Eur. Phys. J. B **38**, 321–330 (2004)
DOI: 10.1140/epjb/e2004-00124-y

THE EUROPEAN PHYSICAL JOURNAL B

Detecting community structure in networks

M.E.J. Newman[a]

Department of Physics and Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI 48109–1120, USA

Received 10 November 2003
Published online 14 May 2004 –

Abstract. There has been considerable recent interest in algorithms for finding communities in networks—groups of vertices within which connections are dense, but between which connections are sparser. Here we review the progress that has been made towards this end. We begin by describing some traditional methods of community detection, such as spectral bisection, the Kernighan–Lin algorithm and hierarchical clustering based on similarity measures. None of these methods, however, is ideal for the types of real-world network data with which current research is concerned, such as Internet and web data and biological and social networks. We describe a number of more recent algorithms that appear to work well with these data, including algorithms based on edge betweenness scores, on counts of short loops in networks and on voltage differences in resistor networks.

PACS. 89.75.Hc Networks and genealogical trees – 87.23.Ge Dynamics of social systems – 89.20.Hh World Wide Web, Internet – 05.10.-a Computational methods in statistical physics and nonlinear dynamics

1 Introduction

In the continuing flurry of research activity within physics and mathematics on the properties of networks, a particular recent focus has been the analysis of communities within networks [1–10]. In the simplest case, a network or graph can be represented as a set of points, or *vertices*, joined in pairs by lines, or *edges*. Many networks, it is found, are inhomogeneous, consisting not of an undifferentiated mass of vertices, but of distinct groups. Within these groups there are many edges between vertices, but between groups there are fewer edges, producing a structure like that sketched in Figure 1.

The ability to find communities within large networks in some automated fashion could be of considerable use. Communities in a web graph for instance might correspond to sets of web sites dealing with related topics [11,12], while communities in a biochemical network or an electronic circuit might correspond to functional units of some kind [4,5,13,14]. In this paper we discuss computer algorithms for the extraction of communities from raw network data.

The outline of the paper is as follows. In Section 2 we describe some of the historical approaches to finding communities including spectral partitioning and hierarchical clustering. Then in Section 3 we describe some newer methods that have appeared in the last few years, including the edge betweenness method of Girvan and Newman and a number of variations on it proposed by other authors. In Section 4 we give our conclusions.

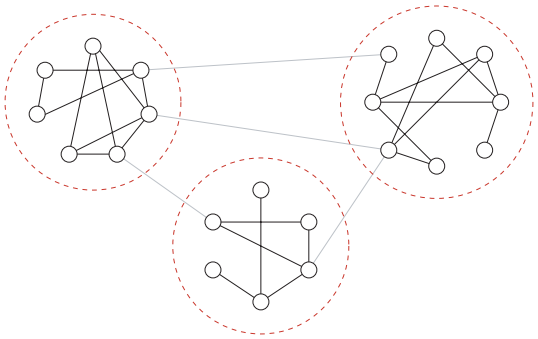

Fig. 1. A small network with community structure of the type considered in this paper. In this case there are three communities, denoted by the dashed circles, which have dense internal links but between which there are only a lower density of external links.

2 Traditional approaches

The methods described in this paper all assume that we are given a network structure that we wish to divide into communities in such a way that every vertex belongs to one of the communities. We assume that the network is of the simplest kind possible, with a single type of undirected, unweighted edge connecting unweighted vertices

[a] e-mail: mejn@umich.edu

of a single type, although generalizations to more sophisticated network types have been given for some of the algorithms described.

The problem of finding good divisions of networks has been studied for some decades now in two fields in particular, computer science and sociology, which have developed quite different approaches as we now describe.

2.1 Computer science approaches

The typical problem in computer science is that of dividing the vertices of a network into some number g of groups with roughly equal size, while minimizing the number of edges that run between vertices in different groups. Computer scientists refer to this task as *graph partitioning*. Graph partitioning problems arise for example in the optimal allocation of processes to processors in a parallel computer. In practice, most approaches to graph partitioning have been based on iterative bisection: we find the best division we can of the complete graph into two groups, and then further subdivide those two until we have the required number of groups. Among the many algorithms suggested for the problem, two have dominated the literature: the spectral bisection method [15,16], which is based on the eigenvectors of the graph Laplacian, and the Kernighan–Lin algorithm [17], which improves on an initial division of the network by optimization of the number of within- and between-community edges using a greedy algorithm.

Spectral bisection: The *Laplacian* of an n-vertex undirected graph G is the $n \times n$ symmetric matrix $\mathbf{L}$ whose diagonal element L_{ii} is the degree of vertex i, and whose off-diagonal element L_{ij} is -1 if vertices i and j are connected by an edge and zero otherwise. Alternatively, one can write $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix of vertex degrees and $\mathbf{A}$ is the adjacency matrix. Since the degree $D_{ii} = \sum_j A_{ij}$, it follows that all rows and columns of the Laplacian sum to zero, and hence that the vector $\mathbf{1} = (1, 1, 1 \ldots)$ is always an eigenvector with eigenvalue zero.

If the network separates perfectly into communities, i.e., divides into g non-overlapping groups of vertices $G_k (k = 1 \ldots g)$ such that there are only within-community edges and no between-community ones—the groups are components of the graph—then the Laplacian will be block diagonal. Each diagonal block will form the Laplacian of its own component, and will therefore also have an eigenvector $\mathbf{v}^{(k)}$ with eigenvalue zero and elements $v_i^{(k)} = 1$ if $i \in G_k$ and 0 otherwise. Thus there will be g degenerate eigenvectors with eigenvalue 0.

If the network separates well but not perfectly into communities—if there are just a few edges that do not fit the block-diagonal pattern—then this will no longer be perfectly true. Instead there will in general be the one eigenvector $\mathbf{1}$ with eigenvalue zero, and $g - 1$ eigenvalues slightly different from zero, indeed slightly greater than zero, since all eigenvalues of the graph Laplacian are non-negative[1]. The corresponding eigenvectors will approximately be linear combinations of the eigenvectors $\mathbf{v}^{(k)}$ defined above. Hence, by looking for eigenvalues of the graph Laplacian only slightly greater than zero and taking linear combinations of the corresponding eigenvectors, one should in theory be able to find the blocks themselves, at least approximately.

A particular special case of this argument is when there are only two blocks. Noting that all eigenvectors corresponding to non-degenerate eigenvalues of a real symmetric matrix are orthogonal, it is clear that all eigenvectors other than that corresponding to the lowest eigenvalue must have both positive and negative elements. And for the case of two weakly coupled communities there will thus be one eigenvector with eigenvalue slightly greater than zero and elements all positive for one community and all negative for the other, since all elements are (nearly) equal within a community (see above). Thus, we can divide the network into its two communities by looking at the eigenvector corresponding to the second lowest eigenvalue and separating the vertices by whether the corresponding element in this eigenvector is greater than or less than zero. This is the spectral bisection method. It works very well in cases where the graph really does split nicely into two communities, and predictably less well when it does not. The second eigenvalue λ_2, which is also called the *algebraic connectivity* of the graph, is a measure of how good the split is, with smaller values corresponding to better splits.

As an example of the application of the spectral bisection method, we consider a well-known graph from the social networks literature. (We will use the same example for many of the algorithms described in this paper.) This is the "karate club" network of Zachary [18], which was studied previously by a number of others in this context [1,10,19]. The network represents the pattern of friendships amongst the members of a karate club at a US university, constructed from ethnographic observations by Zachary over a period of two years in the early 1970s. During the period of study, the club split in two as a result of a dispute between two factions, and previous studies have found that the fault lines along which the split occurred are readily visible in the structure of the network. As a result the network makes a good test of the bisection algorithm—can the algorithm predict the two groups into which the club split, given only the pattern of edges in the network?

In Figure 2a we show the results of a bisection of the karate club network using the algorithm described above. The algebraic connectivity is $\lambda_2 = 0.469$, which is not exactly tiny, but is at least not approaching 1. And in practice, as the figure shows, the algorithm works very well, finding the known split of the network into two groups almost perfectly. Only one vertex is classified wrongly,

[1] This fact follows because the Laplacian can be expressed as the product of the so-called incidence matrix with its own transpose. It can also be regarded as a corollary of the Perron–Frobenius theorem. See, for instance, Bollobás [44].

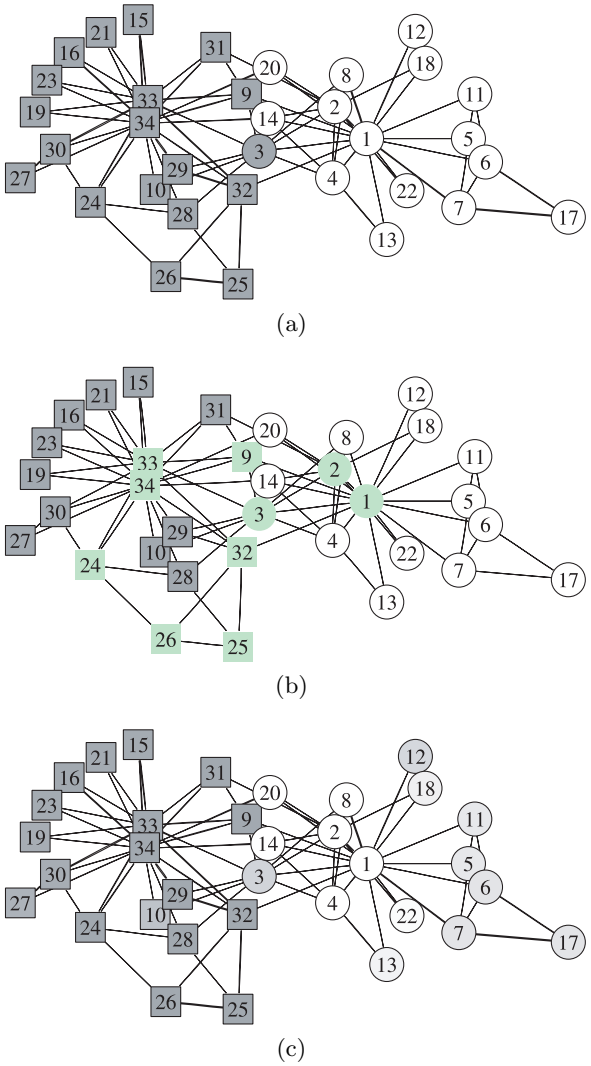

Fig. 2. The karate club network of Zachary [18], with numbered vertices representing the members of the club and edges representing friendships, as determined by observation of interactions. The two factions into which the club split during the course of the study are indicated by the squares and circles, while the dark grey and white show the divisions of the network found by (a) the spectral bisection algorithm of Section 2.1, (b) the hierarchical clustering method of Section 2.2 and (c) the Monte Carlo sampled version of the algorithm of Girvan and Newman proposed by Tyler et al. and discussed in Section 3.1. In (b) the lightly shaded vertices are those not assigned by the algorithm to either of the two principal communities. In (c) shades intermediate between the dark grey and white indicate ambiguously assigned vertices that fall in one community or the other, or neither, on different runs of the algorithm.

vertex 3, which is on the border between the groups and so it is understandable that it might be an ambiguous case.

The spectral bisection method is reasonably fast. Calculation of the eigenvectors of an $n \times n$ matrix takes in general a number of operations $O(n^3)$, which is slow. But in most cases of practical interest the Laplacian is a sparse matrix, in which case the leading eigenvectors can be calculated more rapidly using the Lanczos method [20]. The running time for the Lanczos method to find the second eigenvector goes approximately as $m/(\lambda_3 - \lambda_2)$, where m is the number of edges in the graph, and hence it can be very fast, although it may become slow if λ_2 is not well separated from the other eigenvalues. In other words, convergence is good if the graph separates cleanly into just the two communities but may be poor otherwise.

The principal disadvantage of the spectral bisection method is that it only bisects graphs. Division into a larger number of communities is usually achieved by repeated bisection, but this does not always give satisfactory results. And even if it did, we do not in general know ahead of time how many communities we want to divide the graph into. These issues are discussed further at the end of this section.

The Kernighan–Lin algorithm: A completely different approach to graph bisection was proposed by Kernighan and Lin [17] which, while heuristic, appears to give good results in practice and runs moderately quickly, in worst-case time $O(n^2)$, where n is the number of vertices in the network.

The Kernighan–Lin algorithm is a greedy optimization method that assigns a benefit function Q to divisions of the network and then attempts to optimize that benefit over possible divisions. The benefit function is the number of edges that lie within the two groups minus the number that lie between them. The algorithm requires the user to specify the size of the two groups into which the network should be split and to choose a starting configuration for the groups, for example by dividing the vertices at random. The algorithm then has two stages. First, we consider all possible pairs of vertices in which one vertex is chosen from each of the groups, and calculate the change ΔQ in the benefit function that would result from swapping them. Then we choose the pair that maximizes this change and perform the swap. This process is repeated, with the restriction that any vertex that has previously been swapped is never swapped again. When all vertices in one of the groups have been swapped once, this stage of the algorithm ends.

In the second stage, we go back over the sequence of swaps that were made and find the point during this sequence at which Q was highest. This is taken to be the bisection of the graph.

This two-stage process allows for the possibility that the value of Q does not increase monotonically. Even if Q decreases, a higher value that occurs later in the sequence of swaps will still be found by the algorithm.

The principal disadvantage of the Kernighan–Lin algorithm is that we have to specify the sizes of the two communities before we start. We can apply the algorithm, for example, to the karate club network of Figure 2, and on so doing, we find that it detects the split into the two factions perfectly—every vertex is correctly classified, which is slightly better than the spectral bisection method described above. However, to get this result we need to specify that the algorithm should look for groups of size 16 and 18, which are the known sizes of the groups

into which the network split. Giving any other sizes will of course produce the "wrong" result. This problem makes the Kernighan–Lin algorithm unsuitable for most applications to real-world network data, in which we have no idea *a priori* what the sizes of the groups will be. One might imagine that it would be possible to run the algorithm for a variety of different choices of the group sizes and then choose the one that gave the greatest value of Q overall but, in addition to increasing the run-time of the algorithm to $O(n^3)$, this will not work, since the best values of Q are always achieved for very asymmetric divisions of the network, with the global maximum being reserved for the trivial division in which one group contains all the vertices and the other none.

Even if this shortcoming could be overcome, the Kernighan–Lin algorithm still suffers from the drawback of all bisection algorithms, as mentioned above for the spectral method: it only divides the network into two groups and not an arbitrary number. Division into more than two groups can be achieved by repeated bisection, but there is no guarantee that the best division into three groups (however we choose to define that) can be arrived at by finding the best division into two and then dividing one of those two again. Furthermore, these algorithms give no hint about when we should stop the repeated bisection process, that is, about how many communities there should be in a network. For these reasons, the traditional graph partitioning methods are not ideal for analysing general network data, and so we turn to other sources in search of better methods.

2.2 Sociological approaches

Sociologists, in their study of social networks, have developed a substantial body of wisdom about the interpretation and analysis of graphs. Their approaches to finding communities, which have been directed almost exclusively at the analysis of empirically derived network data, are perhaps better suited to our current purposes than the methods of the previous section. The principal technique in current use is *hierarchical clustering* [21]. The idea behind this technique is to develop a measure of similarity x_{ij} between pairs (i, j) of vertices, based on the network structure one is given. Many different such similarity measures are possible—several are discussed below. Once one has such a measure then, starting with an empty network of n vertices and no edges, one adds edges between pairs of vertices in order of decreasing similarity, starting with the pair with strongest similarity. Note that the edges added in this way have no direct connection with the edges of the original network; in this method the original network is used only for the calculation of the similarity measure.

There are a couple of different ways in which communities can be extracted from this method. The most common method, called the *single linkage* method, is to declare the components formed as the edges are added to be the communities. As we add more and more edges in order of decreasing similarity, the components coalesce and get larger. If the similarity of the most recently added edge is x, then the communities have the definitive property that any two vertices with similarity greater than or equal to x are necessarily in the same community. The method does not however place any general conditions on the similarities of vertices in the same community: such vertices may have similarity either greater or less than x. In other words $x_{ij} \geq x$ is a sufficient but not necessary condition for vertices i and j to be in the same community.

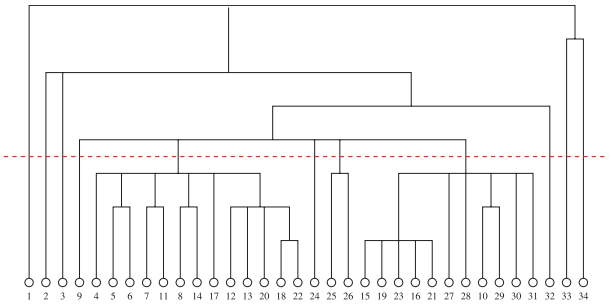

Fig. 3. The hierarchical tree or dendrogram depicting the results of a single linkage hierarchical clustering of the karate club network based on the Euclidean distance measure of structural equivalence, equation (1). A cross-section of the tree at any level will give the communities at that level. The cross-section indicated by the dotted line corresponds to the community division shown in Figure 2b. The vertical height of the branching points in the tree are indicative only of the order in which the joins between vertices take place. Note that the heights of some joins coincide, indicating that the vertices joined at that level have identical similarities.

As x is decreased and the communities join together, the single linkage hierarchical clustering method divides the network into fewer and fewer communities. At the start of the algorithm there are n components consisting of a single vertex each, and at the end there is just one component containing all vertices. The components at each step along the way are perfectly nested inside the components at the next step, so that the entire progress of the algorithm from start to finish can be represented as a tree or *dendrogram*, an example of which is shown in Figure 3. The "leaves" at the bottom of the figure show the individual vertices at the start of the algorithm, and the "root" at the top represents the network after all vertices have been joined into a single component. Horizontal cuts through the tree at various heights represent the communities found if the process is halted at the corresponding point. Like the bisection methods of Section 2.1, the hierarchical clustering method provides no measure of how many communities the network should be split into—it is up to the investigator to make his or her own choice about where the tree should be cut.

The opposite extreme to the single linkage method of defining communities is the *complete linkage* method. In this method edges are once again added to an initially empty graph in order of decreasing similarity, but now the communities are defined as being the maximal cliques in the network rather than the components. A *clique* is a set of vertices each of which is connected directly to all

others in the set. A *maximal clique* is a clique that is not contained inside any larger clique. If the similarity of the most recently added edge is x, then the communities created in this way have the definitive property that any two vertices in the same community have similarity greater than or equal to x. The method however places no general conditions on the similarities of vertices in different communities and the condition $x_{ij} \geq x$ is now a necessary but not sufficient condition for vertices to be in the same community.

Of the two methods described, the complete linkage method has perhaps the more desirable properties, but it is rarely used, for two reasons. First, finding cliques in a graph is a hard problem. The algorithm of choice is the Bron–Kerbosch algorithm [22], which runs in worst-case time scaling exponentially with graph size. Second, the cliques are, in general, not unique. A vertex can belong to two or more different cliques, obliging us to assign it to one community or another according to some rule. Typically one assigns it to the largest clique of which it is a member, or randomly to one such if several cliques tie for the honour, but this choice is to some extent arbitrary.

There are a variety of ways of defining the similarity between vertices. Sociological studies have tended to concentrate on the property known as *structural equivalence*. Two vertices are said to be structurally equivalent if they have the same set of neighbours (other than each other, if they are connected). Thus two individuals in a friendship network are structurally equivalent if they have the same friends. Since exact structural equivalence is rare in real-world networks, one usually defines a measure of the degree of equivalence, which can be done in several ways. The confusingly named *Euclidean distance* [23,24], which has nothing Euclidean about it, is

$$x_{ij} = \sqrt{\sum_{k \neq i,j} (A_{ik} - A_{ij})^2}, \tag{1}$$

where A_{ij} is once again the element of the adjacency matrix for vertices i and j. The Euclidean distance is really a *dissimilarity* measure, being zero for vertex pairs that are precisely structurally equivalent and largest for pairs that share none of the same neighbours at all. Thus a hierarchical clustering performed using this measure should add edges to the network in order of increasing x_{ij}, not decreasing. Notice that two vertices can be perfectly structurally equivalent by this measure without actually being connected to one another—the existence or not of an edge between i and j makes no difference to equation (1).

Another commonly used similarity measure is the Pearson correlation between columns (or rows) of the adjacency matrix [24]. Defining means and variances of the columns thus:

$$\mu_i = \frac{1}{n} \sum_j A_{ij}, \quad \sigma_i^2 = \frac{1}{n} \sum_j (A_{ij} - \mu_i)^2, \tag{2}$$

the correlation coefficient is

$$x_{ij} = \frac{\frac{1}{n} \sum_k (A_{ik} - \mu_i)(A_{jk} - \mu_j)}{\sigma_i \sigma_j}. \tag{3}$$

Vertices that have a high degree of structural equivalence will have high values of this similarity measure, and those that do not will have low values.

A similarity measure not based on structural equivalence is the count of edge- (or vertex-) independent paths between vertices. Two paths are said to be edge-independent if they share none of the same edges. By the max-flow/min-cut theorem, the number of such paths between two vertices is equal to the maximum flow that can be propelled through the network between the same two vertices if each edge can carry a maximum flow of one unit. This quantity can be calculated in $O(m)$ time, where m is the number of edges in the graph, using, for instance, the augmenting path algorithm [25]. Complete linkage communities formed using independent path counts as the similarity measure have the property that any two vertices in the same community have at least k independent paths between them, where k is the similarity for the most recently added edge. In the graph theory literature such communities are called *k-components*, and a number of special-purpose algorithms have been developed specifically for finding them, the best known being the 2- and 3-component (or bicomponent and tricomponent) algorithms of Hopcroft and Tarjan [26,27]. The k-components of a network are sometimes of interest in social network analyses—see references [28] and [29] for two recent examples.

As an example of the hierarchical clustering method, we show in Figure 3 the dendrogram resulting from the application of Euclidean distance single linkage clustering to the karate club network of Section 2.1. Choosing the cut through the tree that corresponds most closely to the known division of the club, as indicated by the dotted line in the figure, we get the separation shown in Figure 2b. As the figure shows, the method finds a substantial number of the members of each of the two groups, but some members get left out of each, and crucially the central members of both groups, vertices 1, 33 and 34, are left out. This is typical of the hierarchical clustering method: it tends to be good at finding parts of communities—those parts corresponding to individuals with high similarity according to whatever similarity measure is chosen—but usually leaves some others unassigned to any major group.

The method is moderately fast: similarity measures like those in equations (1, 3) take $O(mn)$ operations to calculate for all vertex pairs. The actual clustering is limited by the time taken to sort the $O(n^2)$ similarities into decreasing order, which is $O(n^2 \log n)$. Construction of the dendrogram can be achieved in near-linear time using, for example, the tree-based union/find algorithm of Fischer [30,31]. Thus the overall run-time of the algorithm scales as $O(n^2 \log n)$ on a sparse graph.

The hierarchical clustering method has the advantage that it doesn't require us to specify the size or number of groups we want to look for beforehand. It does not however tell us how many groups should be used to get the best division of the network. And even were this not the case, the problems revealed in Figure 2b are sufficient to make hierarchical clustering unsatisfactory for the

analysis of many large real-world networks. For these reasons, researchers have in recent years developed new methods for identifying communities in such networks, a selection of which we now describe.

3 Recent approaches

Observing that while the traditional approaches to finding communities in networks can be very useful for certain types of problems they are not ideal for general network analysis, we now turn to a description of some more recent community structure algorithms. We start by describing the algorithm of Girvan and Newman [1], which divides networks by iterative removal of their edges, and some variations on it that have been proposed by other authors.

3.1 Methods based on edge removal

The algorithm of Girvan and Newman: The basic requirements for a general community finding algorithm are that it should find "natural" divisions among the vertices without requiring the investigator to specify how many communities there should be, or placing restrictions on their sizes, and without showing the pathologies evident in the hierarchical clustering method of Section 2.2. Girvan and Newman [1] have proposed an algorithm that appears to achieve these goals and which has three definitive features thus: (1) it is a *divisive* method, in which edges are progressively removed from a network, by contrast with the *agglomerative* hierarchical clustering method; (2) the edges to be removed are chosen by computing betweenness scores as described in detail below; (3) the betweenness scores are recomputed following the removal of each edge.

The motivation behind the method is as follows. In a network such as that depicted in Figure 1, the few edges that lie between communities can be thought of as forming "bottlenecks" between the communities—traffic of one kind or another that flows through the network will have to travel along at least one of these bottleneck edges if it wishes to pass from one community to another. Thus if we consider some model of traffic on the network and look for the edges with highest traffic, we should find the edges between the communities. Removing these should then split the network into its natural communities.

As a measure of traffic flow Girvan and Newman use "edge betweenness", a generalization to edges of the well-known vertex betweenness of Freeman [32], which in fact seems to predate Freeman's work [33], although its original discoverer never published the discovery. The betweenness of an edge is defined to be the number of geodesic (i.e., shortest) paths between vertex pairs that run along the edge in question, summed over all vertex pairs. This quantity can be calculated for all edges in time that goes as $O(mn)$ on a graph with m edges and n vertices [34,35].

The algorithm of Girvan and Newman then involves simply calculating the betweenness of all edges in the network and removing the one with highest betweenness, and repeating this process until no edges remain. If two or more edges tie for highest betweenness then one can either choose one at random to remove, or simultaneously remove all of them. The entire progress of the algorithm from start to finish can, as with the hierarchical clustering method, be represented as a dendrogram (see Fig. 3 again). The algorithm can be thought of as progressing from the root of the dendrogram to the leaves, rather than the other way round, the branches of the tree representing the order of splitting of the network as edges are removed, and the communities are taken to be the components of the graph, as in the single linkage clustering method. Horizontal cross-sections of the dendrogram represent possible community divisions with a larger or smaller number of communities depending on the position of the cut.

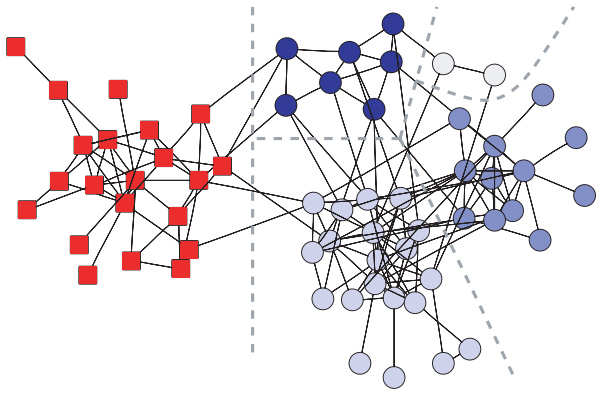

Fig. 4. Community structure in the social network of bottlenose dolphins assembled by Lusseau et al. [36,37], extracted using the algorithm of Girvan and Newman [1]. The squares and circles denote the primary split of the network into two groups and the circles are further subdivided into four smaller groups as shown. After Newman and Girvan [38].

Applying the algorithm to the karate club network, for example, gives precisely the same result as the spectral bisection method (Fig. 2a)—the network is split into two communities, with all vertices save one, number 3, classified correctly. However, the algorithm is considerably more useful than the spectral bisection method for general network analysis because, like the hierarchical clustering method, it also allows us to split the network into any other number of communities, where the bisection method only ever finds two. Furthermore, some networks divide both at a coarse level into a few communities and then subdivide further into a larger number of small communities, and this also can be represented by the dendrogram generated by the algorithm. As one example of this, we reproduce in Figure 4 the results of the application of the algorithm to a network of social interactions within a group of dolphins. The network data are taken from the work of Lusseau et al. [36,37] and the algorithm in this case finds first a split of the network into two groups, represented by the squares and circles in the figure, and then a subdivision of the larger of these two groups into four smaller ones. Some speculations about the origin of these splits are given in reference [38].

While it appears to give good results in many cases, there are two principal disadvantages of the algorithm of

Girvan and Newman. The first is that, like all the others described so far, it provides no guide to how many communities a network should be split into. To address this problem, Newman and Girvan [38] proposed that the divisions the algorithm generates be evaluated using a measure they call *modularity,* which is a numerical index of how good a particular division is. For a division with g groups, we define a $g \times g$ matrix $\mathbf{e}$ whose component e_{ij} is the fraction of edges in the original network that connect vertices in group i to those in group j. Then the modularity is defined to be

$$Q = \sum_i e_{ii} - \sum_{ijk} e_{ij} e_{ki} = \mathrm{Tr}\,\mathbf{e} - \|\,\mathbf{e}^2\,\|\,, \qquad (4)$$

where $\|\,\mathbf{x}\,\|$ indicates the sum of all elements of $\mathbf{x}$. Physically, Q is the fraction of all edges that lie within communities minus the expected value of the same quantity in a graph in which the vertices have the same degrees but edges are placed at random without regard for the communities. A value of $Q = 0$ indicates that the community structure is no stronger than would be expected by random chance and values other than zero represent deviations from randomness. Local peaks in the modularity during the progress of the community structure algorithm indicate particularly good divisions of the network, and this is, for instance, how the division depicted in Figure 4 was chosen. The definition and application of the modularity is independent of the particular community structure algorithm used, and it can therefore also be applied to any other algorithm. We give another example of its use in Section 3.2.

The other main disadvantage of the algorithm of Girvan and Newman is that it is slow. Since there are m edges to be removed in total and each iteration of the algorithm takes $\mathrm{O}(mn)$ time, the worst-case running time of the algorithm is $\mathrm{O}(m^2 n)$, or $\mathrm{O}(n^3)$ on a sparse graph.

To address the slow speed of the algorithm, a number of authors have suggested modifications of the basic approach. We discuss two here, the algorithms of Tyler et al. [6] and of Radicchi et al. [9].

The algorithm of Tyler et al.: In studies of email networks—networks in which the vertices are email addresses and the edges are messages passing between them—Tyler et al. [6] have introduced a variation on the algorithm of Girvan and Newman that improves the speed of the calculation substantially, although it does so at the cost of a reduction in accuracy.

The algorithm employed by Girvan and Newman calculates the contributions to edge betweenness for all paths starting at a single vertex i, which takes $\mathrm{O}(m)$ operations, and then sums the results over all n vertices to derive the total betweenness scores for all edges. Tyler et al. suggest instead that only a subset of vertices i be summed over, giving partial betweenness scores for all edges; if a random sample is chosen, this will give a Monte Carlo estimate of betweenness that tends to the true betweenness as the size of the sample becomes large. This estimate will contain statistical errors, as all Monte Carlo estimates do, but Tyler et al. show that good results can be obtained with reasonably small sample sizes, which could potentially offer substantial speed improvements over the original algorithm. The number of vertices sampled is chosen so as to make the betweenness of at least one edge in the network greater than a certain threshold. (There is also a hard lower limit on the number of samples.) Since one is interested only in which edge has the highest betweenness, this ensures that the error on that highest betweenness falls below some satisfactory level chosen by the investigator.

Tyler et al. were in their calculations primarily interested not in increasing the speed of the community structure algorithm. Rather, their interest was in finding a way of introducing a stochastic element into the algorithm. By doing so, vertices whose community assignment is ambiguous, like vertex 3 in Figure 2, will sometimes be put in one community and sometimes in another, and by repeating the calculation many times one can make an estimate of the extent to which particular assignments are reliable. As an example of this technique, we show in Figure 2c the result of applying the algorithm of Tyler et al. to the karate club network twenty times and then averaging the results. As the figure shows, one community, the one on the left in the figure, is quite unambiguous, while vertex 3, predictably, falls somewhere between the left and right communities. The right community is mostly identified correctly, but contains a number of peripheral vertices whose community assignment is less strong—on some of the runs these vertices are assigned to their own separate communities, indicating that their link to the rest of the group is weaker.

The algorithm of Radicchi et al.: In order to speed up the identification of communities, Radicchi et al. [9] have proposed another algorithm that takes a different approach. Their algorithm, like that of Girvan and Newman, is based on iterative removal of edges, but uses a different measure instead of betweenness to identify the edges to be removed. As in the algorithm of Girvan and Newman, this measure is recalculated after each removal, but it is a local measure that can be calculated quickly, and hence the overall algorithm runs faster, in time $\mathrm{O}(m^4/n^2)$ on a graph with m edges and n vertices, or $\mathrm{O}(n^2)$ on a sparse graph, which is one order of system size faster than the original algorithm.

The algorithm of Radicchi et al. is based on counting short loops of edges in the network—loops of length three, or triangles, in the simplest case. Edges that run between communities (see Fig. 1) are unlikely to belong to many short loops, because to complete a loop containing such an edge we need another edge that runs between the same two communities, and such other edges are, by hypothesis, rare. Thus one should be able to spot the between-community edges by looking for ones that belong to an unusually small number of loops.

Consider an edge that runs between two vertices i and j having degrees k_i and k_j. The maximum number of triangles to which such an edge can belong, assuming that there is at most one edge between any pair of vertices, is $\min(k_i - 1, k_j - 1)$. Radicchi et al. define what they call

the *edge clustering coefficient* C_{ij}, which is roughly the fraction of these triangles that are actually realized:

$$C_{ij} = \frac{z_{ij} + 1}{\min(k_i - 1, k_j - 1)}, \tag{5}$$

where z_{ij} is the measured number of triangles to which the edge belongs. The extra +1 in the numerator is included to avoid penalizing too heavily edges that belong to zero triangles, but which join vertices of low degree.

The quantity C_{ij} will be small for edges between communities, and Radicchi et al. show that it is in fact quite strongly negatively correlated with edge betweenness in the networks they looked at. Their algorithm consists of iterative removal of edges with low values of C_{ij}, followed by recalculation of C_{ij} for the remaining edges[2]. Edges for which either k_i or k_j is 1, so that equation (5) diverges, are excluded from consideration. They give a number of tests of the algorithm for different networks, showing that it is effective at finding known community structure in many cases. They also examine generalizations of the algorithm in which one counts loops of length four or higher, instead of triangles, and find in some cases that these out-perform the simple triangle-based version.

The time taken to calculate the edge clustering coefficient for an edge goes like the product of the degrees k_i and k_j. Assuming these are uncorrelated (which is known not to be true in some networks [39,40]), this time scales as the square of the mean degree, i.e., as m^2/n^2. Repeating the calculation for each of m edges and each of m removals then gives a total running time $O(m^4/n^2)$ as above. In practice, the algorithm is fast enough to analyse some moderately large graphs: Radicchi et al. study the structure of a collaboration network of about 13 000 scientists, some 30% bigger than the largest network that has been tackled with the algorithm of Girvan and Newman, and it seems likely that bigger networks still would be within reach of the patient researcher.

The principal disadvantage of the method of Radicchi et al. is that it relies on the presence of triangles in the network. Clearly if a network has few triangles in the first place, then the edge clustering coefficient will be small for all edges, and the algorithm will fail to find the communities. On the basis of comparisons with a standard Erdös–Rényi random graph, it has been conjectured that essentially all real-world networks have a statistically high proportion of triangles in them [41], but recent results making use of a more accurate null model argue otherwise [42]. In fact, it appears that triangle counts are indeed unusually high in most social networks (with the exception of networks of sexual contacts [29]), but in nonsocial networks they are relatively low. This suggests that the method of Radicchi et al. would probably work well when applied to social networks, but perhaps less well for other network types.

[2] Their paper also discusses possible definitions of a community and ways in which these definitions can be used to improve the performance of the algorithm, but these issues are outside the scope of the present article.

3.2 Other methods

Recently, the present author has proposed an alternative approach to the discovery of community structure based on the modularity Q defined in equation (4) [43]. This quantity, it is claimed, is high for good community divisions and low for poor ones, so one ought to be able to find the communities in a network by optimizing Q over possible divisions. Unfortunately, optimizing Q exhaustively would take an amount of time at least exponential in the number of vertices, so to get an algorithm with reasonable running time one must use some approximate optimization strategy. The simplest such strategy is a greedy algorithm that starts with each vertex in a separate community on its own, and amalgamates communities in pairs, choosing at each step the pair whose amalgamation will give the greatest increase (or smallest decrease) in Q. Since the greatest increase in Q can never be produced by amalgamating groups that are not actually connected by any edges, the largest number of pairs one need ever consider is equal to the number of edges m, and a total of $n - 1$ amalgamations are necessary to connect all n vertices into a single large group, at which point the algorithm stops. Thus the total running time is $O(mn)$, or $O(n^2)$ on a sparse graph. The output of the algorithm can be represented in the form of a dendrogram and the optimal cross-section of the dendrogram found by looking for the optimal value of Q.

The main advantage of the algorithm is its speed, which allows large networks to be analysed; an application to a collaboration network of more than fifty thousand scientists is given in reference [43]. It should also work well on networks of all types, although it appears in general to give results slightly less good than the algorithm of Girvan and Newman.

A quite different approach has been proposed by Wu and Huberman [10], based on the properties of resistor networks. Their algorithm is fundamentally a bisection algorithm, like those described in Section 2.1, although they also give a version that will divide a network into a larger number of communities provided one knows in advance how many communities there are. The idea behind the algorithm is to consider the electrical circuit formed by placing a unit resistor on each edge of the network and then applying a unit potential difference between two vertices chosen arbitrarily. If the network divides strongly into two communities and the vertices in question happen to fall in different communities, then the spectrum of voltages on the rest of the vertices should, the authors argue, show a large gap corresponding to the border between the communities. We can thus identify the communities by finding the largest gap and dividing the vertices according to whether their voltages lie above or below it. Since the largest gap sometimes falls at the end of the spectrum, giving a highly asymmetric division of the network, the authors restrict themselves to looking only within some central portion of the spectrum.

The calculation of the voltages requires the inversion of the graph Laplacian (see Sect. 2.1), which will normally take time $O(n^3)$. However, as Wu and Huberman point

out, a reasonable approximation to the inverse can obtained by expanding it as a power series and truncating at some finite order. On a sparse graph, for which the Laplacian is sparse also, each term in the expansion can be evaluated from the previous one in $O(m)$ time, and the speed of the algorithm then depends on how fast the series converges. Typically we have to take a number of terms of order $1/(\lambda_3 - \lambda_2)$ to get good convergence, where λ_2 and λ_3 are the second and third smallest eigenvalues of the Laplacian. This means the running time goes as $m/(\lambda_3 - \lambda_2)$. Thus, like the Lanczos-based spectral bisection of Section 2.1, the algorithm works well when the network separates cleanly into two communities, but can be slow otherwise[3].

Assuming that the network separates cleanly and convergence is fast, then the rate-determining step in the algorithm is the sorting of the vertices according to their voltages in order to find the largest gap, which takes time $O(n \log n)$. Repeating the calculation for all pairs of vertices to which the initial potential difference is applied gives an algorithm that bisects the graph in time $O(n^3 \log n)$. However, Wu and Huberman find that good results can be obtained by a trick similar to the one employed by Tyler et al. [6] in the algorithm described in Section 3.1, of randomly sampling vertex pairs from the complete set. If only a fixed number of pairs is sampled then one gets an algorithm that runs in $O(n \log n)$ time. Wu and Huberman argue that a pair of vertices need only lie in different halves of the network in order for the algorithm to find the correct bisection, and provided the two communities are roughly equal in size, this will happen about a half of the time, so a reasonably small fixed-size sample should be adequate to bisect the network.

The algorithm appears to work well when applied, for instance, to the karate club network—it successfully finds the two known communities in the network. As with the other bisection methods, it can be applied repeatedly to find divisions of a network into more than two communities, although, as discussed in Section 2.1, this is not always an ideal approach. Wu and Huberman give a more detailed discussion of this point in the final section of their paper.

An interesting further feature of the algorithm of Wu and Huberman is that it can also be used to find the particular community to which a specified vertex belongs, without first finding all communities in the network. There are certainly circumstances (e.g., web searching) in which one would imagine this could be useful. Applying a source voltage to the one vertex of interest and placing a sink at another chosen at random, one can look for the set of vertices with voltages close in some sense to that of the vertex of interest, and regard those as its community.

[3] Indeed, as Wu and Huberman point out, there are close mathematical connections between their method and the spectral bisection method, and it is no coincidence that the two methods converge at the same rate—both fundamentally involve the repeated multiplication of the Laplacian into a given vector.

Interestingly, this is similar to another earlier method for solving the same problem proposed by Flake et al. [12]. They considered a different definition of flow in the network—the computer scientist's max-flow definition, which can also be calculated in linear time—and they averaged over all possible sink vertices rather than choosing one at random. However, the basic idea is the same and the two algorithms appear to give qualitatively similar results.

4 Conclusions

In this paper we have reviewed algorithmic methods for finding communities of densely connected vertices in network data. We have discussed some of the traditional approaches, such as spectral graph partitioning [15,16] and hierarchical clustering [21], but, as we have pointed out, these have a number of shortcomings as far as the analysis of large real-world networks is concerned. In the last few years, therefore, several new methods have been developed that are flexible enough to apply to quite general network structures. We have described a number of methods based on iterative removal of between-community edges, including the betweenness-based method of Girvan and Newman [1,38] and the Monte Carlo resampled variation proposed by Tyler et al. [6], as well as the algorithm based on counts of short loops proposed by Radicchi et al. [9]. We have also discussed briefly two more recent algorithms that are notable for their relative computational efficiency, the modularity maximization algorithm of Newman [43] and the resistor network algorithm of Wu and Huberman [10]. We have compared results produced by these algorithms and outlined their strengths and weaknesses.

As a result of substantial progress in recent years, it appears we now have an effective toolkit for studying community structure in networks. There is certainly still room for improvement however in both the speed and sensitivity of community structure algorithms, and there are many interesting networked systems awaiting analysis using these methods.

The author thanks Michelle Girvan for useful conversations and comments. Thanks also to Oliver Boisseau, Patti Haase, David Lusseau, and Karsten Schneider for providing the data for the dolphin network and to Douglas White for the karate club data. This work was funded in part by the National Science Foundation under grant number DMS–0234188.

References

1. M. Girvan, M.E.J. Newman, Proc. Natl. Acad. Sci. USA **99**, 7821 (2002)
2. D. Wilkinson, B.A. Huberman, preprint `cond-mat/0210147` (2002)
3. R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, A. Arenas, Phys. Rev. E **68**, 065103 (2003)
4. P. Holme, M. Huss, H. Jeong, Bioinformatics **19**, 532 (2003)

5. P. Holme, M. Huss, *Proceedings of 3rd Workshop on Computation of Biochemical Pathways and Genetic Networks*, edited by R. Gauges, U. Kummer, J. Pahle, U. Rost (Logos, Berlin, 2003), pp. 3–9
6. J.R. Tyler, D.M. Wilkinson, B.A. Huberman, in *Proceedings of the First International Conference on Communities and Technologies*, edited by M. Huysman, E. Wenger, V. Wulf (Kluwer, Dordrecht, 2003)
7. P. Gleiser, L. Danon, preprint `cond-mat/0307434` (2003)
8. M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, A. Arenas, preprint `cond-mat/0309263` (2003)
9. F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, preprint `cond-mat/0309488` (2003)
10. F. Wu, B.A. Huberman, preprint `cond-mat/0310600` (2003)
11. D. Gibson, J. Kleinberg, P. Raghavan, in *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia* (Association of Computing Machinery, New York, 1998)
12. G.W. Flake, S.R. Lawrence, C.L. Giles, F.M. Coetzee, IEEE Computer **35**, 66 (2002)
13. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Science **298**, 824 (2002)
14. S. Shen-Orr, R. Milo, S. Mangan, U. Alon, Nature Genetics **31**, 64 (2002)
15. M. Fiedler, Czech. Math. J. **23**, 298 (1973)
16. A. Pothen, H. Simon, K.-P. Liou, SIAM J. Matrix Anal. Appl. **11**, 430 (1990)
17. B.W. Kernighan, S. Lin, Bell Sys. Techn. J. **49**, 291 (1970)
18. W.W. Zachary, J. Anthropological Research **33**, 452 (1977)
19. H. Zhou, Phys. Rev. E **67**, 061901 (2003)
20. G.H. Golub, C.F. Van Loan, *Matrix computations* (Johns Hopkins University Press, Baltimore, MD, 1989)
21. J. Scott, *Social Network Analysis: A Handbook* (Sage, London, 2000), 2nd ed.
22. C. Bron, J. Kerbosch, Comm. ACM **16**, 575 (1973)
23. R.S. Burt, Social Forces **55**, 93 (1976)
24. S. Wasserman, K. Faust, *Social Network Analysis* (Cambridge University Press, Cambridge, 1994)
25. R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Upper Saddle River, NJ, 1993)
26. R.E. Tarjan, SIAM J. Comput. **1**, 146 (1972)
27. J.E. Hopcroft, R.E. Tarjan, SIAM J. Comput. **2**, 135 (1973)
28. D.R. White, F. Harary, Sociological Methodology **31**, 305 (2001)
29. P.S. Bearman, J. Moody, K. Stovel, preprint, Department of Sociology, Columbia University (2002)
30. M.J. Fischer, in *Complexity of Computer Computations*, edited by R.E. Miller, J.W. Thatcher (Plunum Press, New York, 1972), pp. 153–167
31. R.E. Tarjan, J. ACM **22**, 215 (1975)
32. L.C. Freeman, Sociometry **40**, 35 (1977)
33. J.M. Anthonisse, Technical Report BN 9/71, Stichting Mathematicsh Centrum, Amsterdam (1971)
34. M.E.J. Newman, Phys. Rev. E **64**, 016132 (2001)
35. U. Brandes, J. Math. Soc. **25**, 163 (2001)
36. D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, Behavioral Ecology and Sociobiology **54**, 396 (2003)
37. D. Lusseau, Proc. R. Soc. London B (suppl.) **270**, S186 (2003)
38. M.E.J. Newman, M. Girvan, Phys. Rev. E **69**, 026113 (2004)
39. R. Pastor-Satorras, A. Vázquez, A. Vespignani, Phys. Rev. Lett. **87**, 258701 (2001)
40. M.E.J. Newman, Phys. Rev. Lett. **89**, 208701 (2002)
41. D.J. Watts, S.H. Strogatz, Nature **393**, 440 (1998)
42. M.E.J. Newman, J. Park, Phys. Rev. E **68**, 036122 (2003)
43. M.E.J. Newman, preprint `cond-mat/0309508` (2003)
44. B. Bollobás, *Modern Graph Theory* (Springer, New York, 1998)