Chapter 10

Mining Social-Network Graphs

There is much information to be gained by analyzing the large-scale data that is derived from social networks. The best-known example of a social network is the "friends" relation found on sites like Facebook. However, as we shall see there are many other sources of data that connect people or other entities.

In this chapter, we shall study techniques for analyzing such networks. An important question about a social network is how to identify "communities," that is, subsets of the nodes (people or other entities that form the network) with unusually strong connections. Some of the techniques used to identify communities are similar to the clustering algorithms we discussed in Chapter 7. However, communities almost never partition the set of nodes in a network. Rather, communities usually overlap. For example, you may belong to several communities of friends or classmates. The people from one community tend to know each other, but people from two different communities rarely know each other. You would not want to be assigned to only one of the communities, nor would it make sense to cluster all the people from all your communities into one cluster.

Also in this chapter we explore efficient algorithms for discovering other properties of graphs. We look at "simrank," a way to discover similarities among nodes of a graph. We explore triangle counting as a way to measure the connectedness of a community. We give efficient algorithms for exact and approximate measurement of the neighborhood sizes of nodes in a graph. Finally, we look at efficient algorithms for computing the transitive closure.

10.1 Social Networks as Graphs

We begin our discussion of social networks by introducing a graph model. Not every graph is a suitable representation of what we intuitively regard as a social

network. We therefore discuss the idea of "locality," the property of social networks that says nodes and edges of the graph tend to cluster in communities. This section also looks at some of the kinds of social networks that occur in practice.

10.1.1 What is a Social Network?

When we think of a social network, we think of Facebook, Twitter, Google+, or another website that is called a "social network," and indeed this kind of network is representative of the broader class of networks called "social." The essential characteristics of a social network are:

1. There is a collection of entities that participate in the network. Typically, these entities are people, but they could be something else entirely. We shall discuss some other examples in Section 10.1.3.

2. There is at least one relationship between entities of the network. On Facebook or its ilk, this relationship is called *friends*. Sometimes the relationship is all-or-nothing; two people are either friends or they are not. However, in other examples of social networks, the relationship has a degree. This degree could be discrete; e.g., friends, family, acquaintances, or none as in Google+. It could be a real number; an example would be the fraction of the average day that two people spend talking to each other.

3. There is an assumption of nonrandomness or locality. This condition is the hardest to formalize, but the intuition is that relationships tend to cluster. That is, if entity A is related to both B and C, then there is a higher probability than average that B and C are related.

10.1.2 Social Networks as Graphs

Social networks are naturally modeled as graphs, which we sometimes refer to as a *social graph*. The entities are the nodes, and an edge connects two nodes if the nodes are related by the relationship that characterizes the network. If there is a degree associated with the relationship, this degree is represented by labeling the edges. Often, social graphs are undirected, as for the Facebook friends graph. But they can be directed graphs, as for example the graphs of followers on Twitter or Google+.

Example 10.1: Figure 10.1 is an example of a tiny social network. The entities are the nodes A through G. The relationship, which we might think of as "friends," is represented by the edges. For instance, B is friends with A, C, and D.

Is this graph really typical of a social network, in the sense that it exhibits locality of relationships? First, note that the graph has nine edges out of the

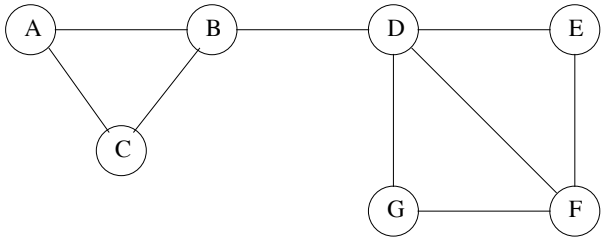

Figure 10.1: Example of a small social network

$\binom{7}{2} = 21$ pairs of nodes that could have had an edge between them. Suppose X, Y, and Z are nodes of Fig. 10.1, with edges between X and Y and also between X and Z. What would we expect the probability of an edge between Y and Z to be? If the graph were large, that probability would be very close to the fraction of the pairs of nodes that have edges between them, i.e., 9/21 = .429 in this case. However, because the graph is small, there is a noticeable difference between the true probability and the ratio of the number of edges to the number of pairs of nodes. Since we already know there are edges (X, Y) and (X, Z), there are only seven edges remaining. Those seven edges could run between any of the 19 remaining pairs of nodes. Thus, the probability of an edge (Y, Z) is 7/19 = .368.

Now, we must compute the probability that the edge (Y, Z) exists in Fig. 10.1, given that edges (X, Y) and (X, Z) exist. What we shall actually count is pairs of nodes that could be Y and Z, without worrying about which node is Y and which is Z. If X is A, then Y and Z must be B and C, in some order. Since the edge (B, C) exists, A contributes one positive example (where the edge does exist) and no negative examples (where the edge is absent). The cases where X is C, E, or G are essentially the same. In each case, X has only two neighbors, and the edge between the neighbors exists. Thus, we have seen four positive examples and zero negative examples so far.

Now, consider $X = F$. F has three neighbors, D, E, and G. There are edges between two of the three pairs of neighbors, but no edge between G and E. Thus, we see two more positive examples and we see our first negative example. If $X = B$, there are again three neighbors, but only one pair of neighbors, A and C, has an edge. Thus, we have two more negative examples, and one positive example, for a total of seven positive and three negative. Finally, when $X = D$, there are four neighbors. Of the six pairs of neighbors, only two have edges between them.

Thus, the total number of positive examples is nine and the total number of negative examples is seven. We see that in Fig. 10.1, the fraction of times the third edge exists is thus 9/16 = .563. This fraction is considerably greater than the .368 expected value for that fraction. We conclude that Fig. 10.1 does indeed exhibit the locality expected in a social network. □

10.1.3 Varieties of Social Networks

There are many examples of social networks other than "friends" networks. Here, let us enumerate some of the other examples of networks that also exhibit locality of relationships.

Telephone Networks

Here the nodes represent phone numbers, which are really individuals. There is an edge between two nodes if a call has been placed between those phones in some fixed period of time, such as last month, or "ever." The edges could be weighted by the number of calls made between these phones during the period. Communities in a telephone network will form from groups of people that communicate frequently: groups of friends, members of a club, or people working at the same company, for example.

Email Networks

The nodes represent email addresses, which are again individuals. An edge represents the fact that there was at least one email in at least one direction between the two addresses. Alternatively, we may only place an edge if there were emails in both directions. In that way, we avoid viewing spammers as "friends" with all their victims. Another approach is to label edges as *weak* or *strong*. Strong edges represent communication in both directions, while weak edges indicate that the communication was in one direction only. The communities seen in email networks come from the same sorts of groupings we mentioned in connection with telephone networks. A similar sort of network involves people who text other people through their cell phones.

Collaboration Networks

Nodes represent individuals who have published research papers. There is an edge between two individuals who published one or more papers jointly. Optionally, we can label edges by the number of joint publications. The communities in this network are authors working on a particular topic.

An alternative view of the same data is as a graph in which the nodes are papers. Two papers are connected by an edge if they have at least one author in common. Now, we form communities that are collections of papers on the same topic.

There are several other kinds of data that form two networks in a similar way. For example, we can look at the people who edit Wikipedia articles and the articles that they edit. Two editors are connected if they have edited an article in common. The communities are groups of editors that are interested in the same subject. Dually, we can build a network of articles, and connect articles if they have been edited by the same person. Here, we get communities of articles on similar or related subjects.

In fact, the data involved in Collaborative filtering, as was discussed in Chapter 9, often can be viewed as forming a pair of networks, one for the customers and one for the products. Customers who buy the same sorts of products, e.g., science-fiction books, will form communities, and dually, products that are bought by the same customers will form communities, e.g., all science-fiction books.

Other Examples of Social Graphs

Many other phenomena give rise to graphs that look something like social graphs, especially exhibiting locality. Examples include: information networks (documents, web graphs, patents), infrastructure networks (roads, planes, water pipes, powergrids), biological networks (genes, proteins, food-webs of animals eating each other), as well as other types, like product co-purchasing networks (e.g., Groupon).

10.1.4 Graphs With Several Node Types

There are other social phenomena that involve entities of different types. We just discussed under the heading of "collaboration networks," several kinds of graphs that are really formed from two types of nodes. Authorship networks can be seen to have author nodes and paper nodes. In the discussion above, we built two social networks by eliminating the nodes of one of the two types, but we do not have to do that. We can rather think of the structure as a whole.

For a more complex example, users at a site like deli.cio.us place tags on Web pages. There are thus three different kinds of entities: users, tags, and pages. We might think that users were somehow connected if they tended to use the same tags frequently, or if they tended to tag the same pages. Similarly, tags could be considered related if they appeared on the same pages or were used by the same users, and pages could be considered similar if they had many of the same tags or were tagged by many of the same users.

The natural way to represent such information is as a k-partite graph for some $k > 1$. We met bipartite graphs, the case $k = 2$, in Section 8.3. In general, a *k-partite graph* consists of k disjoint sets of nodes, with no edges between nodes of the same set.

Example 10.2: Figure 10.2 is an example of a tripartite graph (the case $k = 3$ of a k-partite graph). There are three sets of nodes, which we may think of as users $\{U_1, U_2\}$, tags $\{T_1, T_2, T_3, T_4\}$, and Web pages $\{W_1, W_2, W_3\}$. Notice that all edges connect nodes from two different sets. We may assume this graph represents information about the three kinds of entities. For example, the edge (U_1, T_2) means that user U_1 has placed the tag T_2 on at least one page. Note that the graph does not tell us a detail that could be important: who placed which tag on which page? To represent such ternary information would require a more complex representation, such as a database relation with three columns corresponding to users, tags, and pages. □

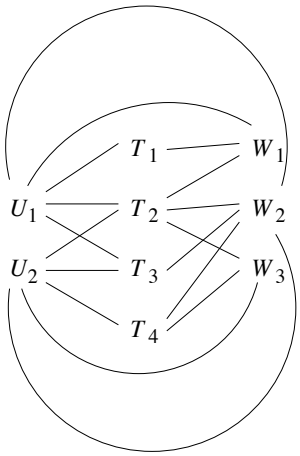

Figure 10.2: A tripartite graph representing users, tags, and Web pages

10.1.5 Exercises for Section 10.1

Exercise 10.1.1: It is possible to think of the edges of one graph G as the nodes of another graph G'. We construct G' from G by the *dual construction*:

1. If (X, Y) is an edge of G, then XY, representing the unordered set of X and Y is a node of G'. Note that XY and YX represent the same node of G', not two different nodes.

2. If (X, Y) and (X, Z) are edges of G, then in G' there is an edge between XY and XZ. That is, nodes of G' have an edge between them if the edges of G that these nodes represent have a node (of G) in common.

(a) If we apply the dual construction to a network of friends, what is the interpretation of the edges of the resulting graph?

(b) Apply the dual construction to the graph of Fig. 10.1.

! (c) How is the degree of a node XY in G' related to the degrees of X and Y in G?

!! (d) The number of edges of G' is related to the degrees of the nodes of G by a certain formula. Discover that formula.

! (e) What we called the dual is not a true dual, because applying the construction to G' does not necessarily yield a graph isomorphic to G. Give an example graph G where the dual of G' *is* isomorphic to G and another example where the dual of G' is *not* isomorphic to G.

10.2 Clustering of Social-Network Graphs

An important aspect of social networks is that they contain communities of entities that are connected by many edges. These typically correspond to groups of friends at school or groups of researchers interested in the same topic, for example. In this section, we shall consider clustering of the graph as a way to identify communities. It turns out that the techniques we learned in Chapter 7 are generally unsuitable for the problem of clustering social-network graphs.

10.2.1 Distance Measures for Social-Network Graphs

If we were to apply standard clustering techniques to a social-network graph, our first step would be to define a distance measure. When the edges of the graph have labels, these labels might be usable as a distance measure, depending on what they represented. But when the edges are unlabeled, as in a "friends" graph, there is not much we can do to define a suitable distance.

Our first instinct is to assume that nodes are close if they have an edge between them and distant if not. Thus, we could say that the distance $d(x, y)$ is 0 if there is an edge (x, y) and 1 if there is no such edge. We could use any other two values, such as 1 and ∞, as long as the distance is closer when there is an edge.

Neither of these two-valued "distance measures" – 0 and 1 or 1 and ∞ – is a true distance measure. The reason is that they violate the triangle inequality when there are three nodes, with two edges between them. That is, if there are edges (A, B) and (B, C), but no edge (A, C), then the distance from A to C exceeds the sum of the distances from A to B to C. We could fix this problem by using, say, distance 1 for an edge and distance 1.5 for a missing edge. But the problem with two-valued distance functions is not limited to the triangle inequality, as we shall see in the next section.

10.2.2 Applying Standard Clustering Methods

Recall from Section 7.1.2 that there are two general approaches to clustering: hierarchical (agglomerative) and point-assignment. Let us consider how each of these would work on a social-network graph. First, consider the hierarchical methods covered in Section 7.2. In particular, suppose we use as the intercluster distance the minimum distance between nodes of the two clusters.

Hierarchical clustering of a social-network graph starts by combining some two nodes that are connected by an edge. Successively, edges that are not between two nodes of the same cluster would be chosen randomly to combine the clusters to which their two nodes belong. The choices would be random, because all distances represented by an edge are the same.

Example 10.3: Consider again the graph of Fig. 10.1, repeated here as Fig. 10.3. First, let us agree on what the communities are. At the highest level,

it appears that there are two communities $\{A, B, C\}$ and $\{D, E, F, G\}$. However, we could also view $\{D, E, F\}$ and $\{D, F, G\}$ as two subcommunities of $\{D, E, F, G\}$; these two subcommunities overlap in two of their members, and thus could never be identified by a pure clustering algorithm. Finally, we could consider each pair of individuals that are connected by an edge as a community of size 2, although such communities are uninteresting.

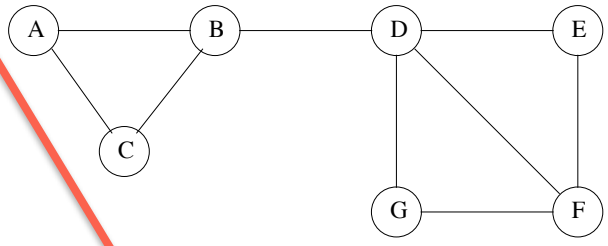

Figure 10.3: Repeat of Fig. 10.1

The problem with hierarchical clustering of a graph like that of Fig. 10.3 is that at some point we are likely to chose to combine B and D, even though they surely belong in different clusters. The reason we are likely to combine B and D is that D, and any cluster containing it, is as close to B and any cluster containing it, as A and C are to B. There is even a 1/9 probability that the first thing we do is to combine B and D into one cluster.

There are things we can do to reduce the probability of error. We can run hierarchical clustering several times and pick the run that gives the most coherent clusters. We can use a more sophisticated method for measuring the distance between clusters of more than one node, as discussed in Section 7.2.3. But no matter what we do, in a large graph with many communities there is a significant chance that in the initial phases we shall use some edges that connect two nodes that do not belong together in any large community. □

Now, consider a point-assignment approach to clustering social networks. Again, the fact that all edges are at the same distance will introduce a number of random factors that will lead to some nodes being assigned to the wrong cluster. An example should illustrate the point.

Example 10.4: Suppose we try a k-means approach to clustering Fig. 10.3. As we want two clusters, we pick $k = 2$. If we pick two starting nodes at random, they might both be in the same cluster. If, as suggested in Section 7.3.2, we start with one randomly chosen node and then pick another as far away as possible, we don't do much better; we could thereby pick any pair of nodes not connected by an edge, e.g., E and G in Fig. 10.3.

However, suppose we do get two suitable starting nodes, such as B and F. We shall then assign A and C to the cluster of B and assign E and G to the cluster of F. But D is as close to B as it is to F, so it could go either way, even though it is "obvious" that D belongs with F.

If the decision about where to place D is deferred until we have assigned some other nodes to the clusters, then we shall probably make the right decision. For instance, if we assign a node to the cluster with the shortest average distance to all the nodes of the cluster, then D should be assigned to the cluster of F, as long as we do not try to place D before any other nodes are assigned. However, in large graphs, we shall surely make mistakes on some of the first nodes we place. □

10.2.3 Betweenness

Since there are problems with standard clustering methods, several specialized clustering techniques have been developed to find communities in social networks. In this section we shall consider one of the simplest, based on finding the edges that are least likely to be inside a community.

Define the *betweenness* of an edge (a, b) to be the number of pairs of nodes x and y such that the edge (a, b) lies on the shortest path between x and y. To be more precise, since there can be several shortest paths between x and y, edge (a, b) is credited with the fraction of those shortest paths that include the edge (a, b). As in golf, a high score is bad. It suggests that the edge (a, b) runs between two different communities; that is, a and b do not belong to the same community.

Example 10.5: In Fig. 10.3 the edge (B, D) has the highest betweenness, as should surprise no one. In fact, this edge is on every shortest path between any of A, B, and C to any of D, E, F, and G. Its betweenness is therefore $3 \times 4 = 12$. In contrast, the edge (D, F) is on only four shortest paths: those from A, B, C, and D to F. □

10.2.4 The Girvan-Newman Algorithm

In order to exploit the betweenness of edges, we need to calculate the number of shortest paths going through each edge. We shall describe a method called the *Girvan-Newman* (GN) Algorithm, which visits each node X once and computes the number of shortest paths from X to each of the other nodes that go through each of the edges. The algorithm begins by performing a breadth-first search (BFS) of the graph, starting at the node X. Note that the level of each node in the BFS presentation is the length of the shortest path from X to that node. Thus, the edges that go between nodes at the same level can never be part of a shortest path from X.

Edges between levels are called *DAG* edges ("DAG" stands for directed, acyclic graph). Each DAG edge will be part of at least one shortest path from root X. If there is a DAG edge (Y, Z), where Y is at the level above Z (i.e., closer to the root), then we shall call Y a *parent* of Z and Z a *child* of Y, although parents are not necessarily unique in a DAG as they would be in a tree.

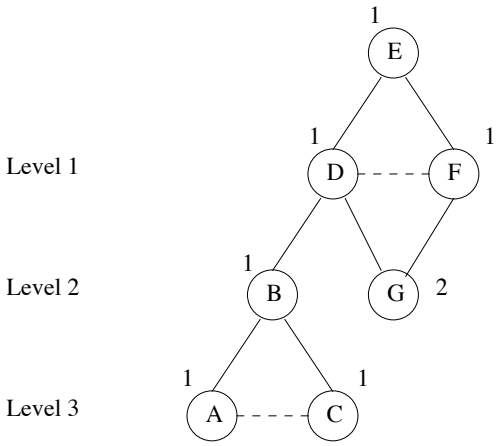

Figure 10.4: Step 1 of the Girvan-Newman Algorithm

Example 10.6: Figure 10.4 is a breadth-first presentation of the graph of Fig. 10.3, starting at node E. Solid edges are DAG edges and dashed edges connect nodes at the same level. □

The second step of the GN algorithm is to label each node by the number of shortest paths that reach it from the root. Start by labeling the root 1. Then, from the top down, label each node Y by the sum of the labels of its parents.

Example 10.7: In Fig. 10.4 are the labels for each of the nodes. First, label the root E with 1. At level 1 are the nodes D and F. Each has only E as a parent, so they too are labeled 1. Nodes B and G are at level 2. B has only D as a parent, so B's label is the same as the label of D, which is 1. However, G has parents D and F, so its label is the sum of their labels, or 2. Finally, at level 3, A and C each have only parent B, so their labels are the label of B, which is 1. □

The third and final step is to calculate for each edge e the sum over all nodes Y of the fraction of shortest paths from the root X to Y that go through e. This calculation involves computing this sum for both nodes and edges, from the bottom. Each node other than the root is given a *credit* of 1, representing the shortest path to that node. This credit may be divided among nodes and edges above, since there could be several different shortest paths to the node. The rules for the calculation are as follows:

1. Each leaf in the DAG (a *leaf* is a node with no DAG edges to nodes at levels below) gets a credit of 1.

2. Each node that is not a leaf gets a credit equal to 1 plus the sum of the credits of the DAG edges from that node to the level below.

3. A DAG edge e entering node Z from the level above is given a share of the credit of Z proportional to the fraction of shortest paths from the root to Z that go through e. Formally, let the parents of Z be $Y_1, Y_2, \ldots, Y_k$. Let p_i be the number of shortest paths from the root to Y_i; this number was computed in Step 2 and is illustrated by the labels in Fig. 10.4. Then the credit for the edge (Y_i, Z) is the credit of Z times p_i divided by $\sum_{j=1}^{k} p_j$.

After performing the credit calculation with each node as the root, we sum the credits for each edge. Then, since each shortest path will have been discovered twice – once when each of its endpoints is the root – we must divide the credit for each edge by 2.

Example 10.8: Let us perform the credit calculation for the BFS presentation of Fig. 10.4. We shall start from level 3 and proceed upwards. First, A and C, being leaves, get credit 1. Each of these nodes have only one parent, so their credit is given to the edges (B, A) and (B, C), respectively.

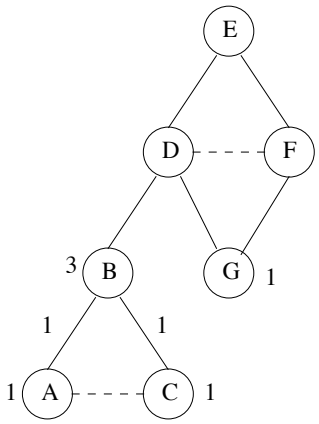

Figure 10.5: Final step of the Girvan-Newman Algorithm – levels 3 and 2

At level 2, G is a leaf, so it gets credit 1. B is not a leaf, so it gets credit equal to 1 plus the credits on the DAG edges entering it from below. Since both these edges have credit 1, the credit of B is 3. Intuitively 3 represents the fact that all shortest paths from E to A, B, and C go through B. Figure 10.5 shows the credits assigned so far.

Now, let us proceed to level 1. B has only one parent, D, so the edge (D, B) gets the entire credit of B, which is 3. However, G has two parents, D and F. We therefore need to divide the credit of 1 that G has between the edges (D, G) and (F, G). In what proportion do we divide? If you examine the labels of Fig. 10.4, you see that both D and F have label 1, representing the fact that there is one shortest path from E to each of these nodes. Thus, we give half the credit of G to each of these edges; i.e., their credit is each $1/(1+1) = 0.5$. Had the labels of D and F in Fig. 10.4 been 5 and 3, meaning there were five

shortest paths to D and only three to F, then the credit of edge (D, G) would have been 5/8 and the credit of edge (F, G) would have been 3/8.

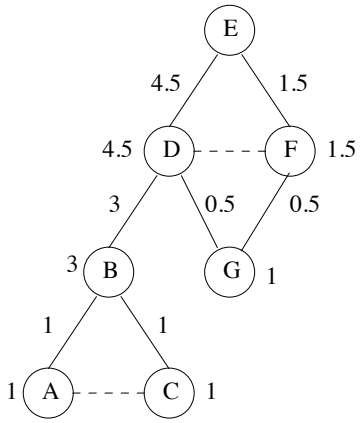

Figure 10.6: Final step of the Girvan-Newman Algorithm – completing the credit calculation

Now, we can assign credits to the nodes at level 1. D gets 1 plus the credits of the edges entering it from below, which are 3 and 0.5. That is, the credit of D is 4.5. The credit of F is 1 plus the credit of the edge (F, G), or 1.5. Finally, the edges (E, D) and (E, F) receive the credit of D and F, respectively, since each of these nodes has only one parent. These credits are all shown in Fig. 10.6.

The credit on each of the edges in Fig. 10.6 is the contribution to the betweenness of that edge due to shortest paths from E. For example, this contribution for the edge (E, D) is 4.5. □

To complete the betweenness calculation, we have to repeat this calculation for every node as the root and sum the contributions. Finally, we must divide by 2 to get the true betweenness, since every shortest path will be discovered twice, once for each of its endpoints.

10.2.5 Using Betweenness to Find Communities

The betweenness scores for the edges of a graph behave something like a distance measure on the nodes of the graph. It is not exactly a distance measure, because it is not defined for pairs of nodes that are unconnected by an edge, and might not satisfy the triangle inequality even when defined. However, we can cluster by taking the edges in order of increasing betweenness and add them to the graph one at a time. At each step, the connected components of the graph form some clusters. The higher the betweenness we allow, the more edges we get, and the larger the clusters become.

More commonly, this idea is expressed as a process of edge removal. Start with the graph and all its edges; then remove edges with the highest between-

ness, until the graph has broken into a suitable number of connected components.

Example 10.9 : Let us start with our running example, the graph of Fig. 10.1. We see it with the betweenness for each edge in Fig. 10.7. The calculation of the betweenness will be left to the reader. The only tricky part of the count is to observe that between E and G there are two shortest paths, one going through D and the other through F. Thus, each of the edges (D, E), (E, F), (D, G), and (G, F) are credited with half a shortest path.

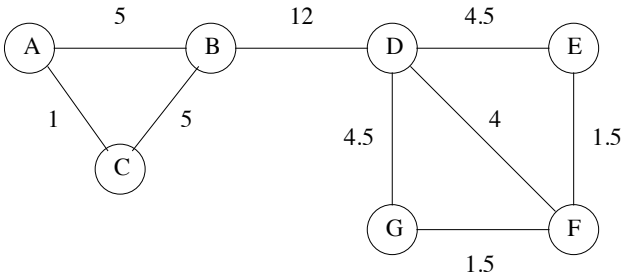

Figure 10.7: Betweenness scores for the graph of Fig. 10.1

Clearly, edge (B, D) has the highest betweenness, so it is removed first. That leaves us with exactly the communities we observed make the most sense, namely: $\{A, B, C\}$ and $\{D, E, F, G\}$. However, we can continue to remove edges. Next to leave are (A, B) and (B, C) with a score of 5, followed by (D, E) and (D, G) with a score of 4.5. Then, (D, F), whose score is 4, would leave the graph. We see in Fig. 10.8 the graph that remains.

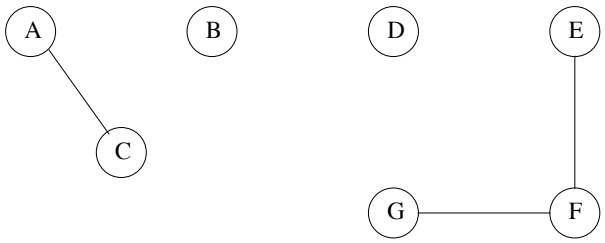

Figure 10.8: All the edges with betweenness 4 or more have been removed

The "communities" of Fig. 10.8 look strange. One implication is that A and C are more closely knit to each other than to B. That is, in some sense B is a "traitor" to the community $\{A, B, C\}$ because he has a friend D outside that community. Likewise, D can be seen as a "traitor" to the group $\{D, E, F, G\}$, which is why in Fig. 10.8, only E, F, and G remain connected. □

Speeding Up the Betweenness Calculation

If we apply the method of Section 10.2.4 to a graph of n nodes and e edges, it takes $O(ne)$ running time to compute the betweenness of each edge. That is, BFS from a single node takes $O(e)$ time, as do the two labeling steps. We must start from each node, so there are n of the computations described in Section 10.2.4.

If the graph is large – and even a million nodes is large when the algorithm takes $O(ne)$ time – we cannot afford to execute it as suggested. However, if we pick a subset of the nodes at random and use these as the roots of breadth-first searches, we can get an approximation to the betweenness of each edge that will serve in most applications.

10.2.6 Exercises for Section 10.2

Exercise 10.2.1: Figure 10.9 is an example of a social-network graph. Use the Girvan-Newman approach to find the number of shortest paths from each of the following nodes that pass through each of the edges. (a) A (b) B.

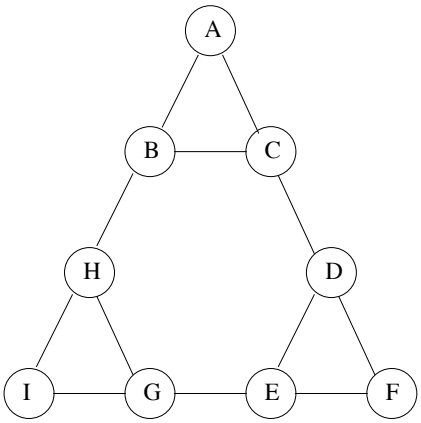

Figure 10.9: Graph for exercises

Exercise 10.2.2: Using symmetry, the calculations of Exercise 10.2.1 are all you need to compute the betweenness of each edge. Do the calculation.

Exercise 10.2.3: Using the betweenness values from Exercise 10.2.2, determine reasonable candidates for the communities in Fig. 10.9 by removing all edges with a betweenness above some threshold.

10.3 Direct Discovery of Communities

In the previous section we searched for communities by partitioning all the individuals in a social network. While this approach is relatively efficient, it does have several limitations. It is not possible to place an individual in two different communities, and everyone is assigned to a community. In this section, we shall see a technique for discovering communities directly by looking for subsets of the nodes that have a relatively large number of edges among them. Interestingly, the technique for doing this search on a large graph involves finding large frequent itemsets, as was discussed in Chapter 6.

10.3.1 Finding Cliques

Our first thought about how we could find sets of nodes with many edges between them is to start by finding a large *clique* (a set of nodes with edges between any two of them). However, that task is not easy. Not only is finding maximal cliques NP-complete, but it is among the hardest of the NP-complete problems in the sense that even approximating the maximal clique is hard. Further, it is possible to have a set of nodes with almost all edges between them, and yet have only relatively small cliques.

Example 10.10: Suppose our graph has nodes numbered $1, 2, \ldots, n$ and there is an edge between two nodes i and j unless i and j have the same remainder when divided by k. Then the fraction of possible edges that are actually present is approximately $(k-1)/k$. There are many cliques of size k, of which $\{1, 2, \ldots, k\}$ is but one example.

Yet there are no cliques larger than k. To see why, observe that any set of $k+1$ nodes has two that leave the same remainder when divided by k. This point is an application of the "pigeonhole principle." Since there are only k different remainders possible, we cannot have distinct remainders for each of $k+1$ nodes. Thus, no set of $k+1$ nodes can be a clique in this graph. □

10.3.2 Complete Bipartite Graphs

Recall our discussion of bipartite graphs from Section 8.3. A *complete bipartite graph* consists of s nodes on one side and t nodes on the other side, with all st possible edges between the nodes of one side and the other present. We denote this graph by $K_{s,t}$. You should draw an analogy between complete bipartite graphs as subgraphs of general bipartite graphs and cliques as subgraphs of general graphs. In fact, a clique of s nodes is often referred to as a *complete graph* and denoted K_s, while a complete bipartite subgraph is sometimes called a *bi-clique*.

While as we saw in Example 10.10, it is not possible to guarantee that a graph with many edges necessarily has a large clique, it *is* possible to guarantee that a bipartite graph with many edges has a large complete bipartite

subgraph.[1] We can regard a complete bipartite subgraph (or a clique if we discovered a large one) as the nucleus of a community and add to it nodes with many edges to existing members of the community. If the graph itself is k-partite as discussed in Section 10.1.4, then we can take nodes of two types and the edges between them to form a bipartite graph. In this bipartite graph, we can search for complete bipartite subgraphs as the nuclei of communities. For instance, in Example 10.2, we could focus on the tag and page nodes of a graph like Fig. 10.2 and try to find communities of tags and Web pages. Such a community would consist of related tags and related pages that deserved many or all of those tags.

However, we can also use complete bipartite subgraphs for community finding in ordinary graphs where nodes all have the same type. Divide the nodes into two equal groups at random. If a community exists, then we would expect about half its nodes to fall into each group, and we would expect that about half its edges would go between groups. Thus, we still have a reasonable chance of identifying a large complete bipartite subgraph in the community. To this nucleus we can add nodes from either of the two groups, if they have edges to many of the nodes already identified as belonging to the community.

10.3.3 Finding Complete Bipartite Subgraphs

Suppose we are given a large bipartite graph G , and we want to find instances of $K_{s,t}$ within it. It is possible to view the problem of finding instances of $K_{s,t}$ within G as one of finding frequent itemsets. For this purpose, let the "items" be the nodes on one side of G, which we shall call the *left* side. We assume that the instance of $K_{s,t}$ we are looking for has t nodes on the left side, and we shall also assume for efficiency that $t \leq s$. The "baskets" correspond to the nodes on the other side of G (the *right* side). The members of the basket for node v are the nodes of the left side to which v is connected. Finally, let the support threshold be s, the number of nodes that the instance of $K_{s,t}$ has on the right side.

We can now state the problem of finding instances of $K_{s,t}$ as that of finding frequent itemsets F of size t. That is, if a set of t nodes on the left side is frequent, then they all occur together in at least s baskets. But the baskets are the nodes on the right side. Each basket corresponds to a node that is connected to all t of the nodes in F. Thus, the frequent itemset of size t and s of the baskets in which all those items appear form an instance of $K_{s,t}$.

Example 10.11 : Recall the bipartite graph of Fig. 8.1, which we repeat here as Fig. 10.10. The left side is the nodes $\{1, 2, 3, 4\}$ and the right side is $\{a, b, c, d\}$. The latter are the baskets, so basket a consists of "items" 1 and 4; that is, $a = \{1, 4\}$. Similarly, $b = \{2, 3\}$, $c = \{1\}$ and $d = \{3\}$.

[1] It is important to understand that we do not mean a *generated* subgraph – one formed by selecting some nodes and including all edges. In this context, we only require that there be edges between any pair of nodes on different sides. It is also possible that some nodes on the same side are connected by edges as well.

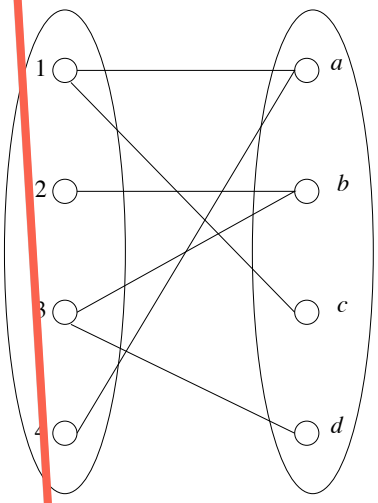

Figure 10.10: The bipartite graph from Fig. 8.1

If $s = 2$ and $t = 1$, we must find itemsets of size 1 that appear in at least two baskets. $\{1\}$ is one such itemset, and $\{3\}$ is another. However, in this tiny example there are no itemsets for larger, more interesting values of s and t, such as $s = t = 2$. □

10.3.4 Why Complete Bipartite Graphs Must Exist

We must now turn to the matter of demonstrating that any bipartite graph with a sufficiently high fraction of the edges present will have an instance of $K_{s,t}$. In what follows, assume that the graph G has n nodes on the left and another n nodes on the right. Assume the two sides have the same number of nodes simplifies the calculation, but the argument generalizes to sides of any size. Finally, let d be the average degree of all nodes.

The argument involves counting the number of frequent itemsets of size t that a basket with d items contributes to. When we sum this number over all nodes on the right side, we get the total frequency of all the subsets of size t on the left. When we divide by $\binom{n}{t}$, we get the average frequency of all itemsets of size t. At least one must have a frequency that is at least average, so if this average is at least s, we know an instance of $K_{s,t}$ exists.

Now, we provide the detailed calculation. Suppose the degree of the ith node on the right is d_i; that is, d_i is the size of the ith basket. Then this basket contributes to $\binom{d_i}{t}$ itemsets of size t. The total contribution of the n nodes on the right is $\sum_i \binom{d_i}{t}$. The value of this sum depends on the d_i's, of course. However, we know that the average value of d_i is d. It is known that this sum is minimized when each d_i is d. We shall not prove this point, but a simple example will suggest the reasoning: since $\binom{d_i}{t}$ grows roughly as the tth

power of d_i, moving 1 from a large d_i to some smaller d_j will reduce the sum of $\binom{d_i}{t} + \binom{d_j}{t}$.

Example 10.12: Suppose there are only two nodes, $t = 2$, and the average degree of the nodes is 4. Then $d_1 + d_2 = 8$, and the sum of interest is $\binom{d_1}{2} + \binom{d_2}{2}$. If $d_1 = d_2 = 4$, then the sum is $\binom{4}{2} + \binom{4}{2} = 6 + 6 = 12$. However, if $d_1 = 5$ and $d_2 = 3$, the sum is $\binom{5}{2} + \binom{3}{2} = 10 + 3 = 13$. If $d_1 = 6$ and $d_1 = 2$, then the sum is $\binom{6}{2} + \binom{2}{2} = 15 + 1 = 16$. □

Thus, in what follows, we shall assume that all nodes have the average degree d. So doing minimizes the total contribution to the counts for the itemsets, and thus makes it least likely that there will be a frequent itemset (itemset with with support s or more) of size t. Observe the following:

- The total contribution of the n nodes on the right to the counts of the itemsets of size t is $n\binom{d}{t}$.
- The number of itemsets of size t is $\binom{n}{t}$.
- Thus, the average count of an itemset of size t is $n\binom{d}{t}/\binom{n}{t}$; this expression must be at least s if we are to argue that an instance of $K_{s,t}$ exists.

If we expand the binomial coefficients in terms of factorials, we find

$$n\binom{d}{t}/\binom{n}{t} = nd!(n-t)!t!/\big((d-t)!t!n!\big) =$$

$$n(d)(d-1)\cdots(d-t+1)/\big(n(n-1)\cdots(n-t+1)\big)$$

To simplify the formula above, let us assume that n is much larger than d, and d is much larger than t. Then $d(d-1)\cdots(d-t+1)$ is approximately d^t, and $n(n-1)\cdots(n-t+1)$ is approximately n^t. We thus require that

$$n(d/n)^t \geq s$$

That is, if there is a community with n nodes on each side, the average degree of the nodes is d, and $n(d/n)^t \geq s$, then this community is guaranteed to have a complete bipartite subgraph $K_{s,t}$. Moreover, we can find the instance of $K_{s,t}$ efficiently, using the methods of Chapter 6, even if this small community is embedded in a much larger graph. That is, we can treat all nodes in the entire graph as baskets and as items, and run A-priori or one of its improvements on the entire graph, looking for sets of t items with support s.

Example 10.13: Suppose there is a community with 100 nodes on each side, and the average degree of nodes is 50; i.e., half the possible edges exist. This community will have an instance of $K_{s,t}$, provided $100(1/2)^t \geq s$. For example, if $t = 2$, then s can be as large as 25. If $t = 3$, s can be 11, and if $t = 4$, s can be 6.

Unfortunately, the approximation we made gives us a bound on s that is a little too high. If we revert to the original formula $n\binom{d}{t}/\binom{n}{t} \geq s$, we see that for the case $t = 4$ we need $100\binom{50}{4}/\binom{100}{4} \geq s$. That is,

$$\frac{100 \times 50 \times 49 \times 48 \times 47}{100 \times 99 \times 98 \times 97} \geq s$$

The expression on the left is not 6, but only 5.87. However, if the average support for an itemset of size 4 is 5.87, then it is impossible that all those itemsets have support 5 or less. Thus, we can be sure that at least one itemset of size 4 has support 6 or more, and an instance of $K_{6,4}$ exists in this community. □

10.3.5 Exercises for Section 10.3

Exercise 10.3.1: For the running example of a social network from Fig. 10.1, how many instances of $K_{s,t}$ are there for:

(a) $s = 1$ and $t = 3$.

(b) $s = 2$ and $t = 2$.

(c) $s = 2$ and $t = 3$.

Exercise 10.3.2: Suppose there is a community of $2n$ nodes. Divide the community into two groups of n members, at random, and form the bipartite graph between the two groups. Suppose that the average degree of the nodes of the bipartite graph is d. Find the set of maximal pairs (t, s), with $t \leq s$, such that an instance of $K_{s,t}$ is guaranteed to exist, for the following combinations of n and d:

(a) $n = 20$ and $d = 5$.

(b) $n = 200$ and $d = 150$.

(c) $n = 1000$ and $d = 400$.

By "maximal," we mean there is no different pair (s', t') such that both $s' \geq s$ and $t' \geq t$ hold.

10.4 Partitioning of Graphs

In this section, we examine another approach to organizing social-network graphs. We use some important tools from matrix theory ("spectral methods") to formulate the problem of partitioning a graph to minimize the number of edges that connect different components. The goal of minimizing the "cut" size needs to be understood carefully before proceeding. For instance, if you