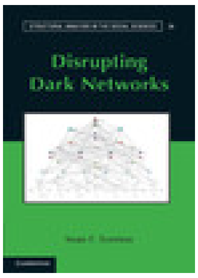


Disrupting Dark Networks

Sean F. Everton

Book DOI: http://dx.doi.org/10.1017/CBO9781139136877

Online ISBN: 9781139136877

Hardback ISBN: 9781107022591

Paperback ISBN: 9781107606685

Chapter

3 - Getting Started with UCINET, NetDraw, Pajek, and ORA pp. 49-75

Chapter DOI: http://dx.doi.org/10.1017/CBO9781139136877.006

Cambridge University Press

3

Getting Started with UCINET, NetDraw, Pajek, and ORA

3.1 Introduction

The advent of the personal computer has played a large role in the growth of social network analysis (SNA). Indeed, it is unlikely that SNA could have developed like it has without it (Freeman 2004:139–141; Wolfe 1978) because SNA relies on complex mathematical and graphical algorithms (Freeman 2004:3, 135–136). Over the years, a number of SNA software packages have been developed, all of which have their own strengths and weaknesses – see Huisman and van Duijn (2011) for a comprehensive review of the available packages. In this book we focus on UCINET (along with its integrated visualization program, NetDraw), Pajek, and ORA, not because they are necessarily the best programs (although they are quite good), but because they are readily available, widely used, and relatively inexpensive (all but UCINET are free). Indeed, in Huisman and van Duijn's (2011:585–590) review of general standalone SNA software packages, these three were among the five packages that were discussed at length.[1] Moreover, users who gain a familiarity with the logic and features of these programs should be able to migrate to other SNA packages with relative ease. This chapter provides a basic introduction to these programs – their interfaces, features, visualization capabilities, strengths, weaknesses, and so on – whereas Chapter 4 discusses the collection, recording, and manipulation of network data.

3.2 UCINET

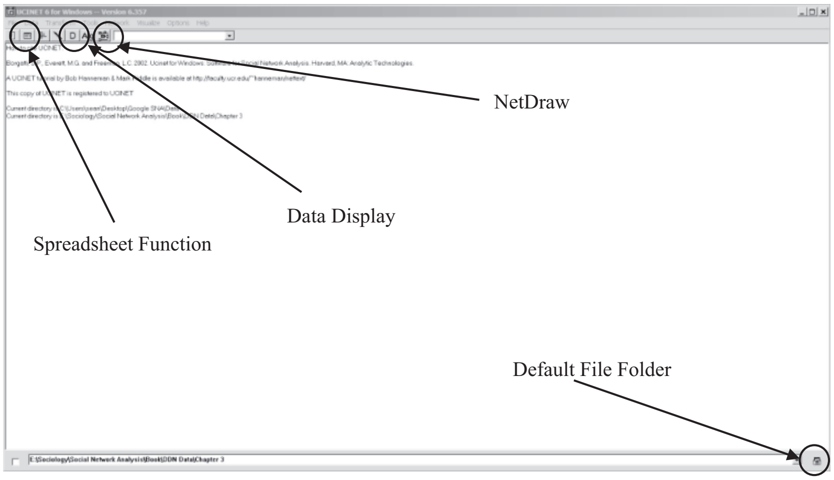
UCINET was initially developed by Linton Freeman at the University of California, Irvine (hence, the "UCI" in "UCINET"), and was later

[1] Multinet (Richards and Seary 2009) and Netminer (Cyram 2009) were the other two.

refined by others, in particular, Steve Borgatti and Martin Everett (Scott 2000:178–179). It is the best-known and most widely used social network software, primarily because it has been around for a long time and it contains a large number of SNA metrics and data management tools (Huisman and van Duijn 2005:275–280, 2011:585–586). Not only does it implement most of the routines needed for estimating measures of network topography (e.g., density, centralization, fragmentation), calculating actor centrality (e.g., degree, betweenness, closeness, eigenvector), identifying subgroups (e.g., cliques, components, factions, Newman groups), and estimating various measures of structural equivalence (e.g., structural, automorphic, regular), but it also includes tools for selecting subsets of files, merging and stacking datasets, transposing and/or recoding data, and importing and exporting of data in a variety of formats (Huisman and van Duijn 2005, 2011). UCINET is menu driven. Its commands call up dialog boxes that specify the inputs needed for it to run its various routines; results are displayed using Window's Notepad program and subsequently saved in log files.

UCINET's comprehensiveness is what makes it so valuable. Not only can you find and estimate most of the metrics you will need for SNA, if you can record social network data in UCINET format, but you can also get it into just about any other format you may need for analyzing with other software programs (e.g., Pajek, ORA, R, Excel, and SPSS). Moreover, UCINET is constantly being updated by its developers, who are adding new routines and fixing any bugs that come to light.

In UCINET, social network data are recorded in matrix format. Users can enter data using either UCINET's internal spreadsheet function or a commercial spreadsheet program, such as Microsoft Excel, which can then be imported (or pasted) into UCINET. UCINET also reads edge and node lists, which are quite useful when working with large datasets or when data are stored in database programs such as Microsoft Access. In storing data, UCINET uses a dual file system: one containing the actual data (extension .##d) and one containing information about the data (extension .##h). Because you need both files in order to analyze social network data in UCINET, this dual file system can occasionally lead to problems. For example, when estimating routines that create new data files, UCINET will sometimes store one of the two newly created files in separate folders, making it impossible to open or analyze the data until the two files are reunited. Because UCINET's creators regularly provide updates to the program in order to fix bugs and glitches, problems such as this are generally corrected relatively quickly. Nevertheless, they can be frustrating when they do happen. A related problem occurs when analysts share social network data with one another and send only one of the two files to the other. All this is to highlight how important it is to be aware of UCINET's dual file system.



How to cite UCINET:

Borgatti, S.P., Everett, M.G. and Freeman, L.C. 2002. Ucinet for Windows: Software for Social Network Analysis. Harvard, MA: Analytic Technologies.

A UCINET tutorial by Bob Hanneman & Mark Riddle is available at http://faculty.ucr.edu/~hanneman/nettext/

This copy of UCINET is registered to UCINET

Current directory is C:\Users\sean\Desktop\Google SNA\Data
Current directory is E:\Sociology\Social Network Analysis\Book\DDN Data\Chapter 3

E:\Sociology\Social Network Analysis\Book\DDN Data\Chapter 3

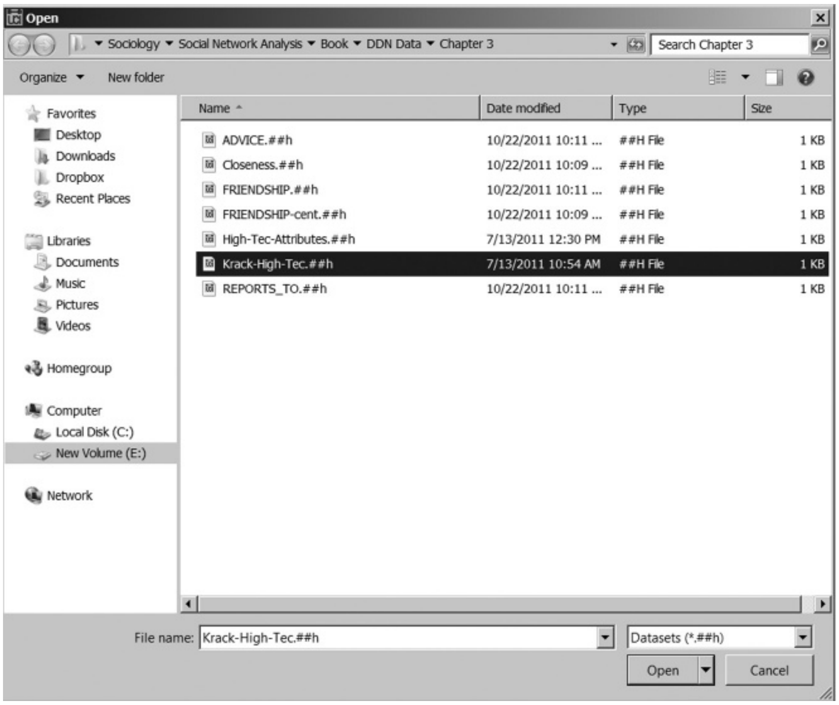
Figure 3.1. UCINET Interface

Getting Started with UCINET

When you open UCINET you encounter an interface similar to the one displayed in Figure 3.1. Across the top of the screen are a series of menus (i.e., *File*, *Data*, *Transform*, *Tools*, *Network*, *Visualize*, *Options*, and *Help*). We will explore some of the commands found under these menus that are useful to know before trying to do too much with UCINET. For now, we will postpone discussing other commands (e.g., commands for estimating various centrality measures). Just below the menus are a handful of speed buttons. Moving from left to right, the first closes UCINET, the second opens UCINET's internal spreadsheet program, the third opens Microsoft's Notepad program (useful for editing some data files), the fourth allows users to display a social network file in matrix format, and the fifth opens NetDraw, a social network drawing program that reads and displays UCINET files (see discussion of NetDraw in Section 3.2). Let's now turn to a discussion of some of the commands found under UCINET's various menus.

The *File* menu contains several useful functions, two of which we will discuss here. One is the *File>Change Default Folder* command, which tells UCINET where to look for social network data (its companion speed button is circled in Figure 3.1). When you open UCINET for the first time, the default folder will be `C:\Program Files\Analytic Technologies\Ucinet6\datafiles` or `C:\Program Files (x86)\Analytic Technologies\Ucinet6\datafiles` (note that UCINET lists the default folder at the bottom of its interface – see Figure 3.1), which is where all of the standard social network datasets that come with UCINET are stored. Issuing this command brings up a dialog

File>Change Default Folder



Name	Date modified	Type	Size
ADVICE.##h	10/22/2011 10:11 ...	##H File	1 KB
Closeness.##h	10/22/2011 10:09 ...	##H File	1 KB
FRIENDSHIP.##h	10/22/2011 10:11 ...	##H File	1 KB
FRIENDSHIP-cent.##h	10/22/2011 10:09 ...	##H File	1 KB
High-Tec-Attributes.##h	7/13/2011 12:30 PM	##H File	1 KB
Krack-High-Tec.##h	7/13/2011 10:54 AM	##H File	1 KB
REPORTS_TO.##h	10/22/2011 10:11 ...	##H File	1 KB

File name: Krack-High-Tec.##h

Datasets (*.##h)

Figure 3.2. UCINET Display Dialog Box

box that allows you to navigate through your computer's folders in order to pick the default folder of your choice. It is probably a good idea to change the default folder to where you are storing the data that are used with this book. Later, when working with your own data, you will want to change it to the folder where your data are stored. Another useful

File>Text Editor

command worth noting is the *File>Text Editor* command; it opens Microsoft's Notepad program, which can be used for editing data files.

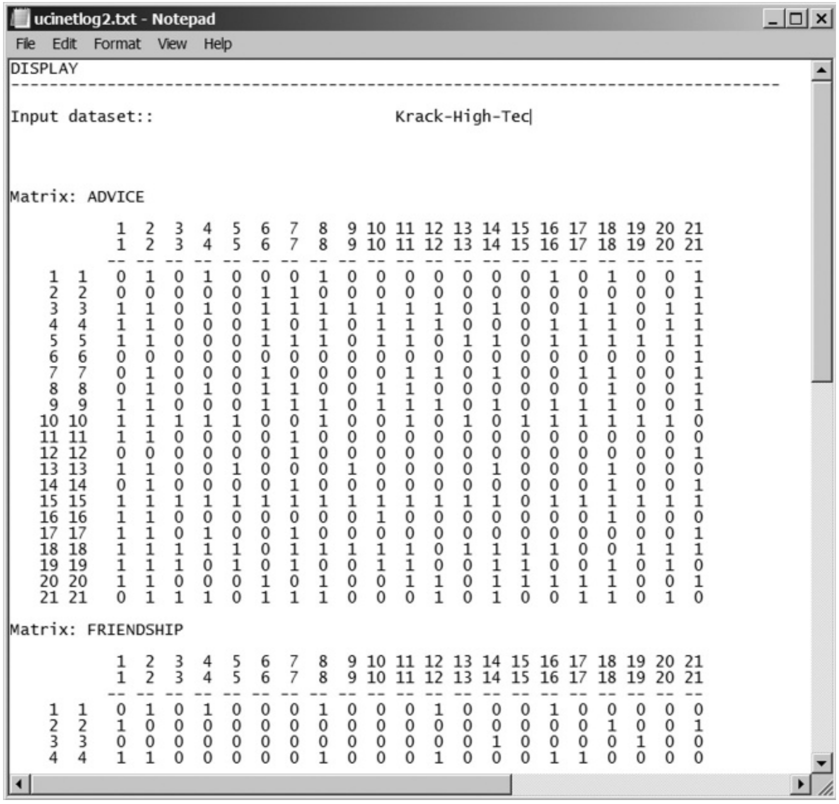
Data>Data Editors>Matrix Editor

The *Data* menu includes several helpful functions. The *Data>Data Editors>Matrix Editor* command accesses UCINET's internal spreadsheet function, which you can use for creating and editing social networks

Data>Display

(its companion speed button is circled in Figure 3.1). The *Data>Display* command (its companion speed button is circled) displays UCINET social network files. To get a sense of what this latter command does, use it to open a dialog box similar to that in Figure 3.2. Select the `Krack-High-Tec.##h` file and click "Open." (Note that although you cannot see the `Krack-High-Tec. ##d` file, it is there – if it was not, you would not be able to display the data file.)

This generates a UCINET output log (see Figure 3.3), which is generally how UCINET displays results and/or information. Note that in the



```
DISPLAY
------------------------------------------------------------------------------

Input dataset::                              Krack-High-Tec


Matrix: ADVICE

             1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
             1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
            -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
    1  1     0  1  0  1  0  0  0  1  0  0  0  0  0  0  0  1  0  1  0  0  1
    2  2     0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1
    3  3     1  1  0  1  0  1  1  1  1  1  1  1  0  1  0  0  1  1  0  1  1
    4  4     1  1  0  0  0  1  0  1  0  1  1  1  0  0  0  1  1  1  0  1  1
    5  5     1  1  0  0  0  1  1  1  0  1  1  0  1  1  0  1  1  1  1  1  1
    6  6     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
    7  7     0  1  0  0  0  1  0  0  0  0  1  1  0  1  0  0  1  1  0  0  1
    8  8     0  1  0  1  0  1  1  0  0  1  1  0  0  0  0  0  0  1  0  0  1
    9  9     1  1  0  0  0  1  1  1  0  1  1  1  0  1  0  1  1  1  0  0  1
   10 10     1  1  1  1  1  0  0  1  0  0  1  0  1  0  1  1  1  1  1  1  0
   11 11     1  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   12 12     0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1
   13 13     1  1  0  0  1  0  0  0  1  0  0  0  0  1  0  0  0  1  0  0  0
   14 14     0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  1
   15 15     1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  1  1  1  1  1  1
   16 16     1  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0
   17 17     1  1  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1
   18 18     1  1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  0  1  1  1
   19 19     1  1  1  0  1  0  1  0  0  1  1  0  0  1  1  0  0  1  0  1  0
   20 20     1  1  0  0  0  1  0  1  0  0  1  1  0  1  1  1  1  1  0  0  1
   21 21     0  1  1  1  0  1  1  1  0  0  0  1  0  1  0  0  1  1  0  1  0

Matrix: FRIENDSHIP

             1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
             1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
            -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
    1  1     0  1  0  1  0  0  0  1  0  0  0  1  0  0  0  1  0  0  0  0  0
    2  2     1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1
    3  3     0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0
    4  4     1  1  0  0  0  0  0  1  0  0  0  1  0  0  0  1  1  0  0  0  0
```

Figure 3.3. UCINET Output Log

upper-left corner of the output log is the label "Matrix #1: ADVICE." If you scroll down the log, you will discover that the file contains two more networks: FRIENDSHIP and REPORTS_TO. We will postpone discussing how you can (and why you might want to) "stack" networks in this way (see Chapter 4). For now, all you need to know is that David Krackhardt (1987a, 1992) collected data on twenty-one managers in a Silicon Valley company that manufactured high-tech equipment. He asked each manager whom they went to for advice and whom they considered their friends. He also determined to whom each manager reported from company documents. A "1" recorded in the cell of the advice network indicates that the manager listed on the far left seeks advice from the manager listed across the top. For instance, you can see in Figure 3.3 that manager #1 seeks advice from manager #2, but manager #2 does not seek advice from manager #1. Similarly, a "1" recorded in the cell of the friendship network indicates that the manager listed on the left considers the manager listed across the top to be a friend, and a "1" recorded in the cell of the reports to network indicates that the manager listed on the left reports to the manager listed across the top. Krackhardt also collected attribute data on the managers: age (in years), length of service or tenure (in years), their level in the corporate hierarchy (1 = CEO, 2 = Vice President, 3 = Manager), and the department to which they belong (coded 1, 2, 3, and 4, with the CEO in department 0). These data, however, are stored in a different file.

Presently, we will not explore any of the commands found under the *Transform*, *Tools*, or *Network* menus. A quick glance at the *Transform* and *Tools* menus indicates that UCINET includes a number of routines for transforming and analyzing social network data. We will not utilize all of these commands, but we will use a few. The commands that will occupy most of our time are found under the *Network* menu. Here you will find routines for estimating measures of network topography, calculating actor centrality, detecting subgroups, and locating structurally equivalent actors.

Help>Help Topics

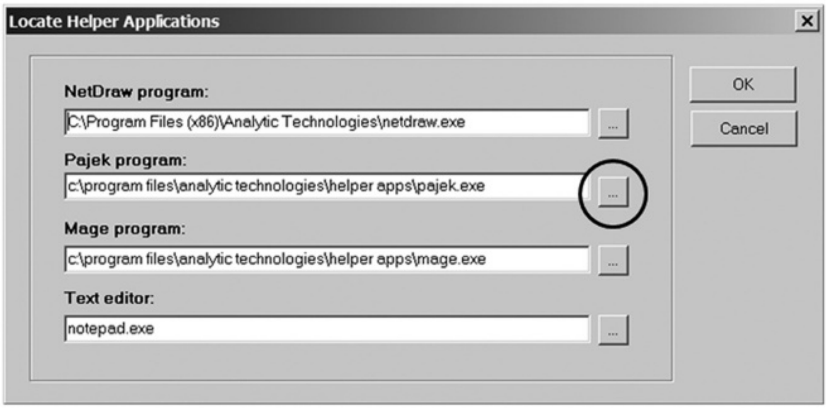
Jumping over to the *Help* menu, the *Help>Help Topics* command leads users to its "Overview of Help" function. The "Introduction Section" link provides access to a series of help topics that covers most (but not all) of UCINET's functions (when new functions are added to UCINET, it sometimes takes a while before they are covered). The "Overview of Help" function also provides a "Standard Datasets" link that provides users with a brief discussion of all the social network datasets that come with UCINET. Finally, the *Help>Hanneman Tutorial* command links users to the very helpful UCINET guide written by Hanneman and Riddle (2005).

Help>Hanneman Tutorial

The *Visualize* menu provides users with access to three visualization programs: NetDraw, Pajek, and Mage. We will focus primarily on NetDraw and Pajek in this book, although it is worth mentioning that before the advent of Pajek and NetDraw, Mage was the program of choice for many social network analysts (Freeman 1999, 2000; Freeman, Webster, and Kirke 1998), which is why this book includes an appendix on its use (see Appendix 3). Mage was initially developed as a device to be used in molecular modeling (Richardson and Richardson 1992), but social network analysts found it attractive because it produces elegant three-dimensional interactive images. In Mage, researchers can rotate images, turn parts of the display off and on, use the computer mouse to select and identify various parts of the network, and animate changes between different arrangements of objects.

Options>Helper Applications

Generally, you will want to leave the default settings included under the *Options* menu untouched. The one exception is the *Options>Helper Applications* command. Clicking on this command brings up a dialog box (see Figure 3.4). As you can see this allows users to change the folders where the four helper applications are currently incorporated



NetDraw program:
C:\Program Files (x86)\Analytic Technologies\netdraw.exe

Pajek program:
c:\program files\analytic technologies\helper apps\pajek.exe

Mage program:
c:\program files\analytic technologies\helper apps\mage.exe

Text editor:
notepad.exe

Figure 3.4. UCINET Helper Application Dialog Box

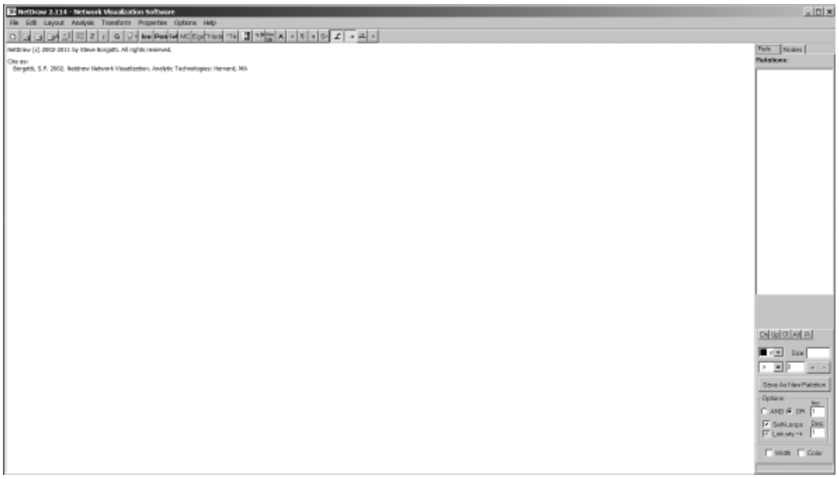
by UCINET (NetDraw, Pajek, Mage, and Notepad). Chances are you will never need to tell UCINET to look for NetDraw and Notepad in a different folder than the default folders set by UCINET. However, you will probably want to change where UCINET looks for Pajek because the version of Pajek distributed by UCINET is typically not the most recent. You will want to download and install Pajek in its default location. So before doing anything else, click on the radio button to the right of the Pajek program location (circled in Figure 3.4) and change Pajek's default location to `C:\pajek\Pajek\`. Doing this will ensure that each time you open Pajek from within UCINET you will open the most recent version (assuming, of course, that you update Pajek on a regular basis).

It is now time to turn our attention to NetDraw, a network visualization program that integrates nicely with UCINET. Because we can open NetDraw from within UCINET, there is no need to close UCINET at this time.[2]

3.3 NetDraw

In recent years, social network analysts have begun using a series of mathematical techniques to locate a social network's actors in such a way that the distances between them are meaningful. These algorithms use the concepts of space and distance in order to represent a network's internal structure, which they hope will reveal, among other things, which actors are "close" to one another and whether potential cleavages exist between sets of actors. NetDraw is a program developed by one of UCINET's

[2] If NetDraw does not open automatically from within UCINET, it is probably because UCINET is looking for it in the wrong place. Using UCINET's Helper Application dialog box (Figure 3.4), you can insert the correct path for finding NetDraw.


NetDraw 2.114 - Network Visualization Software
File Edit Layout Analysis Transform Properties Options Help
Rels Nodes
Relations:
Save As New Relation
Options
AND OR
Self-Loops
Links only
Width Color

NetDraw (c) 2002-2011 by Steve Borgatti. All rights reserved.

Cite as:
Borgatti, S.P. 2002. NetDraw Network Visualization. Analytic Technologies: Harvard, MA

Figure 3.5. NetDraw Home Screen

creators (Steve Borgatti) that is designed to draw networks using some of these algorithms. Network maps created by NetDraw can be rotated, flipped, resized, and stored in several different formats (Huisman and van Duijn 2005:306). In addition to its mapping algorithms, NetDraw also includes a handful of other algorithms for calculating centrality, detecting subgroups, identifying sets of key players, and so on.

Although technically a stand-alone program, NetDraw essentially functions as an extension of UCINET: It is distributed with UCINET, it can be opened from within UCINET, and it reads UCINET files without the need for using any importing and/or exporting functions (Huisman and van Duijn 2005:306). Whereas NetDraw's initial iterations did not handle large networks very well, more recent versions seem to do just fine. Moreover, if you save network data in NetDraw's native *.vna format, it can handle very large network files. Like UCINET, NetDraw's creators continually update the program with new procedures, routines, and bug fixes.

Getting Started with NetDraw

[UCINET] Visualize> NetDraw

To open NetDraw from UCINET, either use the *Visualize>NetDraw* command or click on the NetDraw speed button located just under the *Network* menu in UCINET (see Figure 3.1). This will open NetDraw's home screen, which should look similar to Figure 3.5. Like UCINET, NetDraw has a full set of menus as well as a series of speed buttons across the top of its interface. We will consider of a few of these here.

To get a sense of how NetDraw visualizes networks, open the Krackhardt dataset (`Krack-High-Tec.##h`) using NetDraw's

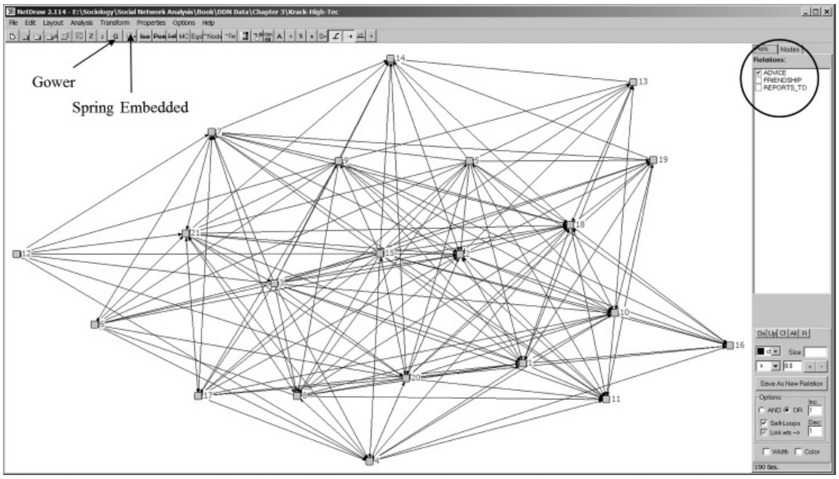


Figure 3.6. NetDraw Display of Krackhardt High-Tech Data

File>Open>Ucinet dataset>Network command. This should yield a network map that looks similar to the one displayed in Figure 3.6. A couple of features are worth noting before proceeding. First, along the upper right side of NetDraw's homepage (circled in Figure 3.6) note the tab entitled Rels, below which is located a list of the three networks (i.e., advice, friendship, reports to) included in the Krackhardt file. Chances are only the box next to the advice network is checked. This indicates that only those ties are currently shown in the graph. If you check the boxes next to "friendship" and "reports to networks," NetDraw will show those ties, as well. Second, NetDraw allows you to assign different colors to the various relations by using the dialog box (not shown) called up by the *Properties>Lines>Color>Relation* command. Note that NetDraw automatically assigns colors to the different types of ties. You can change these defaults by clicking on the color box. Generally, however, NetDraw's defaults tend to provide a nice contrast between the types of ties, so it is advisable to at least begin with the default colors. You can always change them later. Note also that when two actors share more than one tie, NetDraw colors these relationships gray. You can change this default as you do the others by clicking on the color box. Once you click "Apply," NetDraw assigns the different colors to the different types of ties.

[NetDraw] File>Open> Ucinet dataset> Network

Properties>Lines> Color>Relation

Mapping Algorithms in NetDraw

Social network analysts have long used network maps (i.e., sociograms, graphs) to visualize social networks. The goal of graphing networks is to place actors and the ties between them, such that those who share a similar pattern of ties (and often have ties with one another) are located close to one another in the graph and those who do not are located far from one another. It may be helpful to think of network maps as attempts to locate actors in *social space* as opposed to geographic space. That is, actors who are socially close (i.e., they share similar patterns of ties) are placed near one another in a network map, whereas those who are socially distant are placed far from one another.

A common technique in the early days of SNA was to construct the data around the circumference of a circle. NetDraw includes a command that *Layout>Circle* creates this type of network map: *Layout>Circle*. Unfortunately, while this approach is useful, the relations between the graph's points do not reflect any specific mathematical properties. The points are arranged arbitrarily, and the distances between them are meaningless, which, depending on how they were arranged, can lead to varying interpretations of the data (McGrath, Blythe, and Krackhardt 1997). NetDraw also includes an *Layout>Random* algorithm that randomly allocates a network's nodes (*Layout>Random*), but this type of layout suffers from the same weaknesses as a circular layout; that is, the distances between the graph's nodes mean nothing.

In recent years, analysts have begun using a series of mathematical algorithms to locate the points of a network in such a way that the distances between them are meaningful. Multidimensional scaling (MDS) is one such technique. MDS uses the concepts of space and distance to represent a network's structure, which, in turn, can help reveal, among other things, which actors are "close" to one another, as well as potential divisions between them (Wasserman and Faust 1994). The typical input to MDS is a one-mode symmetric matrix (see Chapter 4) that contains measures of similarity or dissimilarity between pairs of actors. The output generally consists of a set of estimated distances among pairs of actors that can be then represented in one-, two-, three-, or higher-dimensional space (Kruskal and Wish 1978; Wasserman and Faust 1994). Distance here differs from distance in graph theory. In graph theory, the distance between two points is measured in terms of path length. With MDS it "is a distance that follows a route 'as the crow flies,' and that may be across 'open space' and need not – indeed, it normally will not – follow a graph theoretical path" (Scott 2000:148–149).

There are different types of multidimensional scaling: metric and non-metric MDS. Metric MDS takes a given matrix of similarities or dissimilarities among actors and calculates a set of points in *k*-dimensional space, such that the distances between them correspond as closely as possible to the input. There are some limitations to using metric MDS for visualizing social networks. Many networks are binary (i.e., dichotomous) in that they only indicate either the presence or absence of a tie. Unfortunately, we cannot use binary data to directly estimate proximities.

Instead, we need to first convert it into other measures, such as correlation coefficients, before calculating distances between actors. However, such data conversion can lead researchers to draw unjustifiable conclusions about the data. For example, although it is reasonable to assume that an actor with four ties is more central than one with only two, we cannot be certain that the former is twice as central as the latter (Scott 2000:157). Even when the data are valued, metric assumptions may be inappropriate. Imagine two actors with four ties between them and two other actors with only one. Although the first set of actors is almost certainly more closely tied than the second set, it is difficult to know how much more closely tied together they are (Scott 2000:157). Nonmetric MDS procedures offer a solution to this problem. Like metric MDS procedures, they use symmetrical adjacency (one-mode) matrices in which the cells reflect the similarities or dissimilarities among actors, but unlike metric MDS procedures, they treat the data as ordinal, seeking "a solution in which the rank ordering of the distances is the same as the rank ordering of the original values" (Scott 2000:157). Moreover, nonmetric MDS is often preferred because it tends to provide a better "goodness-of-fit" (stress) statistic.[3]

Currently, NetDraw only includes metric MDS algorithms although plans for nonmetric MDS algorithms are in the works.[4] Implementing NetDraw's metric MDS routine is relatively straightforward with its *Layout>Graph Theoretic layout>MDS* command. Nodes that are "socially" close to one another (because there is a tie between them or they are tied to a common friend) should be located close to one another in the graph, whereas nodes that are socially distant from one another (i.e., they are not tied to one another nor do they share a common tie) should be located far from one another in the graph. It is important to note that there is no single correct way to graph the data (i.e., there is not a single "solution"), which is why analysts will generally want to implement an algorithm (or various algorithms) multiple times. Another metric MDS algorithm included in NetDraw is the *Layout>Graph Theoretic layout>Gower* command (or click on the "G" speed button – see Figure 3.6). Note that all of these routines provide a slightly different network map from one another, which makes it difficult to choose which network map fits the data best. An alternative is to calculate the coordinates in UCINET and import them into NetDraw (see Appendix 3).

Layout>Graph Theoretic layout>MDS

Layout>Graph Theoretic layout>Gower

[3] The lower the stress (0 = perfect fit), the better. Generally, stress levels below "0.1" are considered excellent while levels above "0.2" are considered unacceptable.

[4] See the options found under the *Layout>Scaling/Ordination* submenu – only the Iterative metric MDS algorithm has been implemented and produces a very different solution than the other MDS routines found in NetDraw. Users can calculate metric and nonmetric MDS coordinates in UCINET and then use these coordinates in NetDraw, however. See Appendix 3 for details.

The advantage of doing this is that UCINET's MDS routines provide users with stress statistics that indicate how well the coordinates fit the data.

Another common set of routines for graphing social networks are spring-embedded algorithms. Pajek, in fact, currently (i.e., through version 2.04) uses only spring-embedded algorithms. These algorithms treat the nodes as pushing and pulling on one another and seek to find an optimum solution wherein there is a minimum amount of stress on the springs connecting the whole set of nodes (Freeman 2000). Generally, ties between nodes are treated as an attractive force (a "spring" pulling them together), whereas nodes that do not share a tie are pushed apart (Moody 2001). The spring-embedding algorithm in NetDraw can be implemented using the *Layout>Graph Theoretic layout>Spring embedding* command. This calls up a dialog box (not shown) that provides a variety of options for using the spring-embedding layout. It is a good idea to use NetDraw's defaults, although varying the options tends not to change the graph's layout too dramatically. The "lightning bolt" speed button is another way to implement this routine.

Layout>Graph Theoretic layout>Spring embedding

Freeman (2005:251) refers to MDS and spring-embedded algorithms as search algorithms because they all involve a search for the optimal location for nodes. He contrasts these with determinate approaches, which are based on the singular value decomposition (SVD) algebraic procedure:

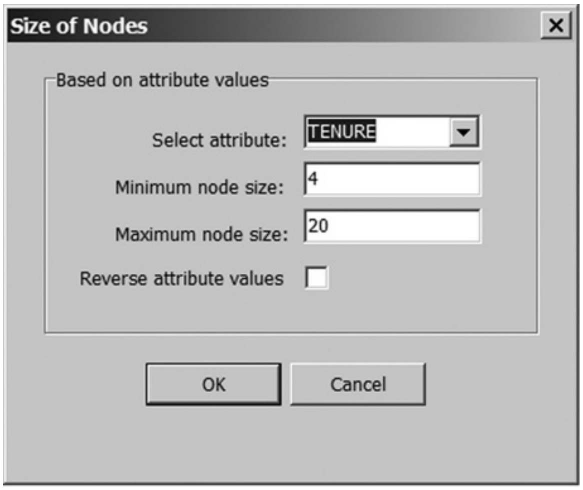
> SVD transforms the N original variables into N new variables, or dimensions. These new dimensions are ordered from largest to smallest in terms of how much of the variance, or patterning, in the original data is associated with each. The most variance is always associated with the first dimension. Each succeeding dimension is, in turn, associated with progressively less of the variance. If a one-, two-, or three-dimensional visual image is going to be useful, the hope is that the first or the first two or three of these new dimensions will be associated with virtually all the variance contained in the original data (Weller and Romney 1990). If, in contrast, the first few dimensions are associated with very little of the original variance, SVD will not yield useful results. (Freeman 2005)

NetDraw includes a mapping algorithm that uses SVD, principal components, which is accessible through the *Layout>Graph Theoretic layout>Principal Component* command and typically produces network maps that look considerably different from the others.

Layout>Graph Theoretic layout>Principal Components

Visualizing Attribute Data in NetDraw

A nice feature of NetDraw is that it allows analysts to incorporate attribute data into its network graphs. Open the attribute file



Select attribute: TENURE
Minimum node size: 4
Maximum node size: 20
Reverse attribute values ☐

Figure 3.7. Node Size Dialog Box

(`High-Tec-Attributes.##h`) associated with the Krackhardt data (which should still be loaded into memory), using the *File>Open>Ucinet dataset>Attribute data* command. You can use attributes (i.e., variables) to vary the size, shape, and color of the nodes. In general, you will want to visualize nominal attributes, that is, variables with two or more categories but where there is no intrinsic ordering to the categories (e.g., gender, ethnicity, country of origin), by varying either the shape or size of the node. By contrast, when visualizing ordered or continuous variables (e.g., age, income, education level), it is better to vary the size of the node.

File>Open> Ucinet dataset> Attribute data

Let's begin by adjusting the size of the nodes based on tenure using the *Properties>Nodes>Symbols>Size>Attribute-based* command. This should call up a dialog box similar to Figure 3.7. Using the drop-down menu choose the attribute you wish to use to vary the size of the node (in this case *TENURE*). Next, adjust the shape of the node using the *Properties>Nodes>Symbols>Shape>Attribute-based* command to reflect the department to which each manager belongs and you should get a network map similar to Figure 3.8. Typically, varying node color rather than shape is more effective for visualizing nominal attributes, but because this book cannot display color, we will leave it for readers to experiment with NetDraw's *Properties>Nodes>Symbols>Color>Attribute-based* command.

Properties> Nodes> Symbols>Size> Attribute-based

Properties> Nodes> Symbols>Shape> Attribute-based

Properties> Nodes>Symbols> Color> Attribute-based

There are a number of other functions available within NetDraw, most of which we will consider further on, but three worth mentioning now. Note that there are three sets of speed buttons included in NetDraw: "Z z," "A a," and "S s." The first increases or decreases the size of the graph,

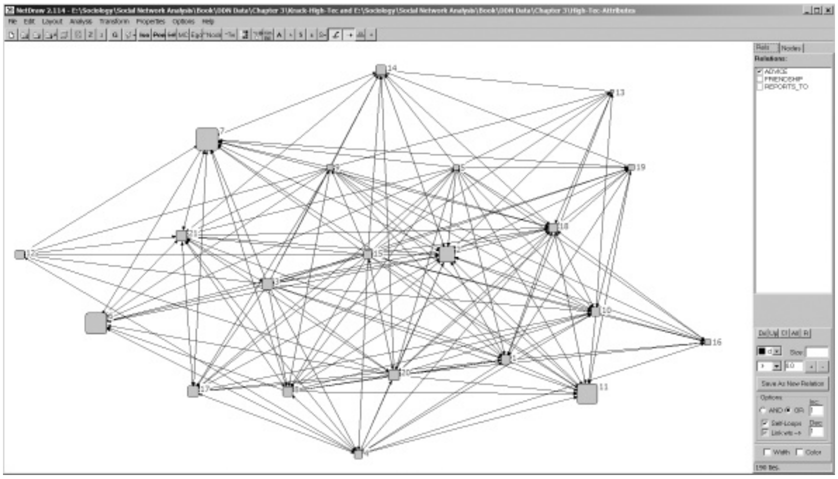


Figure 3.8. NetDraw Map of Krackhardt Data Size Reflecting Tenure

the second increases or decreases the size of the labels in the graph, and the third increases or decreases the size of the nodes in the graph.

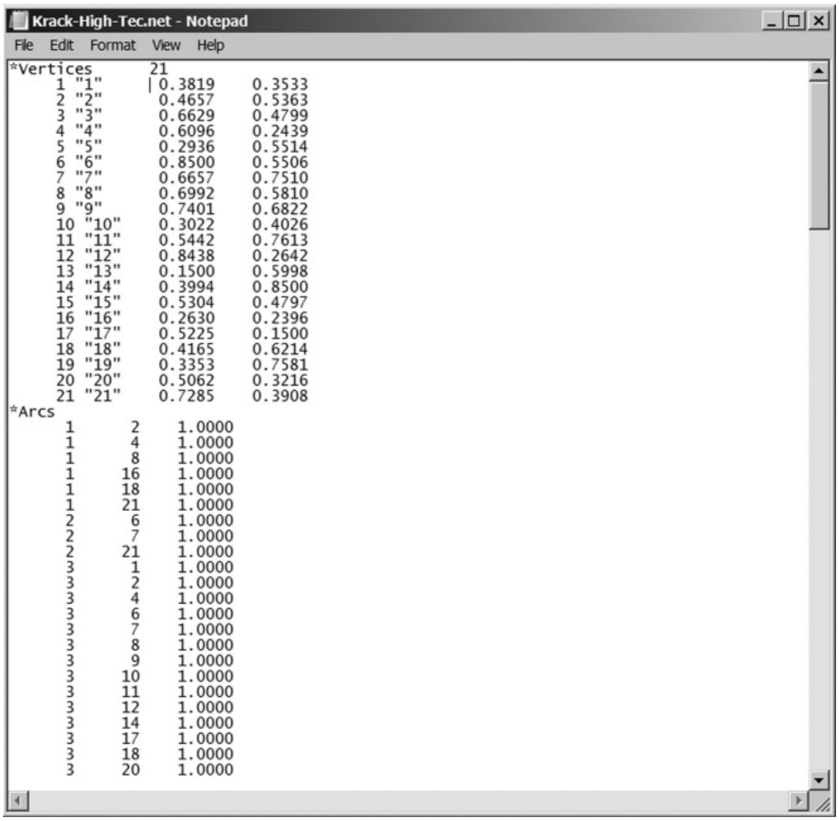
File>Save Data As>Vna

Before turning to our exploration of Pajek, save the Krackhardt data in NetDraw's native format (*.vna) using its *File>Save Data As>Vna* command. A helpful feature of the VNA format is that it saves both relations *and* attributes in a single file, which is useful when working with the same data over and over again. It also avoids the potential problems that UCINET's dual file system may cause.

3.4 Pajek

Pajek – which means "spider" in Slovenian – was created by Vladimir Batagelj and Andrej Mrvar in 1996 and is designed to handle very large datasets (Scott 2000:179–180). Although Pajek does not include as many algorithms as UCINET, it still offers many of those that analysts use. In particular, it includes routines that lend themselves to the visualization and simplification of large networks while allowing users to visualize networks in two or three dimensions (Huisman and van Duijn 2005:280, 2011:587). It runs on Windows-compatible computers, can be downloaded for free, and is routinely updated by its developers. Pajek uses six different types of data objects, or structures:

- Networks (nodes/actors and ties)
- Partitions (discrete classification of actors where each actor is assigned to one *and only one* class – for example, a dark network partition might assign each actor to a specific role – it is here that actors' nominal attributes are stored)



```
*Vertices      21
      1 "1"      0.3819     0.3533
      2 "2"      0.4657     0.5363
      3 "3"      0.6629     0.4799
      4 "4"      0.6096     0.2439
      5 "5"      0.2936     0.5514
      6 "6"      0.8500     0.5506
      7 "7"      0.6657     0.7510
      8 "8"      0.6992     0.5810
      9 "9"      0.7401     0.6822
     10 "10"     0.3022     0.4026
     11 "11"     0.5442     0.7613
     12 "12"     0.8438     0.2642
     13 "13"     0.1500     0.5998
     14 "14"     0.3994     0.8500
     15 "15"     0.5304     0.4797
     16 "16"     0.2630     0.2396
     17 "17"     0.5225     0.1500
     18 "18"     0.4165     0.6214
     19 "19"     0.3353     0.7581
     20 "20"     0.5062     0.3216
     21 "21"     0.7285     0.3908
*Arcs
       1      2   1.0000
       1      4   1.0000
       1      8   1.0000
       1     16   1.0000
       1     18   1.0000
       1     21   1.0000
       2      6   1.0000
       2      7   1.0000
       2     21   1.0000
       3      1   1.0000
       3      2   1.0000
       3      4   1.0000
       3      6   1.0000
       3      7   1.0000
       3      8   1.0000
       3      9   1.0000
       3     10   1.0000
       3     11   1.0000
       3     12   1.0000
       3     14   1.0000
       3     17   1.0000
       3     18   1.0000
       3     20   1.0000
```

Figure 3.9. Pajek Net File (Edge List)

- Vectors (continuous properties of actors, that is, ordered attributes – for example, an actor's age, level of education, centrality score, etc. – are stored here)
- Permutations (reordering of nodes)
- Clusters (subsets of nodes)
- Hierarchies (hierarchically ordered clusters and nodes)

We will not explore all of these data objects – indeed, we will utilize primarily the first three – networks, partitions, and vectors – but it is helpful to at least be aware of them.

Pajek stores network data as an edge list, which is simply a list of vertices (i.e., actors/nodes) and edges/arcs (i.e., ties). Figure 3.9 displays a portion of the Pajek Krackhardt High-Tech network data file. It begins by specifying the number of vertices (i.e., actors); then each vertex is identified on a separate line by a serial number, a label (enclosed in quotation marks), and two numbers between 0 and 1, which are simply two sets of coordinates for visualizing the network data in two-dimensional space.

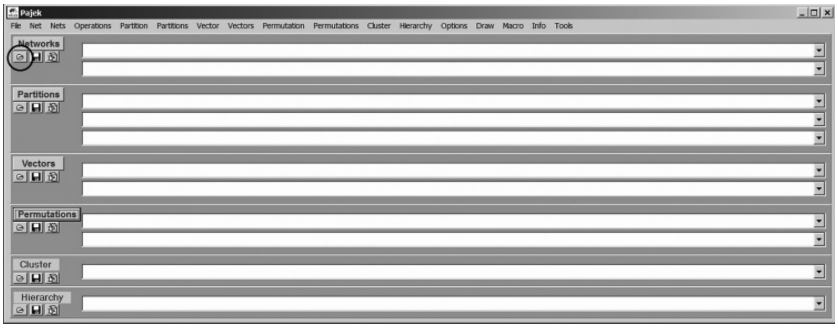

Pajek
File Net Nets Operations Partition Partitions Vector Vectors Permutation Permutations Cluster Hierarchy Options Draw Macro Info Tools
Networks
Partitions
Vectors
Permutations
Cluster
Hierarchy

Figure 3.10. Pajek Main Screen

The list of vertices is then followed by a list of arcs (or edges). Each line identifies the number of the sending and receiving actors and the value of the tie between them. Thus, you can see that the first manager (vertex 1) sought advice from managers 2, 4, 8, 16, 18, and 21. An advantage of edge lists is that they tend to be smaller in size because you do not have to code nonties (i.e., 0s), which means that they can handle very large social networks.

Like NetDraw, Pajek allows users to load and keep multiple networks and other data objects (e.g., partitions) in memory at the same time. This is quite helpful because, like other social network software programs, most of Pajek's routines generate new networks or other data objects. All of these can then be stored in what Pajek calls a "project file," which means that after analyzing one or more social networks, users can save all of their work in a single file. This decreases the likelihood that users will have to "recreate the wheel" in doing their analysis.

Pajek's primary drawback is that it contains fewer algorithms than UCINET (although it does include a few that UCINET does not have), and its network manipulation features are somewhat limited. Thus, many analysts sometimes use UCINET for data manipulation and then Pajek for visualization. To Pajek's credit, however, it now allows users to call up the statistical packages R and SPSS to perform procedures not available in Pajek.

Getting Started with Pajek

The Pajek main screen looks very different from that of UCINET and NetDraw (see Figure 3.10). As you can see, it is organized by the type of data object or structure. Network data are loaded into the Network drop-down menu, partition data into the Partition drop-down menu, vector data into the Vector drop-down menu, and so on. In order to make working with more than one network, partition, vector, or permutation at the same time easier, beginning with version 1.21, Pajek's main screen can display two or three drop-down menus for each of the data objects. You can change the number of visible drop-down menus by clicking on the buttons labeled "Networks," "Partitions," "Vectors," or "Permutations" (see Figure 3.10). As we will see, this is a very helpful feature. However, users need to be careful when working with various objects in Pajek. For example, if you are seeking to obtain information on a particular network using Pajek's *Info>Network>General* command, you have to make sure that *Network* appears in the first or top Network drop-down menu. The same holds true if you are getting information on a partition, vector, or hierarchy using the *Info>Partition*, *Info>Vector*, and *Info>Hierarchy* commands, respectively, because in most cases, Pajek looks to the object that is listed first.

Info>Network>General

Info>Partition

Info>Vector

Info>Hierarchy

To open a network file into Pajek, we use Pajek's *File>Network>Read* command or click on the folder icon located under the "Networks" button (circled in Figure 3.10). Here, however, we are going to open a Pajek project file, which like NetDraw's *.vna file format, allows users to store network and attribute data in a single file. To open the Krackhardt project file, use Pajek's *File>Pajek Project File>Read* command (or press the F1 key) and select the `Krackhardt High Tech.paj` file.[5] Note that Pajek loads all three networks in the Network drop-down menu; the department attribute in the Partition drop-down menu; and the age, level, and tenure attributes in the Vector drop-down menu (not shown). You can open any of these drop-down menus and see the list of networks, partitions, and vectors stored in the project file by left-clicking on the triangle on the right.

[Main Screen] File>Network>Read

File>Pajek Project File>Read

The functions available in Pajek are vast, and probably only Pajek's developers along with their colleague Wouter de Nooy know all that Pajek can do (de Nooy, Mrvar, and Batagelj 2005, 2011). Pajek's functions are organized based on the task analysts plan to use. For example, if you are working with an individual network, then you will primarily use the functions found under the *Net* menu. If you are manipulating two networks, then the algorithms located under the *Nets* menu will probably be useful. Similarly, if you are working with a single partition (or vector), then you will most likely turn to those found under the *Partition* (or *Vector*) menu, whereas if you are working with two or more partitions (or vectors), then you will look to those functions found under the *Partitions* (or *Vectors*) menu. A similar logic holds true if you are manipulating or analyzing permutations, clusters, or hierarchies. Often you will work with a network in conjunction with a vector and/or partition. If you are

[5] Pajek does not currently read UCINET files. However, UCINET allows you to export network and attribute data in formats that Pajek can read. We will explore these features in subsequent chapters.

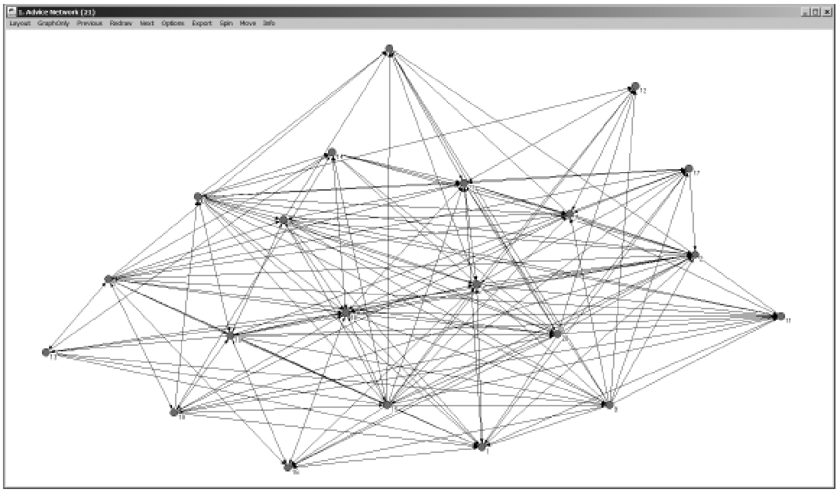

1. Advice Network (21)
Layout GraphOnly Previous Redraw Next Options Export Spin Move Info

Figure 3.11. Pajek Draw Screen

seeking to manipulate network data, then it is likely that you will use one or more of the algorithms located under the *Operations* menu. If you plan to visualize them, then the *Draw* menu will be where you turn.

Mapping Algorithms in Pajek

[Main Screen] Pajek>Draw> Draw

[Draw Screen] Options>Color> Background

Options>Values of Lines> Similarities

Layout> Energy> Starting Positions> Given xy

Options>Size> of Vertices

To view Krackhardt's network, make sure that the Advice network is showing in the first or top Network drop-down menu and then issue *Pajek>Draw>Draw* command. This will open Pajek's Draw screen, which will look similar, but probably not identical, to Figure 3.11. Before exploring Pajek's drawing capabilities, we should choose a couple of options. To change the background use the *Options>Color>Background* command located in Pajek's Draw screen (not Main screen). This brings up a dialog box showing a number of colors from which to choose. Select the background of your choice. Light backgrounds are generally a good choice if you plan to print network screenshots. You will generally want to tell Pajek that a tie between two actors indicates a similarity or closeness between them. This is done with the *Options>Values of Lines>Similarities* command. Next, select the *Layout>Energy>Starting Positions>Given xy* (i.e., horizontal and vertical dimensions) option; this indicates that with each new drawing Pajek begins with the nodes/actors where they are (as opposed to randomly placing them before drawing the network map); this assumes that repeated drawings of networks yield increasingly better solutions. Finally, choose the node size you prefer with Pajek's *Options>Size>of Vertices* command. This brings up a dialog box where you enter the desired size (note that the size of the nodes in

Figure 3.11 is set to 8). Pajek includes an "auto-size" option (type a "0" in the dialog box), which is quite useful when you are varying the size of a node based on an attribute that contains considerable variation (e.g., betweenness centrality).

As noted previously, through version 2.04 Pajek has only implemented spring-embedding algorithms for its layouts. In particular, it uses two: Fruchterman Reingold (1991) and Kamada-Kawai (1989). The Kamada-Kawai algorithm assumes attraction between actors that are tied with one another and repulsion between actors that are not. The Fruchterman Reingold algorithm is similar, except that instead of assuming attraction or repulsion between actors, it attempts to simulate a system of mass particles where the vertices simulate mass points repelling each other while the edges simulate springs with attracting forces. It then tries to minimize the "energy" of this physical system.

[Draw Screen] Layout>Energy>Fruchterman Reingold>2D, 3D

Layout>Energy>Kamada-Kawai>Free

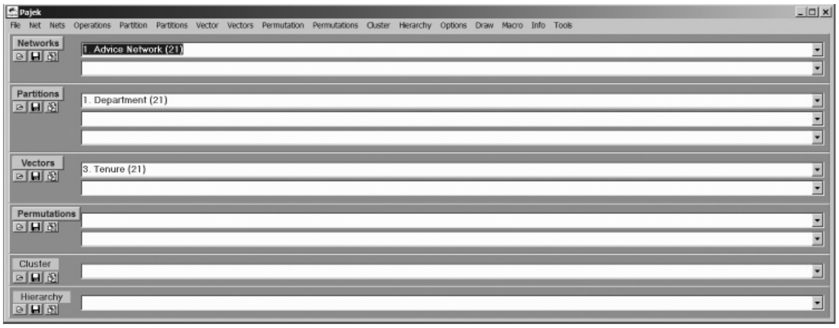
To implement the Fruchterman Reingold layout algorithm, use Pajek's *Layout>Energy>Fruchterman Reingold>2D, 3D* command; to implement Kamada-Kawai, use its *Layout>Energy>Kamada-Kawai>Free* command. Which one should you choose? On the one hand, the Kamada-Kawai algorithm works well with small, connected networks but is not recommended for networks with more than 500 actors (de Nooy et al. 2011:17). It draws layouts similar to nonmetric MDS and tends to be better than Fruchterman Reingold at mapping sparse networks. Unfortunately, it places isolated nodes randomly on the graph, so they can sometimes appear to be in the center of a network when in reality they are not. The Fruchterman Reingold algorithm is faster than Kamada-Kawai and works well with large networks. It also is able to map networks in both two-dimensional and three-dimensional space (whereas Kamada-Kawai only maps networks in two-dimensional space). As long as the network is not too large, it is often helpful to first visualize the network using Fruchterman Reingold, and then (after making sure that the *Layout>Energy>Starting Positions>Given xy* option has been selected) use Kamada-Kawai.

Layout>Energy>Starting Positions>Given xy

Pajek does not provide goodness-of-fit statistics for network maps, which makes it more difficult for analysts to objectively evaluate the accuracy of a network map. It does, however, evaluate the aesthetic properties of a network drawing with the series of commands found under the *Draw>Info* submenu. For example, some argue that the number of crossing lines in a graph should be kept to a minimum and that unconnected vertices should not be drawn too closely to one another (de Nooy et al. 2011:18). And, if you select the *Draw>Info>Closest Vertices* command, Pajek identifies the nodes that perform the worst in this regard and assigns them a different color, making them easy to identify. You can then use the mouse to drag the two vertices farther apart.

[Draw Screen] Draw>Info

[Main Screen] Draw>Info>Closest Vertices


Pajek
File Net Nets Operations Partition Partitions Vector Vectors Permutation Permutations Cluster Hierarchy Options Draw Macro Info Tools
Networks
Partitions
Vectors
Permutations
Cluster
Hierarchy

1. Advice Network (21)

1. Department (21)

3. Tenure (21)

Figure 3.12. Pajek's Main Screen

Visualizing Attribute Data in Pajek

Draw>Draw-Partition-Vector

We will now look at how to work with attribute data in Pajek. Return to Pajek's main screen. With the Krackhardt advice network highlighted in the first Network drop-down menu, the *Department* partition highlighted in the first Partition drop-down menu and the *Tenure* vector highlighted in the first Vector drop-down menu (see Figure 3.12), select Pajek's *Draw>Draw-Partition-Vector* command and energize it using one or both of the visualization algorithms. This should produce a drawing similar (but again, not identical) to the one in Figure 3.13, where the color of the nodes indicates the department to which each manager belongs and the size of the nodes indicates each manager's tenure. In

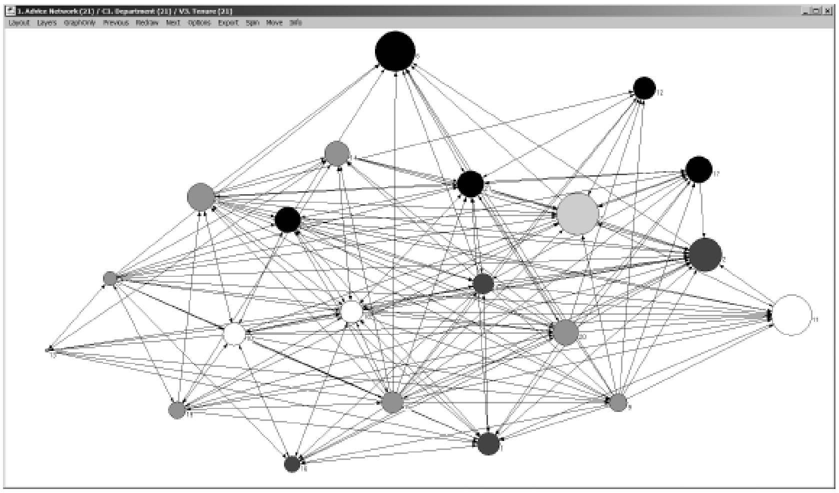

1. Advice Network (21) / C1. Department (21) / V3. Tenure (21)
Layout Layers GraphOnly Previous Redraw Next Options Export Spin Move Info
12
17
11
1

Figure 3.13. Krackhardt Advice Network with Varying Node Size and Color (Pajek)

Table 3.1. *Meta-matrix representation of a meta-network*

	Agents	Knowledge	Tasks	Organizations
Agents	Social Network	Knowledge Network	Assignment Network	Membership Network
Knowledge		Information Network	Needs Network	Organizational Capability
Tasks			Precedence Network	Institutional Support Network
Organizations				Interorganizational Network

Figure 3.13, the layout was obtained using the Fruchterman Reingold two-dimensional algorithm, and the node color has been set using one of Pajek's greyscale options. The latter option is accessed with the *[Draw Screen] Options>Colors>Partition Colors>for Vertices* command, which calls up a dialog box (not shown) where you can select specific colors for various partition classes or use one of Pajek's preset options. Here, the preset "GreyScale 1" option was chosen. As you can see, there is some clustering based on department, and tenure does not appear to determine the centrality of managers in the network. We would, of course, want to test this latter observation by correlating tenure with various centrality measures – but we are getting ahead of ourselves. It is now time to turn our attention to ORA.

[Draw Screen] Options> Colors>Partition Colors>for Vertices

3.5 ORA (Organizational Risk Analyzer)

Organizational Risk Analyzer (ORA; Carley 2001–2011) is a recent entry in the SNA arena that has gained a wide following in a very short time. For instance, in Huisman and Duijn's (2005) original review of available social network packages, ORA was not even mentioned. By 2011, it was one of the five general-purpose packages the authors explored in depth (Huisman and van Duijn 2011). ORA was developed by Kathleen Carley, who is a former student of Harrison White and currently teaches at Carnegie Mellon University. ORA is user-friendly and designed to find those actors, types of skills or knowledge, and tasks that are critical to a network's performance. Lying at the heart of ORA's approach is the notion of a meta-matrix of networks (Carley 2001–2011; Carley, Lee, and Krackhardt 2002) that includes not only social networks but also knowledge networks (who knows what), information networks (what ideas are related to what), assignment networks (who is doing what), need networks (what knowledge is needed to do the task), and so on. Table 3.1 illustrates (but does not entirely capture) ORA's meta-network approach (adapted from Carley 2003b).

Unlike UCINET, NetDraw, and Pajek, ORA is report based; that is, when analyzing networks, you typically do not request a single metric from ORA, but rather a report containing a series of related metrics. For example, if you wish to identify central actors using ORA, you would probably ask for ORA's "Standard Network Analysis" report, which provides the estimated scores of the most commonly used centrality measures. Like Pajek, ORA includes both a main screen, where you can analyze networks using a variety of algorithms, and a draw screen, where you can visualize networks using different mapping algorithms in either two or three dimensions. A helpful feature of ORA's draw screen is that it includes its own analytical capabilities; that is, it implements several algorithms and metrics (in this respect, it is similar to NetDraw), which is an advantage it has over Pajek. ORA also includes features for simulating various scenarios (e.g., the effect of removing or isolating various actors from a network at different points of time), geospatially analyzing and mapping social networks that can then be plotted in various geospatial programs, creating different types of charts (e.g., histograms, bar charts, and scatter plots) that can be quite useful for analysis and presenting results, and analyzing changes over time with its "view measures over-time" feature.

Like the other programs, ORA has its weaknesses. For example, although its report-based approach is quite handy, its reports will sometimes include metrics that are inappropriate for a particular network, such as estimating closeness centrality (Freeman 1979) when analyzing a disconnected network, or calculating Krackhardt's (1994) measure of hierarchy when examining an undirected (i.e., symmetric) network. This could lead analysts to conclude that certain actors are more important than they really are, which of course could lead to using mistaken assumptions when crafting strategies. And although its visualization capabilities are adequate and offer some useful options, the network maps it produces tend not to be as robust as Pajek's, at least when visualizing valued networks (i.e., where ties between actors are ordered rather than just "0s" and "1s"). Nevertheless, ORA is a powerful tool. Moreover, because it has been funded in part by Department of Defense dollars,[6] it is quite popular among those in the intelligence community.

Getting Started with ORA

Like Pajek, ORA has both a main screen and a draw screen (the ORA visualizer). Figure 3.14 presents ORA's main screen. As you can see it is broken down into three sections or panels: the (1) Meta-Network Manager, (2) Network Information/Editor, and (3) Report panels. The

[6] See http://www.casos.cs.cmu.edu/projects/ora/sponsors.php.

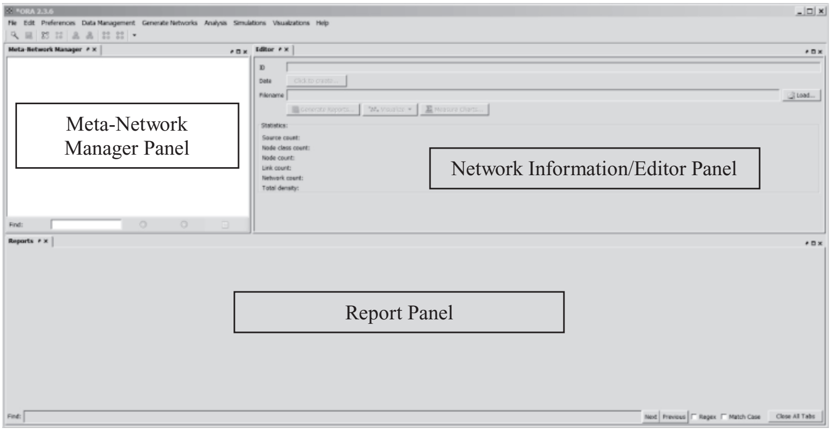


Figure 3.14. ORA Main Screen

first displays a list of the meta-networks that are currently loaded into ORA. In this case, all three of the Krackhardt networks have been loaded into ORA. We can view information about a particular network by single-clicking and highlighting it. Here, the advice network has been selected, causing its basic statistics to appear in the Network Information/Editor panel. If you selected the Editor tab, you could examine (and modify, if you so choose) the ties between the actors in the network. Finally, the Report panel is where ORA displays the results of any analyses that have been run on a network or set of networks.

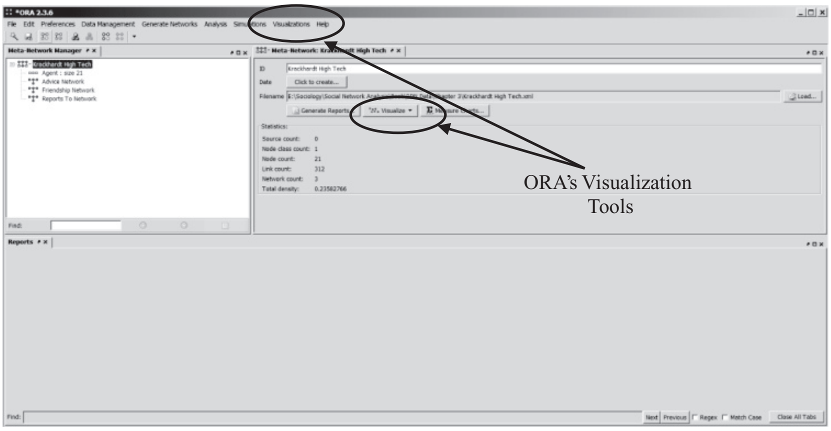
ORA's menus follow a logical pattern. We will spend most of our time utilizing the tools found under the *Analysis* and *Visualizations* menus, although functions found under other menus will be of use to us as well. For example, under the *File* menu you will find commands for reading, importing, saving, and exporting data, and under the *Edit* menu there are tools for copying and pasting data. The *Preferences* menu includes features for preferred displays and fonts. Using, for instance, the *Preferences>Other>General>Select* command, I changed ORA's default color (blue) to gray. The *Data Management* menu has a series of tools for manipulating data, a few of which we will use throughout the course of this book. We will not utilize too many of the features found under the *Simulations* or *Generate Networks* menus, although in the next chapter we will see how to record network data in ORA.

[Main Screen] Preferences> Other>General> Select

Mapping Algorithms in ORA

To read network data into ORA, use ORA's *File>Open Meta-Network* command and select the `Krackhardt High Tech.xml` file that accompanies this book. ORA's visualizer can be accessed by clicking "Visualize" on the Network Information/Editor panel or by accessing the

[Main Screen] File>Open Meta-Network



ID: Krackhardt High Tech
Date: Click to create...
Filename: E:\Sociology\Social Network Analysis\...\ORA Data\Chapter 3\Krackhardt High Tech.xml

Statistics:
Source count: 0
Node class count: 1
Node count: 21
Link count: 312
Network count: 3
Total density: 0.23582766

Figure 3.15. Accessing ORA's Visualizer from Ora's Main Screen

Visualizations> View Networks

Visualizations> View Networks command (see Figure 3.15). Both features allow you to visualize the network in either two or three dimensions. To do so, either hold down the "Visualize" button and select the "Network Visualizer" option or choose one of the two dimensions from the menu command. Because the analytical capabilities of ORA's two-dimensional visualizer are more robust, we will focus more on it. After selecting the two-dimensional option for viewing the Krackhardt data, you should get a network drawing that looks similar to Figure 3.16 (Here I've selected ORA's *Display> Grayscale* option). By default, ORA visualizes all of the networks (i.e., advice, friendship, reports to) in the Krackhardt dataset. If you want to visualize only one, as Figure 3.16 does, then uncheck the boxes of the networks you do not want to visualize.

[Visualizer] Display> Grayscale

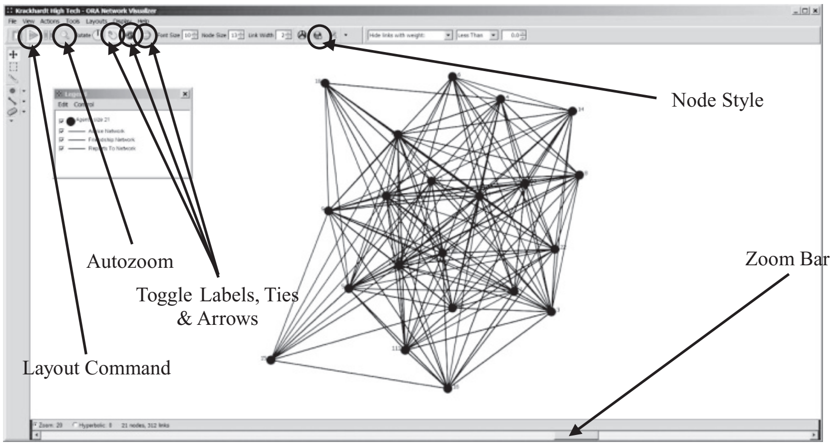
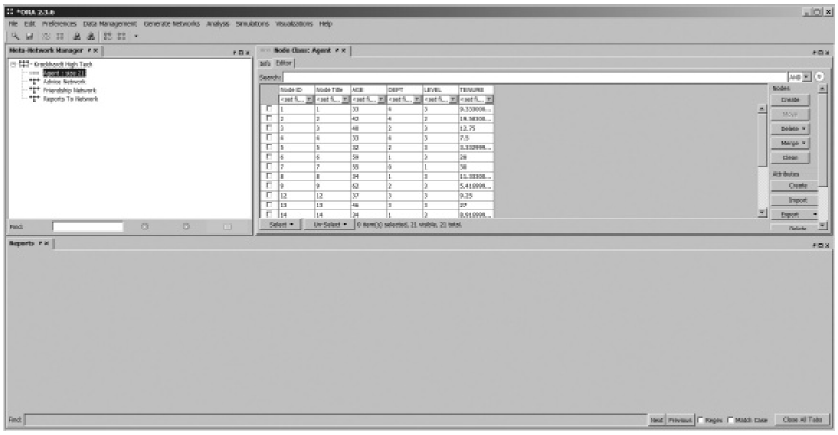


Figure 3.16. ORA Visualizer



	Node ID	Node Title	AGE	DEPT	LEVEL	TENURE
	<set fi...	<set fi...	<set fi...	<set fi...	<set fi...	<set fi...
☐	1	1	33	4	3	9.333008...
☐	2	2	42	4	2	19.58308...
☐	3	3	40	2	3	12.75
☐	4	4	33	4	3	7.5
☐	5	5	32	2	3	3.332998...
☐	6	6	59	1	3	28
☐	7	7	55	0	1	30
☐	8	8	34	1	3	11.33308...
☐	9	9	62	2	3	5.416998...
☐	12	12	37	3	3	9.25
☐	13	13	46	3	3	27
☐	14	14	34	1	3	8.916998...

Select ▾ Un-Select ▾ 0 item(s) selected, 21 visible, 21 total.

Figure 3.17. ORA Editor

ORA's visualizer includes a number of features that are useful in displaying and analyzing social networks. We will not explore all of these, but a few are worth mentioning now. Circled in the upper-left corner, is the layout command. ORA's default is a spring-embedded algorithm, but it can be changed to an alternative spring-embedded algorithm designed for large network data using the *Layouts>Spring Embedded (with enhancements for large data)* command. Another option is to use ORA's MDS routine, which is accessed with its *Layouts>Run MDS Layout* command. This calls up a dialog box (not shown) that asks you to indicate how many iterations you wish to run. A minimum of 100 generally produces good results. Note that there are a number of other visualization options, some of which you can try out on your own. Sometimes after visualizing a network, it becomes too large or too small for the screen. ORA's Autozoom feature, which is just to the right of the layout button (see Figure 3.16), both centers and fits the network in the visualizer. Working from left to right across the menu, you can see that ORA includes a number of additional features that allow you to rotate the network; toggle on and off labels, ties, and arrows; vary the size of fonts, nodes, and ties; change the node style; and hide ties of a particular strength. In addition, at the base of the visualizer, you can vary the size of the network by dragging the zoom bar to the left or right.

Layouts>Spring Embedded (with enhancements for large data)

Layouts>Run MDS Layout

Visualizing Attribute Data in ORA

To visualize attribute data in ORA, first return to ORA's main screen to see where they are stored. Select the node class icon in the Meta-Network Manager panel and click on the Editor tab in the Network Information/Editor panel (see Figure 3.17). ORA uses "node classes" to store characteristics (e.g., attributes) about nodes/actors. Note that in the Editor tab we can (among other things) create and delete nodes as well as create, import, export, and delete attributes. Because the attribute data are already loaded and stored within the Krackhardt data file, we do not need to explore these features here.

[Visualizer] Node Appearance>Node Color>Color Nodes by Attribute or Measure

Display>Grayscale

Display>Node Appearance>Size Nodes by Attribute or Measure

Return to ORA's visualizer. To change the color of the nodes based on the department attribute, under the *Display* menu, choose the *Node Appearance>Node Color>Color Nodes by Attribute or Measure* command. At the resulting dialog box (not shown), use the *Select an Attribute* drop-down menu and choose the "DEPT" attribute. Click "Apply Changes" and close the dialog box. I have chosen ORA's grayscale option, which you can select with the *Display>Grayscale* command. To change the size of the node, choose the *Display>Node Appearance>Size Nodes by Attribute or Measure* command. Select "LEVEL," click "Apply Changes," and close the dialog box. The nodes in your network map should now vary in color and size and look similar to Figure 3.18.

3.6 Summary and Conclusion

In this chapter we have briefly examined some of the basic features of UCINET, NetDraw, Pajek, and ORA. All four programs are widely used within the SNA community. All four have their strengths and weaknesses. Generally, analysts use UCINET for estimating metrics and manipulating data, NetDraw for the basic drawing of social networks, and Pajek for

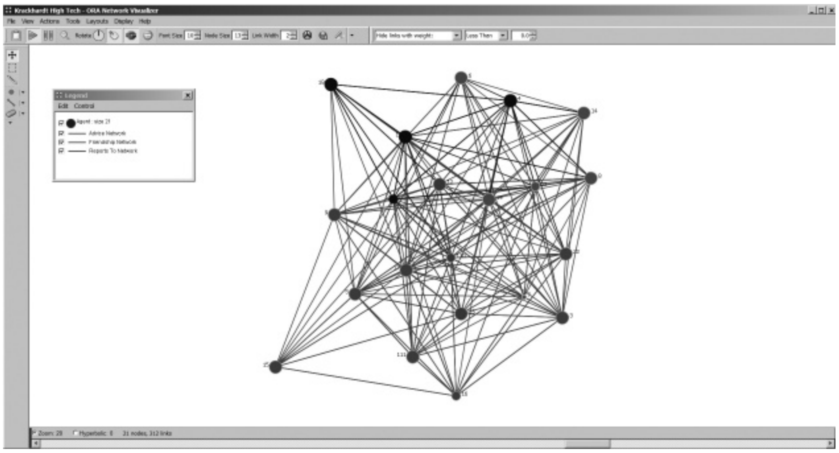


Figure 3.18. Krackhardt Advice Network with Varying Node Size and Color (ORA)

analyzing large networks and robust visualizations of social networks. ORA is the newcomer on the block and offers several useful features, such as the calculation of numerous metrics (some that are unique to ORA), the ability to handle relatively large datasets, and visualization features that lend themselves to presentations and analysis.