Chapter 3
Mapping Networks

Given a complex system under study, one natural goal is to create a network-based representation of the system and to present that representation in the form of a visual image. The process by which such images are produced involves a number of distinct steps, with important decisions to be made along the way, and in fact is not unlike the production of cartographic maps. In this chapter, we address the topic of 'mapping' networks.

3.1 Introduction

As in many other areas of the quantitative sciences, visual imagery plays a fundamental role in network analysis. Such has been the case since even before the advent of computers and computer graphics. Freeman [158] has described, for example, how the initial impetus for the use of visualization in social network analysis can be traced to the seminal work of Moreno and colleagues in the 1930's. These and other early efforts at network visualization were in the form of hand-drawn, annotated graphs and related diagrams of relational data. The computer brought with it the potential to automate, and the motivation to standardize, the drawing portion of this process. Nevertheless, network visualization remains a nontrivial task, for at least two main reasons.

First, in most contexts there is not necessarily one, single network graph representation for a given system of interest – where by 'network graph representation' we refer here to a network graph G and any accompanying weights, annotations, and labels. Phrased another way, we may say that to speak vaguely of "*the* network" for a system is often misleading. There may in fact be many potentially relevant network representations, not unlike the way that a given geographical region may be represented in many different ways (e.g., with respect to its topography, its population patterns, its geology, its land use by humans, etc.). Second, even given a particular network graph representation of a complex system, visualization of that representation is still rendered nontrivial by the substantial challenges inherent in effectively communicating the information in a network graph using just the two-dimensional surface of a printed page or a computer display.

A graph G is just a pairing of two particular sets – a vertex set and an edge set – which itself is lacking in any inherent geometry. A visualization of G can be thought of informally, therefore, as a mapping from the space of such pairs to two- or three-dimensional Euclidean space. Hence, there is both an enormous amount of flexibility in defining such mappings and a correspondingly diverse range of information that may be communicated, or lost, as the case may be. In this sense, the challenges faced here are related to many of those faced in the visualization of high-dimensional datasets – or indeed in the visualization of information quite generally, as eloquently discussed and illustrated by Tufte [382], for example. However, this area is still arguably quite young and research is quite active, with aspects of mathematics, algorithms, aesthetics, and the workings of the human visual system all playing roles.

There is a strong analogy to be made between the creation of a network-based visualization for a complex system, on the one hand, and a cartographic map of a geospatial region on the Earth, on the other. As a result, the former process is more and more frequently being referred to through phrases such as ‘network mapping.’ It is convenient to identify, broadly speaking, three key stages in the production of network maps:

(i) collection of data from a system of interest;
(ii) creation of a network graph representation from the data; and
(iii) rendering of the representation in the form of a visual image.

These stages are illustrated schematically in Figure 3.1. In addition, there is a temptation to define a critical but typically difficult, and often ignored, fourth stage, that of assessing the representative-ness of a network map, a process referred to as ‘validation’ in geography.

In this chapter, we will consider each of the three stages listed above in turn, in Sections 3.2 through 3.4. We will then examine a pair of case studies in Section 3.5, illustrating the overall mapping process in two different contexts – mapping ‘Science,’ in the field of scientometrics, and mapping the Internet. Lastly, the topic of mapping dynamic network graphs will be discussed in Section 3.6. The issue of validation will be revisited again briefly at the end of this chapter; a related discussion, in the context of network inference, is the topic of Chapter 7.

3.2 Collecting Relational Network Data

Network mapping effectively begins with the collection of data from a system that is to be mapped. Whether we collect the data ourselves, or someone else does, an understanding of the data collection protocol is imperative. Of course, what data are to be collected as well as the tools and techniques for their collection vary widely from discipline to discipline. Furthermore, the quality and quantity of data that we

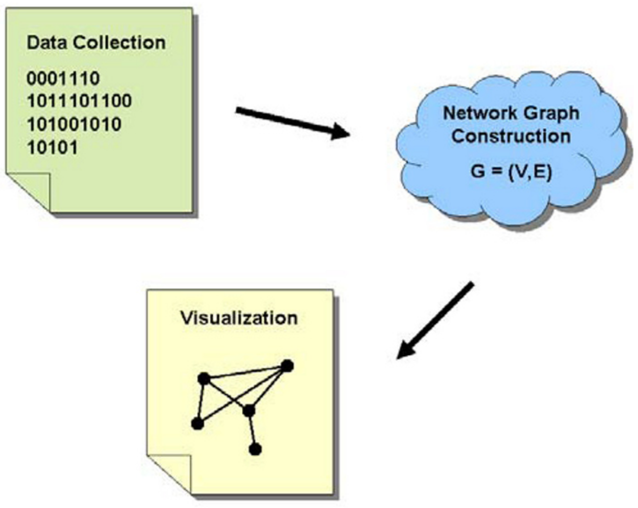


Fig. 3.1 Schematic illustration of the process of mapping a network.

can realistically expect to collect differ among disciplines as well. Our aim here in this section will be to survey some of the common aspects associated with, and typical issues faced in, network data collection across disciplines.

3.2.1 Measurement of System Elements and Interactions

In order to produce a network visualization, it is necessary to have a network graph representation. But it is not generally the case that network graphs are obtained explicitly themselves. Rather, more often than not they are constructed from a set of more basic measurements on some elements and their interactions in an underlying system of interest. Therefore, the choice of what is meant by 'elements' and 'interactions,' as well as what measurements to take for each, is important, since ultimately this choice influences not only the network graph(s) that might be constructed but also *a fortiori* the analyses of such graphs and hence the conclusions that may be drawn thereof. We illustrate this point through the following example.

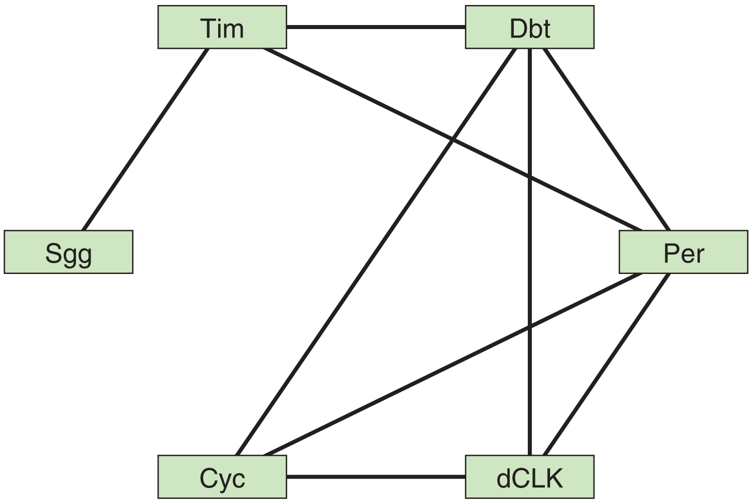
Example 3.1 (Two Alternative Perspectives on Drosophila's *Circadian Clock Mechanism).* Recall from Chapter 1 the network shown in Figure 1.3 for the circadian clock mechanism in the organism *Drosophila*. As mentioned in the text accompanying this figure, there are actually a number of relevant units and sub-processes depicted. The green rectangles (e.g., *Tim, Per, dCLK,* etc.) represent specific proteins, the basic biological molecules underlying this process. Through the various lines and arrows drawn, the proteins are shown to be serving at least two important purposes. First, some of them physically interact with each other, such as to form more sophisticated complexes of proteins. Complexes like these are shown with the corresponding green rectangles stacked upon each other, as with *Dbt* and *Per* in the middle of the figure. Second, some of the proteins, in attaching to specific locations on the organism's DNA, near particular genes, serve to regulate (i.e., activate or repress) the 'expression' of these genes; that is, they regulate the production of the specific proteins for which these genes code. This activity is shown in the figure by short paths of two lines, from the original protein, to a circle with 'DNA' next to it, to another protein.

Hence, there are at least two sets of 'elements' and 'interactions' of potential interest here: (i) proteins, and their affinity for interacting with each other, and (ii) genes, and their regulation among themselves. Each may be represented in the form of a small network graph, as shown in Figure 3.2. Note, however, that the two graphs, although depicting two highly inter-related aspects of the overall circadian clock mechanism, are nonetheless different in many ways. For example, while the protein *Per* interacts with no fewer than four other proteins, the gene coding for *Per* regulates none of the other genes directly.

Current high-throughput methods in genomics allow scientists to obtain measurements that provide – separately – evidence of protein interaction (e.g., protein affinity experiments) and gene regulation (e.g., DNA micro-array experiments). In conjunction with various methods of network inference (some of which we will see in Chapter 7), these measurements allow for the construction of network graphs of protein interaction and gene regulation on large scales. However, as we have seen here, each only provides a partial view of what transpires in the underlying biological system, and hence biologists are necessarily cautious in their interpretation and usage of such networks as they seek to disentangle the biological workings of different organisms. □

So different notions of 'elements' in a system, even if closely related, can produce quite different network graphs. However, even with a single, well-defined notion of what 'elements' will mean in a given study, there may be choices left to be made, such as when there exist different 'scales' at which the elements can be labeled. In studying the topology of the Internet, for example, such as that underlying the Abilene map in Figure 1.1, we might consider the elements to be individual users, or machines and devices (which, depending on their placement in the Internet, can serve varying numbers of users), or groups of such users/machines, such as those falling under the purview of a given administrative unit (e.g., a so-called autonomous system, or AS). Alternatively, in a study of citation patterns in the scientific literature, we might consider individual authors, or the papers they write, or the journals in which the papers are published, or even the disciplines that are represented by the journals. (We will return to each of these examples through the case studies in Section 3.5.)

With the elements of interest chosen, the relevant 'interactions' are sometimes relatively pre-determined, but not necessarily. In the case of Internet topology, for example, as mentioned previously, we might be interested in the topology of the



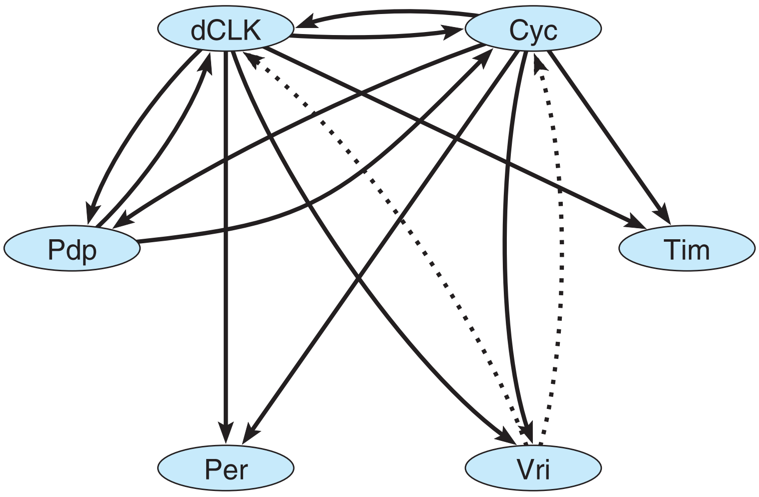


Fig. 3.2 Network graph representations of protein binding (top) and gene regulation (bottom) associated with the circadian clock mechanism in *Drosophila melanogaster* (fruit fly), as extracted from the composite representation in Figure 1.3. Proteins are shown as vertices in the form of green rectangles, and their interactions, by undirected edges. Genes, coding for their respective proteins, are shown as vertices in the form of blue ellipses. The regulation of one gene by another is depicted by a directed edge from the latter to the former (solid indicates activation, and dotted, repression).

literal, physical Internet, or rather, of the effective, logical Internet. Similarly, in studying citation patterns, say, among journals in a given literature, we might use some measure of the frequency of citations of articles in one journal by those in another (termed 'inter-citation'), or rather some measure of the frequency with which two journals cite common papers (termed 'co-citation').

Measurements of interactions can take many forms (e.g., binary, counts, or continuous values), and can be either direct or indirect in nature, in the sense that they may capture the intended notion of 'interaction' to a greater or lesser extent. For example, in the context of friendship networks in social network analysis, we may have the opportunity to interview the actors in a group and directly ask them whether or not they feel they are friends with each other actor (i.e., binary response), or instead we may (perhaps by choice) observe the frequency of particular types of interactions (i.e., count response). Note that even in the case of the seemingly direct measurement (i.e., asking "Who are your friends?") there can be issues, in that the veracity of the response is not guaranteed, the notion of 'friendship' held by actors may not be the same, or it may not coincide with that of the scientist.

Ultimately, it is worth keeping in mind with network studies – as with any study involving assessment of association – that not only is what we choose (or are capable of) to measure in a system important, but also, potentially, so too is what remains unmeasured in the system.

3.2.2 Enumerated, Partial, and Sampled Data

So far in this chapter we have talked as if we could collect whatever data we set our eyes upon. This is an overly optimistic perspective. Instead, it is important to distinguish between network data that are enumerated, partial, or sampled. To facilitate our discussion, and borrowing terminology from statistical sampling theory, we will picture the collection of all possible system elements that we would like to measure as a finite set and refer to that as a 'population,' with the elements themselves referred to as 'units.'

Enumerated data are data collected in an exhaustive fashion from the full population. In some network analysis studies it is possible to collect data in this manner. For instance, in some social network studies, such as those that might involve the dynamics among members of small groups, with full compliance from the group members, we could view the data collected as exhaustive. Network data on friendships among children in a given school are a possible example, as is the network of interactions among members of the karate club in Zachary's study, described in Section 1.2.2. Similarly, in the analysis of citations in scientific journals of a given discipline (e.g., medicine, physics, etc.), thanks to database resources like the Scientific Citation Index, PubMed, and others, it is possible to collect relevant citation information exhaustively as well, at least within certain windows of time.

Partial data are data that derive from a full enumeration of only a subset of the population. Note that, depending on the context and the perspective of those conducting a given study, many datasets that are apparently enumerated may be better considered, for some purposes, as partial. Consider the Abilene network shown in Figure 1.1. The Internet is a global technological network, but in this figure we have only shown a sub-network pertaining to the United States. And in fact, that sub-network in turn is a network that supports the transit of Internet data primarily for users only in U.S. educational and research institutions. Hence the presence of other networks, such as analogous commercial U.S. networks or networks in other countries, is largely absent from this map. To the extent that Abilene is a coherent sub-network under the watchful eye of a single administrative unit, such a perspective may be quite acceptable for many purposes. On the other hand, given that part of the traffic on this network is from Abilene users to and from locations throughout the rest of the world, for some questions ignoring the partial-ness of the network can be misleading. For instance, an observed disruption in traffic patterns in New York may be found to have little to do with any of its immediate Abilene neighbors, but rather to be due to the propagation of disruptions from, say, its European counterpart, Géant.

The concern posed by partial network data is closely related to that posed by finite regions in the analysis of spatial data and the danger of potential ‘boundary effects’ (e.g., see Cressie [102]). In the Abilene example above this connection is of course particularly evident, since the network itself has a strong geo-spatial aspect to it.

Sampled data are data on units selected from the population through some random mechanism (either implicit or explicit). Like partial data, sampled data are only some subset of the possible data to be collected, but unlike partial data, sampled data will not *a priori* yield an exhaustive view of a well-defined sub-population. In many areas of network analysis, sampling is the rule rather than the exception. An example of sampled data is the AIDS blog data in Chapter 1.2.4, where the random choice of a three-day period of observation induces a random selection of bloggers. There are many types of network sampling designs, and it is generally important to understand the nature of the design underlying the collection of network data, as sampling can affect analyses at later stages in ways that are potentially both nontrivial and subtle. The topic of network sampling is substantial enough that we will wait to develop it more fully on its own in Chapter 5.

The three categories of data we have used above – enumerated, partial, and sampled – are convenient, but the reality of data collection can be more nuanced, combining aspects of more than one of these. Alternatively, it can be useful instead to think in terms of *observed data* and *missing data*, a perspective that has been found to be quite powerful in many contexts in statistical analysis generally (e.g., see Little and Rubin [263]), particularly over the past few decades. The missing data paradigm has evolved into an organized approach to categorizing mechanisms for missing-ness of data, studying their effects on a given analysis, and, when possible, adjusting for such effects. However, unfortunately, it seems to have received much less attention so far in the field of network analysis, with most efforts to date found in the context of social networks and relevant primarily to the types of structural analysis methods we will see in Chapter 4. See Kossinets [237], for example, and citations therein.

The discussion above is included here at this point in the book primarily as a cautionary note, with the basic message being that the manner in which the data are collected can be important for their proper analysis and the interpretation of such analyses. At a minimum this suggests that a certain sense of *caveat emptor* (i.e., 'buyer beware') be maintained, by both producers and users of networks and network analyses – even visual analyses like those to be considered next in this chapter. At a more involved level, we may want to incorporate the mechanism(s) of data collection into the analysis, using models like those discussed in Chapters 5, 6, and 7. Modeling can also be useful, as we will see, for attempting to account for the effects of *measurement error,* a phenomenon distinct from those discussed so far. An obvious measurement error of interest in network analysis is the false declaration of the presence or absence of interactions among measured units. The mechanisms for such errors typically are inherent in the process of data collection.

3.3 Constructing Network Graph Representations

We turn now to the task of constructing a network graph representation from a set of network measurements. At its most basic, the representation will consist of just a graph $G = (V, E)$, with vertex set V and edge set E. However, more generally, the representation may also include additional auxiliary information, such as a set of edge weights $\{w_e\}_{e \in E}$, providing some indication of the strength of association between vertices, vectors $\{x_v\}_{v \in V}$ of variables describing vertex attributes, and labels. Attribute variables may be discrete in nature, such as those indicating class membership (e.g., gender in social networks), or continuous (e.g., size of a population in the geographical region serviced by an airport in an airline network). It will be this sort of information specifically that we seek to incorporate into a map, using methods like those to be described in the next section.

Having collected a set of network measurements, in principle the construction of a network graph G consists 'simply' of assigning vertices v to measured elements (or units) and edges $e \in E$ to measured interactions between elements.[1] However, even at this stage there may be nontrivial decisions to be made.

Consider just the issue of specifying vertices. In rendering a map on a printed page or a computer screen, both the space and the resolution available will be finite. For network graphs with more than a few hundred vertices, maps can quickly become cluttered. So the unit of analysis may still become an issue. As we shall discuss more in Section 3.4, we can choose to deal with this problem by displaying only a relevant sub-graph(s). On the other hand, if it is not clear *a priori* just which

[1] If it is desired to assign a direction to the measured interactions, then it will be a digraph that is constructed, with the set E consisting of arcs from one vertex to another. As the treatment of digraphs in this section is analogous to that of undirected graphs, we will confine our discussion here to undirected graphs.

sub-graph(s) to display, or if it is felt that there is value to be had in displaying the entire graph in some form or another, then some reduction of the vertex set may be necessary. Aggregation of vertices is one approach, particularly when there is some relevant sense of scale inherent in the possible choice of vertex definitions, such as in the Abilene map in Figure 1.1.

Now consider the related issue of specifying edges. As mentioned earlier in Chapter 2, the number of edges N_e in measured network graphs frequently has been found to scale linearly with the number of vertices N_v. So if the number of vertices to be displayed is problematic, the task of displaying a similar number of edges will likely be no less so. In fact, since the number of edges can, of course, scale as much as quadratically (i.e., $N_e \leq \binom{N_v}{2}$), we can find ourselves faced with the problem of having too many edges to cleanly display even if the number of vertices does not seem unreasonable. Graph visualizations suffering from excessive density of edges often are justifiably described as looking like a ball of yarn.

The ball-of-yarn phenomenon may be addressed through various means. For example, again, when there is some relevant sense of scale underlying the measurements, choice of scale may be used effectively. In the Abilene network, or more generally in many networks with a connection to the Earth's geography, such as various transportation networks, vertices tend both to be spatially localized and to have few if any interactions with vertices outside the immediate locale. As a result, spatial aggregation will lead to both aggregation of vertices and thinning of edges. This effect is apparent in the map of Abilene and its blow-out, in Figure 1.1. Viewed more formally, rooted sub-trees or DAGs at a given depth in an original candidate graph, say G_0, are trimmed at their root, effectively replacing the entire sub-structures (both vertices and edges) with a single vertex in G.

Alternatively, vertex annotations may sometimes be used to separate an initial, dense candidate network $G_0 = (V_0, E_0)$ into a handful of less dense networks, say G_i, for $i = 1, \ldots, I$. For example, if the annotations are indicators of category or group membership, it may be more visually informative to display the sub-networks of those vertices and their interactions associated with each category or group. In biology, for instance, it can be useful to associate genes or proteins with their biological function(s), when these functions are known. The resulting graphs then describe the gene regulatory relationships or protein interactions among just those genes or proteins involved with a given function. Larger networks are then sometimes built up from these smaller networks by incorporating related functions together. More formally, this process of thinning corresponds to selecting only a certain subset of vertices of V_0, say V_i, as those associated with characteristic or group i, and defining G_i to be the corresponding induced subgraph.

As a final example, when the edges in a candidate graph G_0 are defined according to some measure of similarity among vertices, the edges in G_0 may be thinned by re-assessing the strength of measured interaction judged necessary to merit an edge. For example, it is common to assign edges according to whether or not the correlation between attribute vectors of a given pair of vertices exceeds some threshold in absolute value. Adjustment of the threshold therefore clearly moderates the density of edges.

Note that in all three of the techniques described above, and in any other similar techniques, fundamentally a choice is being made in summarizing the extent of 'interaction' or 'relation' contained in the original measurements that is to be communicated in the corresponding visualization. More sophisticated approaches to reducing an initial candidate graph G_0 to one better suited for visualization involve the use of techniques that summarize graph structure. An illustration, based on the use of clustering, will be presented in Section 3.5.1. These techniques themselves will be discussed in detail in Chapter 4.

3.4 Visualizing Network Graphs

Up until this point in this chapter we have spoken only loosely of displaying network graphs, the final component of the process illustrated in Figure 3.1. Here in this section we consider the problem of display in its own right, which turns out to be far more than a simple matter of plotting the data. Techniques for displaying network graphs fall under the field of *graph drawing* or *graph visualization* and incorporate a combination of elements from mathematics, human aesthetics, and algorithms. Our discussion below is broken into two parts, beginning with an overview of the elements of graph visualization and followed by a brief description of some of the more common graph visualization problems and their solutions.

3.4.1 Elements of Graph Visualization

Suppose we have a set of network measurements that have been encoded in a network graph representation $G = (V,E)$, and we now wish to summarize G in a visual manner. At the heart of the graph visualization problem is the challenge of creating "geometric representations of ... combinatorial structures," [117] using symbols (e.g., points, circles, squares, etc.) for vertices $v \in V$ and smooth curves for edges $e \in E$. For human consumption it is most convenient, of course, if a graph is drawn[2] in two-dimensional space, as opposed to three-dimensional space or on some more abstract surface. Hence, we will restrict our attention to this setting.

Intuitively, it is not hard to see that there are uncountably many ways that we could lay down a candidate set of points and curves on paper to represent a graph G. The important question for any such candidate, however, is whether or not it adequately communicates the desired relational information in G. While in principle this might suggest the drawing of graphs by hand, in practice hand drawings are only realistic for very small graphs. Generally graphs of nontrivial size must be drawn, at least in part, using automated methods.

[2] Here and throughout we use terms like 'draw' only in the colloquial sense, although more formal mathematical treatments of this topic area exist (e.g., see Chapter 8 of Gross and Yellen [186]) which attach more specialized understandings to these terms.

Of course, one approach, which in fact is used frequently, is to just lay out all vertices in G as equi-spaced points around a circle, and to draw straight lines between points for vertices connected by an edge. Unfortunately, although such drawings are simple enough to create, they tend to accentuate any tendency towards the ball-of-yarn phenomenon mentioned earlier. In order to facilitate improved automatic drawings of graphs, various specifications or requirements have evolved – some firm, and some flexible – which have been formally categorized as drawing conventions, aesthetics, and constraints. See di Battista, Eades, Tamassia, and Tollis [117].

Drawing conventions are basic requirements that a given drawing must satisfy. For example, we might specify that only straight line segments be used to draw edges, rather than polylines or smooth curves, that vertices and edges be positioned only along an underlying lattice, or that no edges intersect (i.e., that G be drawn planar). For trees or, more generally, for directed acyclic graphs, it is common to use an 'upward' (or 'downward') convention, in that all edges are drawn as a curve increasing (or decreasing) along the vertical axis.

While drawing conventions are hard requirements, in the sense that they must be satisfied, *aesthetics* are soft, in that they are specified with the intention that they be satisfied to the extent possible, all else being equal. For example, by definition a planar graph G may be drawn using edges that do not intersect anywhere other than at vertices, but such graphs are comparatively rare. Therefore, if edge crossings must be tolerated, it is natural to wish for them to occur as infrequently as possible. Similarly, we might wish to minimize the area used for the drawing, or the aspect ratio, or the total or maximum edge lengths, or the total or maximum number of bends edges may take, and so on.

Finally, *constraints* are requirements that pertain to subgraphs $H \subset G$, rather than the overall graph G itself. These might specify the relative placement of a vertex in the drawing, or a cluster of vertices, the direction of a path, etc.

Together, drawing conventions, aesthetics, and constraints effectively serve as parameters for automatic graph drawing methods, and the determination of a graph drawing becomes a formal optimization problem. Such optimizations frequently are difficult to solve exactly in real time for graphs that are nontrivial in size. Therefore, it is common to develop computationally efficient algorithms that seek an approximate solution, often through the use of heuristics and the imposition of priorities among aesthetics. Such algorithms are complicated enough, and software (both commercial and non-commercial) plentiful enough, that for most users it makes little sense to code the algorithms oneself. All of the network visualizations presented in this book, for example, were constructed using pre-existing software implementations. See Appendix A of the volume edited by Kaufmann and Wagner [221] for a fairly extensive list of packages recommended by those authors.

On a final practical note, we point out that graph drawing – like statistics itself, perhaps – involves not only 'science' but also some 'art.' Few automatically generated graph drawings cannot be improved upon through post-processing 'by hand'.

3.4.2 Methods of Graph Visualization

There are far too many methods of graph visualization for us to present a full survey here. But there are two broad themes under which a large portion of such methods can be argued to fall, namely (i) methods for drawing graphs with special structure, and (ii) methods for graph drawing based on physical analogies. We will focus on describing and illustrating a handful of examples falling under each. We will also discuss extensions relevant to visualizing large graphs and directed graphs. References to more complete treatments of this topic area may be found in Section 3.7.

3.4.2.1 Drawing Graphs with Special Structure

Two special structures whose drawing has received a large amount of attention are planar graphs and trees. Graphs that are planar are especially relevant traditionally to certain important network areas, such as the design of circuits. Networks with a strong connection to the Earth's geography, such as some energy or transportation networks, also are often easily associated with planar graphs. The Abilene network and its representation in Figure 1.1 is another example.

However, planar graphs are also a natural starting point conceptually in the study of graph drawing, being by definition free of any edge crossings. In fact, many graph drawing algorithms for non-planar graphs build upon algorithms for drawing planar graphs, by first drawing a planar subgraph of the non-planar graph, and then augmenting that drawing to display the full graph. Two common techniques for the layout of planar graphs are to use orthogonal paths for edges (i.e., changing direction only by 90-degree turns in the manner of a canonical circuit drawing) or to use k-sided convex polygons for each cycle of length k in the graph. In other words, blocks and polygons, respectively, become the geometric structures upon which the visualization of the graph, as a combinatorial object, is built.

Layout algorithms for planar graphs are generally quite computationally efficient, running in as little as linear time in the number of vertices (i.e., $O(N_v)$ time). Note, however, that underlying the algorithms for drawing planar graphs can also be algorithms for determining whether a candidate graph G has various required properties, starting of course with the property of planarity itself. Algorithms exist for testing whether or not a graph is planar in linear time, but other related problems, such as certain constructions of planar subgraphs, are prohibitively expensive to solve exactly (i.e., NP-complete) and their solution is generally approximated.

Trees, although themselves planar graphs, are both distinct enough and encountered commonly enough that there is a separate body of methods for their drawing. Trees are used, for example, in the representation of various hierarchical structures, such as organizational charts and genealogies. In fact, it is this very notion of hierarchy that is the special aspect of trees that we generally wish to communicate effectively in their display. Figure 3.3 shows examples of three common layouts for rooted trees using layering of edges, horizontal and vertical edges (i.e., so-called *hv*-

layout), and edges that radiate outward on concentric circles (i.e., so-called 'radial layout'), respectively. Algorithms for all three layouts run in linear time.

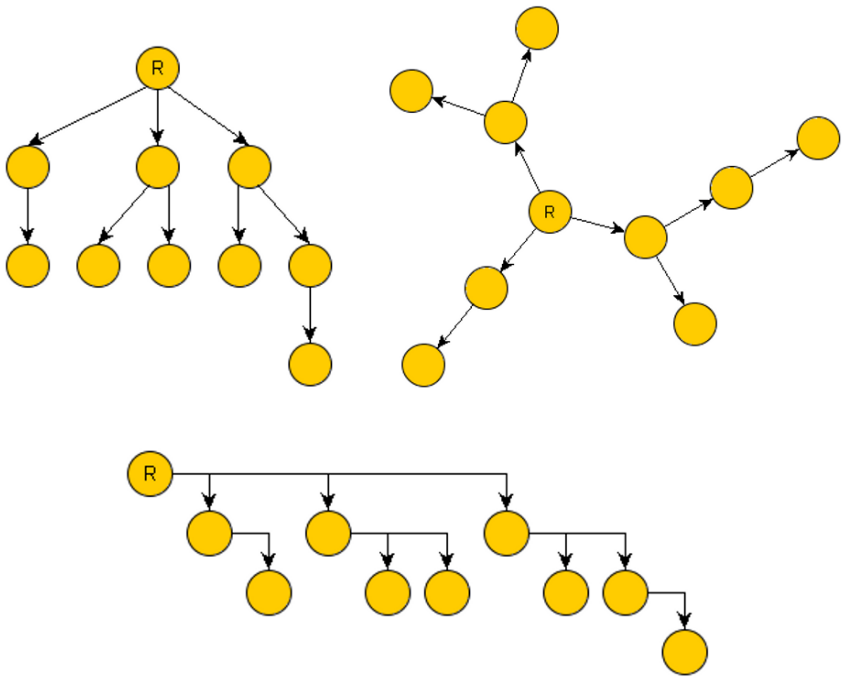


Fig. 3.3 Three displays of the same tree, rooted at the vertex labeled 'R': layered (top left), circular (top right), and horizontal-vertical (bottom).

3.4.2.2 Drawing Graphs Using Analogies to Physical Systems

If a graph *G* does not have a special structure, a popular and effective set of approaches to creating useful drawings is based on exploiting analogies between the relational structure in graphs and the forces among elements in physical systems. The expectation is that attraction and repulsion between 'likes' and 'not likes,' respectively, inherent in the organization of much of the natural world around us, if similarly used to represent a graph, will provide a familiar context within which to interpret the underlying relational information.

One approach in this area, and the earliest proposed, is to introduce attractive and repulsive forces by associating vertices with balls and edges with springs. If a literal system of balls connected by springs is disrupted, thereby stretching some of the springs and compressing others, upon being let go it will return to its natural state. So-called *spring-embedder* methods of graph drawing define a notion of force for each vertex in the graph depending, at the very least, on the positions of pairs of vertices and the distances between them, and seek to iteratively update the placement of vertices until a vector of net forces across vertices converges. The method of Fruchterman and Reingold [164] is a commonly used example of this type, and underlies the network visualization in Figure 1.4. It runs nominally in $O(N_v^2)$ time per iteration, but typically various devices are employed to both speed up the run time and decrease the number of iterations until convergence.

Alternatively, motivated by the fact that it is possible to associate the collection of forces in spring systems with an overall system energy, a second approach in this area is that of *energy-placement* methods. An energy, as a function of vertex positions, ostensibly is defined using expressions motivated by those found in physics. That vertex placement is chosen which minimizes the total system energy. A physical system with minimum energy is typically in its most relaxed state, and hence the assertion here is that a graph drawn according to similar principles should be visually appealing. Methods based on multidimensional scaling (MDS), which have a long history in the social network literature, are of this type. The method of Kamada and Kawai [217] is a popular variant of MDS methods and was used to draw the network visualization of the karate club data in Figure 1.2.

An advantage of energy-placement methods, in specifying vertex placement as a formal optimization, is that standard methods of numerical optimization may be brought to bear. Typically (modified) Newton-Raphson algorithms are used, or eigensystems are solved, depending on the choice of energy function, and hence run in $O(N_v^2)$ time. Alternatively, a number of algorithms have been introduced to compute approximations of these graph layouts in essentially linear time (e.g., Brandes and Pich [56], Ganser, Koren, and North [165]). Another advantage to the energy-placement perspective is that it lends itself well to the formal inclusion of additional requirements (e.g., aesthetics and constraints), through the definition of more complex energy functions. There is generally a trade off between the complexity of the energy functional, as various requirements are incorporated, and the ease with which the corresponding optimization may be performed.

3.4.2.3 Drawing Large Graphs

Despite their sophistication, for all of the methods described so far the graph drawings still tend to look increasingly cluttered as the number of vertices N_v nears 100 or so, due to the finiteness of the available space and resolution. As was discussed above, in Section 3.4.1, if faced with cluttering, we may wish to go back and modify the network graph G itself. Alternatively, specific information to be communicated by the network visualization might suggest that only a relevant subgraph(s) be shown. For example, sometimes it is useful to highlight the structure local to a given vertex, such as in the so-called *egocentric* network visualizations commonly used in the social network literature, which show the vertex, its immediate neighbors, and all edges among them. An illustration is shown in Figure 3.4, based on the karate club network of Figure 1.2. A clever compromise between global and local network visualizations is the use of ‘fish-eye’ views, similar to that achieved by wide-angle lenses in photography, wherein the network local to a vertex is shown in detail, but with the rest of the network shown as well, in successively less and less detail farther away from that vertex.

3.4.2.4 Drawing Directed Graphs

All of the discussion in this section up to this point has focused only on the drawing of undirected graphs. The drawing of directed graphs tends to be more involved. Standard techniques employ layering (as mentioned earlier in the context of trees), with the basic framework going back to Sugiyama, Tagawa, and Toda [375]. To draw a digraph in a layered fashion with, say, up-to-down orientation, the following requirements are generally incorporated: (i) the avoidance of arcs pointing upward; (ii) a relatively even placement of vertices and uniform length of arcs; (iii) as few crossings of arcs as possible; and (iv) reasonably vertical arcs. Common algorithms approach the enforcement of these requirements in a stagewise fashion and frequently employ heuristics in place of attempting to solve some of the demonstrably difficult (i.e., *NP*-hard) computational problems that arise in this context as a result of the directed-ness of the graph. The reader is referred to the resources described in Section 3.7 for details.

Before closing our discussion of graph drawing, it cannot hurt to again point out the importance of the human hand in this process. The production of sophisticated network maps is most often an iterative and interactive process. In fact, the better software packages for network visualization allow for at least some degree of interactive mapping. In addition, automatic methods of graph display typically are not set up to make decisions on the incorporation of additional attributes and labels of vertices and edges with a basic network representation G, when available. User choice of symbols, colors, and line thicknesses and types, for example, can all be used productively to better communicate such information in the original network data.

3.5 Case Studies

In order to better illustrate network mapping as a coherent process, and the challenges faced at the different stages described above, we present in this section two cases studies, both involving the mapping of very large-scale systems. The first case study presents an example of the mapping of ‘Science,’ while the second describes the construction of a particular map of the Internet.

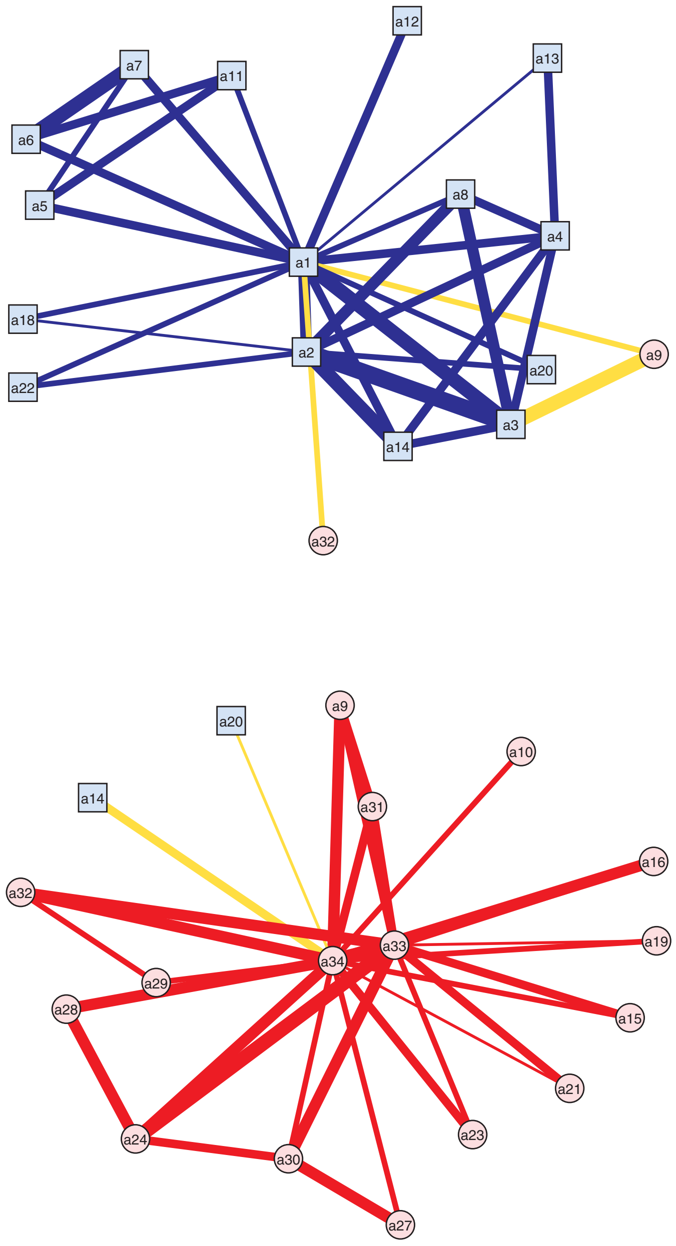


Fig. 3.4 Two ego-centric views of the karate club network of Figure 1.2, from the perspectives of vertices 1 (top) and 34 (bottom), the authority figures about which the club ultimately split.

3.5.1 Mapping ‘Science’

The human enterprise of Science and Technology (which we shall simply call ‘Science’ throughout) is both vast and multi-faceted, and is therefore of significant interest to members of many communities – such as groups and agencies within government, industry, and the sciences themselves. In particular, it is useful to be able to identify and understand patterns and associations in the growth and development of the various branches of Science. The field of *scientometrics,* whose foundations go back at least 40 years, to the work of Price [112] and others, aims to do so by applying the very techniques of scientific analysis to Science itself.

In this case study, we summarize the work of Boyack, Klavans, and Börner [50], whose goal was to produce a map of the ‘backbone’ of Science. Mirroring the development of material in the earlier part of this chapter, we organize our presentation in three parts: measurement, network graph construction, and visualization.

3.5.1.1 Measurement

It is common in scientometrics to study collections of written documents, in the form of articles published in archival journals, as a concrete representation of the knowledge produced by scientific endeavors. If we view such collections of articles as the ‘system’ to be measured, there are a number of natural choices of units. One choice would be the articles themselves, while another would be the journals within which the articles are published. The latter represents a coarser scale of unit than the former, resulting in a noticeably smaller (but still not at all small!) network, and is what we will use here. Ultimately, in the maps produced below, journals will be clustered and clusters of journals will be replaced by units of individual scientific disciplines.

We note that it is also not uncommon in scientometric studies to use the authors of articles as units. But this choice would be more relevant to the goal of mapping the patterns of co-authorship among individuals in a given scientific discipline, perhaps from a social network perspective, rather than that of mapping the relationships among the disciplines themselves.

Having chosen journals as our units, consider the task of defining a notion of interaction among journals that will allow us to meaningfully quantify the pattern of underlying scientific relations we wish to map. Measures based on citations are the standard choice. The citation of one paper by the authors of another is an explicit and universally accepted manner of acknowledging a connection between the two. In addition, detailed citation information for a large fraction of the journals across disciplines is now available in reasonably accessible form in various citation databases. To measure the strength of the citation patterns between two journals, we will use inter-citation frequencies. That is, for each journal pair (i, j), the strength of their relationship will be quantified by the number of times C_{ij} that journal i in a given time period (e.g., one year) cites journal j in any time period.

Note that in choosing to use citation databases, there is an aspect of partial sampling that will enter into our measurement process, in that such databases do not necessarily include all journals published in all disciplines. For most disciplines, we can imagine that the impact of partial sampling on our goal of mapping the 'backbone' of Science will likely be minimal, because presumably human experts have chosen what are felt to be the most visible and important journals. Yet for some disciplines, such as Computer Science, where much of the 'high-impact' publishing is done in the form of refereed conference proceedings papers, which are excluded from many of the main citation databases, this partial sampling is potentially a concern.

Boyack, Klavans, and Börner chose to use databases provided by the Institute of Scientific Information (ISI). Specifically, a complete set of records on 1.058 million articles were obtained from 7,349 journals in these databases for the year 2000. Of the 7,349 journals, only 7,121 of them appeared at least once as both citing and cited, and of the 23.08 million citations within the 1.058 million articles, only roughly 16.24 million of them (or 70%) were to journals similarly within this database. From these 16.24 million citations among these 7,121 journals, the corresponding inter-citation frequencies C_{ij} were calculated. The resulting matrix of frequencies was in fact extremely sparse, with 98.6% of the entries equal to zero.

In addition to the citation information, Boyack, Klavans, and Börner also obtained the ISI journal category assignments, which were used to, among other things, produce disciplinary labellings of clusters on the final maps shown later in Figures 3.5 and 3.6.

3.5.1.2 Network Graph Construction

A network graph G can be defined directly from the inter-citation frequency matrix, where journals i and j, serving as vertices, have an edge (i,j) between them if and only if the sum of their inter-citation frequencies (i.e., $C_{ij}+C_{ji}$) is non-zero.[3]

However, Boyack, Klavans, and Börner found that this measure of similarity did not yield graphs that, according to a validation study, clustered journals in a manner consistent enough with human expectations, as summarized through the ISI categories assigned to each journal. Therefore, a number of other similarity measures were considered, each transforming the inter-citation frequency matrix in a different manner. The so-called Jaccard coefficient (or Jaccard measure),

$$JAC_{ij} = JAC_{ji} = \frac{C_{ij}+C_{ji}}{\sum_{k\neq j} C_{ik} + \sum_{k\neq i} C_{jk}} \ , \tag{3.1}$$

a normalized version of the symmetrized inter-citation frequencies that attempts to adjust for the overall citation frequency of individual journals, was judged to be preferable.

[3] Note that, as the inter-citation frequency matrix is asymmetric, we could also choose to construct a directed graph.

In addition, these authors found it useful to apply some thinning of edges prior to mapping, and therefore chose to trim edges to include, for each journal, only the 15 most similar other journals. That is, the degree of vertices in the graph G was limited to be at most 15.

3.5.1.3 Visualization of the 'Backbone' of Science

The maps presented here were produced through a combined use of methods of automatic display, on the one hand, and human interpretation and annotation, on the other. VxOrd [109], an in-house visualization package produced by Sandia Labs, was used for the automatic display. The underlying drawing algorithm is based on an enhanced version of the spring-embedder methodology discussed in Section 3.4.2, involving the use of sophisticated optimization methods to search the space of possible graph drawings and a grid that helps reduce computation time from the typical $O(N_v^2)$ down to $O(N_v)$. In addition, it employs edge-cutting criteria, designed towards producing drawings that balance the detail with which both local and global structure are shown.

The human interpretation and annotation entered the mapping process at the next stage. The immediate output obtained from VxOrd is characterized by numerous small clusters of highly inter-connected vertices, with a sparse collection of ties between clusters. Given the nature of Science and scientific work, such structure is not unexpected. Boyack, Klavans, and Börner therefore took the VxOrd output and, through examination of the journals within clusters and their corresponding ISI categories, assigned labels to the clusters. The result of the overall two-stage process is shown in Figure 3.5, where the edges have been removed to improve readability.

Lastly, to produce our final map of the 'backbone' of Science, at the level of scientific disciplines, the clusters in Figure 3.5 were each replaced by a single vertex, the size of which is proportional to the number of journals in the cluster. Then, directed edges were placed from one vertex to another if more than 7.5% of the citations by journals corresponding to the first were to journals corresponding to the second, with darker edges reflecting larger percentages. Furthermore, the vertices were colored, on a scale ranging from dark to light, in a manner proportional to the relative frequency of self-citations among journals within a discipline, so that darker vertices may be viewed as more independent and lighter vertices as less independent. The results are shown in Figure 3.6.

Various features are apparent from this map, and we note just a few here. As is not uncommon with output from many automatic drawing methods, more highly linked vertices are located closer to center, while vertices with fewer links lie along the perimeter. Among those near the center, biochemistry and medicine are the two most highly connected and also quite large. The two disciplines differ, however, in their dependency on other disciplines, in that biochemistry is comparatively independent, and medicine, similarly dependent. Among the more mathematically-oriented sciences we note that statistics is both fairly close to the center of the map and comparatively independent, matching well with its reputation as an originating

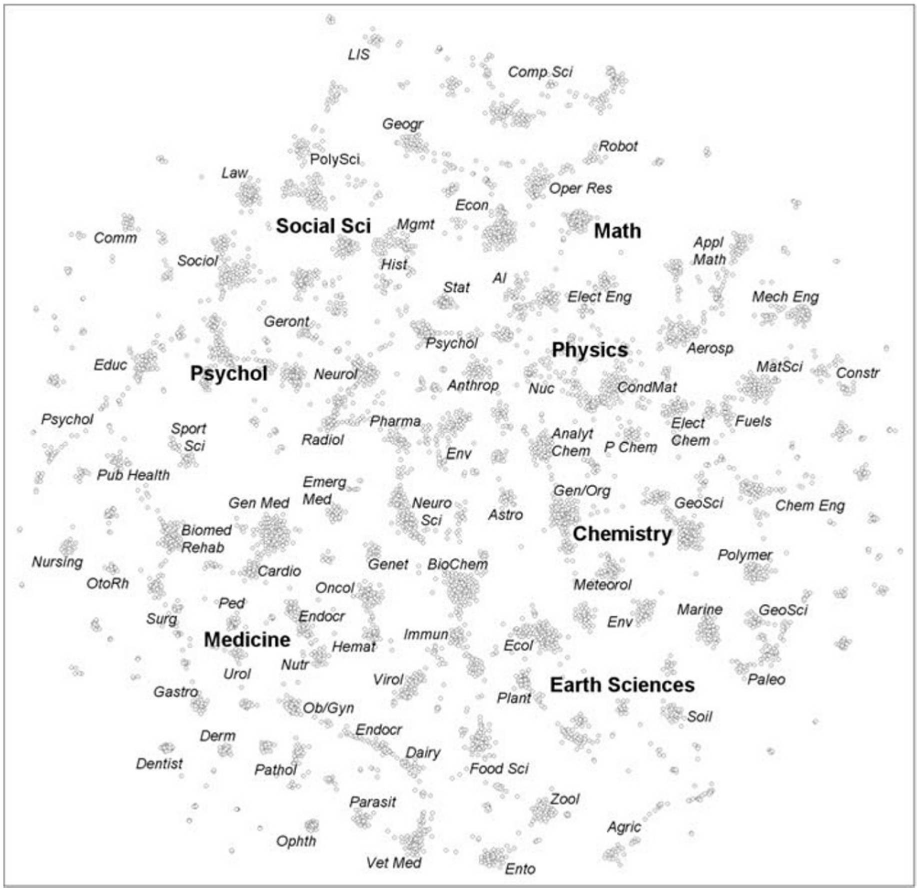


Fig. 3.5 Preliminary map of Science, resulting from a combination of automated drawing software and human interpretation and annotation. All edges have been removed, to improve readability. Figure courtesy of Kevin Boyack.

source of important quantitative concepts and tools that are then used broadly across other fields.

3.5.2 *Mapping the Internet*

While the Internet arguably pervades nearly every aspect of modern society, in ways both obvious and not so obvious, it may nevertheless come as a surprise to many people that a single, comprehensive map of this purely human construction is largely lacking. The lack of such a map is due to a variety of factors, including its dynamic and self-organizing nature, proprietary and security concerns among its

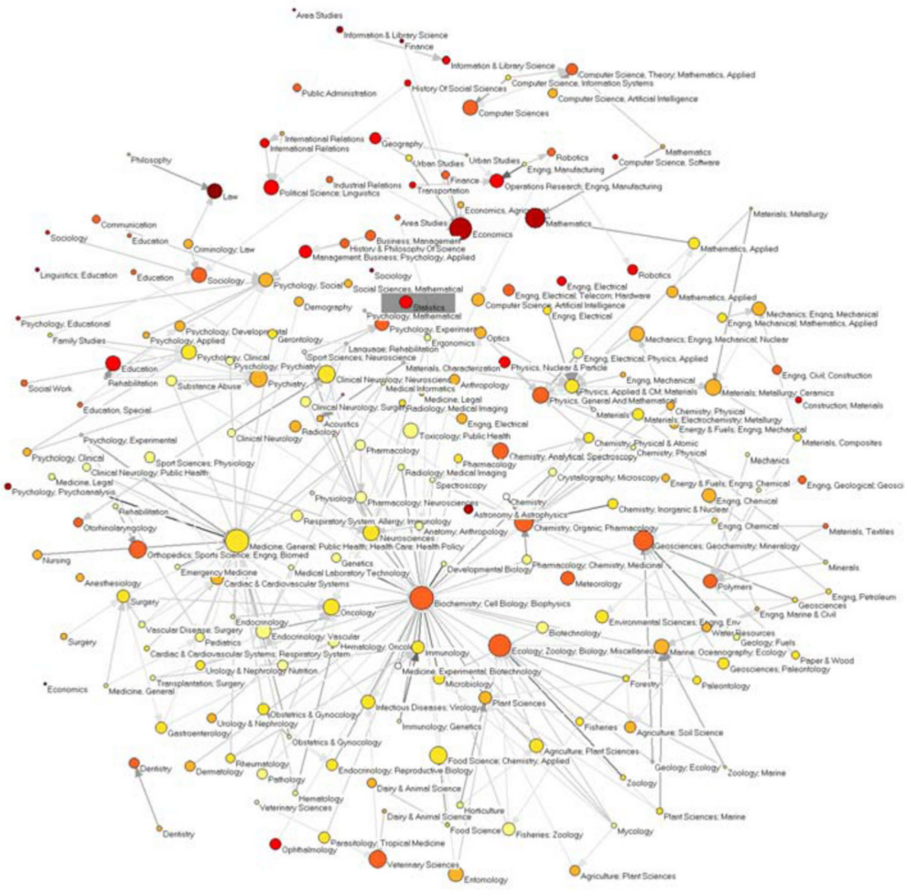


Fig. 3.6 Map of the ‘backbone’ of Science. Figure courtesy of Kevin Boyack.

various service providers (which own and maintain the separate, interacting pieces of the Internet), and simply its sheer size. Because of the dependency of so many interests on the Internet, the construction and maintenance of accurate maps is an important and ongoing activity.

Realistically, there are numerous inter-related aspects of the system colloquially referred to as ‘The Internet,’ including the physical infrastructure, the logical paths by which information flows over that infrastructure, the content underlying that information, the usage patterns of those creating, disseminating, and consuming that content, and the traffic created by such usage. Accordingly, there are many maps that we might think to produce. In this section, we focus on the task of mapping the logical structure of the Internet, drawing on the treatment in Crovella and Krishnamurthy [105] for background on measurement and network graph construction, and on Alvarez-Hamelin, Dall’Asta, Barrat, and Vespignani [8] for visualization.

3.5.2.1 Measurement

Recall that the physical Internet is a network of digital devices communicating over wired[4] connections via a set of communication protocols. Information is transfered over the physical Internet in the form of *packets,* which may be thought of as 'pieces' of a larger information object (e.g., an email, photo, web-page content, etc.), equipped with information identifying the source and the destination. Devices can be categorized as either *end systems,* such as desktop and laptop computers, and *routers,* which are responsible for receiving packets from one location and sending them onward towards their destination (much like a system of post offices). The connections between devices are simply referred to as *links.*

The logical Internet, on the other hand, which will be our focus, refers to the set of paths over which packets are routed through the physical Internet. Common units used in mapping the logical Internet are either the routers themselves or aggregations of routers. Common units of aggregation include administrative units (e.g., grouping according to so-called autonomous systems or ASs) and geographical concentrations (e.g., as in the Abilene map in Figure 1.1). Logical links may then be representative of either the connections between routers or the effective connections between aggregations of routers. Note, however, that as the logical Internet refers to the *used* portion of the Internet, physical nodes and links that are unused at a given point in time in the routing of traffic are essentially invisible from the logical perspective. In this section, we will use the routers themselves as units.

The distinction between physical and logical is important to us because, while there are no databases from which to extract comprehensive information on either, unlike the case when mapping Science in Section 3.5.1, it is the logical Internet that is most amenable to large-scale measurement, albeit by indirect means. Common techniques utilize information pertaining to or generated by the very communication protocols underlying the flow of Internet traffic. For example, the data we use here was generated by use of a certain probing mechanism, called `traceroute`, in a manner not unlike the sending of acoustic signals into the Earth in geological surveys of oil reserves.

Figure 3.7 shows an illustration of how `traceroute`-based measurement works. Part of the usual information with which each Internet packet is equipped, prior to being sent from its source to a destination, is information on its 'time to live' (TTL), an integer value that is decreased by one at each router through which the packet passes and is intended to ensure that misrouted packets do not continue to circulate through the Internet for eternity. If the TTL of a packet reaches zero at a router on its way to a destination, that router 'drops' the packet (i.e., it is sent no further) and, importantly, can generate a message to notify the original sender that includes the router's unique identifying address (i.e., its Internet protocol, or IP, address). The `traceroute` tool exploits this protocol characteristic by sending a sequence of packets towards a destination of interest with successively greater TTLs, and records the corresponding IP address returned by each intermediate router re-

[4] We will ignore the issue of wireless connections in this section since, due to their nature, such connections essentially are found only at the periphery of the Internet.

sponding that it has encountered a packet with a TTL of zero. In Figure 3.7, for example, a sequence of three `traceroute` probes (the communication patterns for which are shown in blue, green, and red) are used in discovering the two routers along the path from the given source to the intended destination.

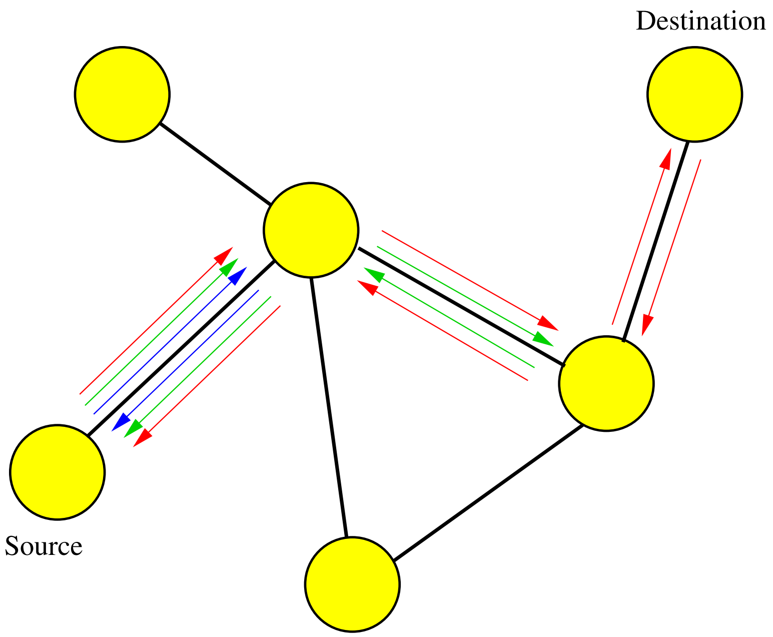


Fig. 3.7 Illustration of the use of `traceroute` to sample the logical topology of the Internet.

By probing in this manner from a number of sources to a number of destinations, it is possible to obtain raw information by which some sense of the logical topology of the Internet may be derived. CAIDA[5] (the Cooperative Association for Internet Data Analysis) is an independent, non-profit group, combining resources from government, industry, and academia, that conducts a number of large-scale Internet measurement projects, including the *Skitter* topology project. For Skitter, CAIDA has established roughly 20 measurement centers around the world, from which it sends `traceroute`-like probes to approximately 800,000 destinations, chosen in an attempt to provide as uniform a coverage of 'destination space' as possible. The maps shown later in Figure 3.8 are based on CAIDA data collected between April 21 and May 8, 2003.

Note that `traceroute`-like studies such as Skitter result in sampled network data, and so it is important to consider the effects of such sampling. An obvious sampling effect is the selection of sources and destinations, which will clearly dictate the paths for which routing information is gathered. We will consider this issue further

[5] *http://www.caida.org/*

in Chapter 5. In addition, there are more subtle effects, such as the role played by certain security measures in the Internet, like the use of firewalls (from behind which topology information cannot be obtained) and the increasing tendency for owners of routers to disable the reply mechanism upon which `traceroute` is based (so as to reduce the overall communication burden on their router). Discussion of effects such as these are beyond the scope of this book.

3.5.2.2 Network Graph Construction

In principle, from each sequence of `traceroute` probes we can infer a logical path in the Internet from a source to a destination, and given a collection of such sequences, from the source to multiple destinations, an entire tree (or, more generally, a DAG) can be inferred. Finally, a collection of such trees, obtained from multiple sources, can be merged to create a network graph G representing the measured topology.

In reality, however, there are a number of practical difficulties that must be dealt with. For example, there is some asymmetry in the manner in which traffic is routed in the Internet, so that, unlike the simple illustration in Figure 3.7, the path from a node A to a node B may not be the same as that from B to A. Hence, such studies realistically produce only information on directed paths. Second, there is the issue of time sensitivity, in the sense that the paths routed on the Internet are dictated by a set of protocols reacting to a combination of human and machine feedback, and hence are dynamic. In merging the results of probes sent over a period of days or weeks, for example, false links can be generated, induced purely by routing changes. Third, routers do not typically have a single interface with the network by which traffic is received and sent, but rather multiple such interfaces, and hence multiple IP addresses. In other words, each router can be encountered by various probes under a number of different 'aliases.' It is therefore necessary to resolve these possibly multiple aliases so as to avoid mapping multiple nodes where only one router is present. The reader is referred to Crovella and Krishnamurthy [105] for additional details on such problems and methods for their solution.

3.5.2.3 Visualization of the Router-Level Internet

After the appropriate post-processing, the CAIDA data yield a network graph G representing $192,244$ discovered routers and $609,066$ discovered edges. As discussed in earlier sections of this chapter, it is a challenge to create informative visualizations of such large graphs. CAIDA itself has produced some visualizations[6] based on aggregation of this sort of router-level data to the scale of autonomous systems. Here we will present another form of visualization, one designed to highlight the

[6] See *http://www.caida.org/tools/measurement/skitter/visualizations.xml* .

known hierarchical structure of the Internet (i.e., as illustrated in the context of the Abilene network in Figure 1.1).

Specifically, we will use the concept of a *k-core decomposition*, which has been at the heart of a number of proposals for the representation and visualization of large network graphs (e.g., Batagelj, Mrvar, and Zaversnik [25], Baur, Brandes, Gaertler, and Wagner [26], and Alvarez-Hamelin, Dall'Asta, Barrat, and Vespignani [8]). A k-core of a graph G is a subgraph H of G for which all vertex degrees are at least k, and such that no other subgraph obeying the same condition contains it (i.e., it is maximal in this property). k-cores are one natural way[7] to look at group structure across a graph G. In a k-core decomposition, the vertex set V is partitioned into 'shells' of vertices, say C_c, such that all vertices in C_c belong to the c-core but not the $(c+1)$-core. Vertices in C_c are said to have a 'shell index' of c. Appropriate visualization of this collection of shells allows us to view the graph G at successive levels of 'core-ness'.

Figure 3.8 shows such a visualization, as well as two close-up views, generated using the k-core visualization tool *LaNet-vi* described in Alvarez-Hamelin, Dall'Asta, Barrat, and Vespignani [8]. *LaNet-vi* is a freely available software tool[8] that uses a combination of different layout concepts, including some from radial and spring-embedder methods introduced earlier, and runs in $O(N_v+N_e)$ time. Vertices, represented as dots, are placed within their respective shells C_c, which in turn are organized in a series of concentric rings. Shells with higher index (i.e., more at the 'core' of the network) are located on rings closer to the center, and those with lower index, farther from the center. The rings are distinguished by their color, ranging across the natural spectrum, from red (high shell index) to purple (low shell index). Within each ring, connected clusters of vertices are kept together and ordered around the circumference. Ring thickness is a function of how connected vertices within a given shell are to those in shells higher or lower than them. Individual vertices with comparatively more neighbors of higher shell index are located closer to the internal part of their ring, and vertices with more neighbors of lower shell index, closer to the external part of their ring. Hence, the more variable this higher/lower connectedness among vertices of a given shell index, the thicker the corresponding ring (relative to a user-defined maximum thickness). The size of individual vertices scales proportional to their degree and is displayed on a logarithmic scale.

Various structural characteristics of the router-level Internet are suggested by the maps in Figure 3.8. For example, the hierarchical structure is quite evident, as links appear to run primarily between vertices of different shell indices, with noticeably fewer links among vertices of the same shell index. Furthermore, differences in shell thickness are apparent, with a number of the shells of lower shell index evidently much thicker than those of higher shell index. This feature indicates a larger range of variability in the shell index of neighbors of vertices in lower-index shells compared to higher-index shells. Note too that, although a large fraction of the high-degree vertices are located near the center, we can see (particularly in the cross-section)

[7] We will encounter k-cores again, as well as other similar measures, in Chapter 4.

[8] Available at *http://xavier.informatics.indiana.edu/lanet-vi* .

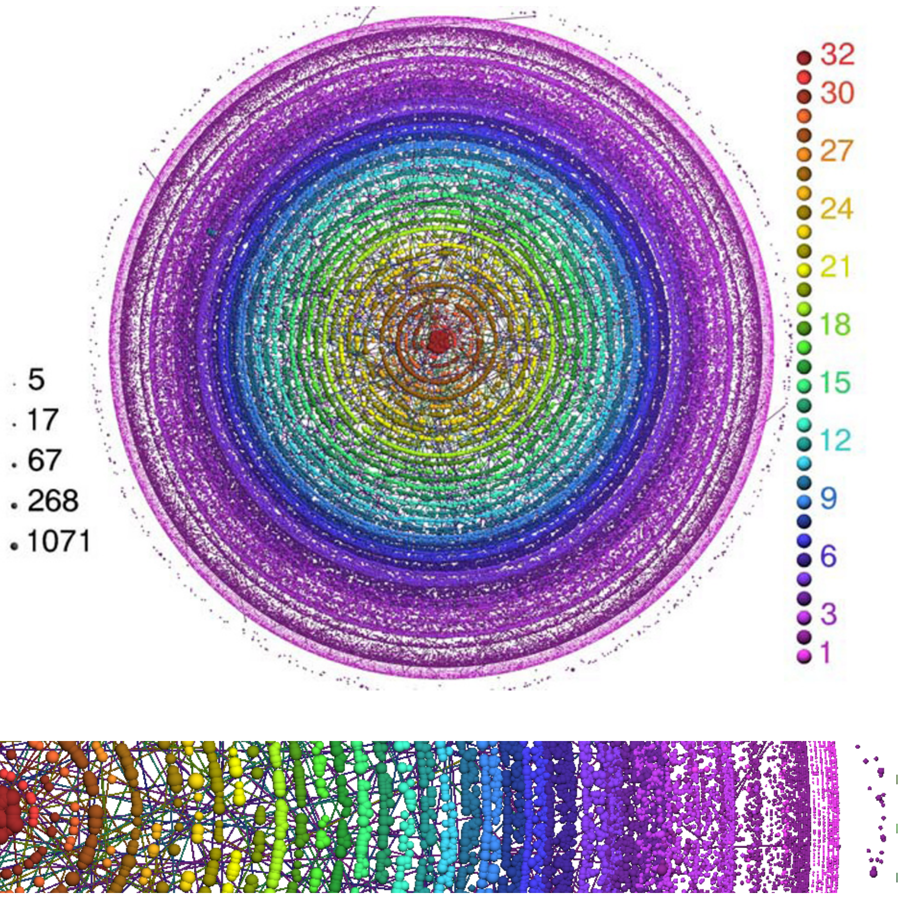


Fig. 3.8 Visualization of a *k*-core decomposition of the CAIDA router-level Internet data. Figure courtesy of Ignacio Alvarez-Hamelin.

that there are nevertheless some high-degree vertices quite far out on the external rings as well, indicating that highly connected routers can in fact be quite peripheral.

3.6 Mapping Dynamic Networks

The material in this chapter has focused almost exclusively so far on the case where the goal is to produce a single map of a static network graph G. But clearly many of the systems studied from a network-based perspective are intrinsically dynamic in nature. Depending on the goals of the researcher, it may therefore be of interest to produce, instead of a single static map, a dynamic network map(s) from a collection

$\{G^{(t)}\}$ of network graphs $G^{(t)}$ indexed over times t in some range $\mathbb{T}$. We close this chapter with a brief look at this topic.

Obviously many of the basic principles of measurement, graph construction, and display discussed for static network mapping continue to hold true in the dynamic case. And in fact the most straightforward way to create a dynamic mapping of a network is as a sequence of separate static maps. More ambitiously, if the data are rich enough temporally, instead of displaying all of the individual static maps, one might animate them to produce a 'movie,' so as to watch the network graphs change over time.

From the point of view of the graph display problem, the basic difference between the static and dynamic cases is the need to respect, in the case of the latter, what is referred to as the user's *mental map*. This term is used to describe the result of the process by which, upon studying a given static network map, a user becomes familiar with it, interprets it, and navigates about it. Dynamic changes in a sequence of network graphs, say from $G^{(t)}$ to $G^{(t+1)}$, involve the addition and deletion of vertices and links. If these changes are small, from one time period to the next, intuitively we would like to see only correspondingly small changes in each of their respective visualizations. That is, we would expect a certain amount of 'stability' across visualizations. However, unless this requirement is incorporated as an additional parameter, automatic drawing methods can produce very different looking displays in the face of even small changes. The result is that the user is then forced to form a new mental map for each graph $G^{(t)}$ and to struggle with the effort of reconciling the sequence of maps.

Although the area of dynamic graph drawing is still arguably in its infancy, there are already a number of approaches that have been developed for automated display of dynamic network graphs with constraints to reinforce a consistent user mental map. Most stringently, we may refuse to allow vertices and edges to move with the addition of new elements to the graph. But this strategy can quickly lead to cluttered maps if the graph changes sufficiently over time. More subtly, we may attempt to allow only 'local' elements in the vicinity of a newly added graph element to change. Use of distance metrics between graphs, in an effort to choose layouts for which consecutive graphs are not 'far' from each other, are also popular. More generally, Brandes and Wagner [58, 59] have suggested a Bayesian framework (based on the use of Gibbs distributions for random fields) within which many of these ideas, and ideas from static network graph display, can be subsumed and dealt with together in a consistent manner. Overall, however, there does not yet appear to be a consensus on a preferred method(s). See Chapter 9 of the edited volume of Kaufmann and Wagner [221] for a recent overview.

Of course, all of the above presupposes that the addition of a temporal index does not adversely affect basic issues of data acquisition and management, nor the construction of effective graph representations. But in some dynamic network contexts this is most certainly not the case, and attention must be given to all of these issues right from the start. Volinsky and colleagues (e.g, Hill, Agarwal, Bell, and Volinsky [198]) have suggested an approximation scheme for dynamic network graphs based on the concept of exponential smoothing, common in statistical time series analysis, wherein the network graph representation at time t (i.e., $G^{(t)}$) is written as a weighted sum of the network graph at the previous time point, $G^{(t-1)}$, and the set of vertices and edges new at time t, say $g^{(t)}$. That is, they propose a recurrence of the form $G^{(t)} = \theta G^{(t-1)} \oplus (1-\theta)g^{(t)}$, where the operation $\oplus$ represents union of vertex and edge sets and weighted arithmetic sum of edge weights. Combined with an edge-thresholding mechanism and a strategy for estimation of the relevant parameters, the methodology is illustrated in the context of massive telephone call graphs.

3.7 Additional Related Topics and Reading

Network mapping is about effective visual communication of relational information. Here we have presented network mapping as a three-stage process, consisting of data collection, network graph representation, and visualization. Brandes and colleagues (e.g., Brandes et al. [60], Brandes [53]) have offered a similar characterization of network visualization. Their characterization is broken down according to the three elements of system, design, and algorithm, which arguably overlap primarily our second and third stages, thereby providing in part a more refined breakdown of those two.

Most methodical expositions of issues associated with the collection of relational data appear to be found in the context of social networks – for example, see the books by Scott [344] and Wasserman and Faust [393], and the review by Marsden [269]. An extensive overview of principles and techniques for network graph visualization may be found in the edited volume of Kaufmann and Wagner [221]. In addition, Chapter 2 of the Ph.D. thesis of Brandes [53] is quite useful in this regard. For detailed development of the corresponding algorithms, see the text of di Battista, Eades, Tamassia, and Tollis [117]. It should be noted that, as Brandes [53] has pointed out, compared to the more established body of work on the visualization of more standard statistical data (e.g., as summarized in the books by Tukey [383], Tufte [382], and Cleveland [93, 94]), the development of formal methods for the visualization of relational data is still in its relative infancy.

Lastly, as was mentioned in Section 3.1, the validation of network maps is an interesting and important topic that merits greater attention. In the automatic construction of geographical maps, such as from measurements obtained by remote sensing instruments, *validation* refers to the process by which one verifies the extent to which the map matches the reality. But in such contexts the comparison is to an appropriate ‘ground truth,’ such as can be obtained by having an ‘expert’ literally visit and assess the spatial content of areas of the Earth. In the network context, it is less clear what constitutes ‘ground truth’ in many cases, and in other cases it is not even clear that ‘ground truth’ is in fact the most useful representation of the information that one wishes to convey. Brandes [53, Ch. 1] provides a nice illustration of this latter issue, through the comparison of the traditional schematic map of a city subway system versus a geographically accurate map of the same. The geographical map, while arguably a more faithful physical replicate of the actual subway, is nevertheless in fact not as useful to the average rider as the schematic map. On the other hand, when some sense of the reality to be captured is available, such as in the form of an alternative set of measurements of the same system of interest, it is tempting to want to ask to what extent the network map formed from the original measurements matches the information captured in the alternative measurements. Some initial efforts in this direction in the context of mapping 'Science' may be found in Klavans and Boyack [228].

Exercises

3.1. For a network graph of interest to you, reflect on the nature of the underlying measurements and the corresponding construction of the graph. Trace out in detail (to the extent you can, if you yourself did not create the network graph) the manner in which vertices and edges were defined from the measurements. To what extent are your data enumerated, sampled, or partial? Is it of interest to be able to 'validate' your network graph and, if so, what information would need to be available to perform such a validation?

3.2. In Figure 3.9 is shown a graph displayed with a circular layout. This graph is in fact a tree. If we assign the vertex 1 to be the root, and we assign an order to the other vertices with respect to (i) their distance from the root and (ii) the order of their labels among vertices of equal distance, then we have an *ordered rooted tree.*

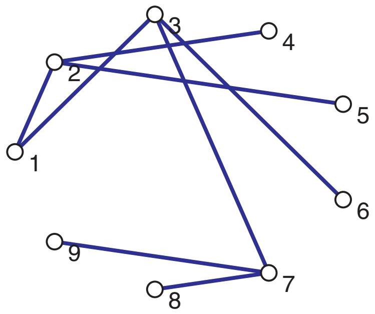


Fig. 3.9 Graph corresponding to Exercise 3.2.

A *preorder traversal* of this tree is a walk that visits the root first, and then visits the vertices of the first sub-tree in preorder, and then the second, in a recursive fashion, yielding the preorder listing $\{1,2,4,5,3,6,7,8,9\}$. A *postorder traversal* of this tree is a walk that traverses in postorder the vertices in the first sub-tree of the root and then the second, in a recursive fashion, yielding the postorder listing $\{4,5,2,6,8,9,7,3,1\}$.

Consider the following tree drawing algorithm. First, we assign a Cartesian coordinate (x_v, y_v) to each vertex v, where x_v is the rank of v in the preorder traversal list, and y_v, the rank in the postorder traversal list. Second, points are connected by a straight line if their corresponding vertices are adjacent in the original underlying graph. Implement this algorithm for the tree shown in Figure 3.9. Is your drawing planar? What is its orientation? What is its effective area? How might you modify the assignment of coordinates to vertices to reduce this area?

3.3. Recall that basic spring-embedder methods of graph drawing assign a measure of 'force' to each vertex $v \in V$ of a graph G and iteratively adjust the position $p_v = (x_v, y_v)^T$ of each vertex until the forces on each have converged to an equilibrium value. A standard setup defines the force $F = (F_x, F_y)^T$ on v in the x direction to be

$$F_x(v) = \sum_{\{u,v\} \in E} k_{uv}^{(1)} \left(||p_u - p_v|| - \mathsf{len}_{uv} \right) \frac{x_v - x_u}{||p_u - p_v||} + \sum_{(u,v) \in V^2} \frac{k_{uv}^{(2)} (x_v - x_u)}{||p_u - p_v||^3} \quad , \tag{3.2}$$

and similarly in the y direction, with y_u, y_v substituted for x_u, x_v. Here $||\cdot||$ indicates Euclidean distance, while len_{uv}, $k_{uv}^{(1)}$, and $k_{uv}^{(2)}$ are user-defined parameters, where len_{uv} is the desired length of the edge to be drawn between u and v, $k_{uv}^{(1)}$ is a 'spring stiffness' (i.e., larger values force $||p_u - p_v||$ closer to len_{uv}), and $k_{uv}^{(2)}$ is a strength of 'electric repulsion' counter-balancing the spring stiffness.

A simple algorithm utilizing these equations is as follows. First, initialize the algorithm by randomly assigning each vertex to a location. Then, for each iteration of the algorithm and each vertex $v \in V$, compute $F(v)$, using equation (3.2) above and your own specified constants len_{uv}, $k_{uv}^{(1)}$, and $k_{uv}^{(2)}$, and move v in the direction of $F(v)$ by an amount $\varepsilon||F(v)||$, for some small $\varepsilon > 0$.

Implement the above algorithm for a small network graph (i.e., a toy graph of your own construction will do). For an appropriate choice of parameters, visualize a succession of iterations of the algorithm (i.e., as in a movie), so as to observe the manner in which the method produces its final graph drawing from the random initialization. Explore the behavior of the method as a function of your parameter settings.

3.4. Using a network graph of interest to you, familiarize yourself with a graph drawing software. While doing so, pay particular attention to issues of aesthetics and the information being conveyed as a function of different layouts and variations on any parameters associated with these layouts. Is it useful or necessary to modify the number of vertices or the density of edges in your graph? Are there interesting visualizations of subgraphs in your network? After a thorough exploration, produce a concise report summarizing your findings.