# SNA Basics #9
# Brokers and Bridges

In this lab, we will learn how to use a variety of algorithms to detect bridges and brokers in a network. Along the way, we'll encounter a number of different terms/algorithms, in particular: bridges, cut-points, bi-components, structural holes, brokerage roles, and edge betweenness. In general, a bridge can be thought of as a tie that connects two subgroups. When used in conjunction with bi-component and cut-point analysis, a bridge is a tie that when deleted, disconnects the network. Similarly, a cut-point is a node that, when deleted, disconnects the network, while a bi-component is a component that is immune to the removal of a single node (i.e., it doesn't contain a cut-point). The concept of structural holes was developed by Ron Burt. It builds on Mark Granovetter's notion of "the strength of weak ties," but rather than focusing on the type of tie (i.e., strong vs. weak) in order to identify bridges, it focuses on finding the gaps (i.e., holes) in the social structure that some ties bridge. The brokerage roles algorithm was developed by Roger Gould and Roberto Fernandez. It is unique in that it not only takes into account an actor's network position, but also the subgroup to which he or she belongs. Finally, edge betweenness identifies edges (rather than a nodes) that lie on between other edges in the network. The assumption here is that edges that score high in betweenness are more likely to be bridges than those that don't.

The data for the lab are in the zipped file, `SNA Basics #9(Data).zip`. It is the communication network of a wood-processing facility where workers rejected a new compensation package and eventually went on strike. Management then brought in an outside consultant to analyze the employee's communication structure because it felt that information about the package was not being effectively communicated to all employees by the union negotiators. The outside consultant asked all employees to indicate, on a 5-point scale, the frequency that they discussed the strike with each of their colleagues, ranging from 'almost never' (less than once per week) to 'very often' (several times per day). The consultant used 3 as a cut-off value in order to identify a tie between two employees. If at least one of two persons indicated they discussed work with a frequency of three or more, a tie between them was included in the network.[1]

## Part I – Bridges and Brokers in NetDraw and UCINET

*[NetDraw]*
*File>Open>Ucinet dataset*
*>Network*

*File>Open>Ucinet dataset*
*>Attribute data*

*Properties>Nodes>Symbols*
*>Color>Attribute-based*

1. In NetDraw read in the `Strike.##h` network and `Strike_groups.##h` attribute (partition) files using the *File>Open>Ucinet dataset>Network* and *File>Open>Ucinet dataset>Attribute data* commands, respectively. Vary the colors of the nodes based on which group each actor belongs to using the *Properties>Nodes>Symbols>Color>Attribute-based* command. The groups are defined by age and language. The Spanish-speaking employees, who are 30 or younger, (class 1), are almost disconnected from the English-speaking young employees (class 2), who communicate with no more than two of the older English-speaking employees 38 years or older (class 3). All ties between groups have special backgrounds. Among the Hispanics, Alejandro is most proficient in English and Bob speaks some Spanish, which helps to explain

---

[1] Adapted from Wouter de Nooy, Andrej Mrvar, and Vladimir Batagelj. 2011. *Exploratory Social Network Analysis with Pajek*. Cambridge, UK: Cambridge University Press.

their tie. Bob owes Norm for getting his job and Ozzie is the father of Karl. Sam and Wendle are the union negotiators and were responsible for explaining the new program proposed by the managers. When the informal communication structure was mapped, management approached two other employees (Bob and Norm) and explained the reforms. Within two days, the young and old employees struck a deal, and the strike soon ended.

2. In the network, the tie between Alejandro and Bob is a bottleneck because it's the only channel for information exchange between the Hispanic employees and all the other employees. Removing it would cut the Hispanic employees off from information circulating among the other employees. Formally, this line is a *bridge* because its removal creates a new component that is isolated from other components. Put differently, when you remove the line between Alejandro and Bob, you disconnect the Hispanic workers from the rest of the communication network. Note that there is one more bridge in the network. If you remove the tie between Frank and Gill, Frank becomes an isolate. Since an isolate is a component, the network consists of two components after removing Frank's tie with Gill.

3. The deletion of an actor can have the same effect as the removal of a tie (since when you delete an actor you effectively remove all ties connected to it). If Bob were to leave or refuse to discuss the strike anymore, all of his ties would disappear, including the bridge to Alejandro. Bob is known as a cut-vertex, cut-point, articulation point, or boundary spanner (depending on which program you use) because deleting it disconnects the network or disconnects a component of the network. Just like a bridge, a cut-point is crucial to the flow of information in a network. How does one identify cutpoints in a network? Cutpoints are actors that belong to two or more bi-components, including bi-components of size two (see below).

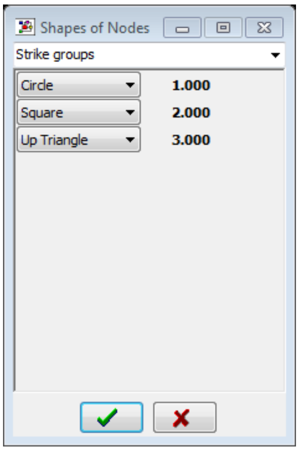
4. Bi-components (aka, blocks) are subnetworks that are invulnerable to the removal of a single actor and are often defined as a component of minimum size three without a cut-point. (However, bi-components of size two function as bridges between components, so we often ask our SNA programs to identify bi-components consisting of two actors because that helps us identify bridges in the network.) Thus, in a bi-component, no actor can completely control the information flow between other actors because there is always an alternative path which information may follow. A bi-component is generally seen as being more cohesive than a strong or weak component because there are at least two different paths between each pair of actors.

*Analysis*
*>Blocks & Cutpoints*

*Properties>Nodes>Symbols*
*>Shape>Attribute-based*

*Properties>Nodes>Symbols*
*>Color>Attribute-based*

5. You can identify bi-components (aka, blocks) and cut-points in NetDraw using the select the *Analysis>Blocks & Cutpoints* command. This command colors your cutpoints one color and the non-cutpoints another. Next, change node shapes to reflect which strike group the actors are a part of with the *Properties>Nodes>Shape>Attribute-based* command. In the dialog box that appears (see Figure 1), choose the "Strike_groups" attribute and click OK. If you want to highlight (by color) the six bi-components identified by NetDraw, select the *Properties>Nodes>Color>Attribute-based* command, and then in the dialog box select the block (i.e., bi-component) you want to highlight.



Strike groups

| | |
|---|---|
| Circle | 1.000 |
| Square | 2.000 |
| Up Triangle | 3.000 |

**Figure 1:** NetDraw "Shapes of Nodes" dialog box

*Analysis*
*>Centrality measures*

*Properties>Nodes>Symbols*
*>Size>Attribute-based*

6. Next, let's adjust the size of the nodes to reflect betweenness centrality. First, calculate betweenness centrality using the *Analysis>Centrality measures* command. Then, select the *Properties>Nodes>Symbols>Size> Attribute-based* command, choose betweenness and click OK. You should get a drawing that looks similar to Figure 2.

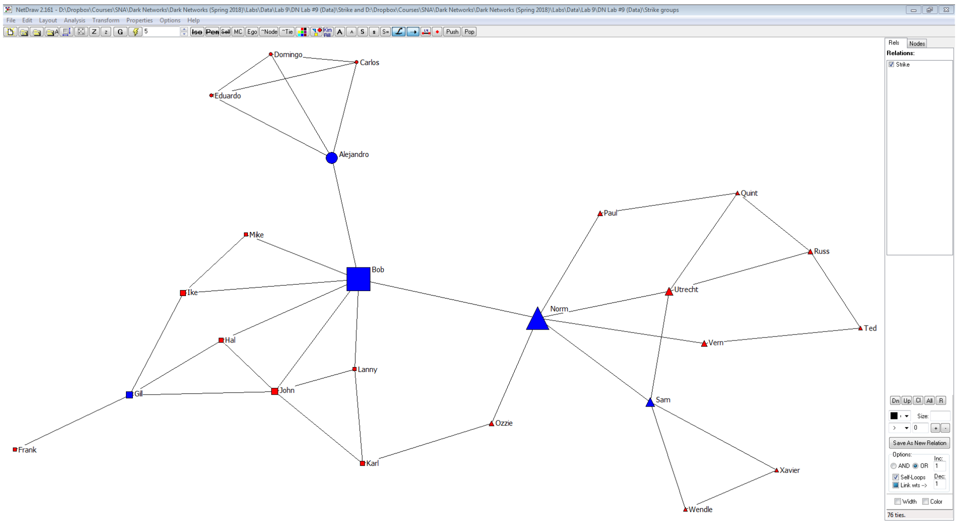
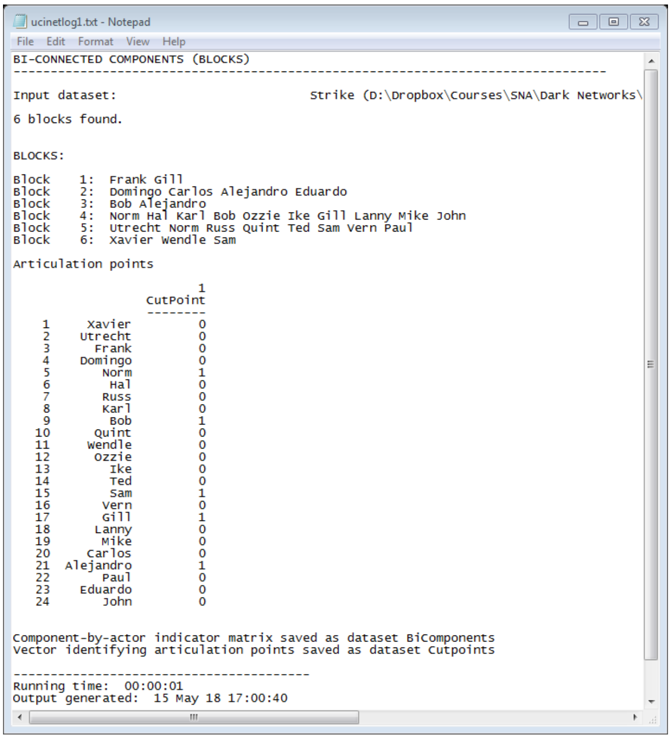


**Figure 2:** NetDraw Drawing of Strike Network

*[UCINET]*
*Network>Regions*
*>Bi-Component*

7. To identify bi-components and cut vertices, use the *Network>Regions>Bi-Component* command. Choose the *Strike* data and click OK. An output log

(Figure 3) appears that shows the various bi-components (blocks) and cut-vertices (cutpoints) are. Note that two of the bi-components are size two – Frank & Gill and Bob & Alejandro – which, as noted above, are bridges in the network, the removal of which disconnects the network.

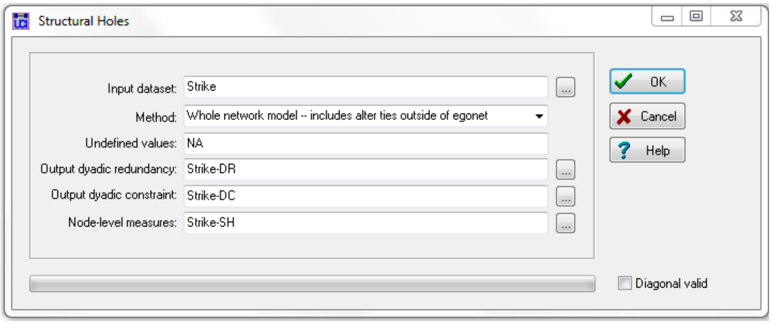


```
BI-CONNECTED COMPONENTS (BLOCKS)
--------------------------------------------------------------------------------

Input dataset:                          Strike (D:\Dropbox\Courses\SNA\Dark Networks\

6 blocks found.


BLOCKS:

Block    1:  Frank Gill
Block    2:  Domingo Carlos Alejandro Eduardo
Block    3:  Bob Alejandro
Block    4:  Norm Hal Karl Bob Ozzie Ike Gill Lanny Mike John
Block    5:  Utrecht Norm Russ Quint Ted Sam Vern Paul
Block    6:  Xavier Wendle Sam

Articulation points

                        1
                 CutPoint
                 --------
    1     Xavier        0
    2    Utrecht        0
    3      Frank        0
    4    Domingo        0
    5       Norm        1
    6        Hal        0
    7       Russ        0
    8       Karl        0
    9        Bob        1
   10      Quint        0
   11     Wendle        0
   12      Ozzie        0
   13        Ike        0
   14        Ted        0
   15        Sam        1
   16       Vern        0
   17       Gill        1
   18      Lanny        0
   19       Mike        0
   20     Carlos        0
   21  Alejandro        1
   22       Paul        0
   23    Eduardo        0
   24       John        0


Component-by-actor indicator matrix saved as dataset BiComponents
Vector identifying articulation points saved as dataset Cutpoints

----------------------------------------
Running time:  00:00:01
Output generated:  15 May 18 17:00:40
```

**Figure 3:** UCINET Bi-Component Output

*[UCINET]*
*Network>Ego Networks*
*>Structural Holes*

8. Now in UCINET, with the strike network file (`Strike.##h`), calculate the structural hole (constraint) measures for the network, using the *Network>Ego Networks>Structural Holes* command and *the "Whole network model – normal method"* option (see Figure 4). The constraint measures are located toward the bottom of the output log.[2]



Input dataset: Strike
Method: Whole network model -- includes alter ties outside of egonet
Undefined values: NA
Output dyadic redundancy: Strike-DR
Output dyadic constraint: Strike-DC
Node-level measures: Strike-SH
☐ Diagonal valid

**Figure 4**: UCINET's Structural Holes Dialog Box

---

[2] For a relatively non-technical explanation of the difference between the whole network and ego network methods for detecting structural holes, see Cunningham, Everton, and Murphy, *Understanding Dark Networks*, pp. 183-184.

*[NetDraw]*
*File>Open>Ucinet dataset*
*>Network*

*Analysis>Structural Holes*
*>Whole Network model*

*Properties>Nodes>Symbols*
*>Size>Attribute-based*

9. Now, re-open the strike data in NetDraw (so that the previous steps are "erased"). Under the *Analysis>Structural Holes* submenu, select the *Whole Network model* option, which tells NetDraw to calculate a series of structural holes measures, which you can inspect in the *Node attribute editor* found under the *Transform* menu. Next, vary the size of the actors' nodes using NetDraw's *Properties>Nodes>Symbols>Size>Attribute-based* command and then choosing the *rConstraint* attribute, which reverses the constraint score, so that nodes with lower constraint have larger nodes. Your sociogram should look similar Figure 5.

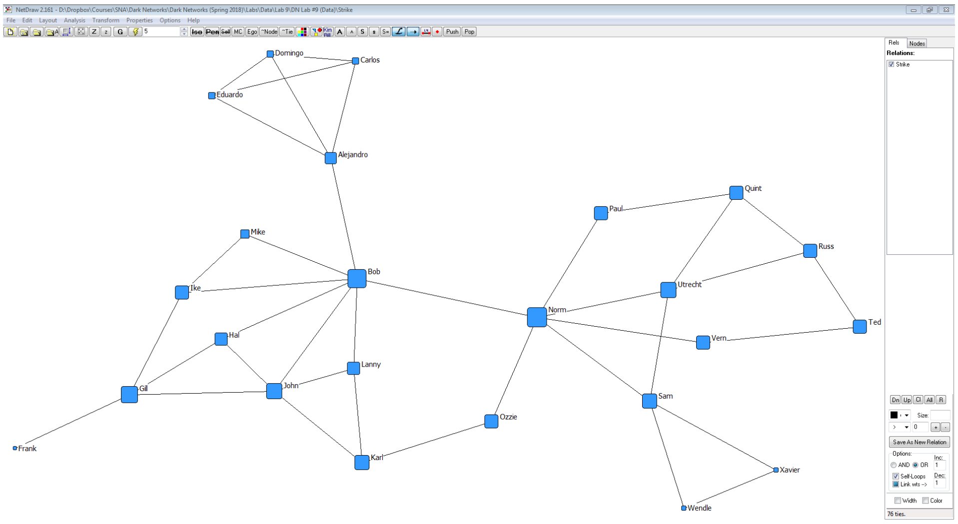
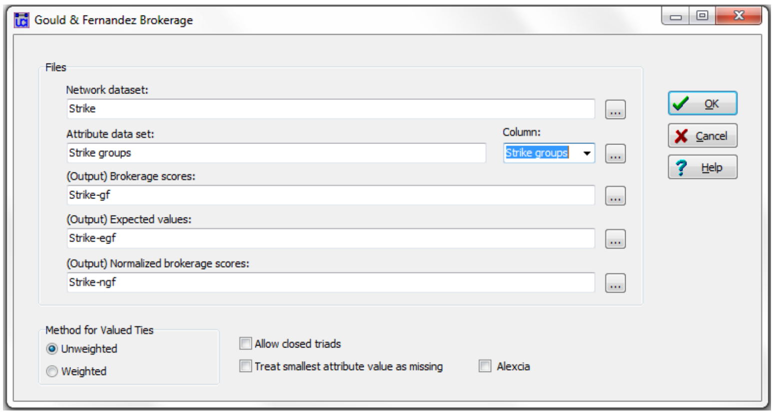


**Figure 5**: NetDraw Sociogram of Strike Network

*[UCINET]*
*Network>Ego Networks*
*>G&F Brokerage Roles*

10. Now, let's consider the Gould and Fernandez brokerage roles. Select the *Network>Ego Networks>G&F Brokerage Roles* command, which brings up the brokerage dialogue box (see Figure 6). For the input dataset, select the strike network; for the partition, choose the strike groups file. Keeping UCINET's defaults, select OK. This brings up an output log and saves the calculations in two attribute files (brokerage and relative-brokerage).



Gould & Fernandez Brokerage

Files

Network dataset: Strike

Attribute data set: Strike groups

Column: Strike groups

(Output) Brokerage scores: Strike-gf

(Output) Expected values: Strike-egf

(Output) Normalized brokerage scores: Strike-ngf

Method for Valued Ties

● Unweighted

○ Weighted

☐ Allow closed triads

☐ Treat smallest attribute value as missing

☐ Alexcia

**Figure 6**: Gould and Fernandez Brokerage Dialogue Box

*[NetDraw]*

11. As of now, NetDraw does not include a G&F brokerage role function. Still, you can visualize the five different brokerage roles in NetDraw by opening

first the strike network file and then the brokerage file (`strike-gf.##h`). Visualize the strike network where node size reflects the number of gatekeeper roles each individual in the network enjoys. Although we used a predetermined group for our partition, here we could have just as easily run a clustering algorithm (e.g., Girvan-Newman), and then used the optimal partition generated by it to calculate the various brokerage scores that we estimated in the step 10 above.

12. Finally, let's examine how to calculate and visualize edge betweenness centrality, which is a great way for detecting bridges and a method used by some community detection algorithms. Specifically, the Girvan-Newman algorithm first estimates betweenness centrality for each edge/tie in the network, then removes the tie with the highest betweenness score, then recalculates edge betweenness centrality, and iteratively repeats this process until no edges/ties remain. And each time the network disconnects, the algorithm calculates a modularity score. Because edges of high betweenness are assumed to be vital for connecting different parts of the network, it is an efficient method for breaking the network down into large subgroups and then into smaller ones. An implicit assumption is that edges that score high in edge betweenness are more likely to span gaps in the social structure. *Put differently, they are more likely to be bridges than are those that do not*.

*[UCINET]*
*Network*
*>Centrality and Power*
*>Freeman Betweenness*
*>Edge (line)*
*Betweenness*

13. Edge betweenness is calculated in UCINET using its *Network>Centrality and Power>Freeman Betweenness>Edge (line) Betweenness* command. Generally, you will want to accept UCINET's defaults and click OK. The command generates a new network where the cell values are edge betweenness scores. If you examine the network matrix, you can determine which edge has the highest betweenness score.

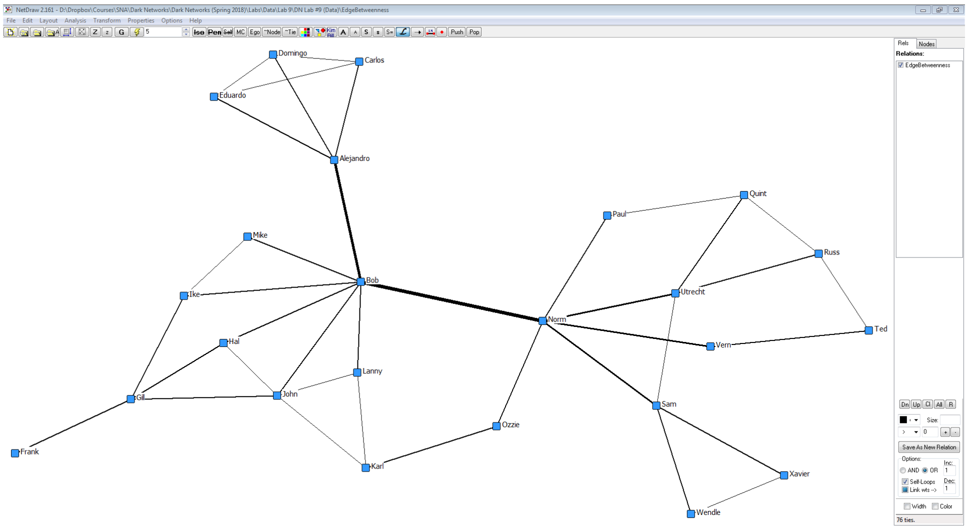


**Figure 7**: Strike Network Where Edges Reflect Betweenness Centrality

*[NetDraw] Properties>Lines>Size >Tie Strength*

14. While inspecting a network matrix of edge betweenness scores is feasible with small networks, it is difficult with large networks. An easier way to examine the network is to visualize it in NetDraw. Open your newly generated network (probably called "`EdgeBetweeness.##h`") in NetDraw and vary the size of the ties based on tie strength (*Properties>Lines>Size>Tie Strength*). To see the edges clearly turn off the arrows, and you should end up with a network map that looks similar to Figure 7. As you can see, some of the ties score higher in terms of edge betweenness (i.e., they are thicker) than do others. Two ties jump out as bridges in the network: the tie between Bob and Norm, and the tie between Bob and Alejandro.

## Part II – Bridges and Brokers in Pajek

*File>Pajek Project File Read>Strike.paj*

1. Open the wood-processing facility ***project file*** (`Strike.paj`), which is included in the zipped data file. You should see two items in the drop lists on the main menu: the strike network (`strike.net`) and a partition (`Strike_groups.clu`) that assigns the various groups into different classes.

*Network >Create New Network >with Bi-Connected Components stored as Relation Numbers*

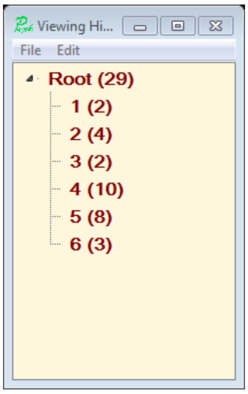
2. Use the *Network>Create New Network>with Bi-Connected Components stored as Relation Numbers* command to find a network's bi-components, bridges, and cut-points. A dialog box will prompt you to specify the minimum size of the bi-components to be identified. While the default value (3) will identify the bi-components within the network, it will only report cut-points that connect two or more bi-components. If you select a minimum size of two, Pajek will identify all bi-components, bridges (bi-components of size 2), and cut-points,[3] so change the default value to 2. Select "Yes" when Pajek asks if you want to create a new network.

*[Draw] Options>Colors>Edges >Relation Number*

3. This command generates a new network, a partition, a vector, and hierarchy (something we haven't seen before). The new network is a multi-relational network—one for each bi-component. If you visualize the network and select the *Options>Colors>Edges>Relation Number* option, the color of the ties between the actors will vary. The new partition indicates the number of the bridge or bi-component to which an actor belongs. Actors that do not belong to one (e.g., isolates) are assigned to class 0, and actors that belong to two or more (i.e., cut-points) are placed in class number 9999998. The vector (articulation points) indicates the number of bridges or bi-components to which an actor belongs: 0 for isolates, 1 for actors that belong to exactly one bridge or bi-component, and 2 for actors belonging to two or more. Finally, the hierarchy indicates the bridges or bi-components to which each actor belongs. Pajek uses a hierarchy to store the bridges and bi-components because cut-points belong to two or more bi-components (see Figure 8).

---

[3] Pajek's *Bi-Components* function treats directed networks as if they were undirected, which means that it identifies weak instead of strong components.



Root (29)
- 1 (2)
- 2 (4)
- 3 (2)
- 4 (10)
- 5 (8)
- 6 (3)

**Figure 8**: Pajek hierarchy of bi-components and bridges

*File>Hierarchy>View/Edit*

4. Since bridges are components of size two in an undirected network without multiple lines, it is easy to find the bridges in the hierarchy of bi-components: open the Edit screen with the hierarchy of bi-components with the command *File>Hierarchy>View/Edit* or with the Edit button on the left of the hierarchy drop-down menu. Next, click on the "+" sign (or arrow) left of the word "Root." This should produce a figure *similar* (but probably not identical) to Figure 8; this lists the six bridges and bi-components among the striking employees. The size of each subnetwork is reported between brackets, so double-click on the two subnetworks of size two in order to see the actors on either side of the bridge.

*Draw>Network + First Partition + First Vector*

5. Next, open the draw screen with the *Draw>Network + First Partition + First Vector* command from the main menu, making sure that the strike network appears in the first network drop down box, the partition "Vertices belonging…" appears in the first partition drop-down menu and the vector "Articulation points…" appears in the first vector drop-down menu. This should create a drawing (Figure 9) where most of the employees are colored according to the bi-component to which they belong and a few (i.e., cut-points) are colored gray. The cut-vertices should also be larger than the other actors.

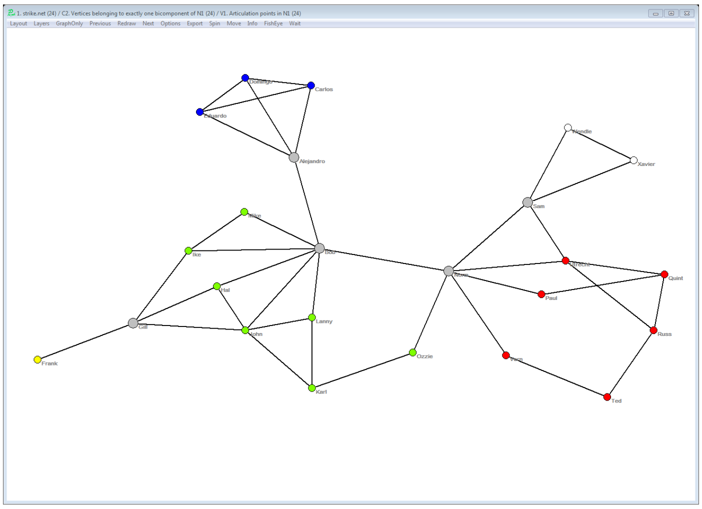


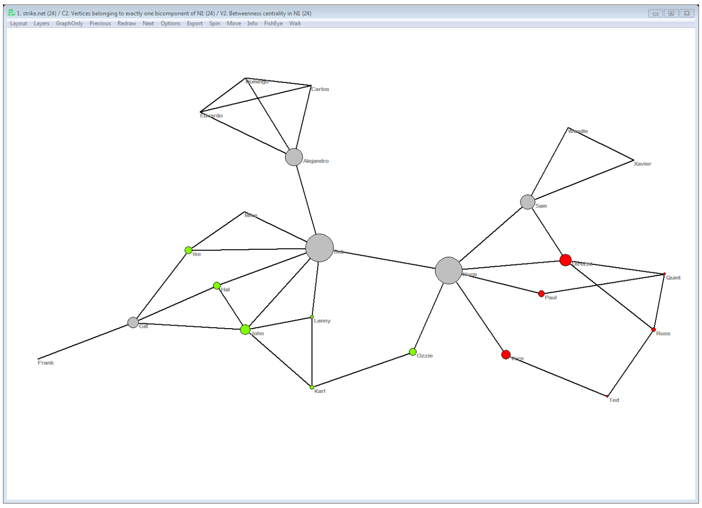
**Figure 9**: Strike Group Bi-Components and Cut-Points

*Network>Create Vector >Centrality>Betweenness*

*Draw>Network + First Partition + First Vector*

*Options>Size>of Vertices*

6. Next, calculate the betweenness centrality of the strike network using the *Network>Create Vector>Centrality>Betweenness* command. Then, with the same partition that you used in the previous step showing in the first partition drop-down box and the new betweenness centrality vector showing in the first vector drop-down box, draw the network with the *Draw> Network + First Partition + First Vector* command. It should look similar to Figure 10. You may need to adjust the options so that Pajek automatically determines the vertex size: *Options>Size>of Vertices*, and then type "0" in the dialog box.



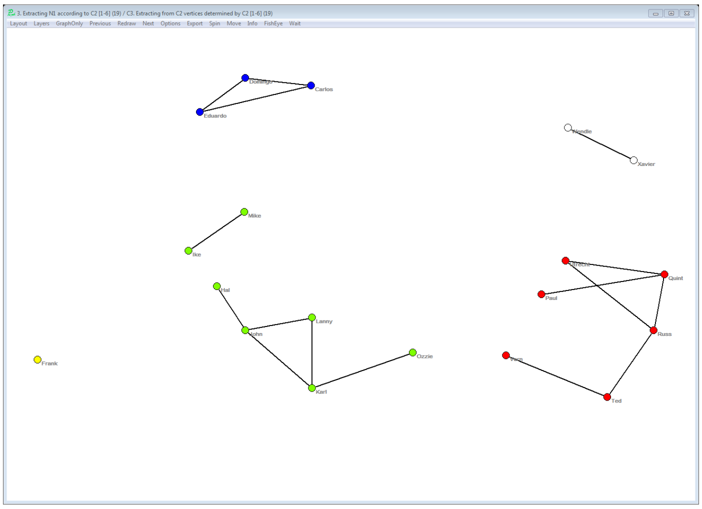
1. strike.net (24) / C2. Vertices belonging to exactly one bicomponent of N1 (24) / V2. Betweenness centrality in N1 (24)

**Figure 10**: Strike Group Bi-Components, Cut-Points, and Betweenness

*Operations >Network + Partition >Extract SubNetwork Induced by Union of Selected Clusters*

*Draw>Network + Partition*

7. To get a sense of what the network would look like if you were to remove the cut-vertices from the network, keep (extract) the non-cut vertices from the network. First, make sure that the original network is highlighted in the network drop box and that the partition, "Vertices Belonging…", is highlighted in the first partition drop box. Then, using the *Operations> Network + Partition>Extract SubNetwork Induced by Union of Selected Clusters* command, extract classes (clusters) 1-6. This should leave you with a network of 19 actors, which you can then visualize.



3. Extracting N1 according to C2 [1-6] (19) / C3. Extracting from C2 vertices determined by C2 [1-6] (19)

**Figure 11**: Strike Network After Removal of Cutpoints

*Operations*
*>Network + Partition*
*>Brokerage Roles*

*Vector>Info*

*File>Vector>View/Edit*

8. To count brokerage roles, make sure that the strike network (`strike.net`) and the group partition (`strike_groups.clu`) appear at the top of their respective drop-down boxes. Next, use the command *Operations> Network + Partition> Brokerage Roles* to obtain five new vectors, one for each brokerage role. A frequency table of each partition can be obtained using the *Vector>Info* command. With the "Coordinators in N1 according to C1 (24)" vector showing in the partition drop box, select the *File>Vector>View/Edit* command, and you can see which actors scores highest in terms of coordinator roles. Repeat for each of the remaining four partitions. How do the Pajek scores compare to UCINET's? They should be the same.

*Draw*
*>Network + First Partition*
*+ First Vector*

9. Finally, visualize the network where node size reflects the number of gatekeeper roles. With the `strike` network data showing in the first network drop-down box, the `Strike_groups` partition showing in the first partition drop-down box, and the gatekeeper vector showing in the first vector drop-down box, visualize the network using the *Draw>Network + First Partition + First Vector* command. The drawing should look similar to Figure 12.

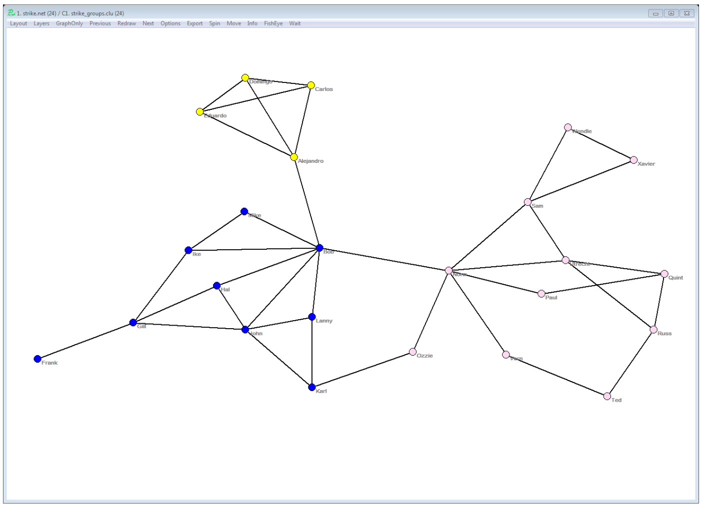


**Figure 12**: Pajek Drawing of Strike Network