In addition to the structural distinctions between these perspectives, they also illustrate different focuses on groups versus individuals, and on the activity within a group versus its contacts with a larger population. The contrasts are also related to Robert Putnam's dichotomy between *bonding capital* and *bridging capital* [344]; these terms, while intended informally, correspond roughly to the kinds of social capital arising respectively from connections within a tightly-knit group and from connections between such groups.

The notion of social capital thus provides a framework for thinking about social structures as facilitators of effective action by individuals and groups, and a way of focusing discussions of the different kinds of benefits conferred by different structures. Networks are at the heart of such discussions — both in the way they produce closed groups where transactions can be trusted, and in the way they link different groups and thereby enable the fusion of different sources of information residing in these groups.

3.6 Advanced Material: Betweenness Measures and Graph Partitioning

This is the first in a series of sections throughout the book labeled "Advanced Material." Each of these sections comes at the end of a chapter, and it explores mathematically more sophisticated aspects of some of the models developed earlier in the chapter. They are strictly optional, in that nothing later in the book builds on them. Also, while these sections are technically more involved, they are written to be completely self-contained, except where specific pieces of mathematical background are needed; this necessary background is spelled out at the beginnings of the sections where it is required.

In this section, we will try formulating more concrete mathematical definitions for some of the basic concepts from earlier in the chapter. The discussion in this chapter has articulated a way of thinking about networks in terms of their tightly-knit regions and the weaker ties that link them together. We have formulated precise definitions for some of the underlying concepts, such as the clustering coefficient and the definition of a local bridge. In the process, however, we have refrained from trying to precisely delineate what we mean by a "tightly-knit region," and how to formally characterize such regions.

For our purposes so far, it has been useful to be able to speak in this more general, informal way about tightly-knit regions; it helps to be flexible since the exact characterization of the notion may differ depending on the different domains in which we encounter it. But there are also settings in which having a more precise, formal definition is valuable. In particular, a formal definition can be crucial if we are faced with a network dataset and actually want to identify densely connected groups of nodes within it.

This will be our focus here: describing a method that can take a network and break it down into a set of tightly-knit regions, with sparser interconnections between the regions.

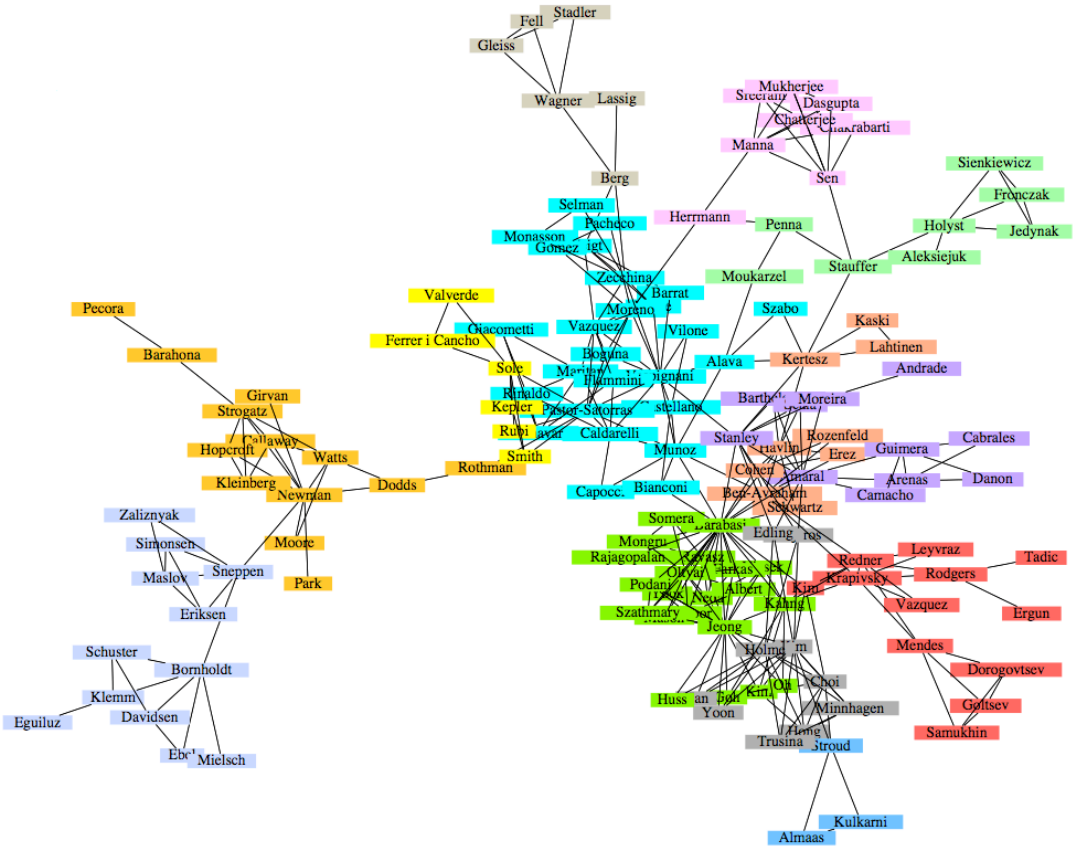


Figure 3.12: A co-authorship network of physicists and applied mathematicians working on networks [322]. Within this professional community, more tightly-knit subgroups are evident from the network structure.

We will refer to this as the problem of *graph partitioning*, and the constituent parts the network is broken into as the *regions* arising from the partitioning method. Formulating a method for graph partitioning will implicitly require working out a set of definitions for all these notions that are both mathematically tractable and also useful on real datasets.

To give a sense for what we might hope to achieve from such a method, let's consider two examples. The first, shown in Figure 3.12, depicts the co-authorships among a set of physicists and applied mathematicians working on networks [322]. Recall that we discussed co-authorship networks in Chapter 2 as a way of encoding the collaborations within a professional community. It's clear from the picture that there are tightly-knit groups within this community, and some people who sit on the boundaries of their respective groups. Indeed it resembles, at a somewhat larger scale, some of the pictures of tightly-knit groups and weak ties that we drew in schematic form earlier, in examples such as Figure 3.11. Is there a general way to pull these groups out of the data, beyond using just our visual intuition?

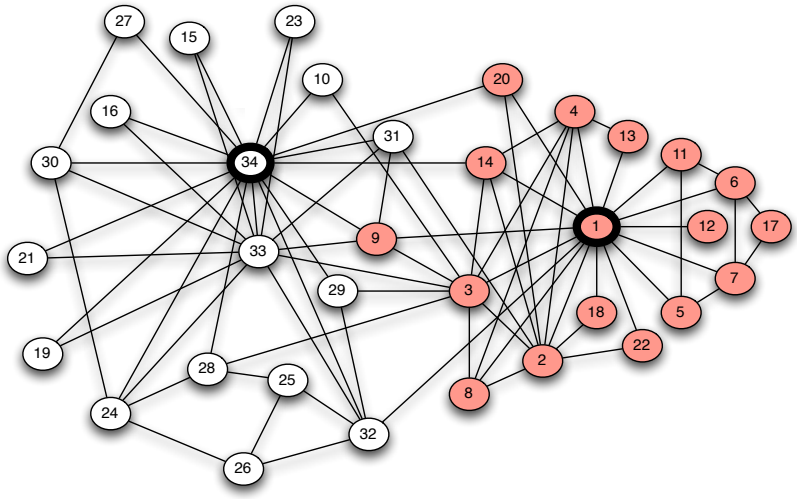


Figure 3.13: A karate club studied by Wayne Zachary [421] — a dispute during the course of the study caused it to split into two clubs. Could the boundaries of the two clubs be predicted from the network structure?

A second example, in Figure 3.13, is a picture of the social network of a karate club studied by Wayne Zachary [421] and discussed in Chapter 1: a dispute between the club president (node 34) and the instructor (node 1) led the club to split into two. Figure 3.13 shows the network structure, with the membership in the two clubs after the division indicated by the shaded and unshaded nodes. Now, a natural question is whether the structure itself contains enough information to predict the fault line. In other words, did the split occur along a weak interface between two densely connected regions? Unlike the network in Figure 3.12, or in some of the earlier examples in the chapter, the two conflicting groups here are still heavily interconnected. So to identify the division in this case, we need to look for more subtle signals in the way in which edges between the groups effectively occur at lower "density" than edges within the groups. We will see that this is in fact possible, both for the definitions we consider here as well as other definitions.

A. A Method for Graph Partitioning

Many different approaches have been developed for the problem of graph partitioning, and for networks with clear divisions into tightly-knit regions, there is often a wide range of methods that will prove to be effective. While these methods can differ considerably in their specifics, it is useful to identify the different general styles that motivate their designs.

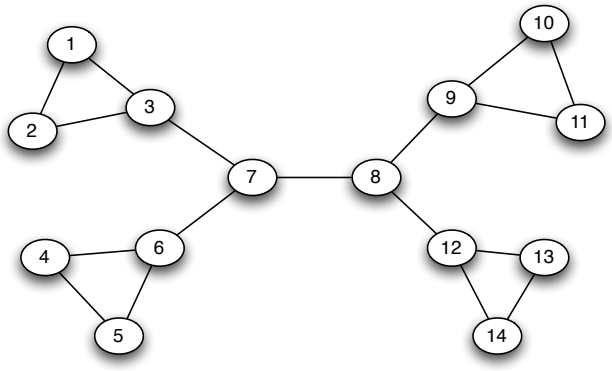
General Approaches to Graph Partitioning. One class of methods focuses on identifying and removing the "spanning links" between densely-connected regions. Once these links are removed, the network begins to fall apart into large pieces; within these pieces, further spanning links can be identified, and the process continues. We will refer to these as *divisive* methods of graph partitioning, since they divide the network up as they go.

An alternate class of methods starts from the opposite end of the problem, focusing on the most tightly-knit parts of the network, rather than the connections at their boundaries. Such methods find nodes that are likely to belong to the same region and merge them together. Once this is done, the network consists of a large number of merged chunks, each containing the seeds of a densely-connected region; the process then looks for chunks that should be further merged together, and in this way the regions are assembled "bottom-up." We refer to these as *agglomerative* methods of graph partitioning, since they glue nodes together into regions as they go.

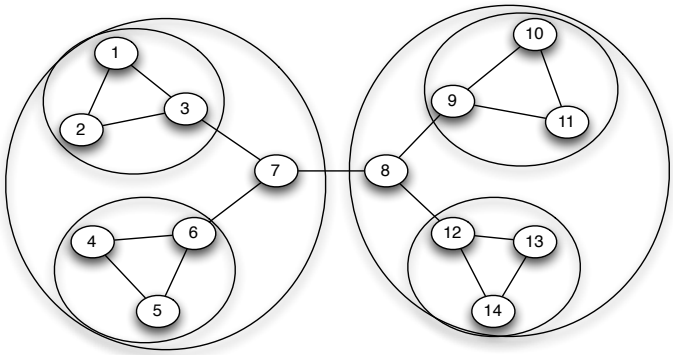
To illustrate the conceptual differences between these two kinds of approaches, let's consider the simple graph in Figure 3.14(a). Intuitively, as indicated in Figure 3.14(b), there appears to be a broad separation between one region consisting of nodes 1-7, and another consisting of nodes 8-14. Within each of these regions, there is a further split: on the left into nodes 1-3 and nodes 4-6; on the right into nodes 9-11 and nodes 12-14. Note how this simple example already illustrates that the process of graph partitioning can usefully be viewed as producing regions in the network that are naturally *nested*: larger regions potentially containing several smaller, even more tightly-knit regions "nested" within them. This is of course a familiar picture from everyday life, where — for example — a separation of the gobal population into national groups can be further subdivided into sub-populations within particular local areas within countries.

In fact, a number of graph partitioning methods will find the nested set of regions indicated in Figure 3.14(b). Divisive methods will generally proceed by breaking apart the graph first at the 7-8 edge, and subsequently at the remaining edges into nodes 7 and 8. Agglomerative methods will arrive at the same result from the opposite direction, first merging the four triangles into clumps, and then finding that the triangles themselves can be naturally paired off.

This is a good point at which to make the discussion more concrete, and to do so we focus on a particular divisive method proposed by Girvan and Newman [184, 322]. The Girvan-Newman method has been applied very widely in recent years, and to social network data in particular. Again, however, we emphasize that graph partitioning is an area in which there is an especially wide range of different approaches in use. The approach we discuss is an elegant and particular widely-used one; however, understanding which types of methods work best in different situations remains a subject of active research.



(a) *A sample network*



(b) *Tightly-knit regions and their nested structure*

Figure 3.14: In many networks, there are tightly-knit regions that are intuitively apparent, and they can even display a *nested* structure, with smaller regions nesting inside larger ones.

The Notion of Betweenness. To motivate the design of a divisive method for graph partitioning, let's think about some general principles that might lead us to remove the 7-8 edge first in Figure 3.14(a).

A first idea, motivated by the discussion earlier in this chapter, is that since bridges and local bridges often connect weakly interacting parts of the network, we should try removing these bridges and local bridges first. This is in fact an idea along the right lines; the problem is simply that it's not strong enough, for two reasons. First, when there are several bridges, it doesn't tell us which to remove first. As we see in Figure 3.14(a), where there are five bridges, certain bridges can produce more reasonable splits than others. Second, there can be graphs where no edge is even a local bridge, because every edge belongs to a triangle — and yet there is still a natural division into regions. Figure 3.15 shows a simple example, where we might want to identify nodes 1-5 and nodes 7-11 as tightly-knit regions, despite

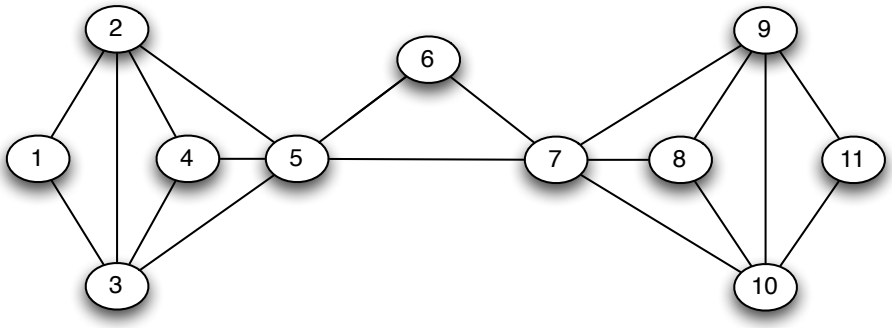


Figure 3.15: A network can display tightly-knit regions even when there are no bridges or local bridges along which to separate it.

the fact that there are no local bridges to remove.

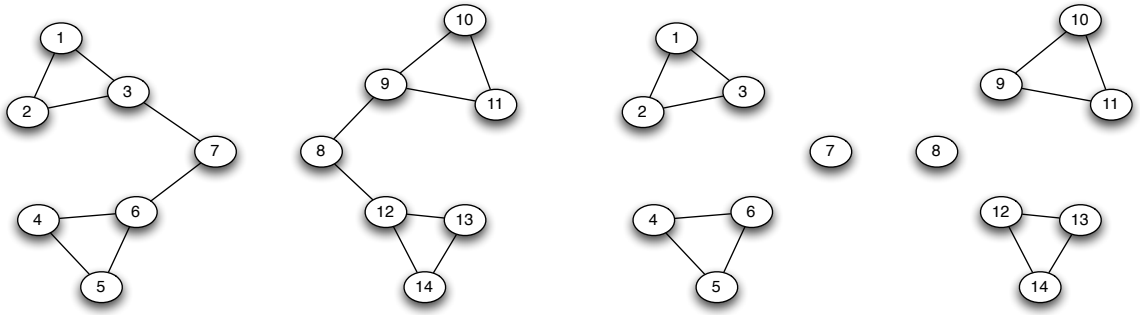
However, if we think more generally about what bridges and local bridges are doing, then we can arrive at a notion that forms the central ingredient of the Girvan-Newman method. Local bridges are important because they form part of the shortest path between pairs of nodes in different parts of the network — without a particular local bridge, paths between many pairs of nodes may have to be "re-routed" a longer way. We therefore define an abstract notion of "traffic" on the network, and look for the edges that carry the most of this traffic. Like crucial bridges and highway arteries, we might expect these edges to link different densely-connected regions, and hence be good candidates for removal in a divisive method.

We define our notion of traffic as follows. For each pair of nodes A and B in the graph that are connected by a path, we imagine having one unit of fluid "flow" along the edges from A to B. (If A and B belong to different connected components, then no fluid flows between them.) The flow between A and B divides itself evenly along all the possible *shortest* paths from A to B: so if there are k shortest paths from A and B, then $1/k$ units of flow pass along each one.

We define the *betweenness* of an edge to be the total amount of flow it carries, counting flow between all pairs of nodes using this edge. For example, we can determine the betweenness of each edge in Figure 3.14(a) as follows.

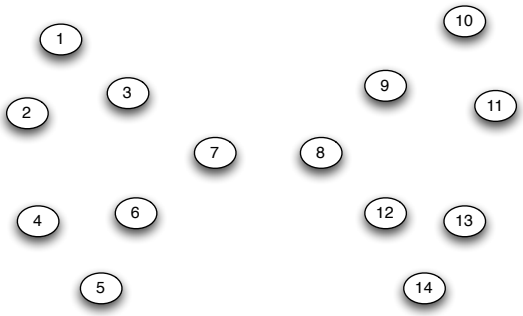
- Let's first consider the 7-8 edge. For each node A in the left half of the graph, and each node B in the right half of the graph, their full unit of flow passes through the 7-8 edge. On the other hand, no flow passing between pairs of nodes that both lie in the same half uses this edge. As a result, the betweenness of the 7-8 edge is $7 \cdot 7 = 49$.

- The 3-7 edge carries the full unit of flow from each node among 1, 2, and 3 to each



(a) *Step 1*

(b) *Step 2*



(c) *Step 3*

Figure 3.16: The steps of the Girvan-Newman method on the network from Figure 3.14(a).

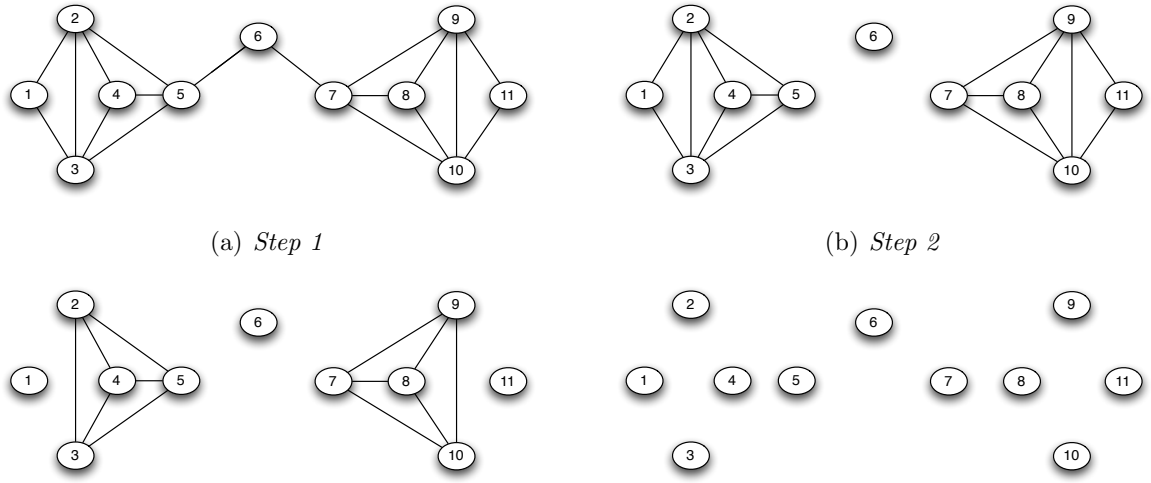
node among 4-14. Thus, the betweenness of this edge is $3 \cdot 11 = 33$. The same goes for the edges 6-7, 8-9, and 8-12.

- The 1-3 edge carries all the flow from 1 to every other node except 2. As a result, its betweennness is 12. By strictly symmetric reasoning, the other edges linked from 3, 6, 9, and 12 into their respective triangles have betweenness 12 as well.

- Finally, the 1-2 edge only carries flow between its endpoints, so its betweenness is 1. This also holds for the edges 4-5, 10-11, and 13-14.

Thus, betweenness has picked out the 7-8 edge as the one carrying the most traffic.

In fact, the idea of using betweenness to identify important edges draws on a long history in sociology, where most attribute its first explicit articulation to Linton Freeman [73, 168, 169]. Its use by sociologists has traditionally focused more on nodes than on edges, where the definition the same: the betweenness of a node is the total amount of flow that it carries, when a unit of flow between each pair of nodes is divided up evenly over shortest paths. Like edges of high betweenness, nodes of high betweenness occupy critical roles in the network



(a) *Step 1* (b) *Step 2*

(c) *Step 3* (d) *Step 4*

Figure 3.17: The steps of the Girvan-Newman method on the network from Figure 3.15.

structure — indeed, because carrying a large amount of flow suggests a position at the interface between tightly-knit groups, there are clear relationships of betweenness with our earlier discussions of nodes that span structural holes in a social network [86].

The Girvan-Newman Method: Successively Deleting Edges of High Betweenness. Edges of high betweenness are the ones that, over all pairs of nodes, carry the highest volume of traffic along shortest paths. Based on the premise that these are the most "vital" edges for connecting different regions of the network, it is natural to try removing these first. This is the crux of the Girvan-Newman method, which can now be summarized as follows.

(1) Find the edge of highest betweenness — or multiple edges of highest betweenness, if there is a tie — and remove these edges from the graph. This may cause the graph to separate into multiple components. If so, this is the first level of regions in the partitioning of the graph.

(2) Now recalculate all betweennesses, and again remove the edge or edges of highest betweenness. This may break some of the existing components into smaller components; if so, these are regions nested within the larger regions.

(...) Proceed in this way as long as edges remain in graph, in each step recalculating all betweennesses and removing the edge or edges of highest betweenness.

Thus, as the graph falls apart first into large pieces and then into smaller ones, the method naturally exposes a nested structure in the tightly-knit regions. In Figures 3.16 and 3.17

we show how the method operates on the graphs from Figures 3.14(a) and 3.15 respectively. Note how smaller regions emerge from larger ones as edges are successively removed.

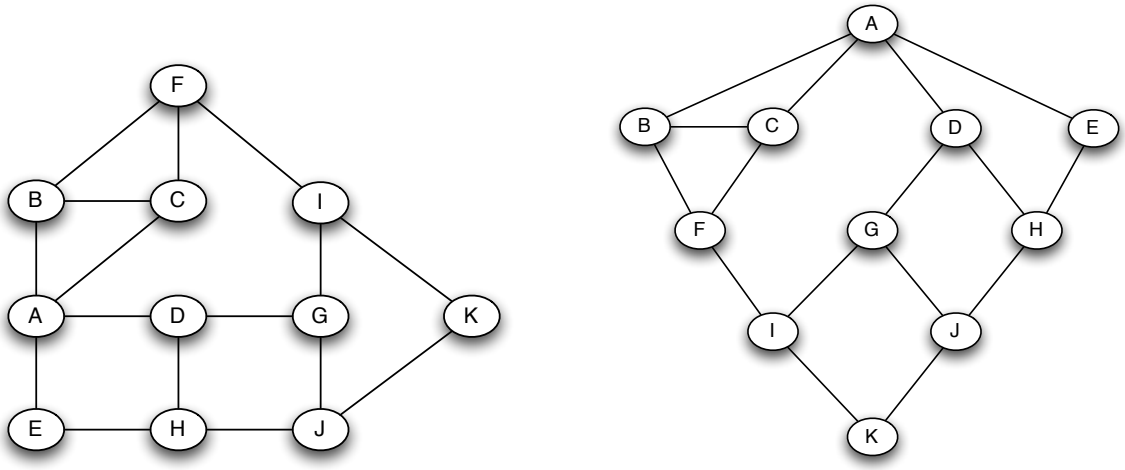
The sequence of steps in Figure 3.17 in fact exposes some interesting points about how the method works.

- When we calculate the betweennesses in the first step, the 5-7 edge carries all the flow from nodes 1-5 to nodes 7-11, for a betweenness of 25. The 5-6 edge, on the other hand, only carries flow from node 6 to each of nodes 1-5, for a betweenness of 5. (Similarly for the 6-7 edge.)

- Once the 5-7 edge is deleted, however, we recalculate all the betweennesses for the second step. At this point, all 25 units of flow that used to be on this deleted edge have shifted onto the path through nodes 5, 6, and 7, and so the betweenness of the 5-6 edge (and also the 6-7 edge) has increased to $5 + 25 = 30$. This is why these two edges are deleted next.

In their original presentation of the method, Girvan and Newman showed its effectiveness at partitioning a number of real network datasets into intuitively reasonable sets of regions. For example, on Zachary's karate club network in Figure 3.13, when the method is used to remove edges until the graph first separates into two pieces, the resulting partition agrees with the actual split that occurred in the club except for a single person — node 9 in the figure. In real life, node 9 went with the instructor's club, even though the graph partitioning analysis here would predict that he would join the president's club.

Zachary's original analysis of the karate club employed a different approach that also used the network structure. He first supplemented the network with numerical estimates of tie strength for the edges, based on his empirical study of the relationships within the karate club. He then identified a set of edges of minimum total strength whose removal would place node 1 and node 34 (the rival leaders) in different connected components, and he predicted this as the split. The approach Zachary used, deleting edges of minimum total strength so as to separate two specified nodes, is known as the problem of finding a *minimum cut* in a graph, and it has the been the subject of extensive research and applications [8, 164, 253]. On the karate-club network, this minimum-cut approach produced the same split as the Girvan-Newman method: it agreed with the split that actually occurred except for the outcome of node 9, an alignment of predictions that emphasizes how different approaches to graph partitioning can produce corresponding results. It is also interesting to note that Zachary traced the anomalous nature of node 9 to a fact that the network structure could not capture: at the time of the actual split, the person corresponding to node 9 was three weeks away from completing a four-year quest to obtain a black belt, which he could only do with the instructor (node 1).



(a) *A sample network* (b) *Breadth-first search starting at node A*

Figure 3.18: The first step in the efficient method for computing betweenness values is to perform a breadth-first search of the network. Here the results of breadth-first from node A are shown; over the course of the method, breadth-first search is performed from each node in turn.

Among the other examples discussed by Girvan and Newman, they provide a partition of the co-authorship network from Figure 3.12, with the top level of regions suggested by the different shadings of the nodes in that figure.

Ultimately, it is a challenge to rigorously evaluate graph partitioning methods and to formulate ways of asserting that one is better than another — both because the goal is hard to formalize, and because different methods may be more or less effective on different kinds of networks. Moreover, a line of recent work by Leskovec et al. has argued that in real social-network data, it is much easier to separate a tightly-knit region from the rest of the network when it is relatively small, on the order of at most a few hundred nodes [275]. Studies on a range of different social and information networks suggest that beyond this size, sets of nodes become much more "inextricable" from the rest of the network, suggesting that graph partitioning approaches on this type of data may produce qualitatively different kinds of results for small networks and small regions than for large ones. This is an area of ongoing investigation.

In the remainder of this section, we address a final important issue: how to actually compute the betweenness quantities that are needed in order to make the Girvan-Newman method work.

B. Computing Betweenness Values

In order to perform the Girvan-Newman method, we need a way to find the edges of highest betweenness in each step. This is done by computing all the betweennesses of all edges and then looking for the ones with the highest values. The tricky part is that the definition of betweenness involves reasoning about the set of *all* the shortest paths between pairs of nodes. Since there could be a very large number of such shortest paths, how can we efficiently compute betweenness without the overhead of actually listing out all such paths? This is crucial for implementing the method on a computer to work with datasets of any reasonable size.

In fact, there is a clever way to compute betweennesses efficiently [77, 317], and it is based on the notion of breadth-first search from Section 2.3. We will consider the graph from the perspective of one node at a time; for each given node, we will compute how the total flow from that node to all others is distributed over the edges. If we do this for every node, then we can simply add up the flows from all of them to get the betweennesses on every edge.

So let's consider how we would determine the flow from one node to all other nodes in the graph. As an example, we'll look at the graph in Figure 3.18(a), focusing on how the flow from node A reaches all other nodes. We do this in three high-level steps; below we explain the details of how each of these steps works.

(1) Perform a breadth-first search of the graph, starting at A.

(2) Determine the number of shortest paths from A to each other node.

(3) Based on these numbers, determine the amount of flow from A to all other nodes that uses each edge.

For the first step, recall that breadth-first search divides a graph into *layers* starting at a given node (A in our case), with all the nodes in layer d having distance d from A. Moreover, the shortest paths from A to a node X in layer d are precisely the paths that move downward from A to X one layer at a time, thereby taking exactly d steps. Figure 3.18(b) shows the result of breadth-first search from A in our graph, with the layers placed horizontally going downward from A. Thus, for example, some inspection of the figure shows that there are two shortest paths (each of length two) from A to F: one using nodes A, B, and F, and the other using nodes A, C, and F.

Counting Shortest Paths. Now, let's consider the second step: determining the number of shortest paths from A to each other node. There is a remarkably clean way to do this, by working down through the layers of the breadth-first search.

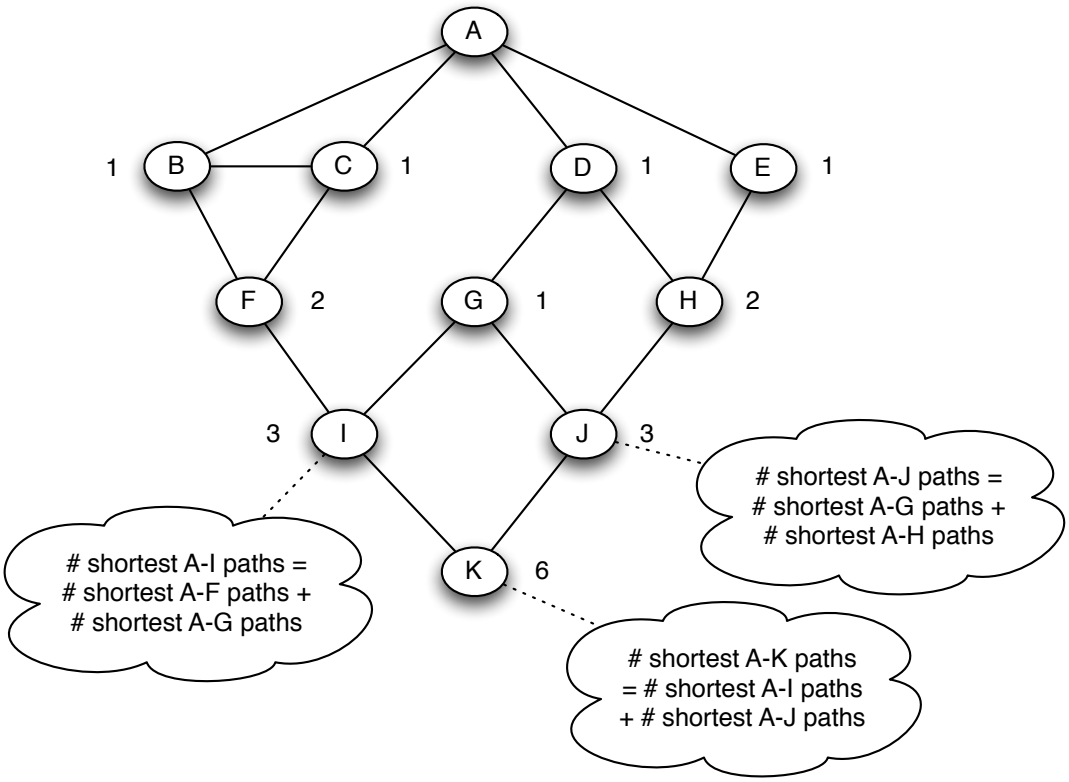


Figure 3.19: The second step in computing betweenness values is to count the number of shortest paths from a starting node A to all other nodes in the network. This can be done by adding up counts of shortest paths, moving downward through the breadth-first search structure.

To motivate this, consider a node like I in Figure 3.18(b). All shortest-paths from A to I must take their last step through either F or G, since these are the two nodes above it in the breadth-first search. (For terminological convenience, we will say that a node X is *above* a node Y in the breadth-first search if X is in the layer immediately preceding Y, and X has an edge to Y.) Moreover, in order to be a shortest path to I, a path must first be a shortest path to one of F or G, and then take this last step to I. It follows that the number of shortest paths from A to I is precisely the number of shortest paths from A to F, plus the number of shortest paths from A to G.

We can use this as a general method to count the number of shortest paths from A to all other nodes, as depicted in Figure 3.19. Each node in the first layer is a neighbor of A, and so it has only one shortest path from A: the edge leading straight from A to it. So we give each of these nodes a count of 1. Now, as we move down through the BFS layers, we apply the reasoning discussed above to conclude that the number of shortest paths to

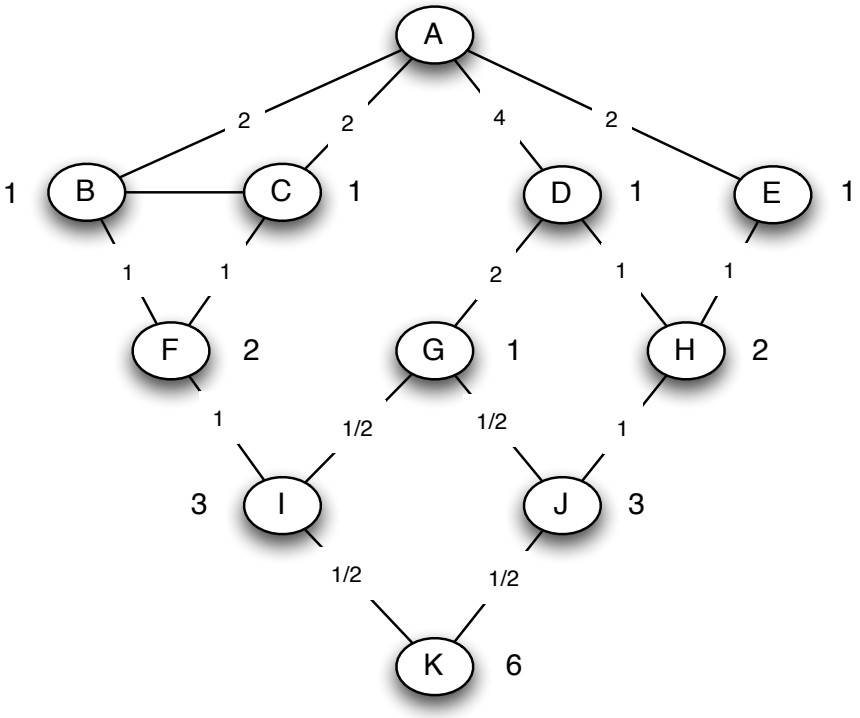


Figure 3.20: The final step in computing betweenness values is to determine the flow values from a starting node A to all other nodes in the network. This is done by working up from the lowest layers of the breadth-first search, dividing up the flow above a node in proportion to the number of shortest paths coming into it on each edge.

each node should be the *sum* of the number of shortest paths to all nodes directly above it in the breadth-first search. Working downward through the layers, we thus get the number of shortest paths to each node, as shown in Figure 3.19. Note that by the time we get to deeper layers, it may not be so easy to determine these number by visual inspection — for example, to immediately list the six different shortest paths from A to K — but it is quite easy when they are built up layer-by-layer in this way.

Determining Flow Values. Finally, we come to the third step, computing how the flow from A to all other nodes spreads out across the edges. Here too we use the breadth-first search structure, but this time working up from the lowest layers. We first show the idea in Figure 3.20 on our running example, and then describe the general procedure.

- Let's start at the bottom with node K. A single unit of flow arrives at K, and an equal number of the shortest paths from A to K come through nodes I and J, so this unit

of flow is equally divided over the two incoming edges. Therefore we put a half-unit of flow on each of these edges.

- Now, working upward, the total amount of flow arriving at I is equal to the one unit actually destined for I plus the half-unit passing through to K, for a total of $3/2$. How does this $3/2$ amount of flow get divided over the edges leading upward from I, to F and G respectively? We see from the second step that there are twice as many shortest paths from A through F as through G, so twice as much of the flow should come from F. Therefore, we put one unit of the flow on F, and a half-unit of the flow on G, as indicated in the figure.

- We continue in this way for each other node, working upward through the layers of the breadth-first search.

From this, it is not hard to describe the principle in general. When we get to a node X in the breadth-first search structure, working up from the bottom, we add up all the flow arriving from edges directly below X, plus 1 for the flow destined for X itself. We then divide this up over the edges leading upward from X, *in proportion to the number of shortest paths coming through each.* You can check that applying this principle leads to the numbers shown in Figure 3.20.

We are now essentially done. We build one of these breadth-first structures from *each* node in the network, determine flow values from the node using this procedure, and then sum up the flow values to get the betweenness value for each edge. Notice that we are counting the flow between each pair of nodes X and Y twice: once when we do the breadth-first search from X, and once when we do it from Y. So at the end we divide everything by two to cancel out this double-counting. Finally, using these betweenness values, we can identify the edges of highest betweenness for purposes of removing them in the Girvan-Newman method.

Final Observations. The method we have just described can be used to compute the betweennesses of nodes as well as edges. In fact, this is already happening in the third step: notice that we are implicitly keeping track of the amounts of flow through the nodes as well as through the edges, and this is what is needed to determine the betweennesses of the nodes.

The original Girvan-Newman method described here, based on repeated removal of high-betweenness edges, is a good conceptual way to think about graph partitioning, and it works well on networks of moderate size (up to a few thousand nodes). However, for larger networks, the need to recompute betweenness values in every step becomes computationally very expensive. In view of this, a range of different alternatives have been proposed to identify similar sets of tightly-knit regions more efficiently. These include methods of approximating the betweenness [34] and related but more efficient graph partitioning approaches using

divisive and agglomerative methods [35, 321]. There remains considerable interest in finding fast partitioning algorithms that can scale to very large network datasets.

3.7 Exercises

1. In 2-3 sentences, explain what triadic closure is, and how it plays a role in the formation of social networks. You can draw a schematic picture in case this is useful.

2. Consider the graph in Figure 3.21, in which each edge — except the edge connecting b and c — is labeled as a strong tie (S) or a weak tie (W).

   According to the theory of strong and weak ties, with the strong triadic closure assumption, how would you expect the edge connecting b and c to be labeled? Give a brief (1-3 sentence) explanation for your answer.

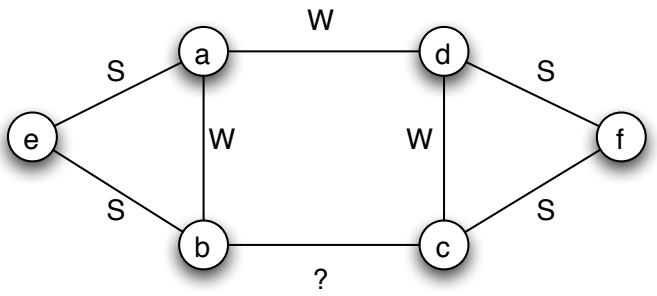


Figure 3.21:

3. In the social network depicted in Figure 3.22, with each edge labeled as either a strong or weak tie, which nodes satisfy the Strong Triadic Closure Property from Chapter 3, and which do not? Provide an explanation for your answer.

4. In the social network depicted in Figure 3.23 with each edge labeled as either a strong or weak tie, which two nodes violate the Strong Triadic Closure Property? Provide an explanation for your answer.

5. In the social network depicted in Figure 3.24, with each edge labeled as either a strong or weak tie, which nodes satisfy the Strong Triadic Closure Property from Chapter 3, and which do not? Provide an explanation for your answer.

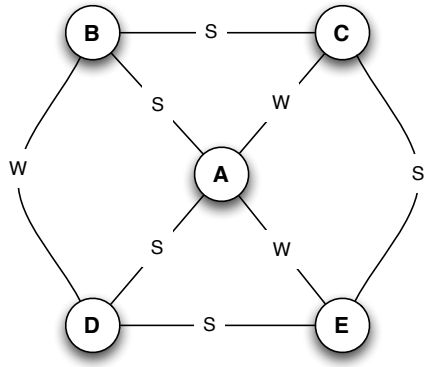


Figure 3.22:

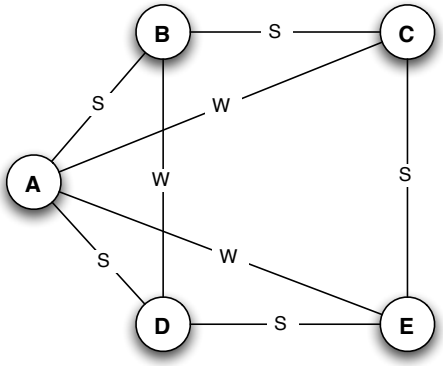


Figure 3.23: A graph with a strong/weak labeling.

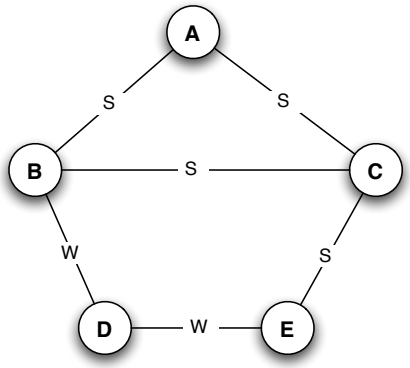


Figure 3.24: