Probabilistic Community Detection in Networks

James P. Ferry  
Metron, Inc.  
Reston, VA, U.S.A.  
Email: ferry@metsci.com

J. Oren Bumgarner  
Metron, Inc.  
Reston, VA, U.S.A.  
Email: bumgarner@metsci.com

Stephen T. Ahearn  
Metron, Inc.  
Reston, VA, U.S.A.  
Email: ahearn@metsci.com

Abstract—Standard community detection methods for networks provide “hard calls”: a specification of which nodes belong to which groups with no indication of the confidence of these assessments. Here, a simple formula is presented which provides the probability of a node belonging to a group. An efficient method is then presented for determining the probability of any pair of nodes being in the same group, without reference to any one, fixed group structure. These pairwise co-membership probabilities can be used directly to enable certain analyses of group structure, or can be converted into a distance metric which enables a different class of analyses. As an example, we demonstrate how this co-membership distance matrix can be used to find a community structure that is both overlapping and hierarchical using a topological technique inspired by Morse theory to partially cluster with respect to the distance metric.

Keywords: Networks, community detection, Bayesian probability, clustering

I. Introduction

In its classical form, data fusion begins with sensors collecting data from the physical world, which is then processed using models of physical law. Often this involves an assumption that there are a small number of objects whose properties and motion are conditionally independent given their environment, aside from a limited number of interactions (e.g., collisions). In the last twenty years there has been a profusion of data arising from human and computer activity (e.g., email, phone calls, and banking and shipping transactions). There can be many objects in such data about which little is known except for their links. One can assemble these links into a network, but this can hardly be called fusion. How, then, can a network of data be fused into useful information?

One answer is to appeal to the large body of network algorithms to provide insight into the data. One could compute various network metrics [1] to assess the network. This would be analogous to processing a sequence of contacts by running various computational geometry algorithms on it: one might compute the area of a convex hull over time, or look for patterns of contacts that lie along a straight line. However, although some useful techniques might be developed in this way, data fusion algorithms often rely on the much more powerful framework of maintaining a probability distribution over some state space of interest. There are many benefits to this approach [2]. Probability distributions retain information that is severely truncated by providing only a single, hard-call solution. Accurate tracking methods, such as the Kalman filter, depend on such probabilistic information in an essential way.

Probabilities provide a universal paradigm for representing uncertainty, which is necessary when interfacing with other sources of information—fusing hard calls tagged with idiosyncratic assessments of confidence (or no such assessments at all) is problematic. Finally, the probabilistic paradigm is well-suited for incorporating the constraints of physical laws.

Although one cannot hope for precise, physical laws, network science has found that networks which arise in practice have a number of universal characteristics. Real-world networks tend to be sparse, to have inhomogeneous degree distribution (which is sometimes well-fit by a power law), and to have a small diameter (a.k.a. the “small world” property or “six degrees of separation”) and high clustering coefficient [3]. Such properties can be categorized as microscopic (i.e., properties of individual nodes) or macroscopic (i.e., properties of the entire network). Recent work, however, has emphasized the importance of mesoscopic properties: those which pertain to organized structures of groups of nodes [4].

Large networks tend to exhibit community structure [5], which is usually characterized by the existence of subsets of nodes that have significantly higher-than-average edge density. In the past decade a variety of algorithms have been developed to discern this structure [6]. With a few exceptions (e.g., CFinder [7]), these methods are restricted to finding a partition \( \pi \) of the nodes into non-overlapping groups. The largest class of methods is based on maximizing a quality function of the partition \( \pi \) for a given graph \( G \)—typically some variant of modularity [8]—and recently a number of methods have been developed that produce multiple partitions, organized hierarchically [9], [10]. All of these methods produce hard calls, however. Even methods that employ probabilistic models [11], [12] do so only to produce a quality function to maximize, and do not provide probabilistic output. This is not surprising. The probability distribution over all possible partitions is vast (e.g., there are \( \Omega^{60} \) partitions of 60 nodes), and the highly correlated nature of network data does not admit a decomposition into tractable state spaces.

The purpose of this paper is to present probability calculations related to community detection, but on spaces of reasonable size. Section II defines a probability measure for a node belonging to a group \( C \) chosen from those specified by some partition \( \pi \) (which is presumably generated by some community detection algorithm). Section III discusses a method for directly estimating the probabilities \( p(v,w) \) that two nodes \( v \) and \( w \) are members of the same group, without...
any need for computing underlying partitions. An example of
the use of $p_{(v,v)}$ is given in Section IV, where a community
detection algorithm is given that allows for hierarchical and
overlapping groups.

II. MEMBERSHIP PROBABILITY

To compute meaningful probabilities, one begins with a
model which assigns probabilities to graph–partition pairs
$(G,\pi)$. This can be done by stipulating a prior probability $\Pr(\pi)$
for each partition $\pi$, then modeling the conditional probability $\Pr(G|\pi)$
for the graph $G$ given $\pi$. In this section and the next, we will use the following model. For a prior
distribution on $\pi$ we assume that (a) all partitions of a given
size $m = |\pi|$ are equally likely, and (b) the prior probability
distribution on $m$ is log-uniform: i.e., $\Pr(m) \propto 1/m$.
For $\Pr(G|\pi)$ we use a planted partition model [13], which
stipulates that the edges are instantiated independently with
probability $p_I$ between nodes in the same group $C \in \pi$, and
with probability $p_O$ between nodes in different groups. For
this model, the probability of $G$ is then

$$\Pr(G|\pi, p_I, p_O) = p_I^{e_I}(1-p_I)^{\tilde{e}_I} p_O^{\tilde{e}_O} (1-p_O)^{\tilde{e}_O},$$

(1)

where $e_I$ is the total number of edges within groups, $\tilde{e}_I$ is the
total number of node pairs within groups without edges,
and similarly for $e_O$ and $\tilde{e}_O$. This notation is summarized
in Table I. The first row is the sum of the other two, and
similarly for columns: e.g., $e(G)$ is the total number of edges
in $G$, $h_O(\pi)$ is the total number of node pairs between groups,
and $h = n(n-1)/2$ is the total number of node pairs.

<table>
<thead>
<tr>
<th></th>
<th>Within Groups</th>
<th>Between Groups</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edges</td>
<td>$e_I(G,\pi)$</td>
<td>$e_O(G,\pi)$</td>
<td>$e(G)$</td>
</tr>
<tr>
<td>Non-Edges</td>
<td>$\tilde{e}_I(G,\pi)$</td>
<td>$\tilde{e}_O(G,\pi)$</td>
<td>$\tilde{e}(G)$</td>
</tr>
<tr>
<td>Total</td>
<td>$h_I(\pi)$</td>
<td>$h_O(\pi)$</td>
<td>$h$</td>
</tr>
</tbody>
</table>

Table I: Edge- and group-count notation

We may integrate out the parameters $p_I$ and $p_O$ by assuming
each is uniformly distributed on $[0,1]$. (More generally, one
could prescribe beta distribution priors for $p_I$ and $p_O$, or one
could impose the restriction $p_O < p_I$, but it is not necessary
to introduce such complications here because large networks
contain enough data to overwhelm any reasonable prior.) The
integral of (1) over $0 \leq p_O \leq 1$ and $0 \leq p_I \leq 1$ is

$$\Pr(G|\pi) = B(e_I + 1, \tilde{e}_I + 1)B(e_O + 1, \tilde{e}_O + 1),$$

(2)

where $B(x, y)$ denotes the beta function. Using Bayesian
inversion with the log-uniform prior on $|\pi|$ yields

$$\Pr(\pi|G) \propto \frac{|\pi|B(e_I + 1, \tilde{e}_I + 1)B(e_O + 1, \tilde{e}_O + 1)}{S(n, |\pi|)},$$

(3)

where $S(n, m)$ denotes the Stirling number of the second kind.

For a given graph $G$ and partition $\pi$ of its nodes, we can use (3)
to assess how well each node $v$ fits into the group to
which it is assigned. To do this, we remove $v$, then consider
the probabilities of the partitions obtained by placing $v$ into
each group $C \in \pi$, or by making $\{v\}$ its own group. We let
$\pi_{v,C}$ denote the partition obtained by moving the node $v$ to
group $C$, with $\pi_{v,\emptyset}$ denoting the special case of putting $v$
into its own group. This includes the partition $\pi$ itself: if we
let $C_v$ denote the group in $\pi$ containing $v$, then $\pi = \pi_{v,C_v}$
(and when $|C_v| = 1$, then this also equals $\pi_{v,\emptyset}$). To use (3)
to compute these probabilities, we need these modified values of
the quantities in Table I:

$$e_{I,v \rightarrow C} = e_I + \delta_v(C), \quad e_{O,v \rightarrow C} = e_O - \delta_v(C),$$

(4)

$$\tilde{e}_{I,v \rightarrow C} = \tilde{e}_I + \delta_v(C), \quad \tilde{e}_{O,v \rightarrow C} = \tilde{e}_O - \delta_v(C).$$

(5)

The numbers of additional edges $\delta_v(C)$ and non-edges $\tilde{\delta}_v(C)$
within groups are given by

$$\delta_v(C) = d_v(C) - d_v(C_v),$$

(6)

$$\tilde{\delta}_v(C) = \eta_v(C) - \delta_v(C),$$

(7)

where $d_v(C)$ is the number of neighbors of $v$ in $C$, and

$$\eta_v(C) = |C| - |C_v| + 1$$

(8)

when $1_v \notin C$ and $0$ when $v \in C$.

Letting $m$ be the number of groups in $\pi$ when $v$ is removed
(i.e., $m = |\pi|$, minus one if $|C_v| = 1$, we may now compute the probabilities of the $m$ partitions $\pi_{v,C}$ (each of which has $m$
groups) and the probability of $\pi_{v,\emptyset}$ (which has $m + 1$).

These probabilities are proportional to

$$\beta_C = \alpha_C \frac{F(e_I, e_O, \tilde{e}_I, \tilde{e}_O, \delta_v(C))}{F(h_I + 1, h_O + 1, \eta_v(C))}.$$

(9)

Here $\alpha_C$ is the correction factor due to the additional group
in $\pi_{v,\emptyset}$. This involves a ratio of Stirling numbers, but using
the approximation $S(n, m) = m^n/n!$ we have

$$\alpha_C = \begin{cases} m (1 + 1/m)^{-n} & \text{for } C = \emptyset, \\ 1 & \text{otherwise.} \end{cases}$$

(10)

Finally, the function $F$ in (9) is obtained by rearranging the
factorials within the beta function in (3):

$$F(n^+, n^-, r) = \frac{(n^+ + r)! (n^- - r)!}{n^! (n^-)^!}.$$ 

(11)

The probabilities of the node $v$ belonging to a group $C$ is then
$\beta_C$ divided by the sum of all the $\beta_C$ (including $C = \emptyset$). It
is difficult to see how these probabilities behave by looking
at (9), so the following approximation is helpful:

$$\beta_C \approx \alpha_C \gamma(C) \tilde{\gamma}(C),$$

(12)

where the effects of a single additional adjacent or non-
adjacent node are given by

$$\gamma = \frac{e_I + \tilde{e}_I}{e_O + \tilde{e}_O}$$

(13)

and

$$\tilde{\gamma} = \frac{\tilde{e}_I}{\tilde{e}_O}$$

respectively.

We let $p_v(C)$ denote the probability of node $v$ being in
group $C$ of some fixed partition. These values are plotted in
Figure 1 for Zachary’s karate club data [14] using the ground-
truth partition $\pi$. This is a social network of 34 members
of a karate club at a university which split into two groups.
model-based probabilities offers important advantages over the alternative of defining metrics based on heuristics or physical analogies. The probabilistic approach is extensible: it states its assumptions clearly, and can be improved by substituting more realistic models. It is interoperable (its interface is well-defined: it produces probabilities of specific events) and modular (its internal methods can be improved without the need for extensive re-testing of algorithms that use it). Finally, it is internally consistent: indeed, provided one requires the use a single number to represent the degree of certainty of an event, along with certain qualitative defining properties (e.g., if the certainty of an event goes up, the certainty of its negation must go down), then the usual laws of probability are the only consistent paradigm for reasoning under uncertainty [2].

For example, the importance of exploiting negative information is well known in the fusion community. Determining how to incorporate it and to what degree can be a delicate problem—and a poorly designed network metric may fail to account for negative information altogether. It is inherent in the solutions produced by a model-based approach, however. This is somewhat difficult to see in (9), but the approximation (12) makes it clear: \( \gamma \) is the multiplier for each unit of positive evidence (i.e., an additional in-group adjacent node), and \( \tilde{\gamma} \) is the multiplier for each unit of negative evidence (i.e., an additional in-group non-adjacent node). In the karate club example, \( \gamma = 6.47 \) and \( \tilde{\gamma} = 0.783 \): as is typical, each unit of positive evidence has a stronger effect, but negative evidence is more abundant.

The values of \( p_v(C) \) can be used to categorize a node \( v \) in a given partition \( \pi \) as a “partisan” (high \( p_v(C_v) \)), or to assess whether \( v \) is drawn to a single group other than \( C_v \), or to a number of groups, or if it wants to be alone. The computation of \( p_v(C) \) is fast, and provides a useful augmentation to the output \( \pi \) of a traditional community detection algorithm.

### III. Co-Membership Probability

The analysis performed in Section II relates only to a single reference partition \( \pi \). In reality, a clear-cut, optimal partition may not exist due to hierarchical or overlapping structure or other ambiguities. The situation can be improved by offering, for example, multiple hierarchical solutions [16]. Here we take a different approach: we estimate the matrix of probabilities \( p^{(v,w)} \) that nodes \( v \) and \( w \) lie in the same group. Computing \( p^{(v,w)} \) is difficult. To do so exactly requires a summation over all partitions. Reichardt and Bornholdt estimated the \( p^{(v,w)} \) matrix by a Monte Carlo sampling of the partition space, but this is slow [17]. We may approximate \( p^{(v,w)} \) directly by considering only the information \( \epsilon_{vw}(G) \) in a network \( G \) that is most relevant to the issue of whether \( v \) and \( w \) are in the same group: the presence or absence of edges on all vertex pairs that intersect \( \{v,w\} \). The likelihood ratio of \( \epsilon_{vw}(G) \) under the hypotheses that \( v \) and \( w \) are in the same group and in different groups, respectively, may be decomposed into factors \( \Lambda_k \) (for the vertex pair \( \{v,w\} \) itself) and \( \Lambda \) (for the rest of \( \epsilon_{vw}(G) \)). This may then be used to convert the prior probability \( \tilde{\mu} \) of two nodes being in the same group to an approximation \( \tilde{p}^{(v,w)} \).
The formulas for the likelihood ratios $\hat{p}(v, w)$ to be in the same group for the karate club data. Nodes 4 and 8 shows which pairs of nodes are particularly likely or unlikely pairs $(w \bar{w} n)$ of the posterior probability $\hat{p}$ be represented in terms of planted partition model parameters $\hat{K}$ the conditional covariance of $v$ and $w$. They are based on quantities $\hat{K} v, w$ $\{I, \phi\}$ or absence $(\phi, \delta) = 0$ if $\phi_0 < 0$, and $f(\delta_0, 0)$ if $\phi_0 > 0$. Finally, the ratio $\hat{\Lambda}$ represents the contribution due to there being an edge ($k = 1$) or not ($k = 0$) between $v$ and $w$. These may be approximated as $\hat{\Lambda}_0 \approx \min(0.7197, 0.46 \delta_0^{−0.15})$, and $\hat{\Lambda}_1 \approx \min(0.5605 n + 1.598, \delta_0^{−0.7})$.

We have now given an approximation based on selecting only a subset of available evidence, and then performed approximations to the integrals involved, but with no indication of their accuracy. Thus there is much room for improvement, if not in the formulas themselves, then certainly in their justification. The primary purpose of this paper, however, is to motivate a probabilistic paradigm for network analysis similar to the one upon which principled data fusion is based, rather than to assert, e.g., that (17)–(19) could not be improved upon. Nevertheless, the reader may rightly question whether (17)–(19) are any good at all. To address this concern, we observe that one can use computed values of $p(v, w)$ to produce a traditional community detection algorithm. This is done by optimizing a certain expected utility function which defines the quality of a proposed partition $\pi$ to be the amount by which the average in-group $p(v, w)$ exceeds a certain threshold $\theta$ (where the number of groups in a computed solution increases with $\theta$) [18], [19]. Because a great many community detection algorithms have been developed, one can assess the quality of our $p(v, w)$ approximation by comparing the performance of the resulting community detection algorithm to those in the literature.

The most comprehensive comparison to date is based on the LFR benchmark graphs which have power-law distributions both on degree and on group size [20]. The conclusion is that all algorithms prior to 2008 were eclipsed by a set of three more recent algorithms: the Louvain algorithm [21], the Ronhovde–Nussinov (RN) algorithm [22], and Infomap [23]. Infomap performed somewhat better than RN, and both somewhat better than Louvain, but all three were much better than

---

**Figure 2:** $\hat{p}(v, w)$ plot for Zachary’s karate club.
  - $\hat{p}(v, w) \geq 60\%$;
  - $\hat{p}(v, w) \leq 1.8\%$

of the posterior probability $p_{v, w}$:

$$
\hat{p}(v, w) = \frac{\tilde{\Lambda} \Lambda_k}{\tilde{\Lambda} \Lambda_k + \mu^{-1}}.
$$

(14)

Using the log-uniform prior on the number of groups (discussed in Section II) we have $\mu = (1/2 - 1/n) / \log(n/2)$. The formulas for the likelihood ratios $\delta$ and $\phi$ defined as follows. Let $K_{vw}$ be a random variable representing the presence ($K_{vw} = 1$) or absence ($K_{vw} = 0$) of an edge between nodes $v$ and $w$. We define $\delta$ to be the mean value of $K_{vw}$, $\phi_t$ to be the conditional covariance of $K_{vx}$ and $K_{wx}$ given that $v$ and $w$ are in the same group, and $\phi_0$ to be the analogous quantity given that $v$ and $w$ are in different groups. The likelihood ratio $\Lambda$ may be decomposed into the product over $x \neq v, w$ of the evidence contained in the pair $(K_{vx}, K_{wx})$. If we let $n_t$ be the number of the $(n - 2)$ nodes other than $v$ and $w$ that are adjacent to exactly $j$ nodes in $\{v, w\}$, then $\Lambda$ may be represented as the ratio of integrals of the function

$$
f(\delta, \phi) =
(1 - \delta)^n \phi^n (\delta - \phi)^n_1 \phi^n_2.
$$

(15)

For small $n$, the integrals over $\delta$ and $\phi$ (which in turn may be represented in terms of planted partition model parameters $p_1$, $p_D$, and $\mu = 1/n$) may be evaluated numerically. Figure 2 shows which pairs of nodes are particularly likely or unlikely to be in the same group for the karate club data. Nodes 4 and 8 are most likely, with $\hat{p}(4, 8) = 98.8\%$, and nodes 1 and 34 least likely to be in the same group with $\hat{p}(1, 34) = 0.65\%$. Being adjacent does not guarantee a high value of $\hat{p}(v, w)$; the node pairs $\{1, 32\}$ and $\{14, 34\}$ each have $\hat{p}(v, w) = 8.9\%$. Nor is it necessary for nodes to be adjacent to have a high value of $\hat{p}(v, w)$; among nodes 15, 16, 19, 21, and 23 $\hat{p}(v, w) = 84.5\%$

for every pair, and $\hat{p}(8, 14) = 96.1\%$. Note that the node pairs $\{8, 14\}$ and $\{1, 34\}$ are each non-adjacent, and each has four common neighbors (i.e., $n_2 = 4$), but their values of $\hat{p}(v, w)$ differ by a factor of 150 because of the degrees of the nodes involved. Finally, although nodes 9 and 31 are in different ground-truth groups, $\hat{p}(9, 31) = 92.1\%$.

For large $n$, $f(\delta, \phi)$ has tight peaks, so the required integrals may be approximated by delta functions at these peaks. The global maximum of $f$ occurs at

$$
\delta_p = \frac{n_1 + 2n_2}{2(n - 2)}, \quad \phi_p = \frac{4n_0n_2 - n_1^2}{4(n - 2)^2}.
$$

(16)

However, because $\phi_0 \geq 0$ and $\phi_D \leq 0$, it is necessary to compute the locations of the maxima constrained to these regions, which, conveniently, turns out to be either $(\delta_p, \phi_p)$ or $(\delta_p, 0)$. This leads to the approximation

$$
\Lambda \approx \left\{
\begin{array}{ll}
\frac{f(\delta_p, \phi_p)}{f(\delta_p, 0)} & \text{if } \phi_p \geq 0, \\
\frac{f(\delta_p, 0)}{f(\delta_p, \phi_p)} & \text{if } \phi_p < 0.
\end{array}
\right.
$$

(17)

Finally, the ratio $\hat{\Lambda}_k$ represents the contribution due to there being an edge ($k = 1$) or not ($k = 0$) between $v$ and $w$. These may be approximated as

$$
\hat{\Lambda}_0 \approx \min(0.7197, 0.46 \delta_0^{−0.15}), \quad \text{and}
\hat{\Lambda}_1 \approx \min(0.5605 n + 1.598, \delta_0^{−0.7}).
$$

(18)

(19)
by first producing a binary dendrogram for the nodes, then selecting the better of the two orderings at each branch point. The standard methods of producing dendrograms require a distance metric between clusters [26]. For this we use

$$\rho(v, w) = 1 - \rho^{(v, w)},$$

which one can verify obeys the triangle inequality (although the approximation (14) sometimes violates it). We then define distance between clusters to be the average over $\rho(v, w)$ (i.e., average-linkage clustering).

To gain an understanding of a large network, it is often necessary to explore it from several viewpoints. Figure 5 shows multiple views of a dataset of Facebook connections on the campus of Georgetown University [27]. This network has 9,414 nodes and 425,638 edges. The dendrogram on which the ordering for the $\hat{p}^{(v, w)}$ matrix is based is shown in the upper left panel. Two levels in this dendrogram have been selected: the lower level is used to coarse-grain the network by merging groups of nodes together into meta-nodes; the upper level is used to determine which sets of meta-nodes to consider groups. The selection of these levels is reflected in the $\hat{p}^{(v, w)}$ matrix panel below. The meta-nodes are indicated by translucent green squares, and groups of nodes are outlined in different colors (corresponding to similar outlines in the dendrogram). The meta-nodes and groups are then displayed in panels on the right: the upper panel corresponding to a coarse-grained version of the original network; the lower, to a variant where the edges have been replaced with averaged $\hat{p}^{(v, w)}$ values between meta-nodes. The sizes of the meta-nodes indicates how many true nodes they contain. In the lower

The approximations (17)–(19) are very fast because one need only compute the value of $n_2$ for all node pairs, which is often 0 for sparse graphs, then compute $\hat{p}^{(v, w)}$ for each distinct value of $(k_{vw}, n_1, n_2)$ (where $k_{vw} = 1$ or 0 denotes the presence or absence of the edge $(v, w)$). Typically there are only a small number of distinct $(k_{vw}, n_1, n_2)$ triples, so the computation is dominated by computing $n_2$, which can be done quite efficiently. For example, in the wb-edu data set (a network of connections between .edu domains) [24], there are 9.5 million nodes and 46 million edges, and a total of 4.2 billion node pairs for which $n_2 > 0$. The $n_2$ computation was performed on a Dell desktop with 8GB of RAM and eight 2.5GHz processors in under three minutes [18].

Direct visualizations like Figure 2 are impractical for larger networks. An alternative is to plot the $\hat{p}^{(v, w)}$ matrix. This is done in Figure 4 for the Enron email communication network, which has 36,692 nodes and 367,662 edges [25]. To produce a useful plot of $\hat{p}^{(v, w)}$, one must first order the nodes in a manner that highlights the group structure. We accomplish this

the previous generation. Figure 3 compares our algorithm to the Infomap, RN, and Louvain algorithms, and to the other algorithms tested in [20]. The method is labeled $U_{opt}$ because numerical optimization over all $\theta \in [0, 1]$ has been used to set $\theta$ to the value that maximizes expected utility: in a deployable algorithm one would need a method for setting $\theta$, but here we are just testing the accuracy the algorithms tested in [20]. The method is labeled $U_{opt}$ because numerical optimization over all $\theta \in [0, 1]$ has been used to set $\theta$ to the value that maximizes expected utility: in a deployable algorithm one would need a method for setting $\theta$, but here we are just testing the accuracy the algorithms tested in [20].

Figure 3: Comparison of community detection algorithms:

(a) 1000 nodes, small groups
(b) 1000 nodes, large groups
(c) 5000 nodes, small groups
(d) 5000 nodes, large groups

Figure 4: Visualization of $\hat{p}^{(v, w)}$ matrix for the Enron email network
We now turn to the problem of using the co-membership probability \( p^{(v,w)} \) to infer a community structure on the graph via the metric \( \rho \) defined in (20). As a first approximation of the community structure on the graph, we could assign each node in the graph to the group defined by

\[
C_v(\tau) = \{ w \in G \mid \rho(v,w) < \tau \} \tag{21}
\]

for a fixed threshold \( \tau \). This approach would result in a collection of overlapping groups which cover the graph. By varying our choice of \( \tau \) we obtain a hierarchy of community structures. These properties are both desirable. However, we have lost much of the information contained in the the probability calculation \( p^{(v,w)} \). When determining the members of each group \( C_v(\tau) \) we have only considered the binary question of whether the value of \( \rho(v,w) \) is above or below \( \tau \). We need a more robust approach which takes into account the entire vector \( (\rho(v,w))_{w \in G} \) of distance values.

Since \( \rho \) is a metric, we could embed the nodes in a large Euclidean space in a manner which preserves the pairwise distances \( \rho(v,w) \). Unfortunately, as we discussed in Section III, the computation of \( p^{(v,w)} \), and thus \( \rho(v,w) \), is difficult. So we must use (14) to approximate \( \rho(v,w) \) as \( \hat{\rho}(v,w) = 1 - p^{(v,w)} \). Because \( \hat{\rho}(v,w) \) is not a true metric we may not be able to find a consistent embedding of the nodes into a Euclidean space. Furthermore, even if we could find an approximate embedding it is not clear where to go from there. Naively clustering in the resulting Euclidean space would produce clusters which have some meaning in that points near each other should be in the same group. However, such an approach would miss more subtle structures by ignoring the topology and geometry of the embedded nodes.

Instead we propose to use Mapper, a topological data analytic technique inspired by the branch of topology called Morse theory. The result of the technique will be a possibly overlapping community structure. The technique can be applied at various resolutions to obtain a hierarchy of such community structures.

### A. Topological inspiration

The overarching goal of topology is to understand the properties of geometric objects which remain unchanged under continuous functions. These properties include connectedness and the number of \( n \)-dimensional holes in the object. Classical Morse theory attempts to study a topological space \( X \) by filtering the space using functional level sets. Rather than attempting to understand \( X \) in its entirety, Morse theory describes how \( X \) can be understood by filtering \( X \) via a continuous function \( f : X \to \mathbb{R} \), for example, and considering the change in the level sets \( f^{-1}(-\infty, a] \) as we increase \( a \). This idea of filtering an object via a continuous function to a parameter space lies at the heart of the Mapper algorithm.

The Mapper algorithm [28]–[30] applies this concept of filtering a geometric object via a function to a parameter space to analyze discrete point cloud data \( D \). The goal of the Mapper algorithm is to replace the complicated, large, and possibly high-dimensional set \( D \) with a much simpler simplicial complex which captures the topological and geometric features of the original data. We choose to use simplicial complexes because, while they are intuitively simple objects, they are capable of capturing the rich array of geometric and topological features which may be present in the point cloud data. Thus Mapper is a data reduction technique, but the algorithm produces more information than the topological structure of the data. At a chosen resolution, the resulting simplicial complex encodes which data points in \( D \) should be grouped together while maintaining the topological and geometric properties of the original data set. It is this additional property of the Mapper algorithm which we will exploit to infer a community structure on the network.

Intuitively, a simplicial complex is a geometric object built from vertices, edges, triangles, and their higher-dimensional analogues. More formally, a simplicial complex \( K \) consists of a set \( K_0 \) of vertices together with sets \( K_n \) of \( n \)-simplices. An \( n \)-simplex is a subset of \( X_0 \) of cardinality \( n+1 \). We require that each \((k+1)\)-element subset of an \( n \)-simplex be in \( X_k \). For more information about simplicial complexes see Björner [31].

Before describing how the Mapper algorithm constructs a simplicial complex from point cloud data, let us examine how to construct a simplicial complex from a topological space \( X \). We start with an open cover \( \mathcal{U} = \{ U_a \}_{a \in A} \) of \( X \). We define the vertex set of the simplicial complex \( N(\mathcal{U}) \) to be the index set \( A \). A finite subset \( \{ \alpha_0, \alpha_1, \ldots, \alpha_k \} \subset A \) is defined to span a \( k \)-simplex in \( N(\mathcal{U}) \) if and only if the intersection
$U_{\alpha_1} \cap U_{\alpha_2} \cap \cdots \cap U_{\alpha_k}$ of sets in the cover $\mathcal{U}$ is non-empty. The resulting simplicial complex $N(\mathcal{U})$ is called the nerve of the cover. The key fact about $N(\mathcal{U})$ of interest to us is the “Nerve Theorem.” Loosely speaking, the Nerve Theorem states that in many situations, $N(\mathcal{U})$ is topologically equivalent to $X$ (see [31, p. 1850]). The Nerve Theorem allows us to understand the topological properties of a space $X$ by studying the much simpler simplicial complex $N(\mathcal{U})$. In a similar way the Mapper algorithm is a technique by which we create a covering of the data, construct the nerve of the covering, and from the nerve create a simplicial complex which captures the key topological features of data.

How do we obtain a covering of the topological space $X$? Returning to the motivating idea in Morse theory, we “parameterize” the space. Suppose we have a continuous function $f : X \rightarrow Z$ from the topological space $X$ to a space $Z$. The function $f$ is called a filter function and the space $Z$ is called the parameter space. Furthermore, suppose that we have a cover $\mathcal{U}$ of $Z$. Then we obtain a cover of $X$ by taking the collection $\{ f^{-1}(U_a) \}_{a \in A}$ of inverse images. In order for the nerve of the cover of $X$ to be topologically equivalent to $X$ we decompose each $f^{-1}(U_a)$ into its path-connected components $f^{-1}(U_a) = \bigcup_{b} V_b$. We denote by $\mathcal{U}_f$ the resulting covering of $X$. Generally, the parameter space $Z$ is chosen so that a cover $\mathcal{U}$ is easy to construct. The appropriate choices for $f$ and $Z$ depend on the nature of the space $X$ we are trying to study. In the next section we describe the filter function and parameter space we used to study the community structure of a network.

B. The technique

The Mapper algorithm is performed in three steps. First, we construct a cover of the network. Second, we cluster on each set in the cover of the network. Finally, we put the clusters together to form a topological reconstruction of the network. We now describe the Mapper algorithm as we have applied it to determine the community structure of a network. Mapper is actually a much more general construction. See [28] for details.

To cover the network we parameterize the network using a filter function as we indicated above. We choose a Gaussian kernel density function as the filter function. Specifically, we define the filter function $f : X \rightarrow [0, 1]$ to be the Gaussian kernel density function

$$f(v) = C \sum_{w} \exp \left( -\frac{\rho(v, w)^2}{\epsilon} \right). \tag{22}$$

The parameter $\epsilon$ is the twice the variance of the Gaussian kernel function and the constant $C$ is the normalizing factor so that $\int f(v) \, dv = 1$.

Let us explore why the density filter is appropriate for identifying group structures in a network. If $u$ and $v$ are nodes in the same group, the distance $\rho(u, v)$ between them should be small. Moreover, we expect the distances from other nodes in the group to also be small. Similarly, nodes outside the group should be roughly the same distance from $u$ and $v$. So intuitively, for $w \neq u$ or $v$ we expect $\rho(u, w) \approx \rho(v, w)$. Therefore as the density value is determined by the metric, the members of a group should have similar density values, and since we are trying to capture the group structure encoded by the metric, we cover the network using sets with similar density values.

The range of the density function is the unit interval $[0, 1]$. To obtain the cover of the network, we cover $[0, 1]$ with overlapping subintervals $\mathcal{U} = \{ U \}$. For convenience and ease of interpretation we choose equal-length subintervals which overlap with adjacent subintervals by a fixed amount. These choices introduce two parameters into the algorithm: the subinterval length and the overlap. The length $\ell$ of each interval controls the resolution of the cover and the overlap $s$ of the subintervals controls the overlap of the cover of the network. Together $\ell$ and $s$ determine the size and overlap of groups in the network. As above, the collection of inverse images $\mathcal{U}_f = \{ f^{-1}(U) \}_{U \in \mathcal{U}}$ covers the network.

To obtain the simplicial complex representing the group structure on the network, we cluster on each inverse image $\mathcal{U}_f = f^{-1}(U)$. Any clustering method can be used and the choice will depend on the application and the data. As was the case for generating dendrograms in Section III, we have found that average-linkage clustering works well for network data (see [26]). A cluster is a collection of nodes in the network with similar density values and which are near enough to each other to be clustered. As we argued above nodes from the same group should have similar density values. Thus, if the nodes have been identified as a cluster, it is reasonable to declare them to be a group in the network.

To complete the complex we must establish the links between the clusters. Because the cover $\mathcal{U}$ of the network was constructed of overlapping sets, the resulting groups will also be overlapping. We draw a link between two clusters if the clusters of nodes in the underlying network overlap. The structure of the resulting simplicial complex informs us about the community structure of the network. We can infer, for example, that two groups which are joined in the simplicial complex by a short path have similarities even though they do not have common members. These groups may share some acquaintances, or be part of larger groups which do overlap.

We can obtain a deeper understanding of the group structure on the network by varying the resolution of the covering. A fine covering (i.e. short interval length) will result in smaller groups; a coarse covering will result in larger groups. By applying the Mapper algorithm at a sequence of resolutions we can realize a hierarchy of community structures in the network. To get a good understanding of the group structure we recommend first using a coarse resolution to get a “big picture” view of the group structure and then applying a finer resolution to explore the more subtle group structure. Comparing this method to others is a subject of future work.

V. Conclusion

The purpose of data fusion is to convert data from various sources into situational awareness. The fusion problem is
usually framed in terms of an unknown state being reflected in the data, but with some degree of measurement error. Sometimes it is possible to identify the unknown state from the data with high confidence; more often the data supports only a posterior probability distribution over the state space. Networks are often studied as intrinsic objects of interest, but to consider them in the context of data fusion is to take a somewhat different approach: i.e., to treat them as the data reflecting an unknown state.

What is this unknown state? In general it could be anything that influences the structure of a network, but this is highly context-dependent. In this paper we treat community structure as a hidden state because this is a fairly universal property of large networks. Most community structure algorithms return as a hidden state because this is a fairly universal property of context-dependent. In this paper we treat community structure that influences the structure of a network, but this is highly

producing outputs that are probabilities is essential for the extension of this work to tracking communities on dynamic networks [32]. It also encourages modular architectures. Probabilities, unlike ad hoc measures of confidence, provide a well-defined interface upon which to design applications, even as improvements are being made to the efficiency and accuracy of the algorithms that produce them. We believe there is interesting work to be done on both sides of this interface: developing new, faster, and more accurate probabilistic assessments of network structure, and developing applications which make creative and fruitful use of such probabilities.

ACKNOWLEDGMENTS

Research partially supported by ONR Contract N00014090C0563. The authors thank Andrea Lancichinetti for the LFR benchmark data shown in Figure 3.

REFERENCES