \ldots, Q\}^N} P(\mathrm{X}, \mathbf{Z} = \mathbf{z}|\Pi)$$

cannot be simplified and the Expectation Maximization (EM) algorithm requires the posterior $P(\mathbf{Z}|\mathrm{X})$ which is also intractable. In [17] a variational approach is taken to parameter estimation while a classification EM approach is taken in [18], in which the complete likelihood is maximized with respect to $(\mathbf{Z}, \Pi)$. An online estimation approach is used where the parameters are incrementally updated using the current value of the parameters and new observations. The algorithm is essentially a greedy maximization strategy. The ERMG assumes a fixed number of communities. To decide between different values of Q, both [17] and [18] use an Integrated Classification Likelihood (ICL) criterion to decide between competing models.

1) Overlapping Stochastic Block Modeling: In [19], the standard ERMG is expanded to allow for overlapping communities and the new model is named the Overlapping Stochastic Blockmodel (OSBM). Now the community assignments of a node i may be described by a vector $\mathbf{Z}_i = (Z_{i1}, \ldots, Z_{iQ})^T$, such that

$$Z_{iq} = \begin{cases} 1 & \text{node } i \text{ in community } q \\ 0 & \text{otherwise.} \end{cases}$$

The full latent structure may be described by the $N \times Q$ matrix Z, with i^{th} column $\mathbf{Z}_i$. As with the ERMG, it is assumed that all the edges are independent, given Z and drawn from a Bernoulli distribution, with the probability $\pi(\mathbf{z}_i, \mathbf{z}_j)$ that an edge exists dependent on the (vector) community assignments $\mathbf{z}_i$ and $\mathbf{z}_j$ of its end-points, leading to a joint distribution of the same form (1), with $\pi_{z_i z_j}$ replaced by $\pi(\mathbf{z}_i, \mathbf{z}_j)$.

The authors assume that the connection probabilities, $\pi(\mathbf{z}_i, \mathbf{z}_j)$ can be written as sigmoid functions of a quadratic form $\mathbf{z}_i \mathrm{A} \mathbf{z}_j$ for a parameter matrix A. Moreover, they choose a prior distribution on Z of the form:

$$P(\mathrm{Z}|\boldsymbol{\alpha}) = \prod_{i=1}^{N} \prod_{q=1}^{Q} \alpha_q^{z_{iq}} (1 - \alpha_q)^{1 - z_{iq}} , \tag{2}$$

for parameters $\alpha_q \in [0, 1]$. The parameters of the model are estimated using a variational strategy similar to that used in [17].

While the models of [16] and [19] allow for a large number of parameters, in practice, when evaluating on real datasets, the parameter space is usually restricted to a much smaller number. In [19] for instance, this is done by considering restricted forms of the matrix A, with only two free parameters. The community-finding algorithm of [19] is shown to out-perform the CPM algorithm of [5] on synthetic data. As the model is fitted using a variational approach, it will not scale to large graphs.

III. The MOSES Model

The model that drives MOSES is essentially an OSBM but with some important differences to that presented in [19]. In particular:

1) The connection probabilities $\pi(\mathbf{z}_i, \mathbf{z}_j)$ take a different form to those used in [19];
2) The prior takes into account that community assignments that differ only by a relabeling of the communities are equivalent;
3) A distribution is placed on the number of communities Q, allowing Q to be integrated from the prior.

We elaborate on these differences in the following:

A. Connection Probabilities

Let $\pi_{qr} \in [0, 1]$ represent the probability that a node in community q connects to a node in community r and let p_o denote a general underlying probability that nodes connect, independent of community structure. Assume that these probabilities are all mutually independent. Hence, the probability that an edge does *not* exist is given by:

$$\begin{aligned} P(X_{ij} = 0|\mathrm{Z}, \Pi) &= 1 - \pi(\mathbf{z}_i, \mathbf{z}_j) \\ &= (1 - p_o) \prod_{q=1}^{Q} \prod_{r=q}^{Q} (1 - \pi_{qr})^{z_{iq} z_{jr}} . \end{aligned} \tag{3}$$

In practice, we use $\Pi = \mathrm{diag}(p_{in})$. Thus, there is a single connection probability p_{in} of within-community connections and there is no tendency for inter-community connections, other than the general tendency of nodes to connect represented by p_o. With this simplification, (3) becomes,

$$P(X_{ij} = 0|\mathrm{Z}, p_{in}, p_o) = (1 - p_o)(1 - p_{in})^{s_Z(i,j)} .$$

where $s_Z(i, j)$ is a count of the number of communities assigned to both node i and node j in Z.

B. Prior on Z

Assuming a uniform distribution on the parameters $\{\alpha_1, \ldots \alpha_Q\}$ in (2) and integrating over them, we obtain a prior of the form

$$P(\mathrm{Z}|Q) = \frac{1}{(N+1)^Q} \left(\prod_{q=1}^{Q} \frac{1}{\binom{N}{n_q}} \right) ,$$

where n_q is the number of nodes assigned to community q. Furthermore, while there are 2^{NQ} possible values for Z, any permutation of the columns of Z results in the same community assignment, with just a different labeling on the communities. The 2^{NQ} possible matrices can be partitioned into equivalence classes of matrices that differ only in a permutation of their columns. Let $c_z(Q)$ be the size of the equivalence class that Z belongs to. Using $\mathcal{C}_\mathrm{Z}$ to denote the community assignment corresponding to the $c_z(Q)$ matrices in this equivalence class, we note that $P(\mathcal{C}_\mathrm{Z}|Q) = c_z(Q)P(\mathrm{Z}|Q)$. Let Q_z be the number of non-empty communities observed in Z. If the actual number of communities is $Q_z + k$, then Z should contain k columns of all zeros. It follows that

$$c_\mathrm{Z}(Q_z + k) = \binom{Q_z + k}{k} c_\mathrm{Z}(Q_z) , \tag{4}$$

since the k communities with no nodes assigned to them must be allocated k labels out of the $Q_z + k$ possible community labels. Furthermore,

$$P(\mathrm{Z}|Q_z + k) = \frac{1}{(N+1)^k} P(\mathrm{Z}|Q_z) . \tag{5}$$

Now, choosing a *Poisson* distribution for Q with mean value m, using (4)and (5), and summing over Q to obtain a prior on $\mathcal{C}_{\mathrm{Z}}$ that is independent of Q, we get

$$\begin{aligned} P(\mathcal{C}_{\mathrm{Z}}) &= \sum_{k=0}^{\infty} P(\mathcal{C}_{\mathrm{Z}}|Q = Q_z + k) \frac{e^{-m} m^{Q_z+k}}{(Q_z+k)!} \qquad (6) \\ &= \frac{c_z(Q_z)}{(N+1)^{Q_z}} \left(\prod_{q=1}^{Q_z} \frac{1}{\binom{N}{n_q}} \right) \frac{e^{-(\frac{N}{N+1})m} m^{Q_z}}{Q_z!} \end{aligned}$$

Finally, if there are p unique non-zero columns in Z, which occur with multiplicity $o_1, \ldots, o_p$, such that $Q_z = \sum_{k=1}^{p} o_k$, we note that $c_z(Q_z)$ is the multinomial coefficient:

$$c_z(Q_z) = \frac{Q_z!}{o_1! \ldots o_p!}$$

With (6) and (1), letting $\mathcal{L}(.) = \log P(.)$, it is now possible to write down the complete data log likelihood as

$$F(\mathcal{C}_{\mathrm{Z}}, p_{in}, p_o) = \mathcal{L}(\mathrm{X}|\mathrm{Z}, p_o, p_{in}) + \mathcal{L}(\mathcal{C}_{\mathrm{Z}}) \,. \qquad (7)$$

As methods such as [17] that attempt to find the maximum likelihood estimators from the observed likelihood $\mathcal{L}(\mathrm{X})$ are too computationally expensive for large-scale networks, we follow an approach similar to [18] and seek the $(\mathrm{Z}, p_{in}, p_o)$ that maximizes (7). Maximization of the complete data likelihood has been shown to result in good clusterings in practice in the context of Gaussian mixture models. In the remainder of the paper, we will simply write $F(\mathcal{C}_{\mathrm{Z}})$, rather than $F(\mathcal{C}_{\mathrm{Z}}, p_{in}, p_o)$, to emphasize our primary objective of finding an optimal $\mathcal{C}_{\mathrm{Z}}$.

IV. The MOSES Maximization Algorithm

MOSES, similarly to algorithms based on modularity, is driven by a global objective function, $F(\mathcal{C}_{\mathrm{Z}})$. Except in the smallest of networks, it is not feasible to exhaustively search every possible community assignment, calculating $F(\mathcal{C}_{\mathrm{Z}})$ for each, and then remembering which got the best score. In order to handle graphs with millions of edges, we use a greedy maximization strategy in which communities are created and deleted, and nodes are added or removed from communities, in a manner that leads to an increase in the objective function.

The change in the objective when an entire community is added or removed can be decomposed into a set of single node updates. A single node update, adding it to, or removing it from, a community, changes z_{iq} to $z'_{iq} = 1 - z_{iq}$. In order to avoid considering a node being connected to itself in the following expression, which is not allowed in this model, we focus on the addition of a community in this discussion. For convenience we define $\psi_{in} = 1 - p_{in}$ and $\psi_o = 1 - p_o$.

The conditional likelihood of X changes where node i is being added to community q, where j iterates over the set of nodes already within q,

$$\begin{aligned} \Delta\mathcal{L}(\mathrm{X}|\mathcal{C}_{\mathrm{Z}}) = n_q \log \psi_{in} - \sum_{z_{jq}=1} x_{ij} \log \psi_{in} \\ + \sum_{z_{jq}=1} x_{ij} \log \left(\frac{1 - \psi_o {\psi_{in}}^{s'_Z(i,j)}}{1 - \psi_o {\psi_{in}}^{s_Z(i,j)}} \right) , \end{aligned}$$

where $s'_Z(i,j) = (-1)^{z_{iq}} + s_Z(i,j)$ is the number of common communities between i and j after the node update has taken place. We note that we need the values of s_Z only for those pairs of nodes that are connected, the edges.

The change in likelihood of $\mathcal{C}_{\mathrm{Z}}$, $\Delta\mathcal{L}(\mathcal{C}_{\mathrm{Z}})$, is more complicated as it depends on whether the node update results in a change to Q_z or not. We estimate m, the mean value of Q to be $\hat{m} = Q_z$, which allows us to simplify (6) when considering small changes to Q_z,

$$P(\mathcal{C}_{\mathrm{Z}}) \propto \frac{c_z(Q_z)}{(N+1)^{Q_z}} \left(\prod_{q=1}^{Q_z} \frac{1}{\binom{N}{n_q}} \right)$$

Moreover, changes to $c_z(Q_z)$ depend on whether the node update results in a change to the number or multiplicity of unique columns in Z. In the MOSES algorithm as currently implemented, we assume that all the communities we have found are unique, estimating $c_z = Q_z!$. This introduces an overestimate of the multinomial c_z, and we would expect that this would introduce a bias towards finding duplicate communities. However, we have not yet observed a duplicate community in the output of the algorithm.

We use a combination of heuristics in an attempt to find good communities. These are edge-expansion, community-deletion and single-node fine-tuning. In the following, it is more useful to think of a community assignment $\mathcal{C}_{\mathrm{Z}}$ as a set of communities, with each community consisting of a set of nodes. We will use $\mathcal{C}_{\mathrm{Z}} \cup C$ for $C \subseteq V$ to denote the addition of a new community to $\mathcal{C}_{\mathrm{Z}}$.

a) Edge expansion: In the initial phase of the algorithm, edges are selected at random from the graph and a community is expanded around each selected edge in turn. Initially the community consists of two nodes $C = \{v, w\}$. New nodes are added to C from its *frontier* i.e. the set of nodes not in C but directly connected to nodes in C. Nodes are added in a greedy manner, selecting the node v^* in the frontier that maximizes $F(\mathcal{C}_{\mathrm{Z}} \cup \{C \cup v\})$. Expansion continues while the objective is the highest found so far. When a community is very small, its contribution to the objective may be negative even if it is a clique. Hence, we use a small lookahead, whereby expansion of a community will continue, even if it would decrease the objective, unless l consecutive expansions fail to raise the objective. In practice, we use $l = 2$ and have found that large values of l slow down the algorithm, without any significant improvement to the quality of the results.

Edges are chosen randomly with replacement to be subject to expansion. Note that each subsequent time an edge is selected, it may expand into a different community, as, with each addition of a new community, the overlap counts $s_Z(i,j)$ change. For the first community expansion v^* is simply the node with most connections to C. Then, as more expansions are performed, and more and more edges are 'claimed' by found communities, and $s_Z(i,j)$ increases, the expansion will favour edges with lower $s_Z(i,j)$. Informally, we can say that $F(\mathcal{C}_{\mathrm{Z}})$ favours finding communities of nodes which are densely connected by edges which are not edges contained within many other communities.

b) Community Deletion: Periodically all the communities are scanned to see if the removal of an *entire* community will result in a positive change in the objective. This check occurs after each 10% of the edges have been expanded, so will happen 10 times. The output of the algorithm will be the assignments after the last community deletion phase.

$$F(\mathcal{C}_Z \setminus \{C\}) > F(\mathcal{C}_Z)$$

c) Single-Node Fine Tuning: The fine tuning phase takes place at the end of the edge expansion phase. It is inspired by the method of Blondel et al. [20]. In this phase, each node is examined in turn by removing it from all the communities it is assigned to and then considering adding it to the communities to which it is connected by an edge. As always, the decision to insert a node into a neighbouring community depends on whether it results in a positive change to $F(\mathcal{C}_Z)$.

d) Estimating p_{in} and p_o: It can be shown that, for a given Z and X, and as a function of p_{in} and p_o, the value of $F(\mathcal{C}_Z, p_{in}, p_o)$ depends on simple summary quantities such as the frequency of various values of $s_Z(i,j)$ across the edges. This allows us to efficiently select the values of p_{in} and p_o which maximize $F(\mathcal{C}_Z)$.

V. Evaluation

A. Evaluation on benchmark data with variable overlap

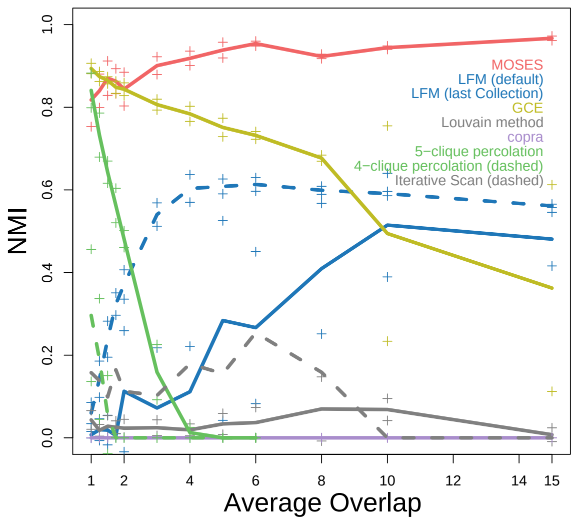


Fig. 2. NMI of various algorithms as average overlap increases. Mean +/- standard deviation of twenty realizations of the graph. IS was run just once, due to time constraints.

To evaluate the accuracy of MOSES and other algorithms, we created a set of simple benchmark graphs with increasing levels of overlap. To generate the graphs, a number of communities of size 20 are initially assigned to a set of 2,000 nodes. For each community, 20 nodes are selected at random, without regard to whether these nodes have been assigned to other communities or not. Hence the community *overlap* of the nodes is not fixed across the network. A note on terminology: we use *overlap*, and *highly overlapping*, to mean nodes that are members of many communities, and not necessarily to mean pairs of communities that share many nodes.

These communities are referred to as the *ground truth communities*. The graph is then generated by joining an edge between every pair of nodes that share a community. Finally, every pair of nodes is joined with probability 0.005 to add a number of non-community edges. This process results in a graph with a large number of 20-cliques. We further confirmed that, in our evaluation, all graphs generated are connected, even those with the smallest number of communities. We use an extension of normalized mutual information (NMI) to calculate how similar the ground truth communities are to the communities found by the various algorithms.[2] Results on these synthetic graphs are shown in fig. 2. We plot the accuracy, as measured by NMI, of a variety of overlapping CAAs. On the horizontal axis, we plot the average *overlap* within the benchmark graph. For example, where the average overlap is 1.0, this means there were 100 communities, each of 20 nodes, placed in the 2,000-node graph.

The algorithms used are LFM [11], COPRA[9], , Iterative Scan (IS) [12] , clique percolation and GCE [1]. We include the Louvain method [20] as an example of a popular partitioning algorithm. We have used implementations supplied by the authors, except for clique percolation where we used our own implementation as existing implementations, [5, 21], were slow on many of the evaluated datasets. The LFM software creates many complete collections from a graph, each of which is a complete community assignment. As recommended by the authors, we select the first such community assignment for use in this comparison. However, we have noticed that the results obtained from LFM when selecting the last collection, instead of the first, can be better. For completeness, we have included this in our comparison. In fig. 3, we plot the average overlap found by the various algorithms. Only MOSES is able to obtain good estimates of the average overlap, up to an average overlap of 15 communities-per-node.

[2]For creating the LFR graphs with fixed overlap-per-node and measuring overlapping NMI, we use the implementations provided by the authors, both of which are freely available at http://sites.google.com/site/andrealancichinetti/software. For the specification of overlapping NMI, see the appendix of Lancichinetti et al. [11].

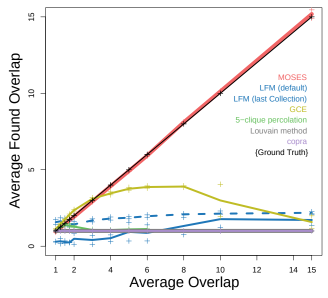


Fig. 3. Estimated overlap of various algorithms as average overlap increases.

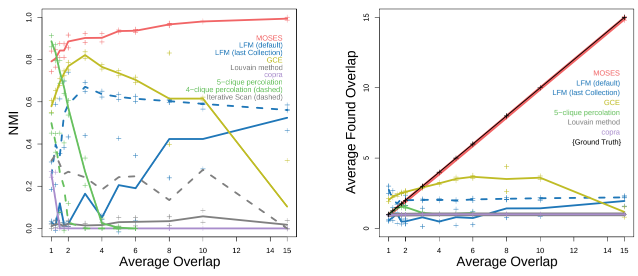


Fig. 4. Graphs with lower levels of "background" edges.

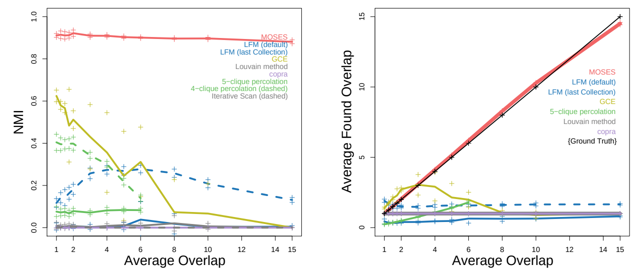
In fig. 4, we consider graphs with a lower probability, 0.001, for the probability of a non-community edge between two nodes. This will assign approximately two non-community edges, on average, to each node. This improves the performance of many algorithms as the number of noisy edges has significantly decreased. We should note that these graphs are not necessarily connected, and some algorithms operate only on the largest connected component. For each of these sparser graphs, at least 90% of the nodes are in the largest connected component.

In the benchmarks described so far, each community was a clique, rendering it simple for MOSES to detect. To investigate further, we generated a series of benchmarks where the edges inside communities are connected with a lower probability. These are presented in fig. 5 where we see that MOSES's performance drops as p_{in} drops below 0.4. Even at p_{in}=0.3 however, when all other algorithms have NMI under 30%, the performance of MOSES is excellent as long as the average overlap in the ground truth is under five communities-per-node. We have not thoroughly investigated the parameter of the k-clique percolation algorithm, but can confirm that 4-cliques give the best NMI in the particular dataset used in fig. 5(a). On much denser graphs, with little noise, clique percolation's performance would improve.

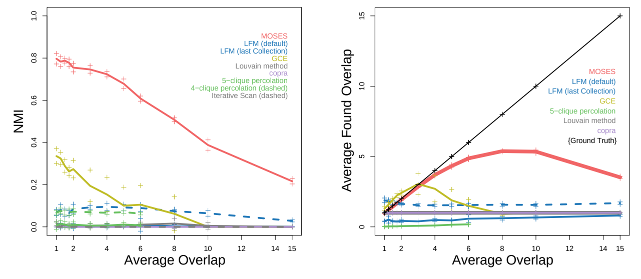
B. Evaluation on LFR Graphs

The LFR benchmark generation software can be used to generate more interesting datasets than those just analyzed. One drawback of the LFR graphs is that all the overlapping nodes must be assigned to the same number of communities. We used the LFR software to generate graphs not unlike those just analyzed in the last section. The number of nodes is again 2,000. The community sizes range uniformly from 15 to 60. The mixing parameter,μ, is 0.2 meaning that 80% of the edges are between nodes that share a community.

We varied the overlap to range from one community per node to ten communities per node. Then the degree of all the nodes was fixed to be 15 times the overlap. We present these results in fig. 6(a), where the horizontal axis is logarithmic. LFR can create graphs where only a portion of the nodes are assigned to more than one community, we use this feature to investigate graphs with on average 1.2, 1.4, 1.6, 1.8

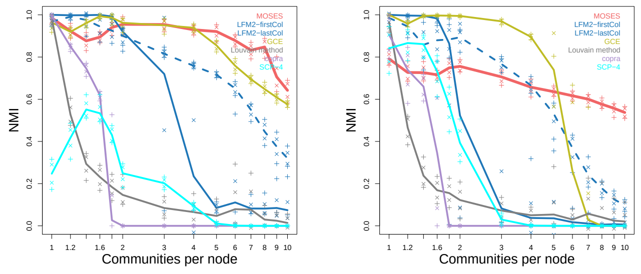


(a) p_{in} = 0.4 (b) p_{in} = 0.4



(c) p_{in} = 0.3 (d) p_{in} = 0.3

Fig. 5. NMI, and average found overlap, for low p_{in}. p_o = 0.005.



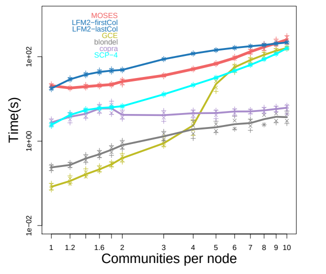
(a) Fixed degree. $\bar{k} = 15 \times$ overlap (b) Maximum degree is triple average

Fig. 6. NMI scores as the amount of overlap increases in the LFR fixed-overlap graphs. We mark the mean +/- standard deviation, along with lines through the mean, over twenty realizations of the benchmark.

communities-per-node. In these graphs, both the degree and the overlap is fixed, making the structure relatively simple. It is not surprising that many algorithms, such as LFM and the partitioning algorithm by Blondel et al. [20], perform well when there is no overlap and each node is assigned to exactly one community.

In the previous section we saw that a partitioning algorithm, such as [20], can fail on graphs with low levels of overlap. This demonstrates that, even in empirical graphs where overlapping communities are not expected to be major feature, it may not be wise to use a partitioning algorithm. Partitioning algorithms succeed only where each node is known to be in *exactly* one community. This is an unrealistic assumption in many empirical datasets.

The LFR software can generate networks with a power law degree sequence. In fig. 6(b) we analyzed that same datasets as in fig. 6(a) but where the maximum degree was set to be three times the average degree. The slope parameters to the


(a) Fixed degree. $\bar{k} = 15 \times$ overlap

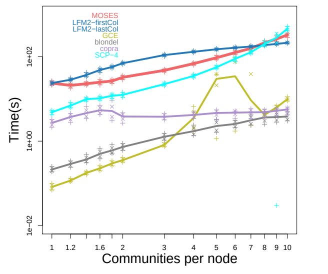

(b) Maximum degree is triple average

Fig. 7. Run time, in seconds, as overlap increases in the LFR benchmarks.

degree distribution is another parameter to the LFR software and we set it to 2.0. In these datasets, when the overlap is low, MOSES does not perform as well as GCE, LFM or clique percolation. On the other hand, MOSES is the only algorithm capable of detecting significant structure when the overlap approaches 10 communities per node. The NMI of the community assignments found by MOSES is consistently above 60% whereas the other algorithms' scores are well below 40% when there are more than six communities per node. The MOSES model does not explicitly model degree distribution, and this may explain its failure to get the highest NMI scores in fig. 6(b). This may be an area for future development of the model. The superior community model of MOSES enables it to detect some structure in the graphs with heaviest overlap.

C. Scalability

In fig. 7 we investigated the run time of these algorithms. The graphs are the same as in fig. 6, but instead we plot the logarithm of the running time on the y-axis. GCE is the fastest of all the algorithms on the less overlapping data. While there are many algorithms faster than MOSES and LFM, the only one of those algorithms capable of getting reasonable NMI scores is GCE. The high quality NMI scores of MOSES do not carry a penalty in performance. MOSES is as fast as many scalable algorithms on overlapping data, and gets the highest quality results on the very highly overlapping data.

D. Evaluation on a real-world social network.

Traud et al. [22] gathered data on Facebook users and friendships in five US universities. There are no *ground truth* communities as such in this dataset. We ran MOSES on the five datasets.

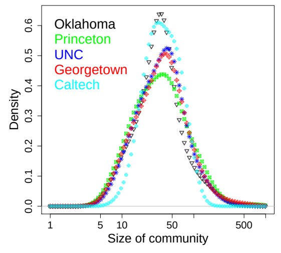
The degree distributions of all five appears to be very approximately log-normal, as can be seen in the logarithmic histograms of fig. 8(c). The distribution does not fit the power law distributions often assumed as an approximation for the degree distribution of empirical graphs. The relative narrowness of this degree distribution may improve the results of MOSES as it is a more reasonable fit for the MOSES model than a strict power law distribution would be. The average degree ranges from 43.3 to 102.4. Assuming that communities are not very large, and that most edges in these networks are community edges, it must be the case that the average node is in many communities.

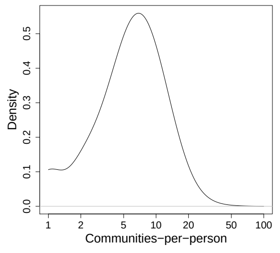
A summary of the results is presented in table I. It suggests that a Facebook user is, on average, a member of seven communities. In an analysis of one of their own Facebook ego-networks, Salter-Townshend and Murphy [23] found it divided into six groups. MOSES assigns nodes each to a different number of communities, and to communities of varying size. In fig. 1, we present the communities of a student at Georgetown. MOSES assigns this student to four communities, and we visualized the subgraph based on all the nodes in those four communities.

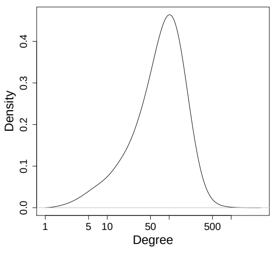
In fig. 8(d) we see a scatter plot of the degree of a node versus the number of communities it is in. It shows that the nodes with higher degree are assigned to more communities. We see there is significant variation in the sizes of the communities found in fig. 8(a). It may be that many of these communities are made up of sets of sub-communities that loosely interact with each other.

TABLE I
SUMMARY OF TRAUD ET AL. [22]'S FIVE UNIVERSITY FACEBOOK DATASETS, AND OF MOSES'S OUTPUT.

	Caltech	**Princeton**	**Georgetown**	**UNC**	**Oklahoma**
Edges	16656	293320	425638	766800	892528
Nodes	769	6596	9414	18163	17425
Average Degree	43.3	88.9	90.4	84.4	102.4
Communities found	62	832	1284	2725	3073
Average Overlap	3.29	6.28	6.67	6.96	7.46


(a) Community size


(b) Communities per node


(c) Degree distribution

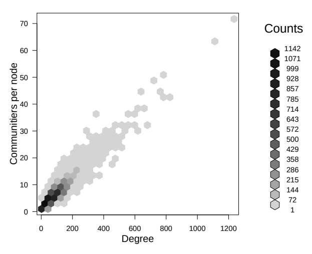

(d) Degree vs. overlap

Fig. 8. Sizes of the communities found, and degree distribution for Georgetown, in (logarithmic) density plots. And a density plot showing high-degree nodes are assigned to more communities by MOSES.

VI. Conclusions

We have demonstrated that it is possible to detect very highly overlapping community structure in large networks using MOSES. Existing algorithms find only relatively low levels of overlapping community structure. It is necessary to be able to detect highly overlapping structure, if only to rule it out for a given observed network. For instance, our analysis on Facebook data has shown that a typical Facebook user can be a member of seven communities. This demonstrates the need for further research into such community structure. Existing algorithms work best where each node is in the same number of communities. But this is not a realistic assumption for social networks and we have demonstrated that MOSES can accurately detect communities in networks where typical nodes are in many communities, and where there is variance in the number of communities a node is in.

Acknowledgments

This research was supported by Science Foundation Ireland (SFI) Grant No. 08/SRC/I1407.

References

[1] C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. *KDD 2010*, 2010. URL http://arxiv.org/abs/1002.1827.

[2] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(4), 2008. URL http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal\&id=PLEEE8000078000004046110000001\&idtype=cvips\&gifs=yes.

[3] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010. ISSN 0370-1573. doi: DOI:10.1016/j.physrep.2009.11.002.

[4] MEJ Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys Rev E*, 69: 026113, 2004.

[5] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[6] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72(2):26132, 2005.

[7] S. Gregory. An algorithm to find overlapping community structure in networks. *Lecture Notes in Computer Science*, 4702:91, 2007.

[8] S. Gregory. Finding Overlapping Communities Using Disjoint Community Detection Algorithms. In *Complex Networks: Results of the 1st International Workshop on Complex Networks (CompleNet 2009)*, page 47. Springer, 2009.

[9] S. Gregory. Finding overlapping communities in networks by label propagation. *Arxiv preprint arXiv:0910.5516*, 2009.

[10] Nina Mishral, Robert Schreiber, Isabelle Stanton, and Robert E. Tarjan. Clustering social networks. *Lecture notes in computer science*, 4863:56, 2007.

[11] Andrea Lancichinetti, Santo Fortunato, and Janos Kertesz. Detecting the overlapping and hierarchical community structure of complex networks. *New Journal of Physics 11, 033015 (2009)*, Mar. 2009. doi: http://dx.doi.org/10%2E1088/1367-2630/11/3/033015.

[12] J. Baumes, M. Goldberg, M. Krishnamoorthy, M. Magdon-Ismail, and N. Preston. Finding communities by clustering a graph into overlapping subgraphs. In *International Conference on Applied Computing (IADIS 2005)*, 2005.

[13] H. Shen, X. Cheng, K. Cai, and M.B. Hu. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 388(8):1706–1712, 2009.

[14] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):16118, 2009.

[15] Anna Goldenberg, Alice X. Zhang, Stephen E.Fienberg, and Edoardo M. Airoldi. A survey of statistical network models. December 2009.

[16] Krzysztof Nowicki and Tom A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, September 2001.

[17] J.J. Daudin, F. Picard, and S. Robin. A mixture model for random graphs. *Statistical Computing*, 18:173–183, 2008.

[18] Hugo Zanghi, Christophe Ambroise, and Vincent Miele. Fast online graph clustering via erdos-renyi mixture, 2007.

[19] Pierre Latouche, Etienne Birmelé, and Christophe Ambroise. Overlapping stochastic block models. Oct 2009. URL http://arxiv.org/abs/0910.2098.

[20] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008.

[21] Jussi M. Kumpula, Mikko Kivela, Kimmo Kaski, and Jari Saramaki. A sequential algorithm for fast clique percolation. Jul 2008. doi: 10.1103/PhysRevE.78.026109.

[22] A.L. Traud, E.D. Kelsic, P.J. Mucha, and M.A. Porter. Community Structure In Online Collegiate Social Networks. *organization*, 88:92.

[23] M. Salter-Townshend and T. B. Murphy. Variational Bayesian Inference for the Latent Position Cluster Model. 2009. URL http://snap.stanford.edu/nipsgraphs2009/papers/townshend-paper.pdf.