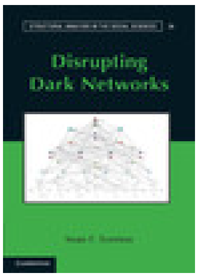


Disrupting Dark Networks

Sean F. Everton

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

Online ISBN: 9781139136877

Hardback ISBN: 9781107022591

Paperback ISBN: 9781107606685

Chapter

9 - Positions, Roles, and Blockmodels pp. 286-316

Chapter DOI: http://dx.doi.org/10.1017/CBO9781139136877.013

Cambridge University Press

9

Positions, Roles, and Blockmodels

9.1 Introduction

In Chapter 1, we noted that social network analysts tend to analyze network data in one of two ways: (1) a relational approach or (2) a positional approach (Emirbayer and Goodwin 1994). The former focuses on the direct and indirect ties between actors and seeks to explain behavior and social processes in light of those ties. It highlights the importance of the topography of networks, the centrality of actors, the cohesiveness of subgroups, and the brokers and bridges between such groups. Up to this point, we have essentially focused on the relational approach of social network analysis. In this chapter, we shift gears and explore the positional approach. It differs from the relational approach in that rather than focusing on the ties between actors, it seeks to identify actors who hold similar positions in the social structure. Why are positions seen as important? A position (e.g., student) is typically connected to a particular role or set of roles (e.g., attending class, writing papers, studying for exams) and located within a larger system of positions (e.g., fellow students, professors, staff, administrators). Consequently, some social network theorists argue that actors occupying a particular position/role will be embedded in a similar pattern of ties and exhibit similar types of behavior.

Actors who hold such equivalent positions are said to be structurally equivalent. A set of structurally equivalent actors is referred to as a "block," and the process by which blocks are identified is referred to as blockmodeling (White, Boorman, and Breiger 1976). Be aware that the terminology in this theoretical area can at times be conceptually confusing because the algorithms that analysts have developed to identify structurally equivalent actors vary in their assumptions and names. For example, structural, automorphic, and regular equivalence algorithms all seek to identify actors who occupy similar positions in the social structure, but they differ in their assumptions of what constitutes a set of equivalent actors and as such identify different sets or blocks (we will explore these differences in detail). Once the blocks of a network have been identified, this information can be used in different ways. For example, analysts can simplify (i.e., collapse) a network based on the sets to which each actor belongs (similar to how we simplified networks using role data in Chapter 4) to see if patterns emerge that were not immediately obvious when looking at all of the actors in the network.

The chapter now turns to an overview of structural, automorphic, and regular equivalence, using the Wasserman and Faust network (1994:468) as a working example. This chapter's final section focuses on the techniques of blockmodeling in UCINET, Pajek, and ORA. In this section we only demonstrate structural equivalence algorithms, not because they are necessarily "preferred," but because they are well established and implemented in all three software programs,[1] and how you blockmodel with them is similar to how you blockmodel with automorphic and regular equivalence algorithms.

9.2 Structural Equivalence

Two actors are said to be structurally equivalent when they have exactly the same relationships to all other actors. That is, they must "have identical ties to and from identical other actors" (Wasserman and Faust 1994:468). Another way to think about this is that in order to be structurally equivalent, two actors must be exactly substitutable for one another (Hanneman and Riddle 2005). Because structurally equivalent actors are tied to exactly the same other actors, they score identically in terms of centrality, prestige, and all other social network metrics (Borgatti and Everett 1992:7).[2]

Perhaps the easiest way to get a handle on the notion of structural equivalence is with the hypothetical network displayed in Figure 9.1 (from Wasserman and Faust 1994:468). Assume that the network displays who supervises whom in a company of managers and employees. In terms of this network, some people supervise others, some are supervised by others, and some are both supervisors and supervisees.

In this example only two sets of actors are considered to be structurally equivalent (5 and 6 and 8 and 9). This is because actors 5 and 6 both have

[1] Whereas UCINET includes a wide array of equivalence algorithms, Pajek includes algorithms for structural and regular equivalence, and ORA includes only one for structural equivalence (i.e., CONCOR).

[2] The reverse is not true, however. "Actors who are indistinguishable on absolutely all graph-theoretic attributes are not necessarily structurally equivalent" (Borgatti and Everett 1992:7).

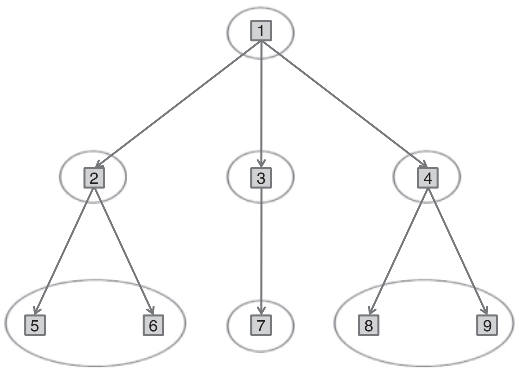


Figure 9.1. Structural Equivalence of Wasserman and Faust Network

ties to actor 2, and actors 8 and 9 both have ties to actor 4. Note that in both sets, the structurally equivalent actors do not have ties to one other. There is no tie between actors 5 and 6, nor is there one between actors 8 and 9. This is important to note because in order for two actors to be structurally equivalent, there does not have to be a tie between them. There can be, but it is not a requirement. What about the remaining actors in the network? They are considered to be in their own equivalence classes (or blocks) because there are no other actors that have exactly the same set of ties that they do. For example, because there is no actor that has exactly the same set of ties as actor 1, it is assigned to its own class, and the same is true of actors 2, 3, 4, and 7. Thus, in the example network there are seven sets of structurally equivalent actors (i.e., classes, blocks).

Structural equivalence is a very restrictive understanding of equivalence. Seldom do we analyze networks in which two or more actors have exactly the same relationships to all other actors; this fact has led some to question the usefulness of the concept. As Wasserman and Faust (1994:468–469) note:

> The fact that structurally equivalent actors must have *identical* ties to and from *identical* other actors is a severe limitation. [For example], two actors can be assigned to the same "manager" position only if they supervise exactly the same employees. Managers from two different companies, or even managers in charge of two different departments, cannot be structurally equivalent. The restriction of identical ties and identical actors, as required by structural equivalence, thus does not provide a general formalization of the theoretical notion of social position. (Borgatti and Everett 1992; Faust 1988)

What this often means in practice is that analysts use some sort of threshold measure to determine who belongs to what equivalence class and what ties exist between equivalence classes (we will see how to do this in Section 9.4).

Moreover, Borgatti and Everett (1992) have noted that structural equivalence is really another form of clustering based on cohesion (as opposed to one based on position), because in order for two actors to be structurally equivalent, they have to have the same ties with themselves, each other, and all other actors, which means that they are always completely contained within the same components of a network. "That is, actors located in different components of a disconnected graph (or in different graphs) can never be structurally equivalent (except isolates). In fact, as a general rule, nodes cannot be structurally equivalent if they are more than two links apart" (Borgatti and Everett 1992:9). Put simply, structural equivalence algorithms identify cohesive subsets rather than "positional" subsets. As a consequence, they and others have argued for more generalized or relaxed definitions of equivalence that they believe possess somewhat more realistic measures of what it means for two actors to be regarded as "equivalent" or "structurally similar" (see, e.g., Borgatti and Everett 1992; Faust 1988; Wasserman and Faust 1994). Automorphic equivalence is one such measure; regular equivalence is another. We consider the former before examining the latter.

9.3 Automorphic Equivalence

With automorphic equivalence, two actors are considered structurally similar if they occupy indistinguishable positions in a network. It is based on the notion of isomorphism, which in graph theory refers to a one-to-one mapping of one set of objects onto another, such that the relationships between the objects are preserved (Borgatti and Everett 1992:11). Isomorphic graphs are identical in terms of all graph theoretic attributes (e.g., density, centralization, number of cliques) and only differ (if at all) in terms of node and edge labels. Whereas the term isomorphism refers to the mapping of one graph onto another graph, automorphism refers to the mapping of a graph onto itself. Actors, then, are considered to be automorphically equivalent with one another if they can be mapped onto one another. For example, two professors would be regarded as automorphically equivalent if they taught the same number of students (although not necessarily the same students) and were supervised by the same number of people (although not necessarily the same people) in identical structural positions.

Consider, once again, the example from Wasserman and Faust (Figure 9.2). In terms of automorphic equivalence, there are five sets or classes of actors. The first thing to note is that actors that are structurally equivalent are also automorphically equivalent (and regularly equivalent).

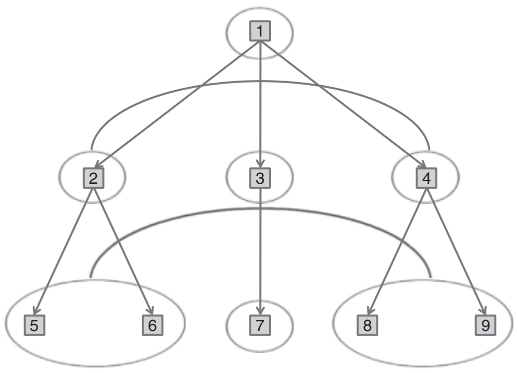


Figure 9.2. Automorphic Equivalence of Wasserman and Faust Network

Actors 2 and 4 are considered automorphically equivalent with one another because although they "supervise" different employees, they supervise the same number of employees. Similarly, actors 5, 6, 8, and 9 are automorphically equivalent because they are supervised by the same number of supervisors even though they are different supervisors. However, actors 2 and 4 are not automorphically equivalent to actor 3 because they "supervise" more employees than does actor 3.

In general then, automorphically equivalent actors have the same indegree and outdegree (if we are examining a directed graph), the same centrality on every possible measure, and so on. But, what about actor 7? Similar to actors 5, 6, 8, and 9, he or she is supervised by the same number of supervisors. The difference is that actors 5 and 6 are colleagues (where "colleague" is defined as someone supervised by the supervisor of oneself) as are actors 8 and 9. Actor 7, however, does not have a colleague, so he or she is sorted into a separate class, at least in terms of a strict or "exact" definition of automorphic equivalence.[3] However, if we are willing to relax the definition somewhat and move to an approximate understanding of automorphic equivalence, then actor 7 positionally looks a lot like actors 5, 6, 8, and 9 in that all five are supervised but they themselves supervise nobody.[4]

Although exact definitions of automorphic equivalence are more common in the literature, UCINET includes three automorphic equivalence algorithms, two of which identify exact automorphic equivalence classes and one that identifies approximate automorphic equivalence classes:

[3] Technically, this means that actor 7 has a different distribution of all types of triads than do actors 5, 6, 8, and 9.

[4] Thanks are due to Ronald Breiger, who helped immensely in delineating the difference between exact and approximate automorphic equivalence, as well as providing terminology (specifically, "colleagues") that should be intelligible to readers unfamiliar with graph theory.

(1) all permutations, (2) optimization, and (3) equivalence of distances. The first two are "exact" automorphic equivalent algorithms. Of these, the first (*Network>Roles & Positions>Automorphic>All Permutations*) compares every possible swapping of nodes (i.e., all permutations) in its search for automorphisms. When applied to the Wasserman and Faust network, it partitions the actors into same set of classes or blocks illustrated in Figure 9.2. Unfortunately, because it literally examines all possible permutations of a network, it is computationally intense and generally only useful with small networks (unless you are willing to wait a long time for the program to come back with an answer).

[UCINET] Network >Roles & Positions >Automorphic >All Permutations

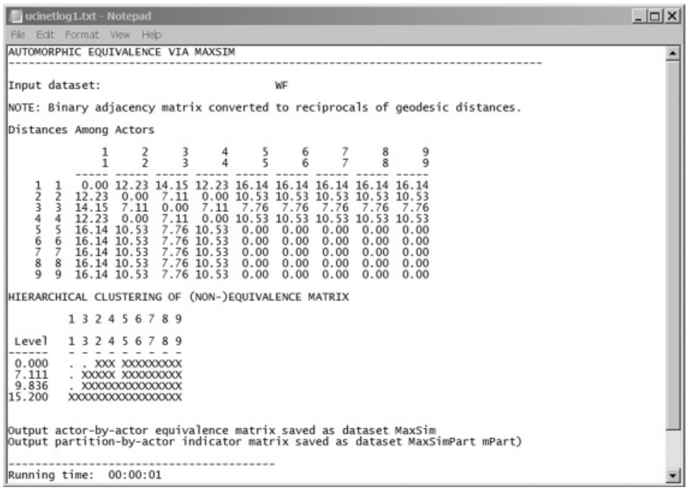
Thus, with large networks you are often better off using the optimization approach (*Network>Roles & Positions>Exact>Optimization*), which is more efficient. It is also similar to factional analysis, which we explored in Chapter 6.[5] Like factional analysis, with the optimization approach for finding automorphically equivalent actors, analysts indicate the number of blocks or classes into which they want to partition the network. The algorithm then begins by randomly sorting actors into blocks and calculating an initial badness-of-fit measure; it continues to sort actors into blocks until it finds a partition that minimizes the badness of fit score. Like factional analysis, analysts will generally want to examine a range of possible numbers of blocks (unless one has a prior theory about this), to determine what number of blocks provides the best measure of fit. When asked to identify five blocks in the Wasserman and Faust network, this algorithm returns a badness of fit score of 0.00. In other words, there is a perfect fit.

Network >Roles & Positions >Exact >Optimization

The final algorithm (*Network>Roles & Positions>Automorphic>MaxSim*), unlike the first two, searches for "approximate" automorphic equivalence classes. Using the Euclidean distance between actors, it estimates the degree of automorphic equivalence for each pair of actors. As noted before, Euclidean distance differs from distance in graph theory. In the latter, the distance between two points is measured in terms of the number of lines in the path that connects the two points (i.e., path distance), whereas in the former the distance between two points is the most direct route between them (Scott 2000). The algorithm begins with a (reciprocal of) distance (for binary data) or strength of tie matrix (for valued data), calculates the distance between all pairs of actors, and regards actors that have similar distance profiles as being more automorphically equivalent. For example, Figure 9.3 depicts the output from an "equivalence of distances" (i.e., MaxSim) analysis of the Wasserman and Faust data. A distance matrix appears at the top of the output, whereas a hierarchical clustering graph appears at the bottom. The distance matrix

Network >Roles & Positions >Automorphic >MaxSim

[5] Recall also that we noted that it is a "special" type of blockmodeling. Structural and regular equivalence optimization algorithms are also available and implemented in UCINET and Pajek.



```
AUTOMORPHIC EQUIVALENCE VIA MAXSIM
--------------------------------------------------------------------------------

Input dataset:                        WF

NOTE: Binary adjacency matrix converted to reciprocals of geodesic distances.

Distances Among Actors

                 1     2     3     4     5     6     7     8     9
                 1     2     3     4     5     6     7     8     9
             ----- ----- ----- ----- ----- ----- ----- ----- -----
    1  1      0.00 12.23 14.15 12.23 16.14 16.14 16.14 16.14 16.14
    2  2     12.23  0.00  7.11  0.00 10.53 10.53 10.53 10.53 10.53
    3  3     14.15  7.11  0.00  7.11  7.76  7.76  7.76  7.76  7.76
    4  4     12.23  0.00  7.11  0.00 10.53 10.53 10.53 10.53 10.53
    5  5     16.14 10.53  7.76 10.53  0.00  0.00  0.00  0.00  0.00
    6  6     16.14 10.53  7.76 10.53  0.00  0.00  0.00  0.00  0.00
    7  7     16.14 10.53  7.76 10.53  0.00  0.00  0.00  0.00  0.00
    8  8     16.14 10.53  7.76 10.53  0.00  0.00  0.00  0.00  0.00
    9  9     16.14 10.53  7.76 10.53  0.00  0.00  0.00  0.00  0.00

HIERARCHICAL CLUSTERING OF (NON-)EQUIVALENCE MATRIX

          1 3 2 4 5 6 7 8 9

 Level    1 3 2 4 5 6 7 8 9
------    - - - - - - - - -
 0.000    . . XXX XXXXXXXXX
 7.111    . XXXXX XXXXXXXXX
 9.836    . XXXXXXXXXXXXXXX
15.200    XXXXXXXXXXXXXXXXX


Output actor-by-actor equivalence matrix saved as dataset MaxSim
Output partition-by-actor indicator matrix saved as dataset MaxSimPart mPart)

----------------------------------------
Running time:  00:00:01
```

Figure 9.3. UCINET MaxSim Output of Wasserman and Faust Network

indicates that the distance between actors 2 and 4 is 0.00, as are the distances between actors 5 through 9. Thus, at a distance level of 0.00 (see the hierarchical clustering matrix in Figure 9.3), actors 2 and 4 are clustered into one equivalence class, actors 5 through 9 are another, and actors 1 and 3 are classes unto themselves.

Actors from different networks could be regarded as automorphically equivalent, although this is technically incorrect because automorphic equivalence is a measure of equivalence *within* a network. When actors from two different networks are regarded as equivalent because they occupy indistinguishable positions (e.g., they supervise and are supervised by the same number of people), this is called isomorphic equivalence. For example, if you regard the network of relations in UCLA's sociology department as distinct from Vanderbilt's, then you would say professors in the two different schools were isomorphically equivalent: They supervised the same number of students, were supervised by the same number of people (e.g., chairs of the department), and (if you are interested in exact automorphic equivalence) have the same number of colleagues. UCINET's automorphic algorithms identify isomorphically equivalent actors as well.

9.4 Regular Equivalence

With regular equivalence, actors do not need to have identical ties to identical other actors nor do they occupy indistinguishable positions in a network. Instead, they must have identical ties to and from regularly

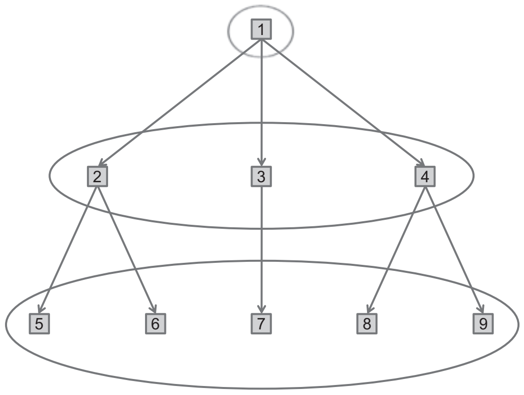


Figure 9.4. Regular Equivalence of Wasserman and Faust Network

equivalent actors (Borgatti and Everett 1988; White and Reitz 1983). Put differently, regularly equivalent actors "do not have to be connected to the same [actors], but they have to be connected to [actors] in the same classes" (de Nooy et al. 2005:280).

> For example, neighborhood bullies occupy the same social position, though in different neighborhoods, because they beat up some kid(s) and are scolded by some irate parent(s), but they do not necessarily beat up the same kid(s) nor are they scolded by the same parent(s). (Wasserman and Faust 1994:474)

In the Wasserman and Faust example (Figure 9.4), there are three classes of regularly equivalent actors: (1) actor 1; (2) actors 2, 3, and 4; and (3) actors 5, 6, 7, 8, and 9.

Actor 1 is in its own class because it is the only actor that has a tie to at least one actor in the second class (all three in this case) and no tie to any actor in the third class. Actors 2 through 4 are regularly equivalent with one another because they each have a tie with at least one member of the first class and one member of the third class. Note that even though actors 2 and 4 have more ties to members of the third class than does actor 3, this does not matter in this case because all that is required is that they have at least one tie to at least one member of the third class. Finally, actors 5 through 9 are regularly equivalent with one another because they have no ties with any actor in the first class and each has a tie with at least one actor in the second class.

Unfortunately, in a given network there can be numerous partitions of actors into classes in order to satisfy the regular equivalence definition. "For example, the partition of actors into structural equivalence classes is a regular equivalence (structurally equivalent actors are also regularly equivalent), and the partition of actors into automorphic (or isomorphic) equivalence classes is also a regular equivalence. But, there may be other regular equivalence partitions in a given network that are neither structural equivalences nor automorphic equivalences" (Wasserman and Faust 1994:475). Consequently, social network analysts typically seek to identify *maximal regular equivalence*, which is the partition of actors that assigns actors into the *fewest number* of equivalence classes. The original regular equivalence algorithm is REGE (accessed by the command *Network>Roles & Positions>Regular>REGE*), which was developed by Douglas White and Karl Reitz (White 1985; White and Reitz 1983). It works best with valued continuous data. For binary and nominal data, the Categorical REGE algorithm (*Network>Roles & Positions>Regular>CATREGE*) is recommended (Borgatti and Everett 1989, 1993). UCINET (and Pajek) also includes a regular equivalence optimization algorithm (*Network>Roles & Positions>Regular>Optimization*) similar to the automorphic optimization algorithm. As before, analysts indicate the number of blocks or classes into which they want the algorithm to partition the network. It then randomly sorts actors into blocks, calculates an initial badness-of-fit measure, and continues to sort until it finds a partition that minimizes the badness-of-fit score. Here again, analysts will generally want to explore a range of possible numbers of blocks in order to determine what number of blocks provides the best measure of fit. Analysts should also be aware that there may be more than one solution; that is, there may be multiple partitions with the same measure of fit. Unfortunately, UCINET only reports one. This is one advantage of Pajek's implementation of regular equivalence optimization algorithm (*Operations>Blockmodeling>Random Start*); it returns all possible solutions, allowing users to visually inspect the various partitions.

[UCINET] Network >Roles & Positions >Regular >REGE

Network>Roles & Positions >Regular >CATREGE

Network >Roles & Positions >Regular >Optimization

[Pajek] Operations >Blockmodeling >Random Start

Another issue to be aware of is that when analyzing symmetric (i.e., nondirectional) networks without isolates, the maximal regular equivalence is a single class that includes all actors (Borgatti 1988; Doreian 1987; Wasserman and Faust 1994). That is because in symmetric networks with no isolates all actors are tied to some other actor that is also in the equivalence class. This has led analysts to think in terms of "neighborhoods" within networks and developing methods (e.g., ones that take into account path distance) for identifying meaningful classes of regularly equivalent actors (Everett, Boyd, and Borgatti 1990). The categorical REGE algorithm can be used here, as can the "exact" categorical algorithm (*Network>Roles & Positions>Exact>ExCatReg*), but in both cases the "Convert data to geodesic distances" option must be selected.

Network >Roles & Positions >Exact >ExCatReg

9.5 Blockmodeling in UCINET, Pajek, and ORA

We now turn to an examination of blockmodeling techniques in UCINET, Pajek, and ORA using structural equivalence algorithms. As noted previously, we focus on structural equivalence algorithms, not because they are necessarily "preferred," but because they are well established, implemented in all three software programs, and similar to how automorphic and regular equivalence algorithms are used to blockmodel. Recall that two actors are considered structurally equivalent if they have identical ties with each other and all other actors in the network. Of course, seldom do two actors exhibit the exact pattern of ties, so we generally use a similarity threshold to identify which actors to consider structurally equivalent. Although on the surface this clustering technique may seem straightforward, there are numerous blockmodeling algorithms for identifying structurally equivalent actors. Some are computationally intense, which limits their use to relatively small datasets.

Roles, Positions, and Structural Equivalence in UCINET

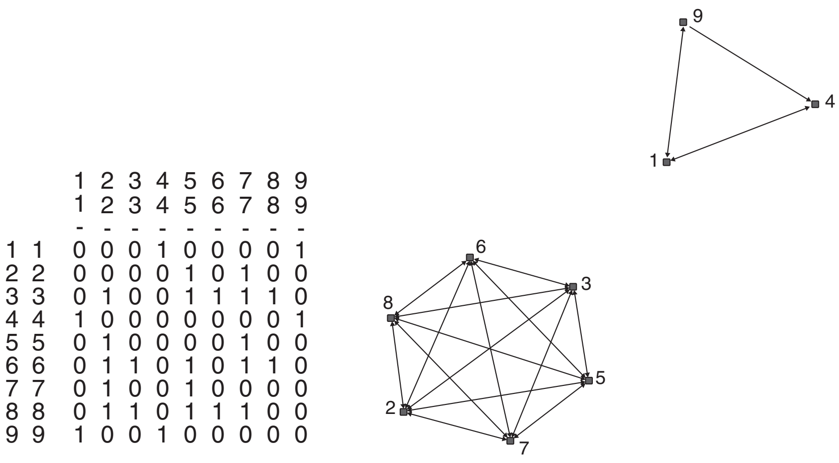
[UCINET] Data>Display

For most of this chapter we will use the Noordin Top trust network, but we will begin by examining a relatively small network (also from Wasserman and Faust 1994:364) because it is somewhat easier to illustrate certain blockmodeling features with smaller datasets than with larger ones. In UCINET, locate and display the Wasserman and Faust structural equivalence network (`WFSE.##h`). It should look similar to Figure 9.5 (a network graph of the dataset appears to the right of the matrix).

Network >Roles & Positions >Structural >Profile

Euclidean distance is a common approach for identifying structurally equivalent actors. Euclidean distance algorithms take network data and calculate a set of points in *n*-dimensional space, such that the distances between them correspond as closely as possible to the input proximities (Scott 2000:157). UCINET's Euclidean distance structural equivalence algorithm can be accessed with the *Network>Roles & Positions>Structural>Profile* command, which brings up a dialog box similar to the one in Figure 9.6. Accept most of UCINET's default settings, including Euclidean distance default[6] – the one exception being that the "For binary data . . . " option should be set to "Yes," not "No." Also, be sure that the Wasserman and Faust file is indicated as the input dataset (`WFSE.##h`). Click "OK." UCINET will generate an output report and

[6] The *Measure of profile similarity/distance* drop-down menu offers an array of methods for estimating structural equivalence, and if you click on UCINET's "Help" button, you will find a brief description of the various methods.



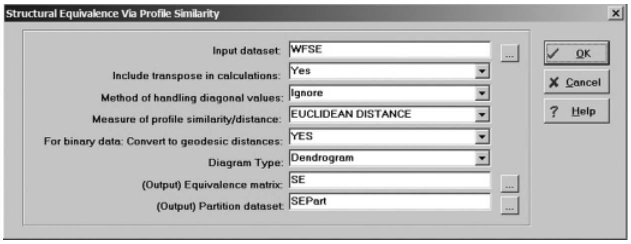
```
        1 2 3 4 5 6 7 8 9
        1 2 3 4 5 6 7 8 9
        - - - - - - - - -
1 1     0 0 0 1 0 0 0 0 1
2 2     0 0 0 0 1 0 1 0 0
3 3     0 1 0 0 1 1 1 1 0
4 4     1 0 0 0 0 0 0 0 1
5 5     0 1 0 0 0 0 1 0 0
6 6     0 1 1 0 1 0 1 1 0
7 7     0 1 0 0 1 0 0 0 0
8 8     0 1 1 0 1 1 1 0 0
9 9     1 0 0 1 0 0 0 0 0
```

Figure 9.5. Wasserman and Faust (1994) Structural Equivalence Network

two new files: a structural equivalence matrix `(SE.##h)` of Euclidean distances between actors and a structural equivalence partition `(SEPart.##h)` that assigns each actor to its respective equivalence class.

Data>Display Figure 9.7 displays the structural equivalence matrix `(SE.##h)` of Euclidean distances between actors (it is also displayed in the report generated by UCINET). The smaller a number in a particular cell, the more similar those two actors are. A score of 0.00 indicates perfect similarity, which means that the two actors share an identical pattern of ties. Looking at the matrix, you can see that actor 2 is perfectly similar to actors 5 and 7 (0.00) and the same distance away from actors 3 and 8 (16.00) and actors 1, 4, and 9 (26.533).

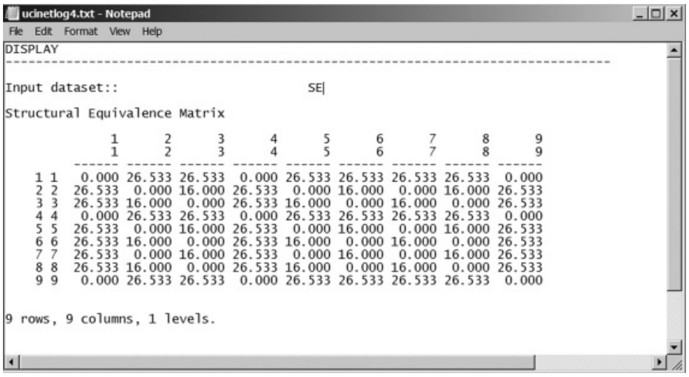
The next step in the blockmodel process is to permute (i.e., reorder the rows and columns) and partition the original matrix so that actors who are assigned to the same block occupy adjacent rows and columns. To do



Structural Equivalence Via Profile Similarity

Input dataset:	WFSE
Include transpose in calculations:	Yes
Method of handling diagonal values:	Ignore
Measure of profile similarity/distance:	EUCLIDEAN DISTANCE
For binary data: Convert to geodesic distances:	YES
Diagram Type:	Dendrogram
(Output) Equivalence matrix:	SE
(Output) Partition dataset:	SEPart

Figure 9.6. UCINET Structural Equivalence Profile Similarity Dialog Box

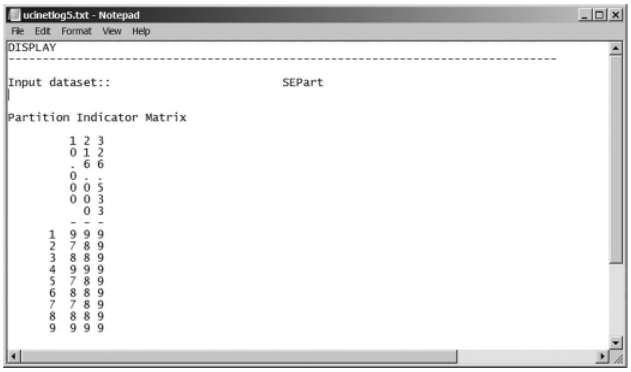


```
DISPLAY
---------------------------------------------------------------------------------

Input dataset::                         SE

Structural Equivalence Matrix

               1      2      3      4      5      6      7      8      9
               1      2      3      4      5      6      7      8      9
          ------ ------ ------ ------ ------ ------ ------ ------ ------
     1 1   0.000 26.533 26.533  0.000 26.533 26.533 26.533 26.533  0.000
     2 2  26.533  0.000 16.000 26.533  0.000 16.000  0.000 16.000 26.533
     3 3  26.533 16.000  0.000 26.533 16.000  0.000 16.000  0.000 26.533
     4 4   0.000 26.533 26.533  0.000 26.533 26.533 26.533 26.533  0.000
     5 5  26.533  0.000 16.000 26.533  0.000 16.000  0.000 16.000 26.533
     6 6  26.533 16.000  0.000 26.533 16.000  0.000 16.000  0.000 26.533
     7 7  26.533  0.000 16.000 26.533  0.000 16.000  0.000 16.000 26.533
     8 8  26.533 16.000  0.000 26.533 16.000  0.000 16.000  0.000 26.533
     9 9   0.000 26.533 26.533  0.000 26.533 26.533 26.533 26.533  0.000

9 rows, 9 columns, 1 levels.
```

Figure 9.7. UCINET Structural Equivalence Matrix

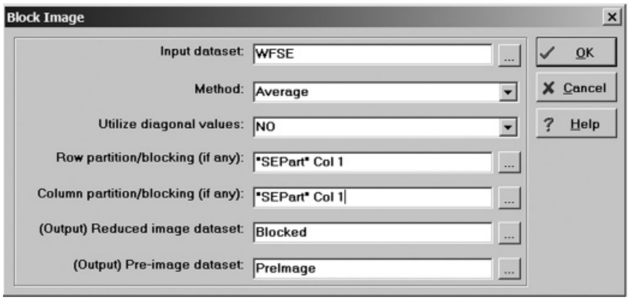
this we use both the original matrix and the structural equivalence partition file (`SEPart.##h`) generated previously. Let's begin by looking at the partition file using UCINET's display command. It should look similar to the Figure 9.8. Note that the file actually includes three partitions: one at similarity level 0.00 (first column; the highest level of similarity), one at 16.00 (second column), and one at 26.53 (third column; the lowest level of similarity). If we wanted to partition the network in structural equivalence classes at the highest level of similarity between actors, then we would want to use the first partition, and if we wanted to partition the network at the lowest level of similarity, then we would use the third partition.

Note that at the lowest level of similarity, all the actors are assigned to the same equivalence class, so it is of little practical use. If we use the second partition, the actors will be assigned to two classes; if we use the first, they will be assigned to three. Here, we will use the first. To do this we use the *Transform>Block* command. In the resulting dialog box (see *Transform>Block*



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

Input dataset::                         SEPart

Partition Indicator Matrix

           1 2 3
           0 1 2
           . 6 6
           0 . .
           0 0 5
           0 0 3
             0 3
           - - -
       1   9 9 9
       2   7 8 9
       3   8 8 9
       4   9 9 9
       5   7 8 9
       6   8 8 9
       7   7 8 9
       8   8 8 9
       9   9 9 9
```

Figure 9.8. UCINET Structural Equivalence Partition

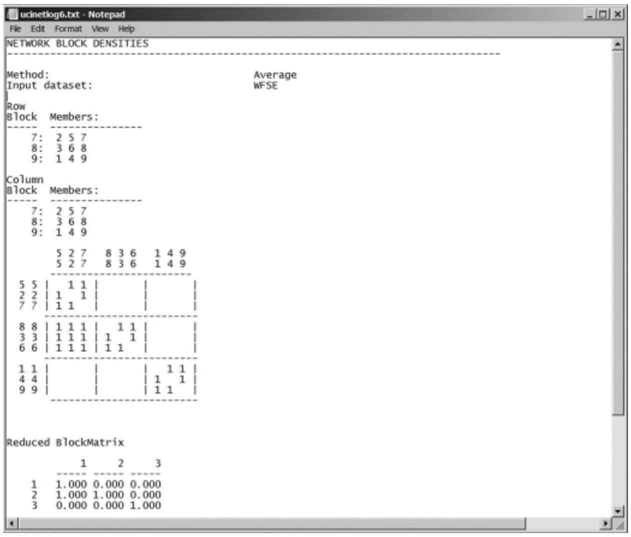


Input dataset: WFSE
Method: Average
Utilize diagonal values: NO
Row partition/blocking (if any): "SEPart" Col 1
Column partition/blocking (if any): "SEPart" Col 1
(Output) Reduced image dataset: Blocked
(Output) Pre-image dataset: PreImage

Figure 9.9. UCINET Block Image Dialog Box

Figure 9.9) you need to indicate the input dataset and the row/column partition file. As you can see, I instructed UCINET to use the first partition (column 1) of the "SEPart.##h" partition file.

The resulting output file (see Figure 9.10) provides three pieces of information. First, it tells you to which blocks the actors have been assigned. Next, it provides a permuted and partitioned matrix in which actors in the same equivalence class are grouped together and partitioned from the other classes. As you can see from the figure, actors 5, 2, and 7 have been assigned to the first block or class; 8, 3, and 6 have been assigned to the second block; and 1, 4, and 9 have been assigned to the third block. You can also see that actors within in the first and third blocks send ties only to other actors within their own block, whereas the actors within the second block send ties to each other and members of the first block.



```
NETWORK BLOCK DENSITIES
------------------------------------------------------------------------------

Method:                                           Average
Input dataset:                                    WFSE

Row
Block   Members:
-----   ---------------
    7:  2 5 7
    8:  3 6 8
    9:  1 4 9

Column
Block   Members:
-----   ---------------
    7:  2 5 7
    8:  3 6 8
    9:  1 4 9

        5 2 7   8 3 6   1 4 9
        5 2 7   8 3 6   1 4 9
       -----------------------
   5 5 |   1 1 |       |       |
   2 2 | 1   1 |       |       |
   7 7 | 1 1   |       |       |
      -------------------------
   8 8 | 1 1 1 |   1 1 |       |
   3 3 | 1 1 1 | 1   1 |       |
   6 6 | 1 1 1 | 1 1   |       |
      -------------------------
   1 1 |       |       |   1 1 |
   4 4 |       |       | 1   1 |
   9 9 |       |       | 1 1   |
       ------------------------


Reduced BlockMatrix

            1     2     3
        ----- ----- -----
    1   1.000 0.000 0.000
    2   1.000 1.000 0.000
    3   0.000 0.000 1.000
```

Figure 9.10. UCINET Permuted and Partitioned Matrix

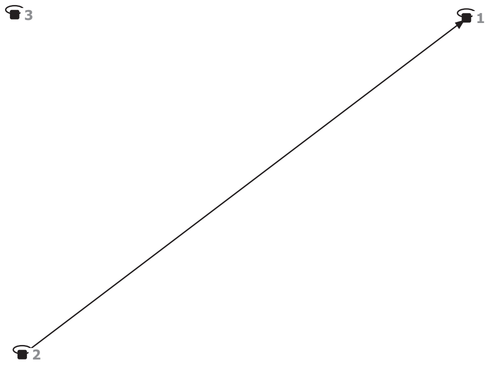


Figure 9.11. Image Matrix Graph (NetDraw)

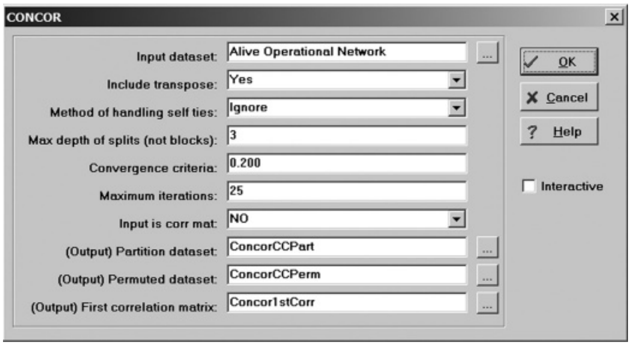
This pattern of where blocks send ties can be captured in two ways: with an image matrix and a graphical representation of the image matrix. The image matrix itself is the last piece of information that the output file provides. The image matrix collapses each block/class of the permuted matrix into a single cell where the number appearing in the cell indicates the density of ties between the actors of that block. In this case in each of the blocks all possible ties are present, so the density within each block is 1.00. Because all of the actors in the second block (i.e., 8, 3, and 6) have ties to all of the actors in the first block (i.e., 5, 2, and 7), the density of that cell is 1.00 as well. However, because ties are completely absent between the other blocks, their density is 0.00.

The image matrix's corresponding graph (Figure 9.11) captures the relationship between blocks. Although it is quite small here, if you look closely you can see that block 2 is the only block that sends ties to any other block (block 1), whereas all three blocks have reflexive ties to themselves (i.e., block members send ties to other block members).

As suggested previously, the density within and between blocks seldom exactly equals either 1.00 or 0.00 (in other words, actors are seldom perfectly similar or dissimilar), so we generally have to choose some sort of criterion that distinguishes "one-blocks" (sometimes referred to as complete blocks) from "zero-blocks" (aka, null blocks). That is one of the topics we will take up in the next section, in which we examine the Noordin Top network.

Roles, Positions, and Structural Equivalence in UCINET (Noordin)

CONCOR. Here we will use Noordin's alive operational network (`Alive Operational Network.##h`) with the idea that if we can identify actors who are substitutable for one another, we may be able to


CONCOR

Input dataset: Alive Operational Network
Include transpose: Yes
Method of handling self ties: Ignore
Max depth of splits (not blocks): 3
Convergence criteria: 0.200
Maximum iterations: 25
Input is corr mat: NO
(Output) Partition dataset: ConcorCCPart
(Output) Permuted dataset: ConcorCCPerm
(Output) First correlation matrix: Concor1stCorr

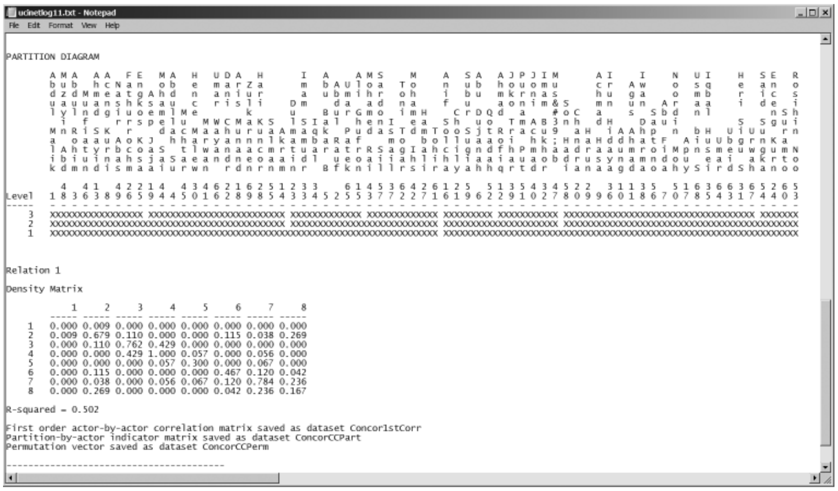
Figure 9.12. UCINET CONCOR Dialog Box

"predict" who might take over for another if the latter is removed from the network. We could examine the network using the Euclidean algorithm that we used previously. However, because the process would be the same, we will instead examine another algorithm: CONCOR, which was one of the earliest approaches to identifying structurally equivalent actors. CONCOR, which stands for "CONvergence of iterated CORrelations," is based on the discovery that repeated calculation of correlations between a matrix's rows (or columns) eventually results in correlation matrix consisting of only +1.00's and −1.00's (Wasserman and Faust 1994:376).[7] It begins by correlating each pair of actors, and then each row of the resulting actor-by-actor correlation matrix is extracted and correlated with each other row. This process is repeated over and over until all of the coefficients approach either +1.00 or −1.00. CONCOR then splits the data into two sets (i.e., the +1.00 set and the −1.00 set). The process is then repeated for each of the sets (or at least those sets with two or more actors) and continues until it runs out of actors to split or arrives at the number of "splits" indicated by analysts at the outset.

The CONCOR command in UCINET is located under the *Network* menu. UCINET now includes two CONCOR algorithms: (1) the standard algorithm where we simply indicate the number of splits we want CONCOR to estimate and (2) an interactive one, which gives us a bit more control over how many and where we want the splits to occur. We will begin with the standard algorithm, which brings up a dialog box that should look similar to Figure 9.12. Note that I have indicated that I want CONCOR to split the data three times, which should give us eight blocks

Network>Roles & Position >Structural >CONCOR >Standard

[7] Recall that correlation coefficients range from −1.00 to 1.00, where −1.00 indicates perfect negative correlation and 1.00 indicates perfect positive correlation. If two actors share an identical pattern of ties (i.e., they are tied to the same actors), their correlation will be 1.00; if two actors have exactly opposite ties to other actors, their correlation will be −1.00; and if two actors ties indicate neither a positive nor a negative association, their correlation will be 0.00.



```
PARTITION DIAGRAM

        A M A   A A   F E   M A   H   U D A   H       I   A     A M S     M     A   S A   A J P J I M       A I     I     N   U I     H   S E   R
        b u b   h c N a n   o b   e   m a r Z a       m   b A U l o a   T o     n   u b   h o u o m u       c r   A w     o   s q     e   a n   o
        d z d M m e a t g A h d   n   a n i u r       a   u b m i h r   o h     i   b u   m k r n a s       h u   g a     o   m b     r   i c   s
        u a u u a n s h k s a u   c   r i s l i     D m     d a   a d   n a     f   u     a o n i m & S     m n   u n   A r   a a     i   d e   i
        l y l n d g i u o e m l M e       k       u     B u r G m o   i m H     C r D Q d   a     # o C   a     s   S b d   n l         n S h
          i   f     r r s p e l u   M W C M a K S   l S I a l   h e n I   e a   S h   u o   T m A B 3 n h   d H     D a u i         S   S g u i
        M n R i S K   r     d a c M a a h u r u a A m a q k   P u d a s T d m T o o S j t R r a c u 9   a H   i A A h p   n   b H   U i U u   r n
        a   o a a u A o K J   h h a r y a n n n l k a m b a R a f     m o   b o l l u a a o i   h k ; H n a H d d h a t F   A i u U b g r n K a
        l A h t y r b c o a S   t l w a n a a c m r t u a r a t r R S a g I a h c i g n d f h P m h a a d r a a u m r o i M p n s m e u w g u m N
        i b i u i n a h s j a S a e a n d n e o a a i d l   u e o a i i a h l i h l i a a i a u a o b d r u s y n a m n d o u   e a i   a k r t o
        k d m n d i s m a a i u r w n   r d n r n m n r   B f k n i l l r s i r a y a h h q r t d r   i a n a a g d a o a h y S i r d S h a n o o

          4   4 1   4 2 2 1 4   4 3 4 6 2 1 6 2 5 1 2 3 3     6 1 4 5 3 6 4 2 6 1 2 5   5 1 3 5 4 3 4 5 2 2   3 1 1 3 5   5 1 6 3 6 6 3 6 5 2 6 5
Level   1 8 3 6 3 8 9 6 5 9 4 4 5 0 1 6 2 8 9 8 5 4 3 3 4 5 2 5 5 3 7 7 2 2 7 1 6 1 9 6 2 2 9 1 0 2 7 8 0 9 9 6 0 1 8 6 7 0 7 8 5 4 3 1 7 4 4 0 3
-----   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    3   XXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXX XXXXXXXXXXXXX XXXXXXXXX XXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX
    2   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


Relation 1

Density Matrix

            1     2     3     4     5     6     7     8
        ----- ----- ----- ----- ----- ----- ----- -----
    1   0.000 0.009 0.000 0.000 0.000 0.000 0.000 0.000
    2   0.009 0.679 0.110 0.000 0.000 0.115 0.038 0.269
    3   0.000 0.110 0.762 0.429 0.000 0.000 0.000 0.000
    4   0.000 0.000 0.429 1.000 0.057 0.000 0.056 0.000
    5   0.000 0.000 0.000 0.057 0.300 0.000 0.067 0.000
    6   0.000 0.115 0.000 0.000 0.000 0.467 0.120 0.042
    7   0.000 0.038 0.000 0.056 0.067 0.120 0.784 0.236
    8   0.000 0.269 0.000 0.000 0.000 0.042 0.236 0.167

R-squared = 0.502

First order actor-by-actor correlation matrix saved as dataset Concor1stCorr
Partition-by-actor indicator matrix saved as dataset ConcorCCPart
Permutation vector saved as dataset ConcorCCPerm

----------------------------------------
```

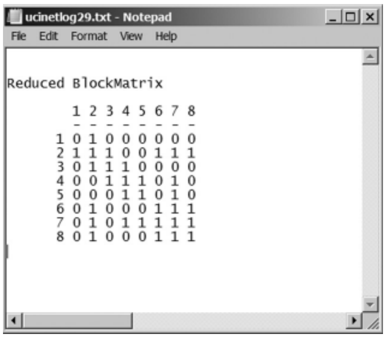
Figure 9.13. UCINET CONCOR Density Matrix

(i.e., 2^3) of structurally equivalent actors (unless after the first or second split, we have a block consisting of only one actor).

UCINET generates a dialog box (not shown – it may be hidden behind the output file) asking you if you want to see a dendogram, which graphically illustrates the splits in the data that have occurred. Click "OK" in order to see this. The output file consists of three panels: the first-order correlation matrix, a partition diagram, and a density matrix. Figure 9.13 displays the last two panels.

The partition diagram indicates where the splits in the data occurred. The bottom row of "x's" (Level 1) indicates the two groups that the first split identified, the second row from the bottom (Level 2) indicates the four groups that the second split identified, and the third row (Level 3) indicates the eight groups that the third split identified. Listed above the rows of x's are the names of the actors, indicating the block to which they have been assigned. As with the profile similarity approach, the density (image) matrix found at the bottom of the output collapses each block/class of the permuted matrix into a single cell where the number appearing in the cell indicates the density of ties between the actors of that block. Note that only one of the densities equals 1.00, so it will not be as straightforward to create an image matrix as it was previously. We take up that question in the next step. Note that UCINET provides a measure of fit (R^2) that compares the partitioned data matrix with an ideal structure matrix (i.e., where cell densities equal block means).[8] In

[8] R^2 (aka, the coefficient of determination) estimates the extent to which a model's variables account for the variance in the dependent variable. See Chapter 11.



```
Reduced BlockMatrix

       1 2 3 4 5 6 7 8
       - - - - - - - -
     1 0 1 0 0 0 0 0 0
     2 1 1 1 0 0 1 1 1
     3 0 1 1 1 0 0 0 0
     4 0 0 1 1 1 0 1 0
     5 0 0 0 1 1 0 1 0
     6 0 1 0 0 0 1 1 1
     7 0 1 0 1 1 1 1 1
     8 0 1 0 0 0 1 1 1
```

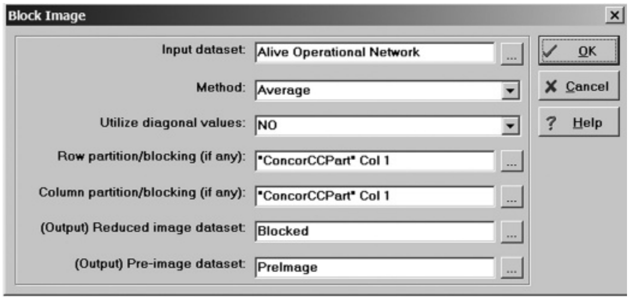
Figure 9.14. Final Image Matrix (Blockmodel), Zero-Block Method (UCINET)

this case, the higher the R^2, the better. UCINET also generates a partition matrix/file that we will use to generate the blockmodel.

How do we identify which blocks to classify as one/complete blocks or zero/null blocks? There are a number of different approaches. The zero-block, or lean fit, approach (Wasserman and Faust 1994:399) only classifies a block as a null block if its density equals 0.00 (i.e., if there is a complete absence of ties) and then classifies the remaining blocks as one (or complete) blocks. If we were to do this with the current network, the final image matrix (i.e., blockmodel) would look like Figure 9.14. What this tells us is, that although most of the blocks share ties with other blocks, the first block does not. It has ties only to the second block. Thus, the gaps between it and all of the other blocks may indicate "holes" in the structure of network that could be exploited. Noordin himself is located in the seventh block, which as we can see from the density matrix in Figure 9.13, is well connected (i.e., its density equals 0.784) and may suggest that if Noordin were to be removed from the network and the network were to reconstitute itself, its future leader would come from this block.

An alternative approach, the one-block approach (Wasserman and Faust 1994:400), only classifies a block as a complete block if a block's density equals 1.00. In the case of the alive operational network, the one-block approach produces a final image matrix (not shown) full of zeros with only one "one-block" (block 4), which in this case (and in most cases) is unhelpful.

Probably the most common approach for distinguishing complete from null blocks is to set a density threshold such that if a particular block's density is greater than or equal to that threshold, it is classified as a complete block, and if it is not, it is classified as a null block (Wasserman and Faust 1994:400–401). The threshold is often set at the density of the overall network (although other densities can be used), which, if you recall


Block Image

Input dataset: Alive Operational Network

Method: Average

Utilize diagonal values: NO

Row partition/blocking (if any): *ConcorCCPart* Col 1

Column partition/blocking (if any): *ConcorCCPart* Col 1

(Output) Reduced image dataset: Blocked

(Output) Pre-image dataset: PreImage

Figure 9.15. UCINET Block Image Dialog Box

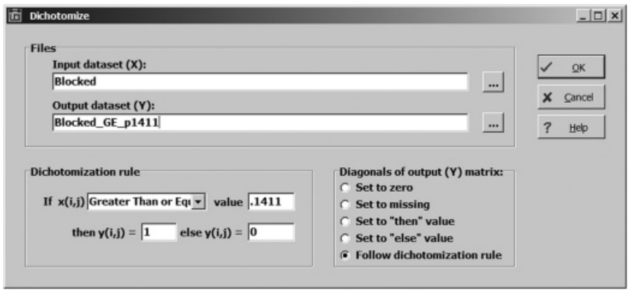
from Chapter 5, can be obtained with the *Network>Cohesion> Density>(new) Density Overall* command. The density of the network is 0.1411, so to create a final image matrix using this as a threshold, we turn (once again) to the *Transform>Block* function (Figure 9.15) to first generate an image matrix. As before, we indicate the input dataset and the partition file (here we use column 1 – you may want to inspect the partition file to see why this is so) and click "OK." Once again UCINET's output (not shown) provides a summary of the blocks to which each actor is assigned, the permuted network, and a reduced block matrix (which should look identical to the density matrix in Figure 9.13).

Network >Cohesion> Density>(new) Density Overall

The final step in the process involves dichotomizing the network so that cells with a density of 0.1411 or greater are classified as complete blocks, whereas the rest are classified as zero blocks. We do this with UCINET's *Transform>Dichotomize* command, which brings up the following dialog box (Figure 9.16).

Transform >Dichotomize

Note that I have set the cutoff operator to be "Greater Than or Equal" and the cutoff value at 0.1411. It is also important to indicate that you want the diagonals of the output matrix to follow the dichotomization rule. This is not UCINET's default setting, so you will need to indicate it yourself.


Dichotomize

Files

Input dataset (X): Blocked

Output dataset (Y): Blocked_GE_p1411

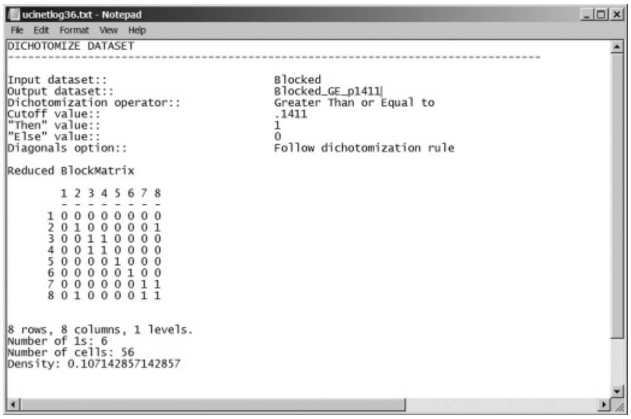
Dichotomization rule

If x(i,j) Greater Than or Eq value .1411

then y(i,j) = 1 else y(i,j) = 0

Diagonals of output (Y) matrix:
- Set to zero
- Set to missing
- Set to "then" value
- Set to "else" value
- [✓ Follow dichotomization rule]

Figure 9.16. UCINET Dichotomize Dialog Box



```
DICHOTOMIZE DATASET
--------------------------------------------------------------------------------

Input dataset::                                  Blocked
Output dataset::                                 Blocked_GE_p1411
Dichotomization operator::                       Greater Than or Equal to
Cutoff value::                                   .1411
"Then" value::                                   1
"Else" value::                                   0
Diagonals option::                               Follow dichotomization rule

Reduced BlockMatrix

         1 2 3 4 5 6 7 8
         - - - - - - - -
       1 0 0 0 0 0 0 0 0
       2 0 1 0 0 0 0 0 1
       3 0 0 1 1 0 0 0 0
       4 0 0 1 1 0 0 0 0
       5 0 0 0 0 1 0 0 0
       6 0 0 0 0 0 1 0 0
       7 0 0 0 0 0 0 1 1
       8 0 1 0 0 0 0 1 1


8 rows, 8 columns, 1 levels.
Number of 1s: 6
Number of cells: 56
Density: 0.107142857142857
```

Figure 9.17. Final Image Matrix, Threshold Method (UCINET)

When you click "OK," UCINET's output will produce a final image matrix that can be analyzed as a matrix (Figure 9.17) or as a network map (Figure 9.18). It now appears that the first, fifth, and sixth blocks are isolated from the rest of the network, whereas the third and fourth blocks and the seventh and eighth blocks are tied with one another, respectively. The eighth block is also connected to the second block, which may place it in a position of brokerage between the seventh and second block, suggesting a possible vulnerability in the network. Analysts may want to consider monitoring this block or explore strategies aimed at isolating it (e.g., through a deception campaign).

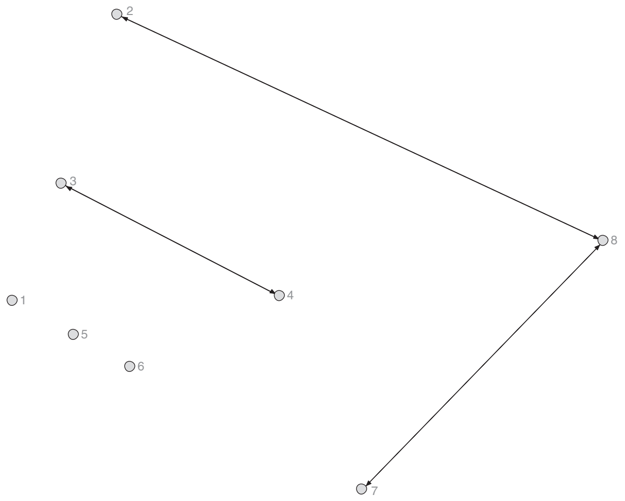
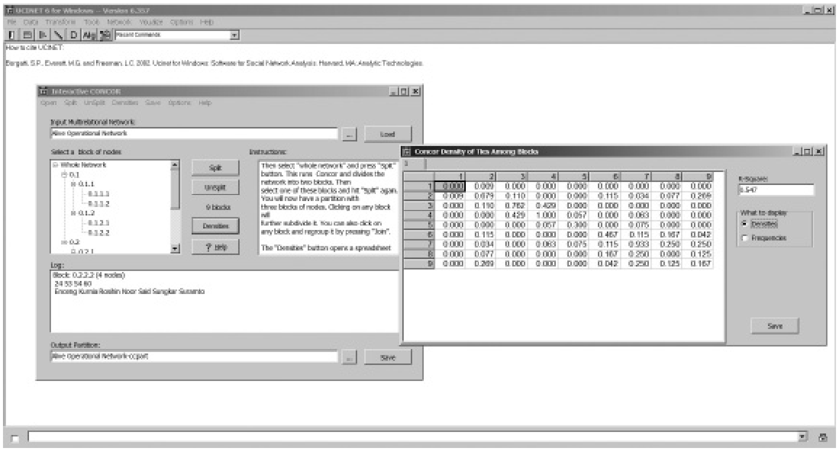


Figure 9.18. Sociogram of Final Image Matrix (NetDraw)


UCINET 6 for Windows -- Version 6.357
File Data Transform Tools Network Visualize Options Help
Recent Commands
Interactive CONCOR
Open Split UnSplit Densities Save Options Help
Split
UnSplit
Densities
? Help
Load
Save
Concor Density of Ties Among Blocks
Save

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.

Input Multirelational Network: Alive Operational Network

Select a block of nodes:

- Whole Network
  - 0.1
    - 0.1.1
      - 0.1.1.1
      - 0.1.1.2
    - 0.1.2
      - 0.1.2.1
      - 0.1.2.2
  - 0.2
    - 0.2.1

9 blocks

Instructions:

Then select "whole network" and press "Split" button. This runs Concor and divides the network into two blocks. Then select one of these blocks and hit "Split" again. You will now have a partition with three blocks of nodes. Clicking on any block will further subdivide it. You can also click on any block and regroup it by pressing "Join".

The "Densities" button opens a spreadsheet

Log:

```
Block: 0.2.2.2 (4 nodes)
 24 53 54 60
 Encang Kurnia Roslin Noor Said Sungkar Susanto
```

Output Partition: Alive Operational Network-ccpart

	1	2	3	4	5	6	7	8	9
1	0.000	0.009	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.009	0.079	0.110	0.000	0.000	0.115	0.034	0.077	0.269
3	0.000	0.110	0.762	0.429	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.429	1.000	0.057	0.000	0.063	0.000	0.000
5	0.000	0.000	0.000	0.057	0.300	0.000	0.075	0.000	0.000
6	0.000	0.115	0.000	0.000	0.000	0.467	0.115	0.167	0.042
7	0.000	0.034	0.000	0.063	0.075	0.115	0.933	0.250	0.250
8	0.000	0.077	0.000	0.000	0.000	0.167	0.250	0.000	0.125
9	0.000	0.269	0.000	0.000	0.000	0.042	0.250	0.125	0.167

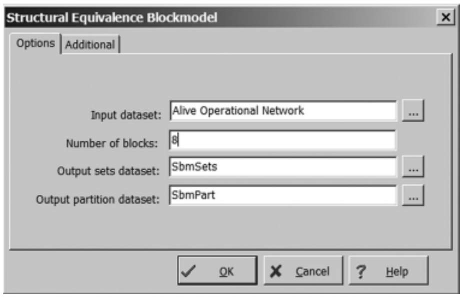
R-Squared: 0.547

What to display: [✓ Densities] Frequencies

Figure 9.19. UCINET Interactive CONCOR Dialog Box

The interactive CONCOR algorithm provides analysts with a little more flexibility in determining how often and where we want to split the network. You access the interactive dialog box (Figure 9.19) using the *Network>Roles & Positions>Structural>CONCOR>Interactive* command. Once you identify the data you intend to analyze, you first need to click "Load" before you can work with the data. You begin by selecting a block of nodes and clicking "Split." This will split the data in two. From this point, you can choose where you want the splits to occur. In the example in Figure 9.19, the network was initially split using the standard CONCOR algorithm, but then the seventh block (see Figure 9.13) was split. Clicking on "Densities" calls up a window that displays the corresponding density matrix (or frequencies) as well as indicates how well the blocking scheme fits the data. In this case, density of the seventh block has been increased to 0.933 because two actors (Adung and Usman bin Sef) were sorted into a separate block (8); the R^2 has increased as well (0.547). Once we are satisfied with how the network is blocked, we

Network>Roles & Positions >Structural >CONCOR >Interactive



Input dataset: Alive Operational Network

Number of blocks: 8

Output sets dataset: SbmSets

Output partition dataset: SbmPart

Figure 9.20. UCINET Optimization Structural Equivalence Dialog Box

simply click "Save" (we need to close the density matrix window first), and UCINET generates a partition that we can use to create a final image matrix as we did before.

Network> Roles & Positions >Structural >Optimization

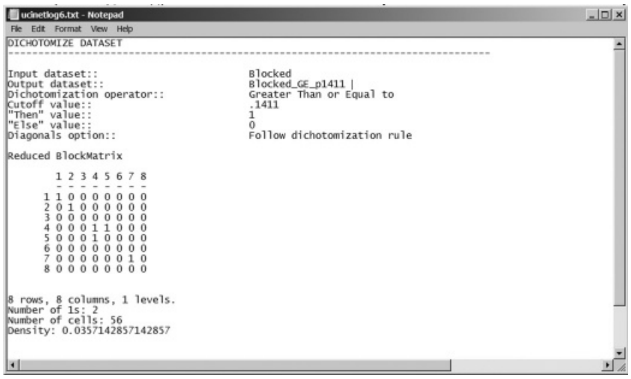
Optimization. We will consider one final structural equivalence algorithm available in UCINET that is accessed with UCINET's *Network> Roles & Positions>Structural>Optimization* command (Figure 9.20).[9] It permutes the rows and columns of the matrix, trying to find the best fit for the number of blocks the analyst designates at the outset ("8" in this case). Fit is determined by an error score (the lower the better) that compares the final model with a perfect model. Recall that in the case of perfect structural equivalence, in a complete block all possible ties are present, whereas in a null block, no ties are present. Consequently, with this approach an error is considered to occur when a tie is present in a null block or one is missing tie in a complete block (this should sound similar to faction analysis, which as we will see is simply a special form of blockmodeling). The optimization algorithm permutes the rows and columns until it can no longer lower the error score.

When you click "OK," UCINET generates a report (not shown) that indicates the final (and beginning) error score as well as an R^2 fit statistic.[10] As did the reports generated by the other equivalence algorithms, this one also indicates the blocks to which each actor is assigned, it provides a permuted and partitioned (blocked) adjacency matrix, and it displays a density table. In addition, it generates a partition that allows analysts to create a final image matrix (Figure 9.21) using the same steps we used previously. This image matrix tells a slightly different story than the previous one (note that the blocks here, both in content and sequence, do not necessarily correspond to blocks in Figure 9.17 – indeed, in this analysis Noordin is located in the second block, which has a density of 1.00). Here, all the blocks are isolated from one another except blocks 4 and 5.

Because the results obtained by the CONCOR algorithm and those with the optimization algorithm are difficult to compare, how might we choose between the two? We could examine them to see which makes more visual sense. Figure 9.22 is a visualization of the network where the colors of the nodes reflect the blocks identified by the standard (noninteractive) CONCOR algorithm, whereas Figure 9.23 is a visualization of the same network using the optimization partition. Because these networks are displayed here in gray scale, it is difficult to compare the two. It is much

[9] Note that this approach provides an option of using either valued or binary data. We will continue to use binary data here so that the results are comparable to the results we got previously.

[10] This feature does not always appear to work; in the previous analysis, UCINET reported an R^2 of 0.00, which is clearly incorrect.



```
DICHOTOMIZE DATASET
------------------------------------------------------------------------------

Input dataset::                         Blocked
Output dataset::                        Blocked_GE_p1411 |
Dichotomization operator::              Greater Than or Equal to
Cutoff value::                          .1411
"Then" value::                          1
"Else" value::                          0
Diagonals option::                      Follow dichotomization rule

Reduced BlockMatrix

          1 2 3 4 5 6 7 8
          - - - - - - - -
        1 1 0 0 0 0 0 0 0
        2 0 1 0 0 0 0 0 0
        3 0 0 0 0 0 0 0 0
        4 0 0 0 1 1 0 0 0
        5 0 0 0 1 0 0 0 0
        6 0 0 0 0 0 0 0 0
        7 0 0 0 0 0 0 1 0
        8 0 0 0 0 0 0 0 0


8 rows, 8 columns, 1 levels.
Number of 1s: 2
Number of cells: 56
Density: 0.0357142857142857
```

Figure 9.21. Final Image Matrix, Optimization Method (UCINET)

easier to compare them using several different colors. Nevertheless, it does appear that in this case the CONCOR algorithm has done a better job of identifying structurally equivalent actors than has the optimization algorithm.

Roles, Positions, and Structural Equivalence in Pajek The three approaches to structural equivalence illustrated above do not exhaust those that are available in UCINET. Hierarchical clustering is another common approach implemented in UCINET, one that is also implemented in Pajek. Nevertheless, in this chapter we will focus on Pajek's optimization algorithm because it is the one for which Pajek (and its developers) are

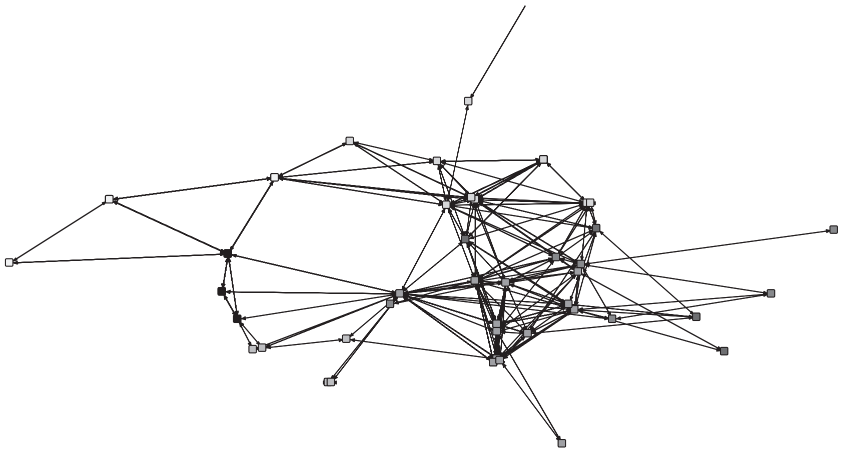


Figure 9.22. Alive Operational Network with CONCOR Partition (NetDraw)

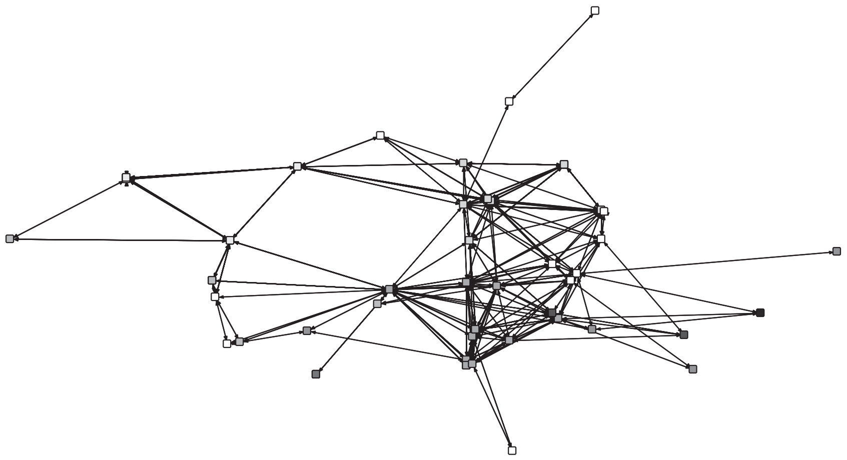


Figure 9.23. Alive Operational Network with Optimization Partition (NetDraw)

best known (Batagelj 1997; Batagelj, Ferligoj and Doreian 1992; de Nooy et al. 2005: 273–291; Doreian, Batagelj and Ferligoj 2005).

[Pajek] File >Network >Read

Begin by loading the trust network file (`Alive Operational Network.net`) into Pajek using Pajek's *File>Network>Read* command. Next, use Pajek's blockmodeling command (*Operations>Blockmodeling>Random Start*) to call up its blockmodeling dialog box (Figure 9.24). A drop-down menu allows you to indicate whether you want to generate structural or regular equivalence, or define one yourself.[11] You can also indicate how many iterations you want Pajek to perform in its search for the blockmodel partitions (yes, there can be more than one) that minimize the error score as well as how many clusters/blocks you want Pajek to identify. Other options are also available (see the bottom of the dialog box), but in most cases you will want to accept Pajek's defaults.

Operations >Blockmodeling >Random Start

Click "Run" and Pajek will generate a report that presents a final image matrix, a final error matrix, and a final error score (Figure 9.25). Note that Pajek's error score is lower than the one found by UCINET. In this case Pajek found only one solution, but if it had found more, it would create a partition for each solution, which means that you can compare them in Pajek's Draw screen (not shown), using Pajek's *Draw>Draw-Partition* command. The solution that Pajek did find differs substantially from any of those found by UCINET and suggests far more interconnection between the various blocks: in particular, blocks 2, 3, and 4.

Draw>Draw-Partition

[11] Patrick Doreian, Vladimir Batagelj, and Anuska Ferligoj (2005) have developed an approach to blockmodeling known as *generalized blockmodeling*, which allows users to combine a variety of equivalence measures into a single model.

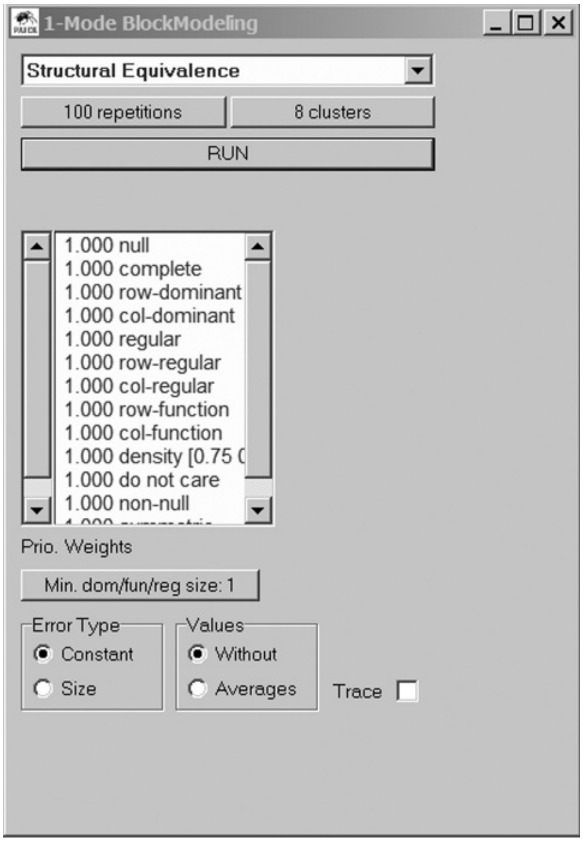
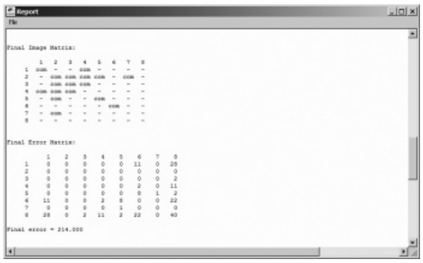


Figure 9.24. Pajek Blockmodeling Dialog Box

Such a conclusion may be misguided, however, because although it is somewhat hard to tell from Figure 9.26, in which the nodes are colored (i.e., gray scale) by the block to which they are assigned, blocks 2, 3, and 4 are relatively small in comparison to the first block. In fact, the fourth block is constituted by a single actor: Noordin Mohammed Top. Perhaps



```
Final Image Matrix:

        1   2   3   4   5   6   7   8
    1  com  -   -  com  -   -   -   -
    2   -  com com com com  -  com  -
    3   -  com com com  -   -   -   -
    4  com com com  -   -   -   -   -
    5   -  com  -   -  com  -   -   -
    6   -   -   -   -   -  com  -   -
    7   -  com  -   -   -   -   -   -
    8   -   -   -   -   -   -   -   -


Final Error Matrix:

          1     2     3     4     5     6     7     8
    1     0     0     0     0     0    11     0    28
    2     0     0     0     0     0     0     0     0
    3     0     0     0     0     0     0     0     2
    4     0     0     0     0     0     2     0    11
    5     0     0     0     0     0     6     1     2
    6    11     0     0     2     6     0     0    22
    7     0     0     0     0     1     0     0     0
    8    28     0     2    11     2    22     0    40

Final error = 214.000
```

Figure 9.25. Pajek Blockmodeling Report

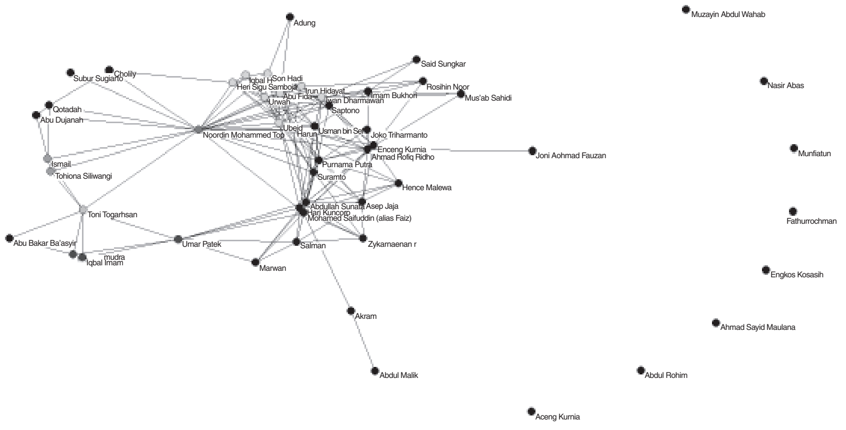


Figure 9.26. Blockmodel of Alive Operational Network (Pajek)

a better way to get a handle on the blocking scheme is to examine the network as a permuted matrix (i.e., a matrix where the rows and columns have been rearranged to reflect the blockmodeling solution).

Partition>Make Permutation

To do this, first ensure that the blockmodel partition found by Pajek is highlighted in the first Partition drop-down menu, and then create a permutation based on that partition, using Pajek's *Partition>Make Permutation* command (or if Pajek found more than one blockmodel solution, ensure that the blockmodel partition you want to use to create the permuted matrix is highlighted in the first Partition drop-down menu). Next, use the *File>Network>Export Matrix to EPS>Using Permutation* command to export the permuted matrix as an EPS file, which can then be displayed similar to Figure 9.27. What this permuted matrix suggests is that perhaps blocks 2, 3, and 4 should be thought of as a single block, and that there are four primary blocks within the network plus a "block" of relatively disconnected actors (blocks 7 and 8). However, blocks 2 and 4 have some unique characteristics, which suggests that we should perhaps treat them separately. Block 2 members (Hambali, Mohamed Ihsan, and Toni Togar), for instance, are connected not only with members of blocks 3 and 4 but also with members of blocks 5 and 7. Block 4 (Noordin) displays a similar pattern. In addition to being tied to members of blocks 2 and 3, he is tied to members of block 1. Indeed, the fact that the Pajek algorithm has identified him as a block unto himself suggests that perhaps he is not easily replaced and that his removal from the network could prove fatal for the group as a whole.

File>Network >Export Matrix to EPS>Using Permutation

As noted previously, faction analysis is actually just a special type of blockmodel in that it only permits ties within a block. If we wanted to estimate factions within Pajek, we would begin as we did previously, by using Pajek's blockmodel dialog box. This time use the drop-down menu

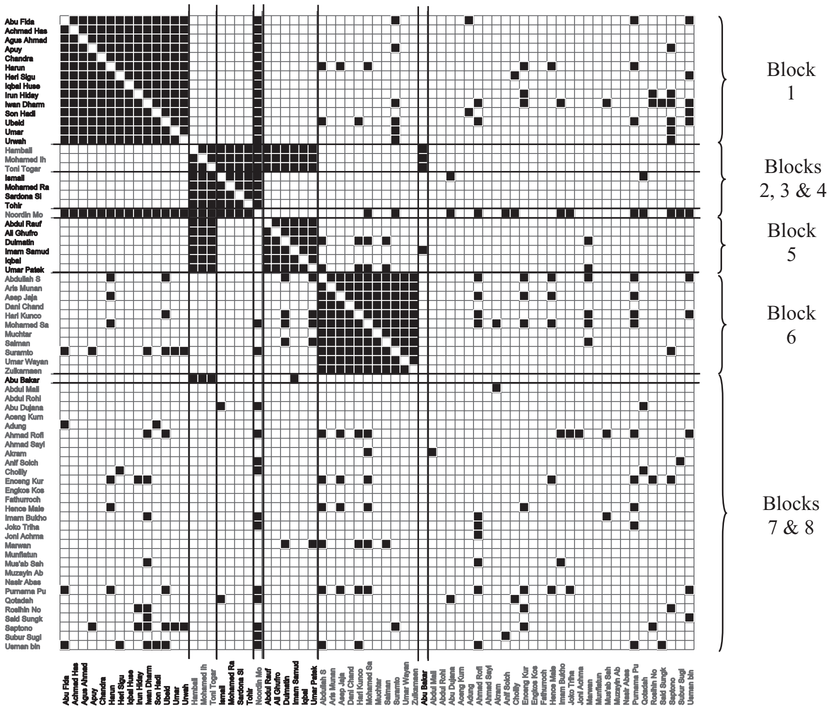
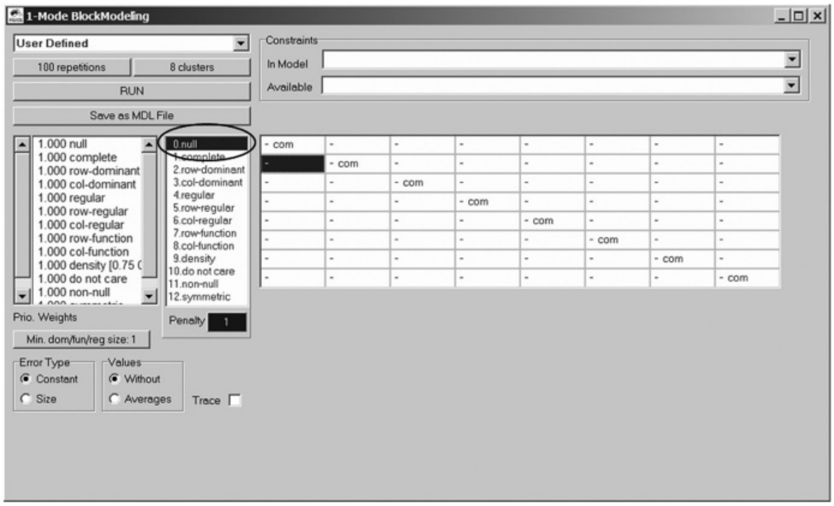


Figure 9.27. Pajek-Generated Permuted Matrix

and select the *User Defined* option. A matrix will appear to the right of the standard options that will allow you to indicate what type of relationship you would like Pajek to estimate within and between blocks (see Figure 9.28). Select each off-diagonal cell and then click on the “0.null” option (circled in Figure 9.28). This will tell Pajek that you want null (i.e., zero) blocks in the off-diagonal cells and complete blocks along the diagonal. Click “Run” and Pajek will conduct a factional analysis that you can visualize in the draw screen.

Roles, Positions, and Structural Equivalence in ORA. In ORA load the alive Noordin meta-network (`Alive Noordin Network.xml`) using either ORA’s *File>Open Meta-Network* command or the *Ctrl+0* quick key. There are two ways of identifying structurally equivalent actors in ORA: either from its main screen or its visualizer. Currently, ORA only implements the CONCOR algorithm, which is something of a limitation, but other algorithms will probably be added in the future. To identify structurally equivalent actors from ORA’s main screen, we need to use ORA’s “Locate SubGroups” report, which can be accessed by

File >Open Meta-Network



User Defined
100 repetitions
8 clusters
In Model
Available

1.000 null
1.000 complete
1.000 row-dominant
1.000 col-dominant
1.000 regular
1.000 row-regular
1.000 col-regular
1.000 row-function
1.000 col-function
1.000 density [0.75 (
1.000 do not care
1.000 non-null

[✓ 0.null]
1.complete
2.row-dominant
3.col-dominant
4.regular
5.row-regular
6.col-regular
7.row-function
8.col-function
9.density
10.do not care
11.non-null
12.symmetric

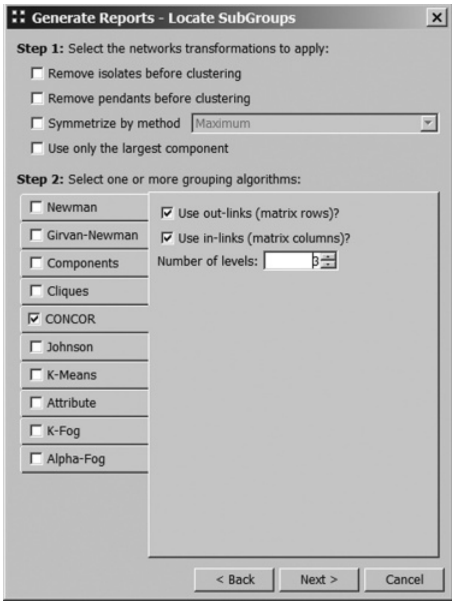
Penalty 1
Min. dom/fun/reg size: 1
Error Type: (● Constant) ○ Size
Values: (● Without) ○ Averages
Trace ☐

- com	-	-	-	-	-	-	-
-	- com	-	-	-	-	-	-
-	-	- com	-	-	-	-	-
-	-	-	- com	-	-	-	-
-	-	-	-	- com	-	-	-
-	-	-	-	-	- com	-	-
-	-	-	-	-	-	- com	-
-	-	-	-	-	-	-	- com

Figure 9.28. Pajek Blockmodeling Dialog Box with User-Defined Option

Analysis >Generate Reports >Groups >Locate SubGroups

ORA's *Analysis>Generate Reports>Groups>Locate SubGroups* command, which calls up the corresponding dialog box (not shown). Click on "Next." At the second dialog box (not shown), indicate which network you want ORA to use for blockmodeling (ORA will use all networks selected to estimate one blockmodel) and click "Next." This will bring up a dialog box similar to one shown in Figure 9.29. Make sure the CONCOR button is chosen and indicate how many levels you want (here, we have selected three). You may remember that ORA allows you to delete isolates prior to identifying subgroups. This can be an attractive option, but for now let's keep our isolates in order to make our results comparable to what we found using UCINET. You may want to come back and rerun the analysis eliminating isolates first. (To do this, simply click on the "Select Network Transformations" tab and then click on the "Remove isolates before clustering" box.) When you click "Next," another dialog box appears (not shown), which asks you what types of output you want. Select all of the options and click "Next." At the next dialog box (not shown), indicate that you want text and HTML output, give your output file a name, click "Finish," and ORA will generate a report for each network that lists each actor by the blocks to which they belong, a hierarchical clustering diagram, a dendogram, a permuted and partitioned matrix, a density table, a reduced block matrix, a graph that corresponds to the density table (and not the reduced block matrix), and a few additional metrics. Note that ORA does not give you a choice in setting your own threshold when it comes to creating the reduced block matrix. It also does not provide you with any measures of fit.


Generate Reports - Locate SubGroups

Step 1: Select the networks transformations to apply:

- ☐ Remove isolates before clustering
- ☐ Remove pendants before clustering
- ☐ Symmetrize by method: Maximum
- ☐ Use only the largest component

Step 2: Select one or more grouping algorithms:

- ☐ Newman
- ☐ Girvan-Newman
- ☐ Components
- ☐ Cliques
- ☑ CONCOR
- ☐ Johnson
- ☐ K-Means
- ☐ Attribute
- ☐ K-Fog
- ☐ Alpha-Fog

- ☑ Use out-links (matrix rows)?
- ☑ Use in-links (matrix columns)?

Number of levels: 3

Figure 9.29. ORA's Locate Subgroups Dialog Box with CONCOR Option

To visualize the results generated by CONCOR's algorithm, return to the main screen, and you will note that ORA has created a new meta-network, probably called "Alive Network Groups." Select this and click "Visualize" and you will see a network map that looks something like Figure 9.30. Because it includes both the actors and the CONCOR groups, as well as the ties between actors and actors, the ties between actors and groups, and the ties between groups and groups, this is a relatively unhelpful visualization. However, because ORA allows us to turn objects and ties off and on, we can play with the visualization to make it more intelligible and useful. For example, if you select only the CONCOR groups and the CONCOR blockmodel ties, you will get a graph of the reduced block matrix. If you select the CONCOR groups and the CONCOR block densities ties, then you get the same graph that appeared in the original CONCOR report.

To run the CONCOR algorithm straight from ORA's visualizer screen, first (at ORA's main screen) select the original Noordin Top trust network (not the meta-networks created by ORA). Next, click "Visualize." Then, at the *Visualizer* screen, select ORA's *Display>Node Color>Display Nodes by Concor Grouping* command, indicate the number of splits, click "OK" and ORA will display the network with the nodes colored by the CONCOR groups (Figure 9.31). Note that ORA treats isolates as a separate group (they have been hidden in the graph using ORA's

[ORA-Visualizer] Display >Node Color>Display Nodes by Concor Grouping

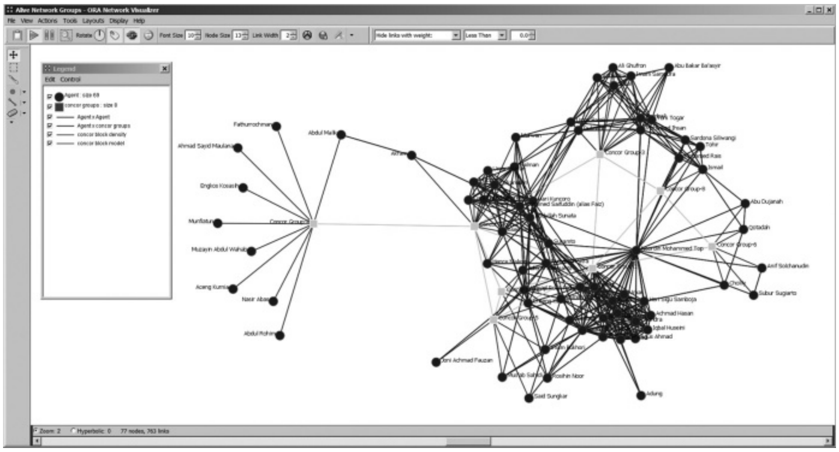


Figure 9.30. ORA Visualization of CONCOR Groups

Actions>Isolates >Hide Isolate Nodes

Actions>Isolates>Hide Isolate Nodes command) and only performs the algorithm on the remaining nodes. Also, note that when you identify the groups from the visualizer, ORA does not provide you with any of the accompanying output that you get when running the algorithm from the main screen. Comparing this blocked network map with the one Figure 9.22 (also based on the CONCOR algorithm) suggests that ORA and UCINET have arrived at similar solutions although one would need to examine the results in detail to know if they are identical.

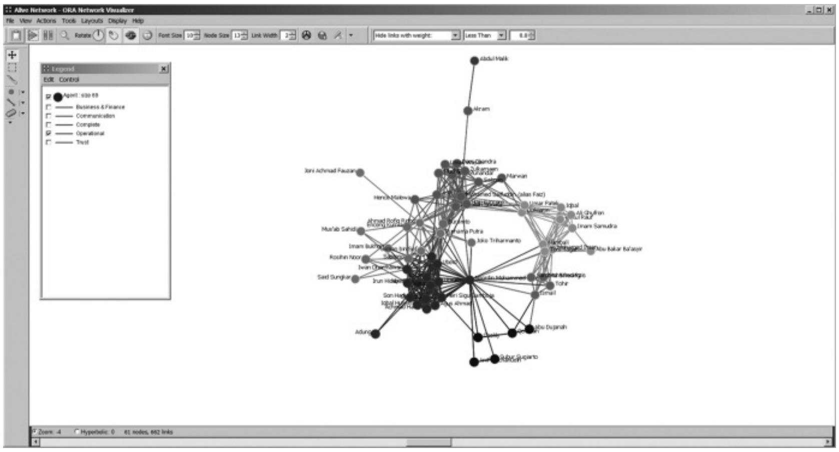


Figure 9.31. ORA Visualization of CONCOR Grouping (from Visualizer)

9.6 Summary and Conclusion

In this chapter we have explored three approaches for identifying actors holding similar positions in a network and sorting them into distinct classes or blocks: structural, automorphic, and regular equivalence.[12] Two actors are said to be structurally equivalent when they have exactly the same relationships to all other actors. Another way to think about it is that, in order to be structurally equivalent, two actors must be exactly substitutable for one another (Hanneman and Riddle 2005). As we saw, structural equivalence is a very restrictive understanding of equivalence. Seldom do two or more actors have exactly the same relationships to all other actors, which is why some have questioned the usefulness of the concept Wasserman and Faust (1994:468–469). What this often means in practice is that analysts use some sort of threshold measure to determine the blocks to which actors belong and the ties that exist between them.

Although structural equivalence is the most widely implemented approach to finding equivalent actors, others have argued on behalf of more generalized or relaxed definitions of equivalence that they believe possess somewhat more realistic measures of what it means for two actors to be regarded as "equivalent" or "structurally similar" (see, e.g., Borgatti and Everett 1992; Faust 1988; Wasserman and Faust 1994). With automorphic equivalence, actors are considered equivalent if they can be mapped onto one another. With regular equivalence, actors are considered to be equivalent if they are connected to actors in the same classes. These three approaches are nested within one another in terms of strictness. That is, structurally equivalent actors are also automorphically and regularly equivalent with one another, and automorphically equivalent actors are also regularly equivalent with one another. The reverse is not true, however. Regular equivalent actors are not necessarily automorphically or structurally equivalent with one another, and automorphically equivalent actors are not necessarily structurally equivalent with one another.

Did we learn anything about Noordin's operational network? Yes, although the results from the various blockmodeling algorithms did not produce identical results, the CONCOR analysis (see Figure 9.18) suggests that there may be a class or block of actors in the network that is in a position of brokerage between other blocks in the network and we may want to monitor this block or craft a deception campaign aimed at isolating or discrediting it. The same analysis also found the block of which Noordin is a highly connected member. As we have noted in

[12] One approach to blockmodeling that we did not consider in this chapter was developed by Patrick Doreian, Vladimir Batagelj, and Anuska Ferligoj in what is referred to as *generalized blockmodeling*, which allows analysts to specify a combination of block types in a given network (see de Nooy et al. 2005; Doreian et al. 2005).

previous chapters, actors that are closely tied to a group's leader are unlikely to defect, so this would be a group that we would probably not want to "waste" resources on, in an attempt to reintegrate members through amnesty or deprogramming campaigns. Instead, we should focus our efforts on actors in other blocks, preferably those blocks whose ties with Noordin's block are minimal. The highly connected nature of this block also suggests that if Noordin was removed from the network, future group leaders could emerge from this block. Thus, we probably would want to dedicate some resources to track members of this block. Finally, the optimization approach to Noordin's network uncovered some interesting structural dynamics of the group. We saw that three blocks are highly connected to one another and may be considered a block unto themselves. We also saw, however, that two of these blocks possessed some interesting connections, which indicated that we may want to treat them separately. In particular, we saw that Noordin was identified as block unto himself, suggesting that he may not be easily replaceable and that his isolation could prove fatal to the group. Of course, we only used the structural equivalence algorithms to examine Noordin's alive operational network. We would most likely want to supplement our analysis with an exploration of Noordin's network using both automorphic and regular equivalence algorithms.