General Exam

Jason Portenoy

The Information School, University of Washington

May 2017

Supervisory Committee:
Jevin D. West (chair)
Emma S. Spiro
Bill Howe
Benjamin Mako Hill (GSR)

Contents

History of community detection

Across many disciplines of science, it is common to encounter data that can be represented as a network, with entities linked to each other in some meaningful way, such as through association or flow. These entities are represented by *nodes* or *vertices* connected to each other with *links* or *edges*. This overall representation is known as a *network* or *graph*.[1] The idea of community detection as a research topic comes from a basic intuition that there exist in these networks groups of nodes that are structurally more related to each other than they are to members of other groups.

The field of *network science* has emerged recently to study this and related topics. This field is highly interdisciplinary, comprising physicists, applied mathematicians, computer scientists, sociologists, and others. This interdisciplinarity arises both from the variety of methods that can be applied, and the breadth of potential applications, often requiring domain-specific knowledge [44]. Within this new field, the concept of *community* has been somewhat more formalized from the idea above as a group of nodes (a *subgraph*) with a high concentration of edges connecting vertices within the group, and a low concentration of edges with nodes outside the group [13].

The earliest analyses of communities were made by social scientists in the early- to mid-twentieth century—for example Weiss and Jacobson's analysis of the organizational structure of a government agency [64]. More developments were made by computer scientists, who began developing graph partitioning algorithms in the early 1970s to apply to problems in parallel computing and circuit layout. In 2002, a seminal paper by Girvan and Newman [20] marked the entrance of the physics community, and ushered in the modern age of community detection [30]. The Girvan and Newman algorithm introduced in their paper involved successively calculating the *edge betweenness*—the number of shortest path between all nodes that run along the edge—of all edges, then removing the edge with the highest betweenness and repeating. The idea is that the edges with the highest betweenness centralities are the ones that connect communities, and the communities can be separated by this divisive algorithm. This work inspired the development of modularity as a quality measure (see section "The clustering perspective" below). Since then, the field has seen rapid growth and the development of many new methods.

Community detection methods

What follows is an overview of some of the many community detection methods currently in use. The overview follows the taxonomy laid out in a recent paper by Schaub et al. [55]. The authors identify four different perspectives on the problem of community detection: (i) the *cut-based perspective*, (ii) the *(data) clustering perspective*, (iii) the *stochastic equivalence perspective*, and (iv) the *dynamical perspective*. The different perspectives represent different approaches to the problem, often with different kinds of data, different methods, and different goals. They also represent, to some degree, the different research communities that have been working on the problem. While this is not the only way to classify the problem space, and

[1] The term *graph* generally refers to a mathematical representation of data, while *network* usually has additional connotations related to the meaning and context associated with the data [44]; however, as is the case with many terms in this area, the distinction is not always made and the two are often used interchangeably.

there is some overlap due to the many connections between methods, it is helpful in examining different approaches that have seen use.

The cut-based perspective

Some of the earliest work in community detection was in the area of circuit layout and design. A circuit can be represented as a graph describing the signal flows between its components. The efficient layout of of a circuit depends on partitioning the circuit into a fixed number of similarly sized groups with a small number of edges between groups—these inter-group edges are known as the *cut*. Similar problems can be found in load scheduling and parallel computing, where tasks must be divided into different portions with minimal dependencies between them. The need for these methods led to the development of the Kernighan-Lin algorithm in 1970 [25], which has become a classical method that is still frequently used. It starts with an initial partition and attempts to optimize a quality function Q representing the difference between intra-cluster edges and inter-cluster edges, by swapping equal-sized subsets of vertices between groups. The method works best if it starts with a decent initial partition, so in modern use it is often used to refine partitions obtained using other methods [13].

The cut-based perspective has also seen the development of spectral methods for graph partitioning. The spectrum (eigenvalues) of a graph's adjacency matrix tend to be related to the connectivity of the graph, and the associated eigenvectors can be used for both cut-based partitioning and clustering. These methods typically make use of the Laplacian matrix L of a connected graph G: $L = D - A$ where A is the adjacency matrix of G, and D is the diagonal degree matrix with $D_{ii} = \sum_j A_{ij}$. The *Fiedler vector* is the second-smallest eigenvector associated with the Laplacian matrix L; the spectral bisection method uses this vector to quickly partition a graph into two groups with a low cut size [12].

A cut-based measure for the quality of a partition is the *conductance*. The conductance of a subgraph $S \in V$ for a graph $G(V, E)$ is:

$$\phi(S) = \frac{c(S, V \setminus S)}{\min(k_S, k_{V \setminus S})}$$

where $c(S, V \setminus S)$ is the cut size of S, and k_S and $k_{V \setminus S}$ are the total degrees of S and the rest of the graph, respectively [54]. While this measure was originally used globally to optimize a bisection of a graph, it has also seen use as a local quality function to find good clusters around certain nodes; in this way it can also be viewed as part of the clustering perspective [55]. Conductance has its roots in computer science, and its use in network science appears still to be especially popular in the computer science community [54, 68].

The clustering perspective

The clustering perspective comes from the world of data clustering, in which data points are thought of as having "distance" between each other based on their (dis)similarity, and the goal is to group together data points that are close to each other. For community detection, this distance is in relation to the connections between nodes in the network. This perspective is related to but different from the cut-based perspective above, which seeks to place divisions among the nodes so as to form balanced groups with weak connections between groups.

A classical method with this perspective is *hierarchical clustering*, which when used on graphs yields a hierarchical partitioning that can be viewed as a dendrogram. The common method uses an agglomerative approach in which each node starts in its own cluster, and they are joined together one by one based on some similarity measure calculated using the graph's adjacency matrix. This approach to community detection has several weaknesses: It necessarily infers a hierarchical community structure even if one does not exist; the hierarchy is not always easy to interpret; it often misclassifies nodes, especially nodes with only one neighbor, which it tends to put it in its own cluster; and it does not scale well to large networks [13].

The *modularity* is a quality function originally developed as a stopping criterion for the Girvan-Newman algorithm (see section on history above) [39, 40]. The algorithm partitions a graph by successively removing links, but the original formulation did not have a clear stopping point at which the communities have been identified. The modularity compares an overall partitioning against a null model. For a given graph $G(V, E)$ and partitioning $\mathcal{C}$, the modularity is:

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j)$$

where m is the total number of edges in G, A is the adjacency matrix of G, k_i is the degree of vertex i, and $\delta(C_i, C_j)$ is a function that yields one if vertices i and j are in the same community in $\mathcal{C}$, zero otherwise. The term $k_i k_j / 2m$ represents the standard null model used for modularity—the configuration model which preserves the degree sequence of the original graph.

Once modularity was introduced, network researchers, especially those in the physics community, seized on the idea and began developing clustering algorithms that aim to maximize it—it has become the most popular and best known quality function [13]. Finding the optimal modularity for a graph is an NP-complete problem; finding the global maximum for modularity on all but the smallest networks is computationally infeasible, so algorithms attempt to approximate the global maximum of modularity in a reasonable amount of time. Algorithms tend to have a tradeoff between speed and accuracy. Greedy algorithms, such as the original method of Newman [38], tend to be fast but inaccurate—they discard much of the problem space in order to arrive at a solution more quickly. Simulated annealing techniques improve accuracy at the expense of speed by introducing stochastic noise to nudge the algorithm away from local maxima; the standard simulated annealing algorithm for modularity can be found in [21]. Many other techniques, including spectral optimization and genetic algorithms, have been employed in attempts to get accurate solutions quickly—see a summary in [13].

Modularity became popular in part because it was one of the first attempts to formalize the concept of community, including in its mathematical form the definition of community in relation to a defined null model. The measure has been shown to have some problems, however—most notably its resolution limit. Modularity-maximizing techniques have a lower limit on the size of communities it is able to detect which depends on the number of links and the interconnectedness of communities [14]. This can be a problem in many real-world networks, which can contain communities of many different sizes. In addition, modularity can be very sensitive to even individual connections, which can be a problem in any noisy dataset which can contain false positives [13].

The stochastic equivalence perspective

The stochastic equivalence perspective groups nodes by their connectivity profile—their probability to connect to nodes in other groups. The most famous model to come from this perspective is the *stochastic block model* (SBM), a probabilistic generative model with roots in statistics and social network analysis. The model sees the community assignment g_i for each vertex i as a latent variable, with nodes in each group having equal probability of connecting to nodes in any other given group (or to other nodes in the same group). These probabilities form what is called the *stochastic block matrix* (or affinity matrix), a $q \times q$ matrix, q being the number of groups, with diagonal elements p_{kk} being the probabilities that nodes of block k are neighbors, and the off-diagonal elements p_{lm} being the edge probabilities between different blocks l and m. The model is then fit to the data by estimating the parameters that give the highest likelihood [15, 55]. This view of equivalence between nodes is not bound by definitions of community that describe the relative density of edges within and across groups, so these models can capture a broader range of structure than the previously described techniques. For example, if the diagonal of the affinity matrix $p_{kk} = 0, \forall k$, the model describes a multipartite graph, in which members of a given group can only link to members of a different group.

There are several advantages to using a generative model such as the SBM to describe the community structure of a network. A generative model views the observed data as one realization of an ensemble of possible networks generated by a stochastic process. Thus, it allows us to make statistical claims about the network data and its properties, such as calculating p-values and confidence intervals for node properties. By generating new instances of the model, we are also able to speculate about possible noise in the data, including predicting missing links. However, these methods have disadvantages as well. For one, they do not scale well to very large networks. They also require that the number of communities to identify be specified, and while ideas have been proposed for how one can select this number, it remains an open problem [15].

The dynamical perspective

The fourth perspective that Schaub et al. identify is the dynamical perspective, and it differs from the above three in that it is not directly concerned with the structure of the network, but rather the dynamical processes acting on the network. In other words, it aims to find coarse-grained, simplified representations of the *behavior* of a system, rather than its topology. Since the structure of a network has a strong influence on its behavior, the results of these methods can actually be very related to the above methods. For example, when modeling the network dynamics using a random walk, the random walker tends to get "stuck" in communities that look very similar to those identified using the clustering perspective, e.g., modularity. So a modular description of the random walker's behavior will closely resemble modularity methods' clustering of the network topology.

The *map equation* uses elements of information theory to describe a random walker's movements on a network. It does this by quantifying the amount of information (in bits) it takes to describe the walker's movements given a particular modular structure of the nodes. The goal is to exploit the relationship between compression and relevant modular structure of a system. The optimal compression means that the smallest number of bits of information is needed to describe the random walker's movements. The map equation is the

minimal description length (MDL) of the diffusion process modeled by the random walker given a modular structure.

To make this idea clearer, an analogy can be made to the world of cartography, which is where the map equation derives its name. Assigning an address to a location in a city is a way of encoding that location. Parts of an address can be reused in different locations—there can exist multiple Main Streets in different cities. Reusing these names makes describing an individual's movement within a city more efficient (fewer bits), since as long as we know the individual is staying within the city limits we can just use street names without causing confusion. The map equation formalizes this idea mathematically, providing a function that can be optimized to find good community structure given a network and a model of the dynamics on that network. The method does not actually find the description of the network (the codewords that would be used to compress the network), but rather the lower theoretical limit on the number of bits that would be needed to do the encoding.

Information theory states that this lower bound for the average length of a codeword to describe a random variable X with n possible states that occur with frequences p_i is equal to the entropy of that random variable: $H(X) = -\sum_{i=1}^{n} p_i \log p_i$ [7]. (It is standard to use base-2 for the logarithms, which yields calculations in bits.) The map equation imagines that there are separate codebooks for each module (community), and is thus the combined entropy of each codebook plus an additional index codebook that allows for switching between modules, rated by their rates of use:

$$L(\mathsf{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{P}^i)$$

where M is the module partitioning; the left term is the average length of codewords (entropy) in the index codebook weighted by the rate of use of the index codebook $q_{\curvearrowright}$; and the right term is the average length of codewords in module codebook i weighted by the rate of use of this module $p^i_{\circlearrowright}$. This equation can be expanded in terms of the individual node visit rates; see section "Infomap implementation in Python" for details.

The node-visit probabilities that define the p's and q's in the map equation can be calculated using the PageRank algorithm [42], originally developed to rank web pages in the hyperlinked world wide web. PageRank is essentially eigenvector centrality—the node-visit probabilities after an infinite random walk are equal to the leading eigenvector of the adjacency matrix (the eigenvector associated with the largest eigenvalue). PageRank includes an additional element: at any step of the random walk, with a given probability, the walker can "teleport" to a random node on the network. This teleportation is necessary to avoid the possibility of the random walker getting stuck forever in an area of the network with no outward links. The random walker represents an ergodic Markov chain—a time process in which the state at the next step only depends on the current state, and the probability distribution of all states as time approaches infinity is well defined.

For undirected networks, this method gives results that are very similar to the clustering perspective above, since the network structure and first-order dynamics are so tightly coupled [55]. However, for directed networks, differences can be seen, as the clustering perspective is more concerned with pair-wise structural relationships between nodes, while the dynamical perspective deals with directed flow. This is illustrated in

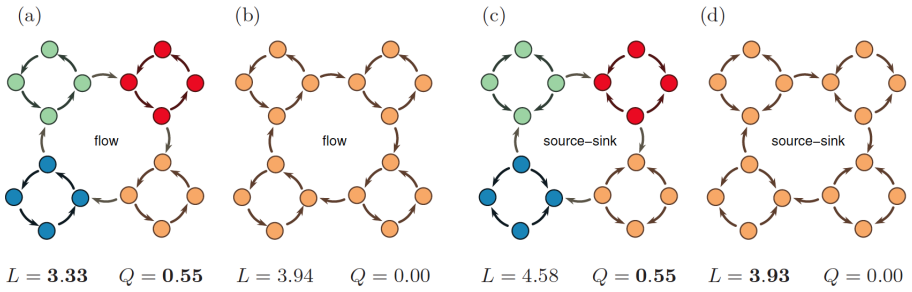

Figure 1: Comparing community detection found using the map equation vs. using modularity. For each network diagram, L is the map equation, Q is the modularity, and the colors of the nodes represents the partitioning. All four networks are the same, except (a) and (b) have differently directed links than (c) and (d)—the former allows flow between clusters; the latter traps flow so that the clusters act as flow sinks. The optimal partitioning—(a) and (c)—are identical for optimizing modularity, but different for optimizing the map equation ((a) and (d)). Figure from [47].

Fig. 1, in which the difference between directed links that allow flow between clusters vs. links that trap flow within clusters has an effect on the optimal partitioning for the map equation, but not for modularity.

The map equation approach is flexible and can give different results given different dynamics induced on the system. For example, instead of the standard PageRank implementation of the random walker, the teleportation can be tweaked, e.g., by employing teleportation to links instead of nodes, or changing the direction of links in strategic ways—this changes the node visit probabilities used in calculating the map equation [28]. Another possibility is the use of higher-order dynamics: instead of modeling the flow as a first-order Markov process, in which every step of the walker depends only on the current position, we can take into account the influence of the position before the current one, or include several time points before the present. This can have important impacts on the communities found [51].

The algorithm used to find the community structure that optimizes the map equation is called Infomap, and the code is available at `http://www.mapequation.org/`. It works similarly to the Louvain algorithm used to optimize modularity [2]. Each node starts out in its own module; the modules are joined in a greedy fashion (examining each node one at a time, in a random order) to yield the largest increase of the map equation. These modules are joined into super-modules, the network is reconstructed with these super-modules as the nodes, and the process is repeated until no further improvement can be made. This greedy approach has a danger of finding a local optimum, so at the end, the solution space is further explored by reapplying the algorithm for each module, and by rerunning the algorithm but allowing for single-node movements between modules. This method is fast, and so can be repeated multiple times (with different random seeds) to broaden the search space more [49, 47, 50]. See section "Infomap implementation in Python for further discussion on Infomap. The method has been extended to allow for hierarchical partitions [50], overlapping communities [60], and higher-order Markov dynamics [51].

Evaluation of community detection methods

The lack of consensus on exactly what a community is and what is meant to be achieved by its detection has presented problems for the evaluation of community detection methods. Still, attempts have been made to systematically evaluate the performance and output of different methods.

Evaluating the results of any community detection method can be thought in terms of either *internal* or *external* validity. Measures of *internal* validity evaluate the output of a clustering algorithm according to some quality measure that uses only the properties of this output; these include modularity [39], minimum description length [47], and conductance [32]. These measures are designed to measure how well the communities identified by a method adhere to some mathematical definition of what a proper community structure should look like. The problem with using these measures to evaluate and compare methods is that these measures often serve as the objective functions for the very algorithms we want to evaluate. While it may be useful to use these quality measures to compare algorithms that are trying to optimize the same function, it may not be fair to compare more broadly than this. As there is no strict mathematical definition for a community, different algorithms use different quality functions to surface community structure, and those algorithms that optimize for whatever measure we are using to evaluate would have an unfair advantage. Some of these quality measures are discussed in the previous section on existing community detection algorithms.

Because of this, much of the work around evaluating community detection methods has focused on *external* validity measures, in which the input is a network with a known community structure. The evaluation in this case measures how well the community detection method matches this ground truth. These "gold standard" networks used for evaluation are either (1) synthetic benchmark networks created with planted communities, or (2) real-world networks with known metadata which are treated as ground-truth communities. In either case, the results of a community detection algorithm can be evaluated against the expected structure using some comparison measure.

Comparison measures

Evaluating a community detection method against either a synthetic benchmark network or a real-world network with known community structure requires some measure of comparison between the clustering found by the method and the ground truth clustering. The popular measures that have been adopted fall into one of three categories: (1) measures based on *pair counting*, (2) measures based on *set matching*, or (3) measures based on *information theory* [35, 62]. These are all general measures comparing data (not just network) clusterings; they work by viewing the network as data points with communities as cluster assignments.

Pair counting measures work by taking every possible pair of nodes in the network and classifying them based on their co-occurrence in the clusterings. Each of these categories is then counted:

- N_{11}: the number of pairs that co-occur in the same cluster in both clusterings
- N_{00}: the number of pairs that do not co-occur in either clustering
- N_{10} or N_{01}: the number of pairs that co-occur in one clustering but not the other.

Examples of measures that use these counts include the Fowlkes-Mallows index [17] and the Rand index [45]. The Rand index, for example, is the ratio of pairs correctly classified in both clusterings to the total number of pairs:

$$\frac{N_{11} + N_{00}}{N_{11} + N_{00} + N_{10} + N_{01}}$$

This measure has a value between zero and one, with one representing perfect agreement between the clusterings and zero representing no agreement whatsoever. In practice, it is rare to see values on the lower end of this range, so a transformation is usually applied that sets a baseline that accounts for chance—this is known as the adjusted Rand index.

Set matching measures compare clusterings by finding matches between the clusters—for example, by treating the cluster assignments as labels and calculating the classification error rate. This approach has problems when the two clusterings to be compared have different numbers of clusters, however. Even within the clusters that match, these measures only consider the matched part of each cluster pair, leaving out the parts that do not match. For these reasons, these measures are not very widely used [35, 62].

Information theoretic measures use elements of information theory to compare clusterings. The *entropy* $H(\mathcal{C})$ of clustering $\mathcal{C}$ is the average amount of information (in bits) needed to encode and transmit each label. The *mutual information* between two clusterings $\mathcal{C}$ and $\mathcal{C}'$ is the entropy of $\mathcal{C}$ minus the conditional entropy of $\mathcal{C}$ given $\mathcal{C}'$, or vice versa: $H(\mathcal{C}) - H(\mathcal{C}|\mathcal{C}') = I(\mathcal{C}, \mathcal{C}') = H(\mathcal{C}') - H(\mathcal{C}'|\mathcal{C})$. If we let $P(k), k = 1, \ldots, K$ and $P'(k'), k' = 1, \ldots, K'$ be the random variables associated with the clusterings $\mathcal{C}$ and $\mathcal{C}'$, respectively, and $P(k, k')$ be the joint probability—the probability that a point belongs to C_k in clustering $\mathcal{C}$ and to $C'_{k'}$ in clustering $\mathcal{C}'$, then:

$$I(\mathcal{C}, \mathcal{C}') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} P(k, k') \log \frac{P(k, k')}{P(k)P'(k')}$$

The mutual information tells us, on average, how much knowing the cluster assignment of a point in $\mathcal{C}$ reduces our uncertainty of which cluster it belongs to in $\mathcal{C}'$. Several variations of the mutual information measure have been proposed, including normalized versions that are meant to vary between 0 (the clusterings are independent of one another) and 1 (the clusterings are identical); and versions adjusted for chance [62]. Another measure is the *variation of information*:

$$\begin{aligned} VI(\mathcal{C}, \mathcal{C}') &= H(\mathcal{C}) + H(\mathcal{C}') - 2I(\mathcal{C}, \mathcal{C}') \\ &= H(\mathcal{C}|\mathcal{C}') + H(\mathcal{C}'|\mathcal{C}) \end{aligned}$$

Finally, Lancichinetti et al. proposed a version of the normalized mutual information that can handle the case of *covers*, clusterings in which a node can be assigned to more than one cluster [31]. While the authors point out that their measure is not a true extension of normalized mutual information because it gives different values when used to compare normal "hard" clusterings, they contend that the difference is small [30].

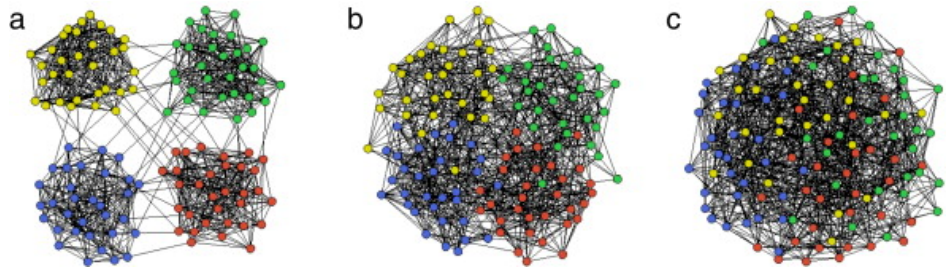

Figure 2: Examples of the Girvan-Newman benchmark. From left to right, the parameter z_{out} increases (and z_{in} decreases), and the community structure becomes less apparent. In (a) $z_{in} = 15$; in (b) $z_{in} = 11$; and in (c) $z_{in} = z_{out} = 8$ and the four clusters are not visually apparent. Image from [21].

Synthetic benchmark networks

Evaluating community detection methods can mean comparing their results with the expected results on some network with known community structure, using the above similarity measures. One way to get such a network is by using computer-generated synthetic benchmarks networks, which are created with some notion of built-in community structure. While there is no clear consensus on exactly what this structure should look like, some standard benchmarks have emerged.

The *planted ℓ-partition model* is one method of generating such a benchmark network. In this model, a graph is created with a certain number (ℓ) of clusters, each having the same number of nodes. Nodes are connected with edges to other nodes in the same cluster with probability p_{in}, and to nodes in other clusters with probability p_{out}; as long as $p_{in} > p_{out}$, the graph has some community structure. The version of this model that has become standard is known as the Girvan and Newman (GM) benchmark [20]. In this version, the number of clusters is set at 4, with 32 nodes per cluster, and an average total degree of 12. The parameter one uses to change how pronounced is the community structure of the generated network is z_{out}—the expected external degree of a node. (Because the expected total degree of a node is fixed, z_{in}—the expected internal degree of a node—depends on z_{out}). As z_{out} increases, the community structure of the graph becomes less apparent, until $z_{out} = z_{in} = 8$, in which case the internal and external degrees are equal (see Fig. 2). Many algorithms are able to properly identify communities up to this limit [13].

The weakness of the GN benchmarks and the planted ℓ-partition model in general lies in assumptions that all communities have the same size and all nodes have the same degree on average. Real-world networks tend to exhibit skewed, power-law-like distributions for cluster size and node degree. Lancichinetti et al. have taken steps to address these issues with their LFR benchmark, which is generally seen as an improvement over the GN benchmark [29, 13]. When an LFR network is generated, both the degree and cluster size are assumed to have power law distributions, with exponents γ and β, respectively. The mixing parameter μ specifies, for each node, the fraction of its links that go outside its community. A graph of N vertices is generated using a configuration model which assigns links given the proper degree sequence; the graph is then rewired with an iterative algorithm until it can be assigned a community structure with the proper parameters.

Evaluation on real-world networks

A second approach to evaluate community detection methods is to use some real-world network data and compare the results of the method against known metadata attributes, which are treated as ground-truth community structure. The most famous of these networks is the Zachary karate club dataset [69], which is a network representation of the social interactions between 34 members of a university karate club. A conflict between the club president and the instructor resulted in the formation of two groups of members who had allegiance to one leader or the other. Community detection methods can be judged according to how well they are able to infer this group division from the network structure alone. This network has become so widely used as a benchmark in network science that researchers who are first to use it as an example at a conference are awarded a trophy and inducted into the prestigious "Zachary Karate Club Club."[2] Several other networks have also emerged as standard benchmarks: Lusseau's network of interaction between 62 bottlenose dolphins in New Zealand's Doubtful Sound is compared against a biological classification [34]; and a network in which nodes represent college football teams and edges represent games played between them is compared against the known conference divisions of the teams [20].

The assumption that known metadata can be equated with community structure has recently begun to be called into question, amid several findings that most community detection algorithms tend to do a fairly poor job of recovering these classifications on many large networks [68, 23]. This may represent something of a crisis for the field.

A very recent paper by Peel et al. breaks down these issues [43], providing an interesting and provocative perspective. While it has been assumed that a method's failure to recover the metadata associated with the network means that the method has performed poorly, the authors point out that there are actually three alternative interpretations: (i) that the metadata do not actually correspond with network structure, (ii) that the metadata correspond to a different aspect of the network structure than that revealed by the community detection method, or (iii) that the network actually has no detectable community structure. People have been aware of these issues for some time. Peel et al. point out one example, a node in the Zachary Karate Club data that is frequently "misclassified" by algorithms actually represents a student who had strong ties to the club's president after the split, but nominally remained aligned to the instructor so as not to lose his progress toward his black belt. Girvan and Newman, in their original paper on their novel method in 2002, also seem aware of this, clarifying some results on the football conference network, "Naturally, our algorithm fails in cases like this where the network structure genuinely does not correspond to the conference structure. In all other respects, however, it performs remarkably well" [20]. Despite this awareness, however, these issues have been largely ignored when evaluating methods.

Peel et al. propose a No Free Lunch theorem for community detection, claiming that no one method can outperform any other on all cases, implying that methods that have an advantage for a certain set of cases will have decreased performance in other cases. They stress that this does not imply that community detection is a useless endeavor, but they do contend that it is impossible to find one optimal method, and that efforts should instead be put into understanding the problem space of different community detection tasks, so that different

[2] See http://networkkarate.tumblr.com/

algorithms can be developed for each one. The recent review paper by Schaub et al. [55] is one example of this kind of effort, classifying and distinguishing the different perspectives on community detection.

The authors of this study introduce two interesting new statistical methods to address cases (i) and (ii) in the paragraph above. The first identifies the extent to which the metadata relates to the network structure by comparing the entropies of two probabilistic models: one representing the metadata partitions, and the other a null model with random permutations of the metadata (this null model preserves network structure and relative frequencies of metadata values but removes the correlation between the two). The second statistical method is meant to shed light on the relatedness between the structural aspects of the metadata versus the structural aspects that a community detection algorithm identifies. It does this by imposing a constraint on the community detection algorithm that fixes some portion of the community structure produced based on the metadata partitions. By varying this constraint, one can explore (visually, using a graph) how the imposition of the metadata affects the likelihood of the community detection method. Both of these methods are general enough to work with any sort of probabilistic generative network model; the authors implement them using stochastic blockmodels (see previous section on community detection methods), calling the first method the blockmodel entropy significance test (BESTest), and the second one the neoSBM.

Results of comparative analyses of algorithms

A 2009 paper by Lancichinetti and Fortunato [30] has gained some acceptance in the network science community as a comparative analysis of the performance of some of the most popular community detection algorithm. The 12 algorithms studied included most of those previously discussed (see section "Community detection methods"), including the original Girvan-Newman algorithm, several greedy and more exhaustive modularity-optimizing algorithms, an Expectation-Maximization Bayesian model-fitting approach similar to stochastic block models, a spectral algorithm, the Markov cluster algorithm popular in bioinformatics, and the flow-compression-based Infomap algorithm of Rosvall and Bergstrom, among several others. They also included one algorithm that can find overlapping partitions, called Cfinder.

The algorithms were tested on random graphs generated using the LFR method (see section "Synthetic benchmark networks"); these benchmark graphs are more rigorous than the Girvan-Newman benchmarks that came before. The results were compared using their variant of normalized mutual information (see section "Comparison measures"). Briefly, the results were as follows: The original Girvan-Newman algorithm suffered from too many performance issues to be considered, failing to finish on most networks (the authors included this method mostly for historical reasons; the other methods are more modern and improved). The modularity-based methods tended to suffer from the resolution limit (see section "The clustering perspective"), failing to detect smaller planted communities on larger graphs. The authors concluded that Infomap was overall the best performing algorithm, with good performance on different sized graphs, and also directed and weighted graphs. It was also one of the fastest methods, along with the Louvain method—the authors were able to test these two on larger graphs (up to 100,000 nodes), and only Infomap maintained good performance. The study did not study hierarchical partitions, or the overlapping or higher-order versions of Infomap.

The authors also test the methods on random graphs. An algorithm taking as input a random graph, in which every pair of nodes having an equal probability of being linked, should not find a community structure,

but this is not always the case (see section "Confidence in communities"). Methods based on modularity fall short here, finding some communities in random graphs. Infomap tends to perform better, finding a single module comprising all nodes. However, Infomap still does find some community structure when the average degree of the nodes is low—i.e., the graph is sparse. This could pose a problem, as many real-world graphs are sparse.

A recent paper by Emmons et al. [9] is an update to the Lancichinetti and Fortunato paper. Computation has advanced, and they were able to test LFR benchmark graphs up to 1 million nodes on four algorithms: Louvain, Smart Local Moving (SLM), Infomap, and Label Propagation. Louvain and SLM are both modularity-optimizing methods that work on large networks; Label Propagation is a method equivalent to the Pott's model approach (another clustering method I have not extensively investigated in this work). The authors found results that contradict Lancichinetti and Fortunato, with Louvain (and SLM) outperforming Infomap. The most likely reason for this is discussed in the results: the authors choose to use benchmark graphs with relatively large clusters, so the resolution limit of the modularity-based methods (Louvain and SLM) actually work in their favor, when usually they are a detriment. The results of Infomap are based on the lowest level of the hierarchical partitioning—at this level Infomap finds smaller clusters than those planted, leading to the lower information recovery scores.

Applications of community detection

The extensive work behind developing and validating methods for community detection is presumably meant to work toward a goal of using community detection for some concrete applications. In this area, the field shows its (young) age—it is somewhat difficult to find published examples where the methods have been applied to solve a specific problem or gain significant new insight into a system. Below, I discuss some of the examples that do exist in the research literature, in the fields of social network analysis, networks of scholarship, biological networks, and others. Besides published research, I speculate on the (mostly unpublished) applications of community detection outside of academia, and present an example of using community detection in the context of a small data science project to address a question of interest.

Social network analysis

Social networks often contain intuitive and potentially interesting community organization, so it is not surprising that there has been interest in applying community detection to various forms of social data. The enormous popularity of online social network platforms like Facebook and Twitter have made detailed social network data available at a massive scale. These data are not available equally to everybody—most popular social networking applications are proprietary and much of their data are reserved for internal research. It is likely that these companies are using community detection to explore patterns in their data, but not publishing all of their results.

One published study of Facebook was conducted by Traud et al. [57]. They examined the Facebook friend network from 100 American colleges and universities (at the time of this study, Facebook was restricted to these institutions). They performed community detection using several modularity-optimizing techniques and compared the community structure against several (self-reported) categorical demographic variables—gender, class year, high school, major, and residence. Fig. 3 shows a visualization of community structure with class year—the variable most strongly associated with community structure. The authors find some patterns that we might reasonably expect, such as class year being associated with friend communities, and high school being more associated with communities at larger institutions (where one is more likely to find multiple people from the same high school). They also identify some patterns that would be interesting to study further, such as that females were more likely to have friends within their same residence.

Weng et al. [65] used community detection to study the virality of memes on Twitter. Since the question of interest was one of information flow, they used Infomap, a flow-based community detection method (see subsection "The dynamical perspective" in section "Community detection methods" above). They identified communities in networks of reciprocal-follower relationships among Twitter users. Fig. 4 shows the evolution of a viral meme vs. a non-viral meme (here a meme corresponds to a hashtag), with the communities represented as nodes. The results suggest that a meme that is less concentrated in one community is more likely to spread, and that community structure might be helpful in predicting the virality of memes. They go on to apply a random forest classifier for new memes that takes into account community features, and find that these features are helpful in predicting whether a meme will go viral.

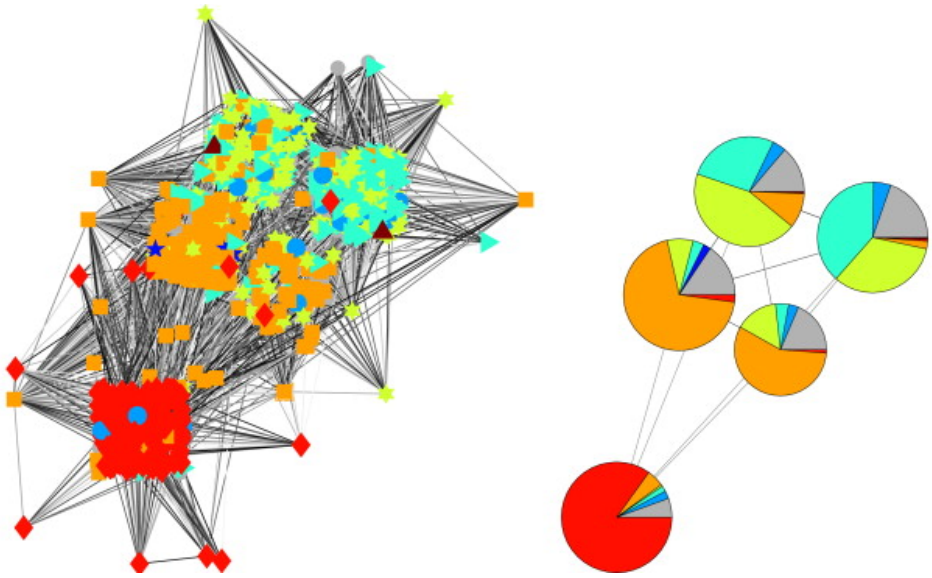

Figure 3: Community structure from the Reed College Facebook friend network. On the left, colors and shapes of nodes indicate different class years. On the right, the communities are condensed into pies. The size of the pies correspond to the number of nodes, and darker edges mean more connections between communities. A correlation between class year and community structure can be seen. Figure from [57].

Networks of scholarship

The collective human endeavor of knowledge generation and organization can be represented as a directed network of scholarly publications with citations between them. The citation links between publications can been viewed as a proxy for influence or information flow. De Solla Price recognized the potential of this representation in the mid 20th century [56], and over the years much work has been done in this meta-scientific research area, which has been given terms such as "bibliometrics," "scientometrics," and "science of science". In these networks, publications (or journals, or authors) form communities, which intuitively represent different fields of scholarship. Rosvall and Bergstrom [49] applied the Infomap algorithm on the journal citation network to build a map of science, visualizing information flows between disciplines (see fig. 5). Rosvall and Bergstrom also applied community detection at different time points to build maps of how science changes over time, revealing the formation of the standalone field of neuroscience by 2007 from various other disciplines starting in 2001, shown in Fig. 6 [48]. Communities in citation networks have also been used in a recommendation system for scholarly articles [66]; and as a way of representing distance between scientific fields, in combination with other measures of distance such as language barriers arising from the use of jargon [61].

Biological networks

Many biological systems can be thought of as structured interactions between functional elements, often with (possibly hierarchical) modular structure. It is natural to think of these systems as networks, and to see promise in the prospect of identifying communities in this network representation.

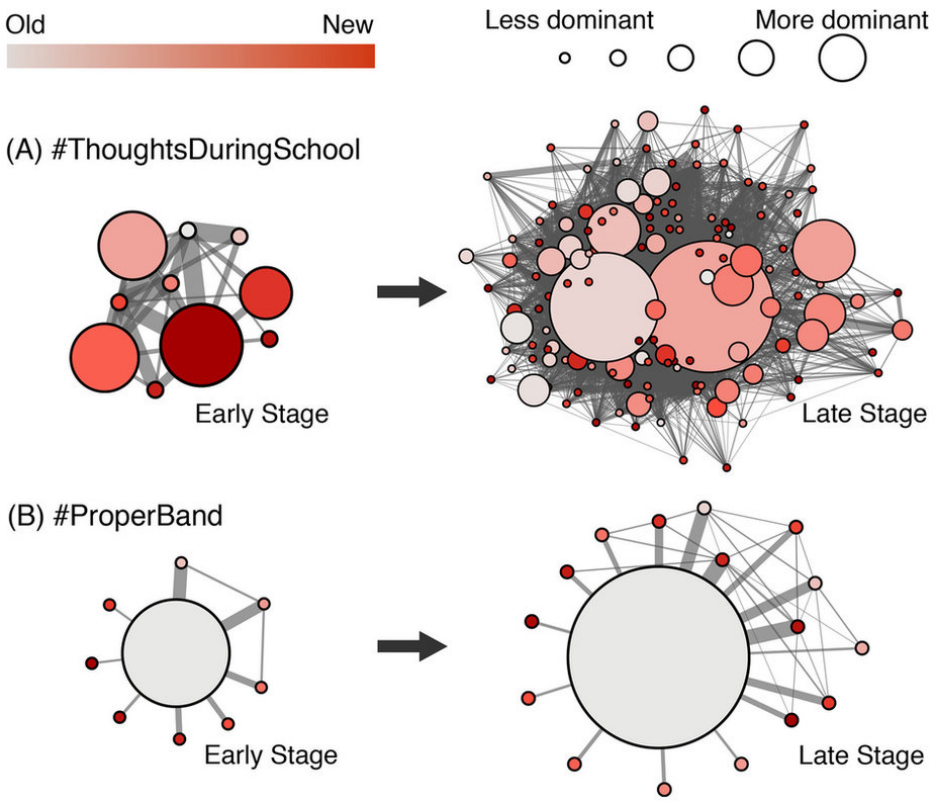

Figure 4: The evolution of a viral meme (A) vs. a non-viral meme (B). Each node is a community, with size proportional to the number of tweets produced by the community, and color indicating the relative time that the hashtag was first used by the community. Figure from [65].

Protein-protein interaction networks are constructed from data collected in experiments that identify molecular interactions between proteins. Community detection on these networks, for example using the Girvan-Newman edge betweenness method, has been shown to be effective in identifying what are known as "functional modules" in these networks—groups of proteins that interact in the service of a particular cellular process. Clusters found in these networks correspond to existing annotations, suggesting promise for the automated analysis of experiments. These methods were also found to be robust against false positive interactions, which is important considering that experimental results can contain considerable noise [8]. Chen and Yuan [5] integrated multiple protein interaction datasets containing hundreds of microarray expression profiles for *Saccharomyces cerevisiae* (brewer's yeast) to form a weighted graph, in which the weights correspond to dissimilarity between genes' expression profiles. By classifying the genes into functional modules using a modified version of the Girvan-Newman method, they were able to predict the function of the as-yet not annotated yeast gene *YLR419w* to be chromosome segregation.

Another biological network that has been studied is the directed network of neuronal connections—the "connectome". Dynamically-minded community detection methods (see subsection "The dynamical perspective" in section "Community detection methods" above) are especially relevant for these networks as the connectome represents a system of information flow, which is what these methods model. Bacik et al. [1]

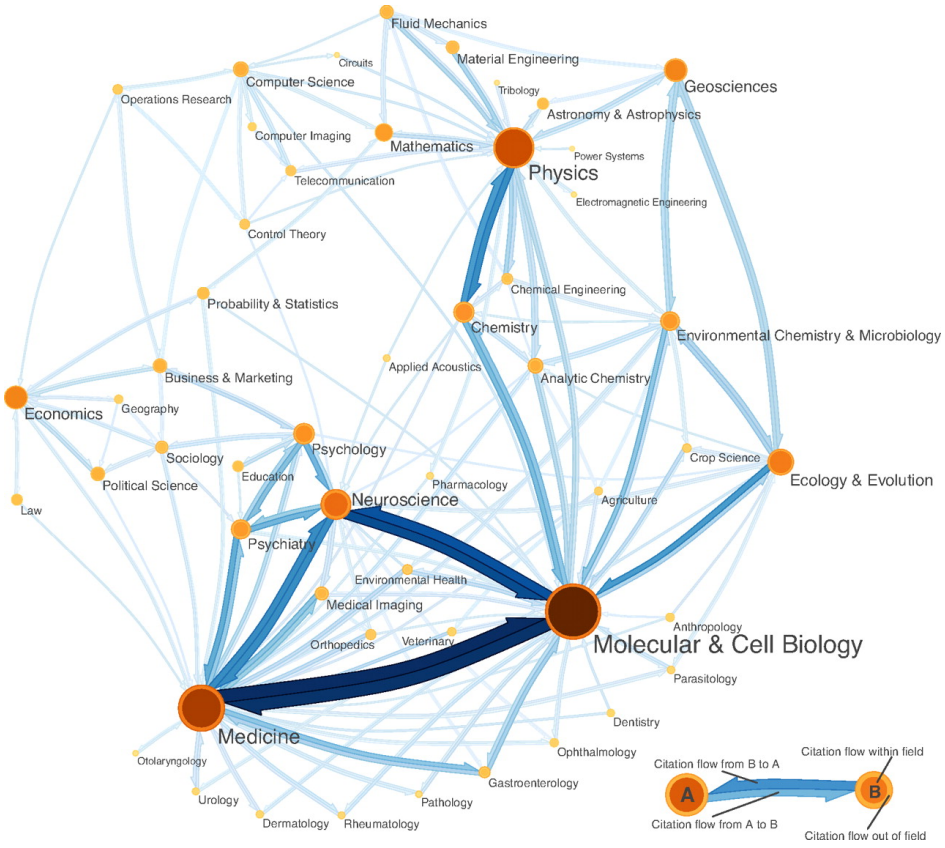

Figure 5: A map of science based on a journal citation network of articles published in 2004, and citations to articles published in the previous 5 years.. Nodes represent communities found using the Infomap algorithm; they are hand-labeled according to the research topic of the journals they contain. The size of the nodes reflects the fraction of time a random walker spends within each community, and directed and weighted links between nodes represent citation flow between communities. We can see a "U" shaped pattern, with social sciences and engineering at opposite ends and medicine, molecular biology, and physics forming the bridge. Figure from [49]

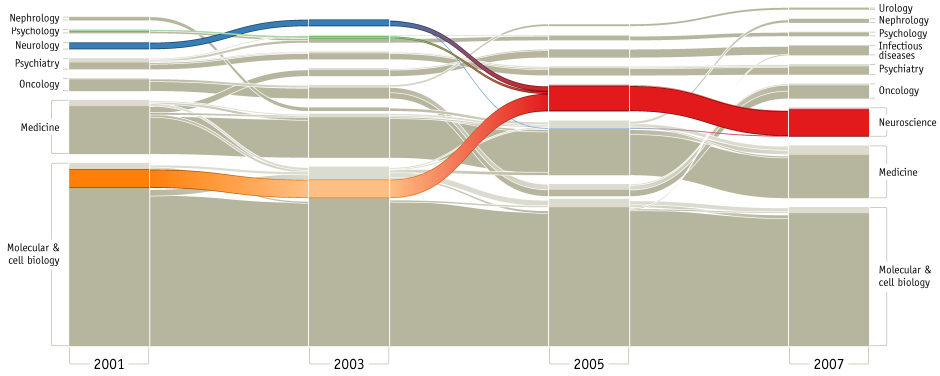

Figure 6: Alluvial diagram of scientific disciplines changing over time, identified by detecting communities in the journal citation network using Infomap. All journals that appear as neuroscience by 2007 are highlighted to show how that field emerged from parts of other disciplines. Figure from [48].

grouped the neurons of *C. elegans* using flow-based methods. These detected communities showed good agreement with previous understanding of functional neuronal groups. They were then able to perform *in silico* ablations of neurons—computer simulations in which they removed nodes and looked at the resulting disruption on community structure. By doing this, they identified neurons important to the network flow, and presumably important to the neuronal function of the organism. These included neurons known to be important, as well as previously uninvestigated neurons that can be candidates for future study.

Other examples in the research literature

Zhang et al. [70] analyzed Congressional cosponsorship networks in the United States Congress in order to study partisan polarization. They created a two-mode (bipartite) network between legislators and legislation, and then projected this network to a one-mode (unipartite) network in which nodes are legislators, and links between legislators represent the fact that they have cosponsored bills together. They then compared the optimal modularity of these networks at different time points with the observed modularity using party affiliation as community membership. In other words, they were interested in how close the structural communities seen in the network were to the "communities" of political party. They found that the difference between these two values decreased over time (1979–2004), indicating that the cosponsorship behavior of legislators was increasingly aligning with party affiliation—i.e., that polarization was increasing.

Lupu and Traag [33] used community detection to study the impact of trade networks among nations on the potential for conflict between nations. States that trade with each other are known to have less conflict with each other. The authors argue for extending this to trade communities. Two states may also see a reduced risk of conflict even if they do not have a strong trade relationship, if they are indirectly linked by trade. The authors test this theory by applying the Louvain modularity-optimizing community detection algorithm on a network of trade flows between states. They find that states in the same trade community are significantly less likely to experience conflict with each other, even controlling for direct trade relationships.

Non-research applications

Besides the published examples of applications of community detection, there are likely many that have not been published, either due to secrecy around industry or state knowledge, or simply because the people involved are not oriented toward academic publishing. One such area is marketing: community information on various types of consumer networks could be useful for targeted marketing [27]. This is also related to recommendation systems, as discussed in the context of scholarly publications above; graph clustering approaches to recommendation have appeared in the published literature [46, 16], and it is probable that more exists unpublished. Another likely use for community detection is in the identification of criminal or terrorist organizations, and while the potential for this is evident in the literature [26, 37, 4, 6, 63, 11], it is likely that the successes seen from following such approaches would not be made public.

Finally, it is possible that community detection methods have become another part of the data analysis toolkit, in a way that its direct applications may often be part of the routine practice of data scientists (a term not reserved for academic scientists) and not published as research. As an example, I direct the reader to the section "Visualizing community structure in networks" below, in which I use community detection to help

answer a question of interest: Which research labs are actively working on research related to visualizing group structure in networks?

Visualizing community structure in networks

Applying community analysis and visualization to identify visualization researchers

A recent paper in the EuroVis conference by Vehlow et al. [58] gives a thorough overview of the state of the art in visualizing group structure in networks. In addition to giving a literature survey as well as a taxonomy for these visualizations, they provide a detailed curated bibliography at http://groups-in-graphs.corinna-vehlow.com/. There one finds a visualization tool for exploring the surveyed literature on this topic, and also the full tagged bibliography available for download as a BibTeX file.

One question was somewhat difficult to answer given the format this bibliography was presented: What are the labs doing work in this area? While the bibliography presented important papers related to the topic, they were not organized by lab. In order to address this question, I perform a community analysis and visualization on the bibliography data provided. The visualization is available at http://students.washington.edu/jporteno/groups/. I also use this endeavor as a first-hand illustration of the utility and challenges around detecting and visualizing communities in network data.

Starting with data on papers and seeking insight into the organization of research in different labs seemed like an appropriate situation to apply community detection methods. I first constructed a co-authorship network in which nodes represent authors, and weighted links represent the number of papers in which a pair appear as authors. In addition, I used the Microsoft Academic API[3] to attempt to assign an affiliation to each author by querying Microsoft Academic Graph for the author and retrieving the most prevalent affiliation for that author among the papers returned. I then used Infomap to find a hierarchical clustering of the nodes. To visualize the results (Fig. 7), I include only the connected components with at least 10 nodes. Each node represents a single author; 238 authors appear in the visualization. The sizes of the nodes correspond to the amount of flow—the relative importance of an author in this co-authorship network. The color of each node is assigned based on the top-level cluster assignment of that author. Clicking on any node of a connected component will shift the focus to that component, and reassign colors, this time based on the bottom-level cluster assignment. I also provide a search box which makes it easier to find specific authors or affiliations, and a pop up box on mouseover with details about the author and titles of papers. Note that there may be some errors due to using author names as unique IDs.

I represent the co-authorship network using D3 [3] as a node-link diagram, which is a very common way of visualizing networks. The nodes are visualized using a force-directed algorithm, in which forces of attraction and repulsion are applied to nodes and links and then placed so as to minimize the energy. Interestingly, the force-directed layout has been shown to be equivalent to the commonly used modularity quality function for communities [41]—this is why a visualization of a network that uses this layout can show implicit community structure by grouping similar nodes together visually.

Labs involved in visualizing group structure in networks

Using this system can help to identify some of the key researchers in this area and the relationships between them. To find the labs, I can search the web for their lab or university websites. By doing this, I was able to

[3] https://docs.microsoft.com/en-us/azure/cognitive-services/academic-knowledge/home

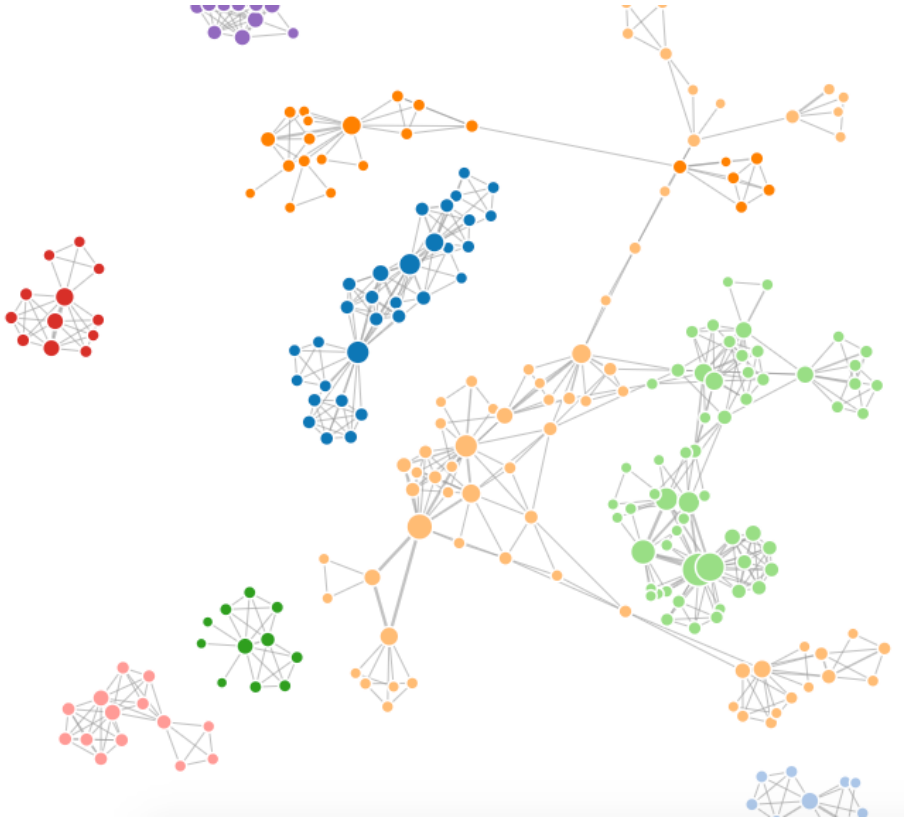

Figure 7: Coauthorship network of the visualization methods papers surveyed in [58]. Nodes represent authors, with weighted links indicating co-authorship. Nodes in this view are colored by the top level of the hierarchical Infomap community assignment, and roughly correspond to labs and institutions. Available at http://students.washington.edu/jporteno/groups/

find a number of labs doing interesting work.

One community that pops out appears in the middle of the largest connected component, with several large and well-connected nodes. Perhaps unsurprisingly, this community represents the authors of the survey paper (and the curators of the bibliography data behind the visualization). This is the University of Stuttgart Visualisation Research Centre (VISUS), and includes Daniel Weiskopf, Fabien Beck and others. It also used to include then-PhD student and lead author of the survey paper Corinna Vehlow. An interesting recent paper from this lab is [59], in which dynamically evolving community structure is visualized as rectangular blocks (in a way resembling the alluvial diagrams in Rosvall and Bergstrom [48] see Fig. 6), and these are combined with node-link representations of the community subgraphs. Fig. 8 shows the evolution of a network of soccer teams over time.

Close to the Stuttgart lab is the team at LaBRI at the University of Bordeaux, France. Researchers here include Romain Bourqui and David Auber. Dr. Auber appears to work closely with researchers at the University of British Columbia including Tamara Munzner and Daniel Archambault. An interesting paper from the LaBRI group is [53], where one technique they use to show hierarchical structure involves assigning similar color to similar nested groups.

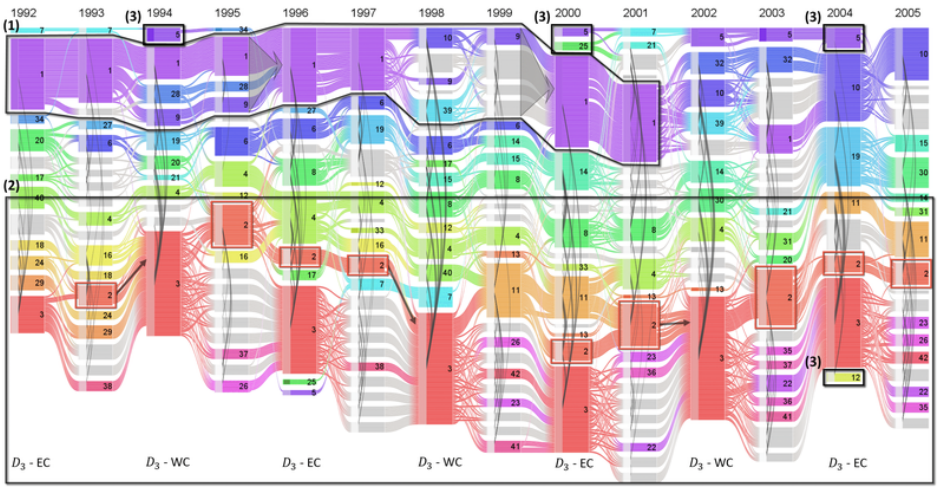

Figure 8: Evolution of the community structure of a network of soccer matches from 1992 to 2005 (teams are connected if they played each other) From Vehlow et al. [59].

Other groups can be seen in the largest connected component. A group of researchers appears as a maroon community; this is the MArVL group at Monash University in Melbourne, Australia, which includes Kim Marriott and Tim Dwyer (I have seen Dr. Dwyer's work before—he created cola.js, a constraint-based graph visualization layout that works with D3). The UC Davis Center for Visualization appears as a mostly blue community. The University of Arizona Graph and Map Algorithm (GAMA) Lab, headed by Stephen Kobourov, is well connected in the component. An interesting paper out of this last lab is [18], in which communities are visualized to look like cartographic maps. This method, which the authors call GMap has a pleasing aesthetic that people might find preferable to the standard node-link diagram, as many people are turned off by the "hairball" that tends to result from the standard approach. Indeed, the authors use their mapping technique to visualize a co-authorship network (Fig. 9), and this may be a superior way to visualize this data than what I have done. Another paper [52] found that the cartographic map technique improved people's recall of data.

Some labs are represented in the smaller components as well. One such component contains two labs in China—the Tong Ji Intelligent Big Data Visualization Lab, headed by Nan Cao; and the VisLab at Hong Kong University, headed by Huamin Qu. One paper from the VisLab is [67], in which contour overlays using Voronoi cells are superimposed over nodes to show communities.

Visualizing group structure in graphs: State of the art and open problems

The review article ([58]) proposes a taxonomy of methods for visualizing group structure in graphs. This taxonomy characterizes techniques along two axes. The group structure taxonomy identifies the type of groups being visualized—disjoint flat groups, overlapping flat groups, disjoint hierarchical groups, or overlapping hierarchical groups. The group visualization taxonomy characterizes the way that the group structure is presented in the visualization—node attributes (such as color), juxtaposed views, superimposed views, or embedded views. The authors organize all of the papers in their review according to this taxonomy and

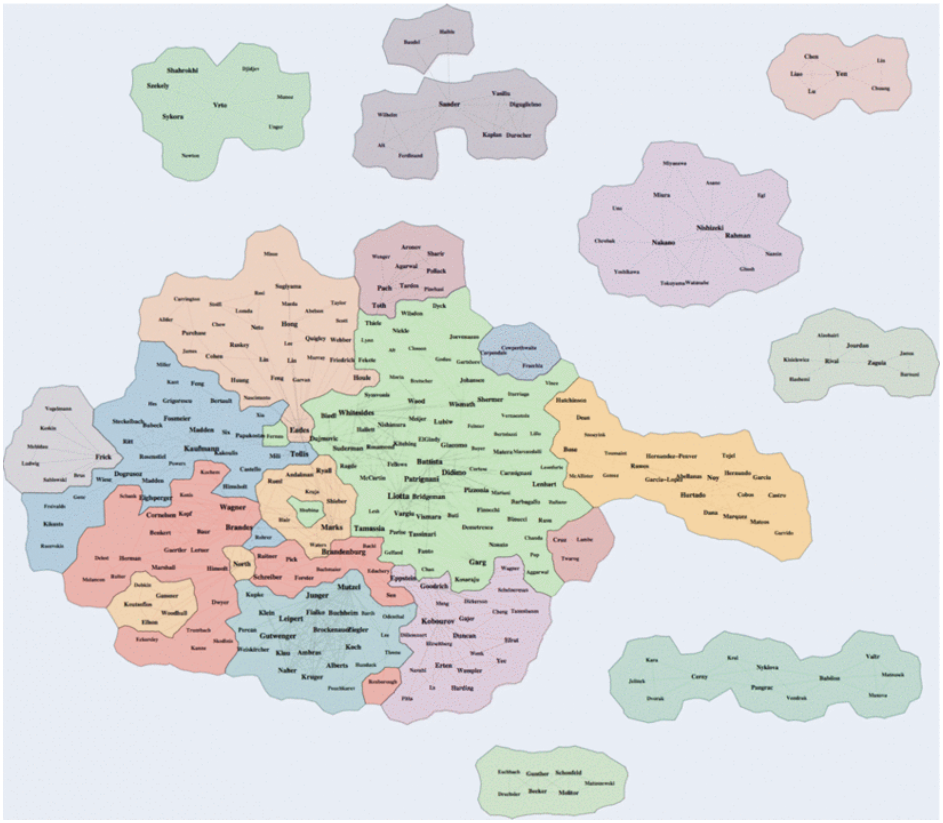

Figure 9: GMap [18] representation of an author collaboration network among visualization ressearchers.

include this information in their curated bibliography. This taxonomy is useful for navigating the space of visualizing communities, and can help find examples of techniques that have been used.

Vehlow et al. also include a discussion of open research challenges in visualizing group structure in graphs, informed by interviews with experts in the field. They identify five main areas with work to do:

1. Representating *time-varying* group structure, especially the simultaneous representation of group evolution and graph topology. Similarly, we need ways of *comparing* sets of graphs.
2. More *data complexity*—for instance, visualizing fuzzy overlapping groups, in which any node can be assigned to multiple communities, possibly with different probabilities.
3. *Scalable approaches*—for example, representing community assignment by coloring the nodes becomes difficult for more than seven groups.
4. *Interactive visualizations* provide many possibilities, and there already exist many examples, but much of this space remains to be explored.
5. The question of how to best *evaluate* visualization techniques is still open, including the identification of *tasks* that these visualizations can support.

Confidence in communities

Statistical significance

Although it is counterintuitive, completely random graphs, when viewed a certain way, can be seen to have community structure. In a random graph, known as an Erdős–Rényi graph [10], every node has an equal probability of being linked to every other node. Fig. 10 shows how a random graph can show group structure. The top left shows the adjacency matrix of a random graph with nodes in arbitrary order; this looks like random white noise. By rearranging the order of the nodes in the same random graph, as in the other panes of the figure, a community structure becomes apparent. This is not actual community structure we are interested in, but rather artifacts of the randomness. Nevertheless, community detection methods can pick up on them and find artificial structure in what should be random. These effects are even more pronounced for sparse graphs, a category which includes most real-world networks we would like to analyze [15].

This raises the question: how can we be confident that the communities detected by an algorithm reflect actual community structure, and not an artifact of random noise? This is an open problem in network science, but several attempts have been made to address it.

One approach is to compare the results to an appropriate null model. The most popular null model has been the configuration model, which can generate any configuration of a network given a number of nodes, number of edges, and degree assignment for every node. One can use an ensemble of these networks and compare them against empirical results. Any measure of the graph, for example a centrality measure of a given node, can be compared with the same measure on a number of generated null models; a p-value can then be calculated as the fraction of model configurations that yield the same value for the measure as the observed network [15]. (It may be difficult to operationalize "same value" here—requiring an exact match might make this test too insensitive, but allowing for variation would introduce more noise.) One could apply this approach to the community structure as a whole, or to individual communities, or to community membership of individual nodes, measuring these phenomena against their presence in an ensemble of generated graphs.

Another approach is to consider the robustness of a given network to perturbations in the network structure. From a conceptual standpoint, this approach differs from the above one in that it does not consider the network structure as having been generated from a stochastic process, and attempt to model that process. Rather, it takes the structure as given and observes the effects of random noise introduced into this structure on the communities that are detected. This approach has the advantages that it is applicable to any method, and it is not restricted to a particular null model [48, 36].[4]

Several strategies for perturbing networks have been proposed. Gfeller et al. [19] employ a strategy on weighted graphs in which edge weights are increased or decreased by a relative amount $0 < \delta < 1$. After choosing δ, multiple realizations of the graph are clustered using any method, and the *in-cluster* probability of each pair of adjacent vertices p_{ij} is calculated as the fraction of realizations in which they were clustered together. Edges with $p_{ij} < \theta$ are considered to be *external* edges. This approach has as a weakness the need to choose values for δ and θ. The authors also provide a way to measure the overall stability of the partition using these in-cluster probabilities; this measure needs to be compared against a null model.

[4] In practice, however, a particular null model is often implied in a perturbation strategy, as can be seen in the discussed examples.

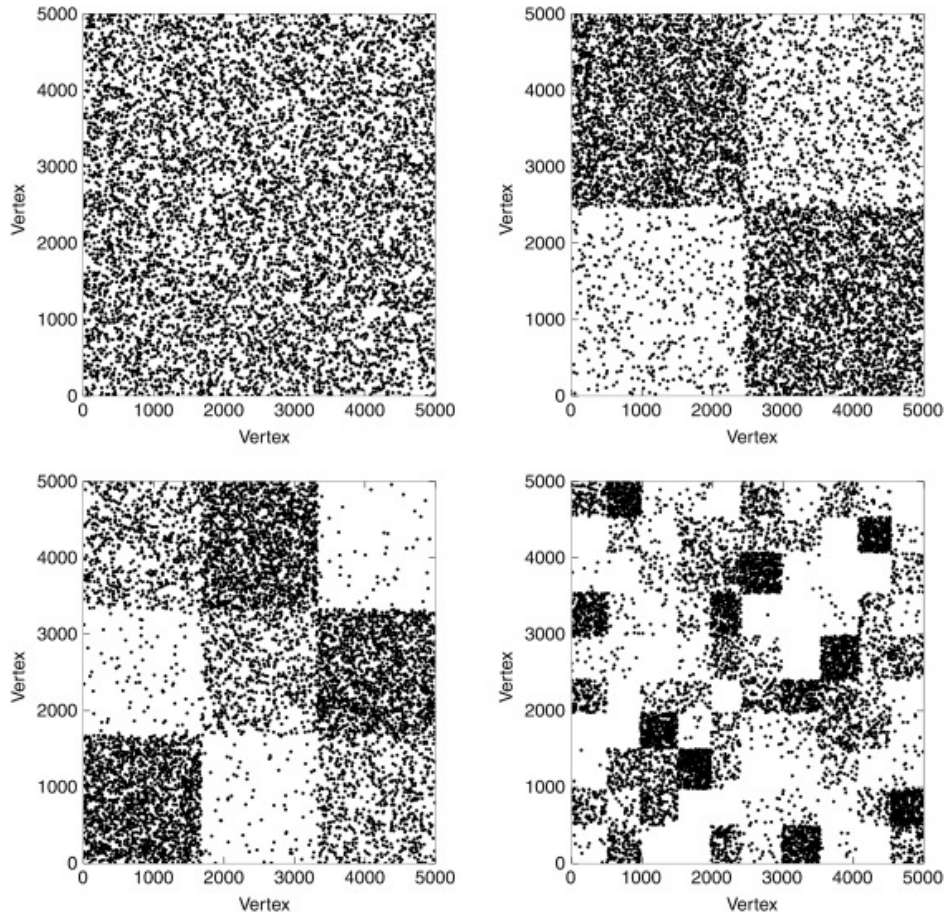

Figure 10: Artificial community structure in a random graph. Each of the four graphs represent the adjacency matrix of the same 5000-node random graph, in which every pair of nodes has equal probility of sharing a link. The top right matrix shows the nodes in arbitrary order, and the impression is one of random white noise. The others are simply rearrangements of the nodes for the same random graph. A community structure can be seen from what is actually random fluctuations in the network construction process. Figure from [15].

Karrer et al. [24] use a strategy for unweighted graphs of network rewiring. Each edge is considered and either left alone or, with probability α, it is removed and replaced with another edge between a pair of vertices (i, j) chosen at random with probability $p_{ij} = k_i k_j / 2m$ (k_i and k_j are the degrees of i and j; m is the number of edges in the graph). This probability is the same one used in the null model of modularity. The authors generated multiple graphs for different values of α, and compared their clusterings to each other using the variation of information V. They then plotted V against α to see how different levels of perturbation would affect the stability of the network structure. They compared the function $V(\alpha)$ against a null model—again, the null model of modularity.

Rosvall and Bergstrom use a parametric bootstrap resampling method to measure significance of communities. To resample the graph, each edge is assigned a weight from a Poisson distribution with mean equal to the original edge weight. The authors then cluster both the original and resampled networks (any clustering method can be used; the authors used Infomap). Each cluster of the original graph is considered,

and of these nodes the largest subset clustered in the same group in at least 95% of the resampled networks is deemed a significant cluster. By applying this technique on citation graphs at different time points, the authors track the change in the community structure over time (see subsection "Networks of scholarship" in section "Applications of community detection")

Mirshahvalad et al. [36] propose a *constructive* perturbation approach suitable for sparse networks. These networks have a danger that communities can be "shattered" into small modules if there are missing links due to noise in the data. *Link prediction* is employed to add links to the network, strengthening the communities. The link prediction strategy used is triangle completion, in which a fraction of open triangles are completed by adding a link. This is a relatively simple strategy, and more sophisticated methods can be used, but the authors show that it performs well on benchmark graphs that have been shattered. This method can be used as a bootstrap resampling technique, and significance can be measured based on the resampled distribution. A weakness of this method is that the number of links to be added needs to be chosen, but the benchmark experiments suggest that results are robust to this choice.

In summary, measuring our confidence in community detection techniques seems to be a hard problem. Network data are highly interrelated, so typical independence assumptions that underlie many statistical methods do not apply. One important issue is the lack of a realistic null model. Typical null models assume that any node can be connected to any other with equal probability, but this assumption does not intuitively hold for large networks. More appropriate would be to have a "horizon" of nodes that each node is more likely to interact with, but such horizons have yet to be defined [13]. The lack of appropriate null models is a problem for generative models that rely on them explicitly, but they tend to underlie assumptions behind link perturbation and resampling strategies as well. More work remains to be done to arrive at standard ways of measuring our confidence in the results of community detection.

Algorithmic inaccuracy

Our confidence in the performance of a given algorithm due to, for instance, using different random seeds is a distinct issue. The statistical significance of community detection is a question of confidence in the *theory* behind our method. Variation in the results due to the fact that we are necessarily using approximating algorithms (due to the complexity of the problem), on the other hand, affects our confidence in the *implementation* of our method. In the case of Infomap, during each pass of the core algorithm, the nodes are considered one at a time in random order, and merged with one of its neighbors in order to give maximum improvement to the objective function (the map equation). In principle, there should be one global minimum for the map equation, corresponding to the one optimal community structure. In practice, it does seem like the order that the nodes are considered does make a difference in the final value of the map equation—i.e., how well Infomap is able to approximate this global optimum. For example, in a recent experiment in which I performed 106 runs of Infomap on the JSTOR article citation network, I found an approximately normal distribution of final values between 11.334 and 11.364 (mean 11.348, standard dev 0.00577). These numbers are of course difficult to interpret, but they do show that there is at least some variation.

The Infomap algorithm is based on the Louvain method for modularity maximization. In the paper introducing the Louvain method, Blondel el al. report on this issue. While they see the same type of variation

in relation to the random order, they claim the effect is small, and they are more concerned with the effect the order has on computation time. They leave further understanding of the issue to future study [2].

In light of this, it would not be a bad idea for someone using any approximating method to perform multiple runs if she is interested in finding the best solution possible. More work is needed to understand specifically the effect of the random seed on the performance of Infomap.

Infomap implementation in Python

The code I am submitting is my work on a Python implementation of Infomap: https://github.com/h1-the-swan/pyinfomap

As far as I know, there does not yet exist a pure Python implementation of Infomap, the algorithm to detect communities in network data by minimizing the map equation. The Infomap software, available at http://www.mapequation.org/code.html, is written in C++, although it has extensions in other languages including Python. A Python repository was created by Daniel Halperin in 2013.[5] This code, deemed version 1 by Dr. Halperin, is capable of calculating the map equation for a given network and a given two-level clustering of its nodes.[6] My code is a fork of this repository: I coded the algorithm starting with this base code that can calculate the function to optimize.

The map equation is (see section "The dynamical perspective" for background):

$$L(\mathsf{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{P}^i)$$

where M is the module partitioning; the left term is the average length of codewords (entropy) in the index codebook weighted by the rate of use of the index codebook $q_{\curvearrowright}$; and the right term is the average length of codewords in module codebook i weighted by the rate of use of this module $p_{\circlearrowright}$. Using $q_{\curvearrowright} = \sum_{i=1}^{m} q_{i\curvearrowright}$; $p^i_{\circlearrowright} = \sum_{\alpha \in i} p_\alpha + q_{i\curvearrowright}$ where $\alpha \in i$ means every node in module i (the $q_{i\curvearrowright}$ is added as the probability that the random walker exits the module and the exit codeword is used); and the definition of entropy[7], the map equation can be expanded to:

$$\begin{aligned} L(\mathsf{M}) = &\left(\sum_{i=1}^{m} q_{\curvearrowright} \right) \log \left(\sum_{i=1}^{m} q_{\curvearrowright} \right) - 2 \sum_{i=1}^{m} q_{\curvearrowright} \log(q_{\curvearrowright}) \\ &- \sum_{\alpha=1}^{n} p_\alpha \log(p_\alpha) + \sum_{i=1}^{m} \left(q_{\curvearrowright} + \sum_{\alpha \in i} p_\alpha \right) \log \left(q_{\curvearrowright} + \sum_{\alpha \in i} p_\alpha \right) \end{aligned}$$

The node visit probability p_α is related to the dynamics being modeled. Dr. Halperin's code uses PageRank with teleportation probability $\tau = 0.15$. This is modeling a random walker on the network that has a 15% chance, on every step, of teleporting to a random node instead of following a link as normal. The code uses the Python package NetworkX [22] to handle graph storage and operations; this package has its own method for calculating PageRank for every node. The exit probability $q_{i\curvearrowright}$ is then calculated as [47]:

$$q_{i\curvearrowright} = \tau \frac{n - n_i}{n} \sum_{\alpha \in i} p_\alpha + (1 - \tau) \sum_{\alpha \in i} \sum_{\beta \notin i} p_\alpha w_{\alpha\beta}$$

where n_i is the number of nodes in module i, and $w_{\alpha\beta}$ is the normalized weight of the link from α to β (if α is a dangling node, this weight is replaced by $1 - n_i/n$).

[5] https://github.com/uwescience/pyinfomap

[6] By "two-level clustering", I mean a non-hierarchical, non-overlapping partitioning of each node into exactly one of any number of clusters.

[7] For a random variable X that can have n states with probability p_i, the entropy is $H(X) = -\sum_{i=1}^{n} p_i \log p_i$.

Version 1 implements a `Module` class and a `Clustering` class. I build a `PyInfomap` on top of these in order to be able to calculate the map equation for different clusterings of an input graph. What follows is my work implementing an optimization algorithm.

The test example I use is network (a) in Fig. 1 in this document (in the original work [47] it is fig. 3). A pajek version of this network was included in the forked repository as `2009_figure3ab.net`. We know from the text [47] that the clustering seen in the figure should have a value for the map equation $L = 3.33$, and that this is the optimal (minimum) value for this network. Thus, my algorithm should be able to find this clustering.

One approach to find the clustering would be to calculate the map equation for every possible partition of the network. I implement this in `search_all_possible_partitions.py`. However, there are far too many combinations of partitions for network data, even for the small 16-node network I use to test. As of this writing, this code has not yet finished after running for 212 hours (almost 9 days), having tried over 6.15×10^9 different partitions.

A more realistic option is to use some sort of search algorithm. I implement the Louvain method, which was originally developed as a modularity-optimizing community detection algorithm [2], and which is the starting point for the official version of Infomap [47]. This method consists of two phases, which are performed repeatedly. It starts with every node in its own separate module. In the first phase, each node, one at a time in random order, is moved to be in the same module as its neighbor that gives the lowest value of the map equation. This is repeated until no improvement can be made. In the second phase, a new network is constructed in which the nodes represent the clusters found in the previous step, edges are weighted by the number of between-cluster edges in the original graph, and within cluster edges become self-loops (i.e., edges from one node in the new network to itself). The algorithm is then repeated on this new graph, with each calculation of the map equation done using the original graph. Phase one and two are repeated until no improvement is seen. Fig. 11 shows a schematic of this procedure.

The official Infomap algorithm takes additional steps after this point to broaden the search space and avoid local optima. These include recursively running the algorithm on the clusters, and freeing individual nodes to move between modules. My implementation does not yet include this.

My implementation can find the optimal four-way partition of the test network. It can also find the optimal partition for network (d) in the same figure, in which all nodes are grouped into the same module. It runs in under 1 second on these test networks (after loading the needed libraries). It is able to run the larger karate club network (34 nodes) in under 5 seconds.

Next steps would include tests on synthetic benchmarks, and further analysis of the results of the karate club network, as well as other standard examples (see section "Evaluation of community detection methods").

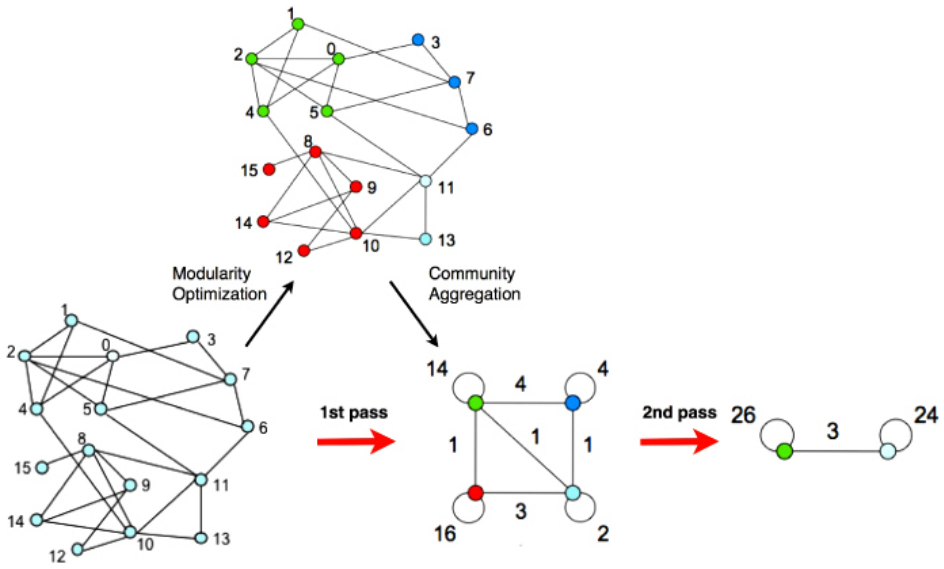

Figure 11: Schematic of the Louvain algorithm from [2]. Each pass consists of a phase in which local merges of communities are made to optimize the objective function, and a phase in which a new meta-graph is constructed with the communities as nodes. The two phases are repeated until no improvement is found.

References

[1] Karol A. Bacik et al. “Flow-Based Network Analysis of the Caenorhabditis elegans Connectome.” In: *PLOS Computational Biology* 12.8 (Aug. 5, 2016), e1005055. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005055.

[2] Vincent D Blondel et al. “Fast unfolding of communities in large networks.” In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 9, 2008), P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/P10008.

[3] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. “D3: Data-Driven Documents.” In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011).

[4] Carlos André and Reis Pinheiro. “Community detection to identify fraud events in telecommunications networks.” In: *SAS SUGI Proceedings: Customer Intelligence*. SAS Global Forum 2012. 2012.

[5] Jingchun Chen and Bo Yuan. “Detecting functional modules in the yeast protein–protein interaction network.” In: *Bioinformatics* 22.18 (Sept. 15, 2006), pp. 2283–2290. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btl370.

[6] Aaron Clauset and Kristian Skrede Gleditsch. “The Developmental Dynamics of Terrorist Organizations.” In: *PLOS ONE* 7.11 (Nov. 21, 2012), e48633. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0048633.

[7] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[8] Ruth Dunn, Frank Dudbridge, and Christopher M. Sanderson. “The Use of Edge-Betweenness Clustering to Investigate Biological Function in Protein Interaction Networks.” In: *BMC Bioinformatics* 6.1 (Mar. 1, 2005), p. 39. ISSN: 1471-2105. DOI: 10.1186/1471-2105-6-39.

[9] Scott Emmons et al. “Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale.” In: *PLOS ONE* 11.7 (July 8, 2016), e0159161. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0159161.

[10] Paul Erdős and Alfréd Rényi. “On the evolution of random graphs.” In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960), pp. 17–60.

[11] Emilio Ferrara et al. “Detecting criminal organizations in mobile phone networks.” In: *Expert Systems with Applications* 41.13 (Oct. 2014), pp. 5733–5750. ISSN: 09574174. DOI: 10.1016/j.eswa.2014.03.024. arXiv: 1404.1295.

[12] Miroslav Fiedler. “Algebraic connectivity of graphs.” In: *Czechoslovak mathematical journal* 23.2 (1973), pp. 298–305.

[13] Santo Fortunato. “Community detection in graphs.” In: *Physics Reports* 486.3 (Feb. 2010), pp. 75–174. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2009.11.002.

[14] Santo Fortunato and Marc Barthélemy. “Resolution limit in community detection.” In: *Proceedings of the National Academy of Sciences* 104.1 (Jan. 2, 2007), pp. 36–41. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0605965104.

[15] Santo Fortunato and Darko Hric. “Community detection in networks: A user guide.” In: *Physics Reports*. Community detection in networks: A user guide 659 (Nov. 11, 2016), pp. 1–44. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2016.09.002.

[16] F. Fouss et al. “Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation.” In: *IEEE Transactions on Knowledge and Data Engineering* 19.3 (Mar. 2007), pp. 355–369. ISSN: 1041-4347. DOI: 10.1109/TKDE.2007.46.

[17] E. B. Fowlkes and C. L. Mallows. “A Method for Comparing Two Hierarchical Clusterings.” In: *Journal of the American Statistical Association* 78.383 (1983), pp. 553–569. DOI: 10.1080/01621459.1983.10478008.

[18] Emden R. Gansner, Yifan Hu, and Stephen G. Kobourov. “GMap: Visualizing graphs and clusters as maps.” In: *Proceedings of the IEEE Pacific Visualization Symposium*. PacificVis ’10. IEEE, 2010, pp. 201–208. DOI: 10.1109/PACIFICVIS.2010.5429590.

[19] David Gfeller. “Finding instabilities in the community structure of complex networks.” In: *Physical Review E* 72.5 (2005). DOI: 10.1103/PhysRevE.72.056135.

[20] M. Girvan and M. E. J. Newman. “Community structure in social and biological networks.” In: *Proceedings of the National Academy of Sciences of the United States of America* 99.12 (June 11, 2002), p. 7821. DOI: 10.1073/pnas.122653799.

[21] Roger Guimerà and Luís A. Nunes Amaral. “Functional cartography of complex metabolic networks.” In: *Nature* 433.7028 (Feb. 24, 2005), pp. 895–900. ISSN: 0028-0836. DOI: 10.1038/nature03288.

[22] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX.” In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[23] Darko Hric, Richard K. Darst, and Santo Fortunato. “Community detection in networks: Structural communities versus ground truth.” In: *Physical Review E* 90.6 (Dec. 9, 2014), p. 062805. DOI: 10.1103/PhysRevE.90.062805.

[24] Brian Karrer. “Robustness of community structure in networks.” In: *Physical Review E* 77.4 (2008). DOI: 10.1103/PhysRevE.77.046119.

[25] B. W. Kernighan and S. Lin. “An efficient heuristic procedure for partitioning graphs.” In: *The Bell System Technical Journal* 49.2 (Feb. 1970), pp. 291–307. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1970.tb01770.x.

[26] Valdis E. Krebs. “Mapping networks of terrorist cells.” In: *Connections* 24.3 (2002), pp. 43–52.

[27] Vincent Labatut and Jean-Michel Balasque. “Detection and interpretation of communities in complex networks: Practical methods and application.” In: *Computational Social Networks*. Springer, 2012, pp. 81–113.

[28] R. Lambiotte and M. Rosvall. “Ranking and clustering of nodes in networks with smart teleportation.” In: *Physical Review E* 85.5 (May 8, 2012). ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.85.056107.

[29] Andrea Lancichinetti. “Benchmark graphs for testing community detection algorithms.” In: *Physical Review E* 78.4 (2008). DOI: 10.1103/PhysRevE.78.046110.

[30] Andrea Lancichinetti and Santo Fortunato. “Community detection algorithms: A comparative analysis.” In: *Physical Review E* 80.5 (Nov. 30, 2009). ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.80.056117.

[31] Andrea Lancichinetti, Santo Fortunato, and János Kertész. “Detecting the overlapping and hierarchical community structure in complex networks.” In: *New Journal of Physics* 11.3 (2009), p. 033015. ISSN: 1367-2630. DOI: 10.1088/1367-2630/11/3/033015.

[32] Jure Leskovec, Kevin J. Lang, and Michael Mahoney. “Empirical Comparison of Algorithms for Network Community Detection.” In: *Proceedings of the 19th International Conference on World Wide Web*. WWW ’10. New York, NY, USA: ACM, 2010, pp. 631–640. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772755.

[33] Yonatan Lupu and Vincent A. Traag. “Trading Communities, the Networked Structure of International Relations, and the Kantian Peace.” In: *Journal of Conflict Resolution* 57.6 (Dec. 1, 2013), pp. 1011–1042. ISSN: 0022-0027. DOI: 10.1177/0022002712453708.

[34] David Lusseau. “The emergent properties of a dolphin social network.” In: *Proceedings of the Royal Society of London B: Biological Sciences* 270 (Suppl 2 2003), S186–S188. ISSN: 0962-8452. DOI: 10.1098/rsbl.2003.0057.

[35] Marina Meilă. “Comparing clusterings—an information based distance.” In: *Journal of Multivariate Analysis* 98.5 (May 1, 2007), pp. 873–895. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2006.11.013.

[36] Atieh Mirshahvalad et al. “Significant Communities in Large Sparse Networks.” In: *PLOS ONE* 7.3 (Mar. 30, 2012), e33721. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0033721.

[37] S. V. Nath. “Crime Pattern Detection Using Data Mining.” In: *2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops*. 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops. Dec. 2006, pp. 41–44. DOI: 10.1109/WI-IATW.2006.55.

[38] M. E. J. Newman. “Fast algorithm for detecting community structure in networks.” In: *Physical Review E* 69.6 (June 18, 2004), p. 066133. DOI: 10.1103/PhysRevE.69.066133.

[39] M. E. J. Newman. “Finding and evaluating community structure in networks.” In: *Physical Review E* 69.2 (2004). DOI: 10.1103/PhysRevE.69.026113.

[40] M. E. J. Newman. “Modularity and community structure in networks.” In: *Proceedings of the National Academy of Sciences* 103.23 (June 6, 2006), pp. 8577–8582. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0601602103.

[41] Andreas Noack. “Modularity clustering is force-directed layout.” In: *Physical Review E* 79.2 (2009). DOI: 10.1103/PhysRevE.79.026102.

[42] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web.* Stanford InfoLab, 1999.

[43] Leto Peel, Daniel B. Larremore, and Aaron Clauset. “The ground truth about metadata and community detection in networks.” In: *Science Advances* 3.5 (May 1, 2017), e1602548. ISSN: 2375-2548. DOI: 10.1126/sciadv.1602548.

[44] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. “Communities in networks.” In: *Notices of the AMS* 56.9 (2009), pp. 1082–1097.

[45] William M. Rand. “Objective Criteria for the Evaluation of Clustering Methods.” In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. DOI: 10.1080/01621459.1971.10482356.

[46] P. Krishna Reddy et al. “A Graph Based Approach to Extract a Neighborhood Customer Community for Collaborative Filtering.” In: *Proceedings of the Second International Workshop on Databases in Networked Information Systems*. DNIS ’02. London, UK, UK: Springer-Verlag, 2002, pp. 188–200. ISBN: 978-3-540-00264-2.

[47] M. Rosvall, D. Axelsson, and C. T. Bergstrom. “The map equation.” In: *The European Physical Journal Special Topics* 178.1 (Apr. 17, 2010), pp. 13–23. ISSN: 1951-6355, 1951-6401. DOI: 10.1140/epjst/e2010-01179-1.

[48] Martin Rosvall and Carl T. Bergstrom. “Mapping Change in Large Networks.” In: *PLoS ONE* 5.1 (Jan. 27, 2010). Ed. by Fabio Rapallo, e8694. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0008694.

[49] Martin Rosvall and Carl T. Bergstrom. “Maps of random walks on complex networks reveal community structure.” In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123.

[50] Martin Rosvall and Carl T. Bergstrom. “Multilevel Compression of Random Walks on Networks Reveals Hierarchical Organization in Large Integrated Systems.” In: *PLoS ONE* 6.4 (Apr. 8, 2011). Ed. by Fabio Rapallo, e18209. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0018209.

[51] Martin Rosvall et al. “Memory in network flows and its effects on spreading dynamics and community detection.” In: *Nature Communications* 5 (Aug. 11, 2014), p. 4630. DOI: 10.1038/ncomms5630.

[52] Bahador Saket et al. “Map-based Visualizations Increase Recall Accuracy of Data.” In: *Computer Graphics Forum* 34.3 (2015), pp. 441–450. DOI: 10.1111/cgf.12656.

[53] Joris Sansen et al. “Adjasankey: Visualization of huge hierarchical weighted and directed graphs.” In: *Proceedings of the 19th International Conference on Information Visualisation*. IV ’15. IEEE, 2015, pp. 211–216.

[54] Satu Elisa Schaeffer. “Graph clustering.” In: *Computer Science Review* 1.1 (Aug. 2007), pp. 27–64. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2007.05.001.

[55] Michael T. Schaub et al. “The many facets of community detection in complex networks.” In: *Applied Network Science* 2.1 (Dec. 1, 2017), p. 4. ISSN: 2364-8228. DOI: 10.1007/s41109-017-0023-6.

[56] D. J. de Solla Price. “Networks of Scientific Papers.” In: *Science* 149.3683 (July 30, 1965), pp. 510–515. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.149.3683.510.

[57] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. “Social structure of Facebook networks.” In: *Physica A: Statistical Mechanics and its Applications* 391.16 (Aug. 15, 2012), pp. 4165–4180. ISSN: 0378-4371. DOI: 10.1016/j.physa.2011.12.021.

[58] Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. “The state of the art in visualizing group structures in graphs.” In: *Eurographics Conference on Visualization (EuroVis)-STARs*. Vol. 2. The Eurographics Association, 2015.

[59] Corinna Vehlow et al. “Visualizing the Evolution of Communities in Dynamic Graphs.” In: *Computer Graphics Forum* 34.1 (2015), pp. 277–288. DOI: 10.1111/cgf.12512.

[60] Alcides Viamontes Esquivel and Martin Rosvall. “Compression of Flow Can Reveal Overlapping-Module Organization in Networks.” In: *Physical Review X* 1.2 (Dec. 29, 2011). ISSN: 2160-3308. DOI: 10.1103/PhysRevX.1.021025.

[61] Daril A. Vilhena et al. “Finding Cultural Holes: How Structure and Culture Diverge in Networks of Scholarly Communication.” In: *Sociological Science* 1 (June 9, 2014), pp. 221–239. ISSN: 2330-6696. DOI: 10.15195/v1.a15.

[62] Nguyen Xuan Vinh, Julien Epps, and James Bailey. “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance.” In: *Journal of Machine Learning Research* 11 (Oct 2010), pp. 2837–2854.

[63] Todd Waskiewicz. “Friend of a Friend Influence in Terrorist Social Networks.” In: *Proceedings on the International Conference on Artificial Intelligence (ICAI)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012, p. 1.

[64] Robert S. Weiss and Eugene Jacobson. “A Method for the Analysis of the Structure of Complex Organizations.” In: *American Sociological Review* 20.6 (1955), pp. 661–668. ISSN: 00031224.

[65] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. “Virality Prediction and Community Structure in Social Networks.” In: *Scientific Reports* 3 (Aug. 28, 2013), p. 2522. ISSN: 2045-2322. DOI: 10.1038/srep02522.

[66] J. D. West, I. Wesley-Smith, and C. T. Bergstrom. “A Recommendation System Based on Hierarchical Clustering of an Article-Level Citation Network.” In: *IEEE Transactions on Big Data* 2.2 (June 2016), pp. 113–123. DOI: 10.1109/TBDATA.2016.2541167.

[67] Yanhong Wu et al. “Interactive visual summary of major communities in a large network.” In: *Proceedings of the IEEE Pacific Visualization Symposium*. PacificVis ’15. IEEE, 2015, pp. 47–54. DOI: 10.1109/PACIFICVIS.2015.7156355.

[68] Jaewon Yang and Jure Leskovec. "Defining and evaluating network communities based on ground-truth." In: *Knowledge and Information Systems* 42.1 (Jan. 1, 2015), pp. 181–213. ISSN: 0219-1377, 0219-3116. DOI: 10.1007/s10115-013-0693-z.

[69] Wayne W. Zachary. "An Information Flow Model for Conflict and Fission in Small Groups." In: *Journal of Anthropological Research* 33.4 (1977), pp. 452–473. DOI: 10.1086/jar.33.4.3629752.

[70] Yan Zhang et al. "Community structure in Congressional cosponsorship networks." In: *Physica A: Statistical Mechanics and its Applications* 387.7 (Mar. 1, 2008), pp. 1705–1712. ISSN: 0378-4371. DOI: 10.1016/j.physa.2007.11.004.