XI

Link Analysis

Based on the outcome of the preprocessing stage, we can establish links between entities either by using co-occurrence information (within some lexical unit such as a document, paragraph, or sentence) or by using the semantic relationships between the entities as extracted by the information extraction module (such as family relations, employment relationship, mutual service in the army, etc.). This chapter describes the link analysis techniques that can be applied to results of the preprocessing stage (information extraction, term extraction, and text categorization).

A social network is a set of entities (e.g., people, companies, organizations, universities, countries) and a set of relationships between them (e.g., family relationships, various types of communication, business transactions, social interactions, hierarchy relationships, and shared memberships of people in organizations). Visualizing a social network as a graph enables the viewer to see patterns that were not evident before.

We begin with preliminaries from graph theory used throughout the chapter. We next describe the running example of the 9/11 hijacker's network followed by a brief description of graph layout algorithms. After the concepts of paths and cycles in graphs are presented, the chapter proceeds with a discussion of the notion of centrality and the various ways of computing it. Various algorithms for partitioning and clustering nodes inside the network are then presented followed by a brief description of finding specific patterns in networks. The chapter concludes with a presentation of three low-cost software packages for performing link analysis.

XI.1 PRELIMINARIES

We model the set of entities and relationships as a graph, and most of the operations performed on those sets are modeled as operations on graphs. The following notation is used throughout the chapter:

Let $V = \{V_1, V_2, V_3, \ldots V_n\}$ be a set of entities extracted from the documents.

A *binary relation* R over V is a subset of $V \times V$.

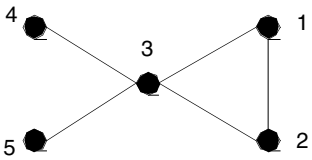

Figure XI.1. A simple undirected network with $V = \{1, 2, 3, 4, 5\}$, $R_1 = \{(1, 2), (1, 3), (2, 3), (3, 4), (3, 5)\}$ and $N = (V, R_1)$.

We use the prefix notation for relations – that is, if X and Y are related by relation R_1, then it will be denoted by $R_1(X, Y)$.

Examples of such binary relations are friendship, marriage, school mates, army mates, and so on.

A *network* N is a tuple $(V, R_1, R_2, R_3 \ldots R_m)$, where R_i $(1 \leq i \leq m)$ is a binary relation over V.

A visual representation of N is shown in Figure XI.1.

We can also describe a binary relation R using a binary matrix M, where $M_{ij} = 1$ if $R(V_i, V_j)$, and 0 otherwise. For example, the matrix that represents the relation R

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

shown in Figure XI.1 is as follows:

Each row in the matrix corresponds to the connection vector of one of the vertices. The ith row $(M_{i1}, \ldots, M_{in})$ corresponds to the connection vector of the ith vertex.

The set of edges connecting all vertices in the undirected graph is denoted by E, and $|E|$ is the number of edges in the graph. If the graph is directed, then the lines that connect the vertices are called arcs. Our focus is mostly on undirected networks and hence also on undirected graphs, and so we use vertices and edges. The network can also have weights or values attached to each of its edges. The weight function denoted $W : E \to R$ (the real numbers) is attaching a real value to each edge. If there are no values for any of the edges, then $\forall e \in E$, $W(e) = 1$.

If the relations R are not symmetric, then $G = (V, E)$ is a directed graph:

A sequence of vertices $(v_1, v_2, \ldots, v_k)$ in G is called a ***walk*** if $(v_i, v_{i+1}) \in E; i = 1 \ldots k-1$.

A sequence of vertices $(v_1, v_2, \ldots, v_k)$ in G is called a ***chain*** if $((v_i, v_{i+1}) \in E || (v_{i+1}, v_i) \in E) i = 1 \ldots k-1$.

In a walk, we care about the direction of the edge, whereas in a chain we do not.

A ***path*** is a walk in which no vertices, except maybe the initial and terminal ones, are repeated.

A walk is ***simple*** if all its edges are different.

A ***cycle*** is a simple path of at least three vertices, where $v_1 = v_k$.

The length of the path $(v_1, v_2, \ldots, v_k)$ is $k-1$.

A special type of network is a ***two-mode network***. This network contains two types of vertices, and there are edges that connect the two sets of vertices. A classic example would be a set of people and a set of events as vertices with edges connecting a person vertex to an event vertex if the person participated in the event.

If there are no self-loops in the network (i.e., a vertex can not connect to itself), then the maximal number of edges in an undirected network with n vertex is $n(n-1)/2$. Such network, in which each vertex is connected to every other vertex, is also called a ***clique***. If the number of edges is roughly the same as the number of vertices, we say that the network is ***sparse***, whereas if the network is close to being a clique we say that it is ***dense***.

We can quantify the density level of a given undirected network by using the following formula:

ND (Network Density) $= \frac{|E|}{\frac{n(n-1)}{2}} = \frac{2|E|}{n(n-1)}$
Clearly $0 \leq \text{ND} \leq 1$.
Similarly, ND for a directed network would be $\frac{|E|}{n(n-1)}$
For example ND for the network of Figure XI.1 is $\frac{2 \cdot 5}{5 \cdot 4} = 0.5$.

XI.1.1 Running Example: 9/11 Hijackers

We have collected information about the 19 9/11 hijackers from the following sources:

1. Names of the 19 hijackers, and the flights they boarded were taken from the FBI site <http://www.fbi.gov/pressrel/pressrel01/091401hj.htm> (see Table XI.1).
2. Prior connections between the hijackers are based on information collected from the *Washington Post* site given below. If there was a connection between $n \geq 2$ people, it was converted to $C(n, 2)$ symmetric binary relations between each pair of people. <http://www.washingtonpost.com/wp-srv/nation/graphics/attack/investigation_24.html.>

The undirected graph of binary relations between the hijackers is shown in Figure XI.2. The graph was drawn using Pajek dedicated freeware link analysis software (Batagelj and Mrvar 2003). More details on Pajek are presented in Section XI.7.1.

The 19 hijackers boarded 4 flights, and in Table XI.1 we can see the names of the hijackers who boarded each flight. The flight information is used when we discuss the various clustering schemes of the hijackers.

XI.2 AUTOMATIC LAYOUT OF NETWORKS

To display large networks on the screen, we need to use automatic layout algorithms. These algorithms display the graphs in an aesthetic way without any user intervention.

The most commonly used aesthetic objectives are to expose symmetries and to make the drawing as compact as possible or, alternatively, to fill the space available for

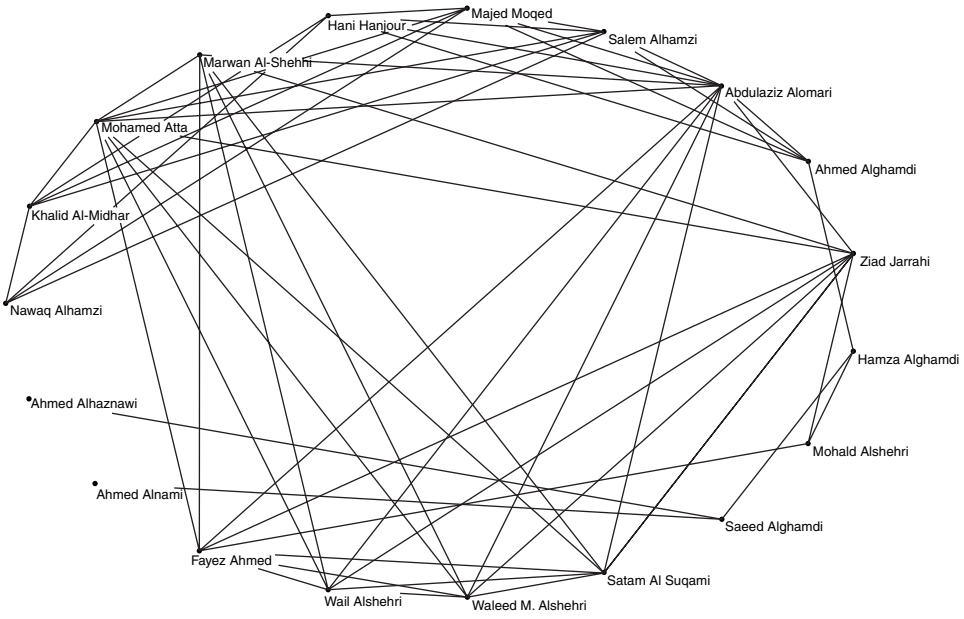

Figure XI.2. Connections between the 9/11 hijackers.

the drawing. Many of the "higher level" aesthetic criteria are implicit consequences of the

- minimized number of edge crossings,
- evenly distributed edge length,
- evenly distributed vertex positions on the graph area,
- sufficiently large vertex-edge distances, and
- sufficiently large angular resolution between edges.

XI.2.1 Force-Directed Graph Layout Algorithms

Force-directed or spring-based algorithms are among the most common automatic network layout strategies. These algorithms treat the collection of vertices and edges as a system of forces and the layout as an "equilibrium state" of the system. The edges between vertices are represented as an attractive force (each edge is simulated by

Table XI.1. The 19 Hijackers Ordered by Flights

Flight 77: Pentagon	Flight 11: WTC 1	Flight 175: WTC 2	Flight 93: PA
Khalid Al-Midhar	Satam Al Suqami	Marwan Al-Shehhi	Saeed Alghamdi
Majed Moqed	Waleed M. Alshehri	Fayez Ahmed	Ahmed Alhaznawi
Nawaq Alhamzi	Wail Alshehri	Ahmed Alghamdi	Ahmed Alnami
Salem Alhamzi	Mohamed Atta	Hamza Alghamdi	Ziad Jarrahi
Hani Hanjour	Abdulaziz Alomari	Mohald Alshehri	

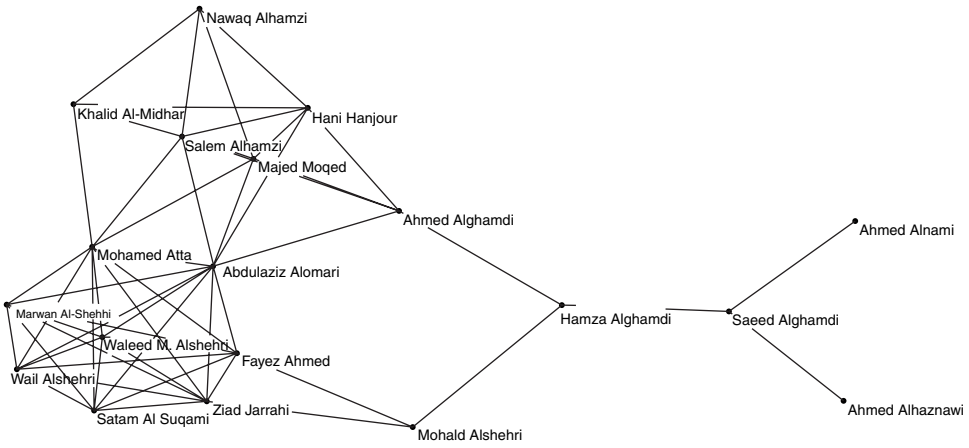

Figure XI.3. KK layout of the hijackers' graph.

a spring that pulls the vertices together), whereas distinct vertices are pushed apart by some constraint to help prevent them from being drawn at the same point. The method seeks equilibrium of these contradicting constraints. The first such algorithm was introduced by Eades (Eades 1984). Following Eades, two additional layout algorithms were introduced by Kamada and Kawai (KK) (Kamada and Kawai 1989) and Fruchterman and Reingold (FR) (Fruchterman and Reingold 1991).

Kamada and Kawai's (KK) Method

Utilizing Hooke's law, Kamada and Kawai modeled a graph as a system of springs. Every two vertices are connected by a spring whose rest length is proportional to the graph-theoretic distance between its two endpoints. Each spring's stiffness is inversely proportional to the square of its rest length. The optimization algorithm used by the KK method tries to minimize the total energy of the system and achieves faster convergence by calculating the derivatives of the force equations. One of the main benefits of the KK method is that it can be used for drawing weighted graphs if the edge lengths are proportional to their weights. The KK method proceeds by moving a single vertex at a time, choosing the "most promising" vertex – that is, the one with the maximum gradient value.

In Figure XI.3 we can see the graph shown in Figure XI.2 drawn by using the KK layout. Unlike the circular drawing of Figure XI.2 in which it is hard to see who the leaders of the groups are, we can see here that the main leaders are Mohamed Atta, Abdulaziz Alomari, and Hamza Alghamdi.

Fruchterman–Reingold (FR) Method

This method utilizes a simple heuristic approach to force-directed layout that works surprisingly well in practice. The underlying physical model roughly corresponds to electrostatic attraction in which the attractive force between connected vertices is balanced by a repulsive force between all vertices. The basic idea is just to calculate the attractive and repulsive forces at each vertex independently and to update all vertices iteratively. As in simulated annealing, the maximum displacement of each

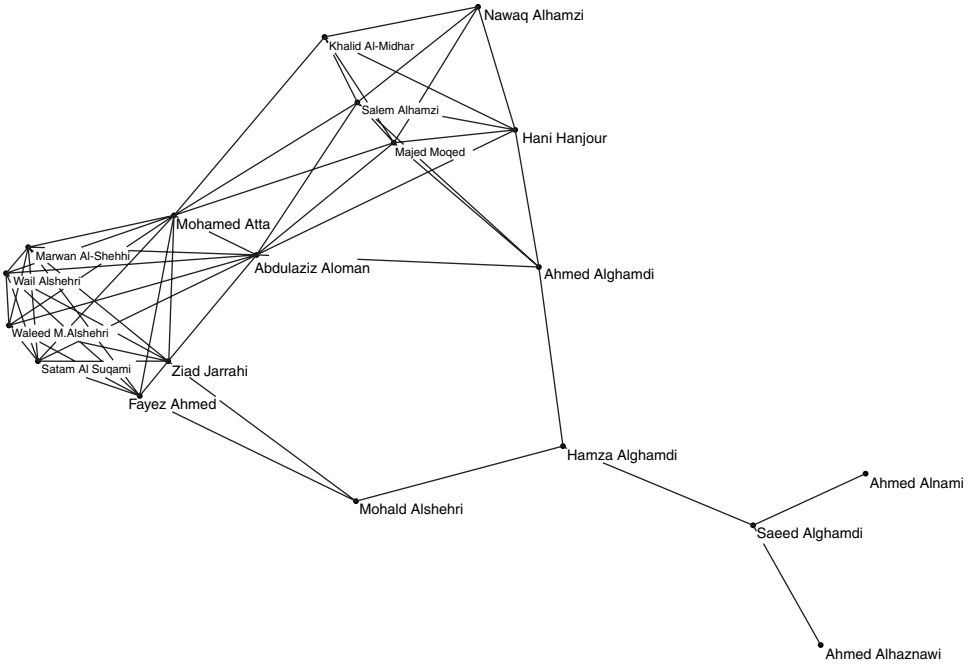

Figure XI.4. FR layout of the hijackers' graph.

vertex in any iteration is limited by a constant that is slightly decreased with each iteration. In Figure XI.4 we can see the graph shown in Figure XI.2 drawn by using the FR layout.

For both KK and FR, the relations between vertices must be expressed as distances between the vertices. For both algorithms we need to build a "dissimilarity" matrix. In the KK algorithm this matrix is constructed from geodesic distances between vertices, whereas in the FR algorithm the matrix is constructed directly from adjacencies between the vertices. Spring-based methods are very successful with small-sized graphs of up to around 100 vertices.

Simulated annealing has also been successfully applied to the layout of general undirected graphs (Davidson and Harel 1996).

Although force-directed methods are quite useful in automatically exposing most of the symmetries of the given graphs, they share several disadvantages:

- They are computationally expensive, and hence minimizing the energy function when dealing with large graphs is computationally prohibitive.
- Because all methods rely on heuristics, there is no guarantee that the "best" layout will be found.
- The methods behave as black boxes, and thus it is almost impossible to integrate additional constraints on the layout (such as fixing the positions of certain vertices or specifying the relative ordering of the vertices)
- Even when the graphs are planar it is quite possible that we will obtain edge crossings.

- The methods try to optimize just the placement of vertices and edges while ignoring the exact shape of the vertices or the possibility the vertices have labels (and hence the labels, vertices, or both may overlap each other).

XI.2.2 Drawing Large Graphs

A fast algorithm for drawing general graphs with straight edges was proposed by Harel and Koren based on the work of Hadany and Harel (Hadany and Harel 2001). Their algorithm works by producing a sequence of improved approximations of the final layout. Each approximation allows vertices to deviate from their final place by an extent limited by a decreasing constant r. As a result, the layout can be computed using increasingly coarse representations of the graph in which closely drawn vertices are collapsed into a single vertex. Each layout in the sequence is generated very rapidly by performing a local beautification on the previously generated layout. The main idea of Hadany and Harel's work is to consider a series of abstractions of the graph called coarse graphs in which the combinatorial structure is significantly simplified but important topological features are well preserved. The energy minimization is divided between these coarse graphs in such a way that globally related properties are optimized on coarser graphs, whereas locally related properties are optimized on finer graphs. As a result, the energy minimization process considers only small neighborhoods at once, yielding a quick running time.

XI.3 PATHS AND CYCLES IN GRAPHS

Given two vertices in a directed graph, we can compute the shortest path between them. The diameter of a graph is defined as the length of the longest shortest path between any two vertices in the graph. Albert et al. (Albert, Jeong, and Barabasi 1999) found that, when the Web contained around 8×108 documents, the average shortest path between any 2 pages was 19. The interpretation of the shortest path in this case is the smallest number of URL links that must be followed to navigate from one Web page to the other.

There are many kinds of paths between entities that can be traced in a dataset. In the Kevin Bacon game, for example, a player takes any actor and finds a path between the actor and Kevin Bacon that has less than six edges. For instance, Kevin Costner links to Kevin Bacon by using one direct link: Both were in *JFK*. Julia Louis-Dreyfus of TV's *Seinfeld*, however, needs two links to make a path: Julia Louis-Dreyfus was in *Christmas Vacation* (1989) with Keith MacKechnie. Keith MacKechnie was in *We Married Margo* (2000) with Kevin Bacon. You can play the game by using the following URL: <http://www.cs.virginia.edu/oracle>.

A similar idea is also used in the mathematical society and is called the Erdös number of a researcher. Paul Erdös (1913–1996) wrote hundreds of mathematical research papers in many different areas – many in collaboration with others. There is a link between any two mathematicians if they coauthored a paper. Paul Erdös is the root of the mathematical research network, and his Erdös number is 0. Erdös's coauthors have Erdös number 1. People other than Erdös who have written a joint paper with someone with Erdös number 1 but not with Erdös have Erdös number 2, and so on.

In Figure XI.5 we can see the split of the hijackers into five levels according to their distance from Mohammed Atta. The size of the little circle associated with each hijacker manifests the proximity of the hijacker to Atta; the larger the circle, the shorter the geodesic (the shortest path between two vertices in the graph) between the hijacker and Atta. There are ten hijackers who have a geodesic of size 1, four hijackers who have a geodesic of size 2, one hijacker who has a geodesic of size 3, one hijacker who has a geodesic of size 4, and finally two hijackers who have a geodesic of size 5. A much better visualization of the different degree levels can be seen in Figure XI.6. The diagram was produced by using Pajek's drawing module and selecting Layers | in *y* direction. The various levels are coded by the distance from the nodes with the highest degree. Connections are shown just between entities of different levels.

XI.4 CENTRALITY

The notion of centrality enables us to identify the main and most powerful actors within a social network. Those actors should get special attention when monitoring the behavior of the network.

Centrality is a structural attribute of vertices in a network; it has nothing to do with the features of the actual objects represented by the vertices of the network (i.e., if it is a network of people, their nationality, title, or any physical feature). When dealing with directed networks we use the term *prestige*. There are two types of prestige; the one defined on outgoing arcs is called *influence*, whereas the one defined on incoming arcs is called *support*. Because most of our networks are based on co-occurrence of entities in the same lexical unit, we will confine our attention to undirected networks and use the term centrality. The different measures of centrality we will present can be adapted easily for directed networks and measure influence or support.

Five major definitions are used for centrality: degree centrality, closeness centrality, betweeness centrality, eigenvector centrality, and power centrality. We discuss these in the next several sections.

XI.4.1 Degree Centrality

If the graph is undirected, then the degree of a vertex $v \in V$ is the number of other vertices that are directly connected to it.

Definition: $\text{degree}(v) = |\{(v1, v2) \in E \mid v1 = v \text{ or } v2 = v\}|$

If the graph is directed, then we can talk about in-degree or out-degree. An edge $(v1, v2) \in E$ in the directed graph is leading from vertex $v1$ to $v2$.

$\text{In-degree}(v) = |\{(v1, v) \in E\}|$

$\text{Out-degree}(v) = |\{(v, v2) \in E\}|$

If the graph represents a social network, then clearly people who have more connections to other people can be more influential and can utilize more of the resources of the network as a whole. Such people are often mediators and dealmakers in exchanges among others and are able to benefit from this brokerage.

When dealing with undirected connections, people differ from one another only in how many connections they have. In contrast, when the connections are directed,

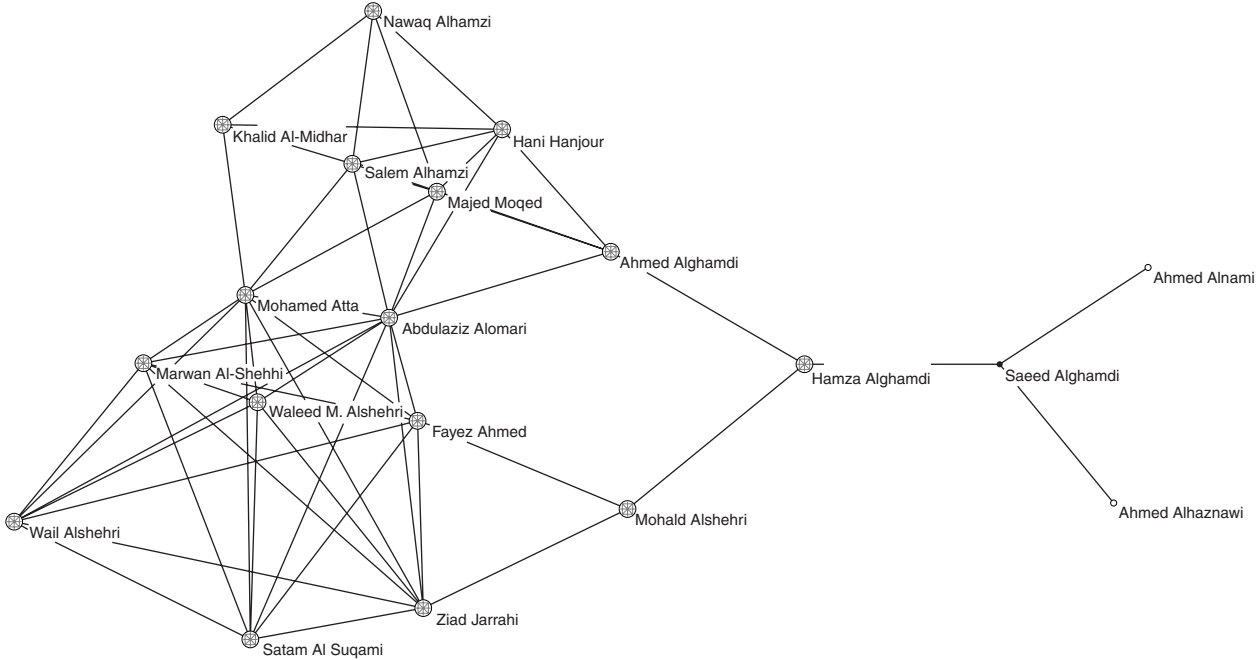

Figure XI.5. Computing the shortest distance between Atta and all other 18 hijackers.

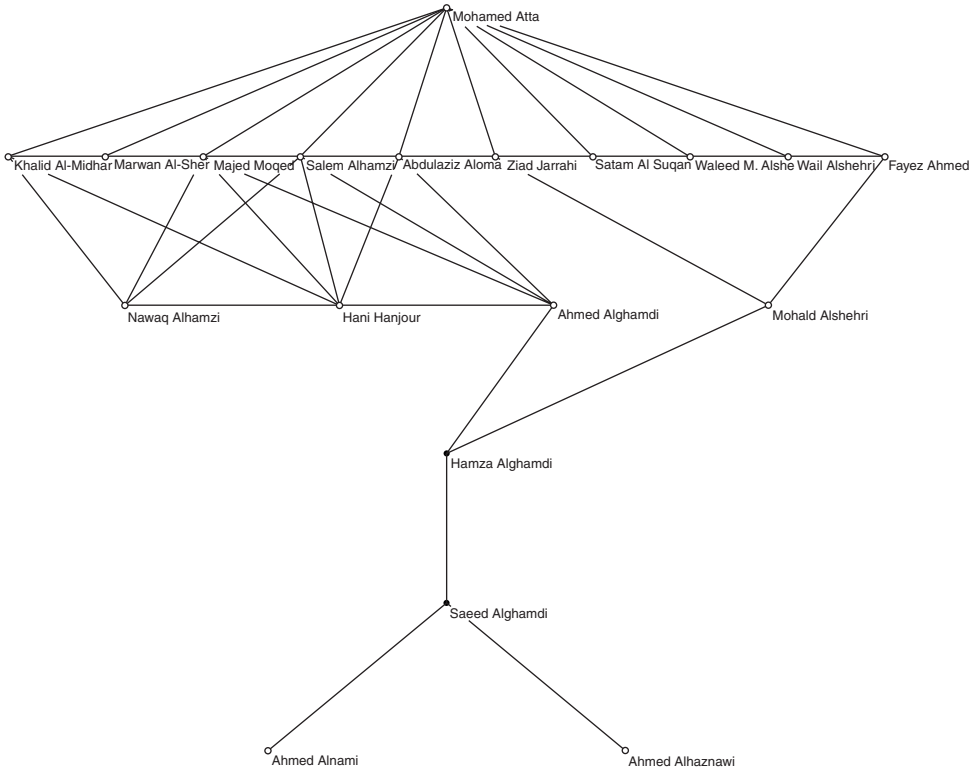

Figure XI.6. Layered display of the geodesic distance between Atta and the other hijackers.

it is important to distinguish centrality based on in-degree from centrality based on out-degree. If a person has a high in-degree, we say that this person is *prominent* and has high *prestige*. Many people seek direct connections to him or her, indicating that persons's importance. People who have high out-degree are people who are able to interact with many others and possibly spread their ideas. Such people are said to be *influential*. In Table XI.2, we can see the hijackers sorted in decreasing order of their (undirected) degree measures. We can see that Mohamed Atta and Abdulaziz Alomari have the highest degree.

XI.4.2 Closeness Centrality

Degree centrality measures might be criticized because they take into account only the direct connections that an entity has rather than indirect connections to all other entities. One entity might be directly connected to a large number of entities that might be rather isolated from the network. Such an entity is central only in a local neighborhood of the network.

To solve the shortcomings of the degree measure, we can utilize the closeness centrality. This measure is based on the calculation of the geodesic distance between the entity and all other entities in the network. We can either use directed or undirected geodesic distances between the entities. In our current example, we have decided to look at undirected connections. The sum of these geodesic distances for each entity

Table XI.2. All Degree Measures of the Hijackers

Name	Degree
Mohamed Atta	11
Abdulaziz Alomari	11
Ziad Jarrahi	9
Fayez Ahmed	8
Waleed M. Alshehri	7
Wail Alshehri	7
Satam Al Suqami	7
Salem Alhamzi	7
Marwan Al-Shehhi	7
Majed Moqed	7
Khalid Al-Midhar	6
Hani Hanjour	6
Nawaq Alhamzi	5
Ahmed Alghamdi	5
Saeed Alghamdi	3
Mohald Alshehri	3
Hamza Alghamdi	3
Ahmed Alnami	1
Ahmed Alhaznawi	1

is the "farness" of the entity from all other entities. We can convert this into a measure of closeness centrality by taking its reciprocal. We can normalize the closeness measure by dividing it by the closeness measure of the most central entity.

Formally, let $d(v_1, v_2)$ = the minimal distance between v_1 and v_2 – that is, the minimal number of vertices we need to pass on the way from v_1 to v_2.

The closeness centrality of vertex v_i is defined as $C_i = \frac{|V|-1}{\sum_{j \neq i} d(v_i, v_j)}$. This is the reciprocal of the average geodesic distance between v_i and any other vertex in the network. In Table XI.3, we can see the hijackers sorted in decreasing order of their closeness.

XI.4.3 Betweeness Centrality

Betweeness centrality measures the effectiveness in which a vertex connects the various parts of the network. Entities that are on many geodesic paths between other pairs of entities are more powerful because they control the flow of information between the pairs. That is, the more other entities depend on a certain entity to make connections, the more power this entity has. If, however, two entities are connected by more than one geodesic path and a given entity is not on all of them, it loses some power. If we add up, for each entity, the proportion of times this entity is "between" other entities for transmission of information, we obtain the betweeness centrality of that entity. We can normalize this measure by dividing it by the maximum possible betweeness that an entity could have had (which is the number of possible pairs of entities for which the entity is on every geodesic between them $= \frac{(|V|-1)(|V|-2)}{2}$).

Table XI.3. Closeness Measures of the Hijackers

Name	Closeness
Abdulaziz Alomari	0.6
Ahmed Alghamdi	0.5454545
Ziad Jarrahi	0.5294118
Fayez Ahmed	0.5294118
Mohamed Atta	0.5142857
Majed Moqed	0.5142857
Salem Alhamzi	0.5142857
Hani Hanjour	0.5
Marwan Al Shehhi	0.4615385
Satam Al Suqami	0.4615385
Waleed M. Alshehri	0.4615385
Wail Alshehri	0.4615385
Hamza Alghamdi	0.45
Khalid Al Midhar	0.4390244
Mohald Alshehri	0.4390244
Nawaq Alhamzi	0.3673469
Saeed Alghamdi	0.3396226
Ahmed Alnami	0.2571429
Ahmed Alhaznawi	0.2571429

Formally,
g_{jk} = the number of geodetic paths that connect v_j with v_k;
$g_{jk}(v_i)$ = the number of geodetic paths that connect v_j with v_k and pass via v_i.

$$B_i = \sum_{j<k} \frac{g_{jk}(v_i)}{g_{jk}}$$

$$NB_i = \frac{2B_i}{(|V|-1)(|V|-2)}$$

In Table XI.4, we can see the hijackers sorted in decreasing order of their between measures.

XI.4.4 Eigenvector Centrality

The main idea behind eigenvector centrality is that entities receiving many communications from other well-connected entities will be better and more valuable sources of information and hence be considered central. The eigenvector centrality scores correspond to the values of the principal eigenvector of the adjacency matrix M.

Formally, the vector $\mathbf{v}$ satisfies the equation $\lambda\mathbf{v} = M\mathbf{v}$, where λ is the corresponding eigenvalue and M is the adjacency matrix.

The score of each vertex is proportional to the sum of the centralities of neighboring vertices. Intuitively, vertices with high eigenvector centrality scores are connected to many other vertices with high scores, which are, in turn, connected to many other vertices, and this continues recursively. Clearly, the highest score will be obtained

Table XI.4. Betweeness Measures of the Hijackers

Name	Betweeness (B_i)
Hamza Alghamdi	0.3059446
Saeed Alghamdi	0.2156863
Ahmed Alghamdi	0.210084
Abdulaziz Alomari	0.1848669
Mohald Alshehri	0.1350763
Mohamed Atta	0.1224783
Ziad Jarrahi	0.0807656
Fayez Ahmed	0.0686275
Majed Moqed	0.0483901
Salem Alhamzi	0.0483901
Hani Hanjour	0.0317955
Khalid Al-Midhar	0.0184832
Nawaq Alhamzi	0
Marwan Al-Shehhi	0
Satam Al Suqami	0
Waleed M. Alshehri	0
Wail Alshehri	0
Ahmed Alnami	0
Ahmed Alhaznawi	0

by vertices that are members of large cliques or large p-cliques. In Table XI.5 we can see that the members of the big clique (with eight members) are those that got the highest scores. Atta and Al-Shehhi got much higher scores than all the other hijackers mainly because the connection between them is so strong. They were also the pilots of the planes going into WTC1 and WTC2 and are believed to have been the leaders of the hijackers.

XI.4.5 Power Centrality

Power centrality was introduced by Bonacich. Given an adjacency matrix M, the power centrality of vertex i (denoted c_i) is given by

$$c_i = \sum_{j \neq i} M_{ij}(\alpha + \beta \cdot c_j),$$

where α is used to normalize the score (the normalization parameter is automatically selected so that the sum of squares of the vertices's centralities is equal to the number of vertices in the network) and β is an attenuation factor that controls the effect that the power centralities of the neighboring vertices should have on the power centrality of the vertex.

As in the eigenvector centrality, the power centrality of each vertex is determined by the centrality of the vertices it is connected to. By specifying positive or negative values to β, the user can control whether a vertex's being connected to powerful vertices should have a positive effect on its score or a negative effect. The rationale for specifying a positive β is that, if you are connected to powerful colleagues it makes you more powerful. On the other hand, the rationale for a negative β is

Table XI.5. Eigenvector Centrality Scores of the Hijackers

Name	E1
Mohamed Atta	0.518
Marwan Al-Shehhi	0.489
Abdulaziz Alomari	0.296
Ziad Jarrahi	0.246
Fayez Ahmed	0.246
Satam Al Suqami	0.241
Waleed M. Alshehri	0.241
Wail Alshehri	0.241
Salem Alhamzi	0.179
Majed Moqed	0.165
Hani Hanjour	0.151
Khalid Al-Midhar	0.114
Ahmed Alghamdi	0.085
Nawaq Alhamzi	0.064
Mohald Alshehri	0.054
Hamza Alghamdi	0.015
Saeed Alghamdi	0.002
Ahmed Alnami	0
Ahmed Alhaznawi	0

that powerful colleagues have many connections and hence are not controlled by you, whereas isolated colleagues have no other sources of information and hence are largely controlled by you. In Table XI.6, we can see the hijackers sorted in decreasing order of their power measure.

XI.4.6 Network Centralization

In addition to the individual vertex centralization measures, we can assign a number between 0 and 1 that will signal the whole network's level of centralization. The network centralization measures are computed based on the centralization values of the network's vertices; hence, we will have for each type of individual centralization measure an associated network centralization measure. A network structured like a circle will have a network centralization value of 0 (because all vertices have the same centralization value), whereas a network structured like a star will have a network centralization value of 1. We now provide some of the formulas for the different network centralization measures.

Degree

$$Degree^*(V) = \mathrm{Max}_{v \in V} Degree(v)$$

$$\mathrm{NET}_{\mathrm{Degree}} = \frac{\sum_{v \in V} Degree^*(V) - Degree(v)}{(n-1) * (n-2)}$$

Table XI.6. Power Centrality for the Hijackers Graph

	Power : $\beta = 0.99$	**Power : $\beta = -0.99$**
Mohamed Atta	2.254	2.214
Marwan Al-Shehhi	2.121	0.969
Abdulaziz Alomari	1.296	1.494
Ziad Jarrahi	1.07	1.087
Fayez Ahmed	1.07	1.087
Satam Al Suqami	1.047	0.861
Waleed M. Alshehri	1.047	0.861
Wail Alshehri	1.047	0.861
Salem Alhamzi	0.795	1.153
Majed Moqed	0.73	1.029
Hani Hanjour	0.673	1.334
Khalid Al-Midhar	0.503	0.596
Ahmed Alghamdi	0.38	0.672
Nawaq Alhamzi	0.288	0.574
Mohald Alshehri	0.236	0.467
Hamza Alghamdi	0.07	0.566
Saeed Alghamdi	0.012	0.656
Ahmed Alnami	0.003	0.183
Ahmed Alhaznawi	0.003	0.183

Clearly, if we have a circle, all vertices have a degree of 2; hence, $\mathrm{NET}_{\mathrm{Degree}} = 0$. If we have a star of n nodes (one node in the middle), then that node will have a degree of $n-1$, and all other nodes will have a degree of 1; hence,

$$\mathrm{NET}_{\mathrm{Degree}} = \frac{\sum_{v \in V \setminus v^*} (n-1) - 1}{(n-1)(n-2)} = \frac{(n-1)(n-2)}{(n-1)(n-2)} = 1.$$

For the hijackers' graph, $\mathrm{NET}_{\mathrm{Degree}} = 0.31$

Betweenness

$$NB^*(V) = \mathrm{Max}_{v \in V} NB(v)$$

$$\mathrm{NET}_{\mathrm{Bet}} = \frac{\sum_{v \in V} NB^*(V) - NB(v)}{(n-1)}$$

For the hijackers' network, $\mathrm{NET}_{\mathrm{Bet}} = 0.24$

XI.4.7 Summary Diagram

Figure XI.7 presents a summary diagram of the different centrality measures as they are applied to the hijacker's network. We marked by solid arrows the hijackers who got the highest value for the various centrality measures and by dashed arrows

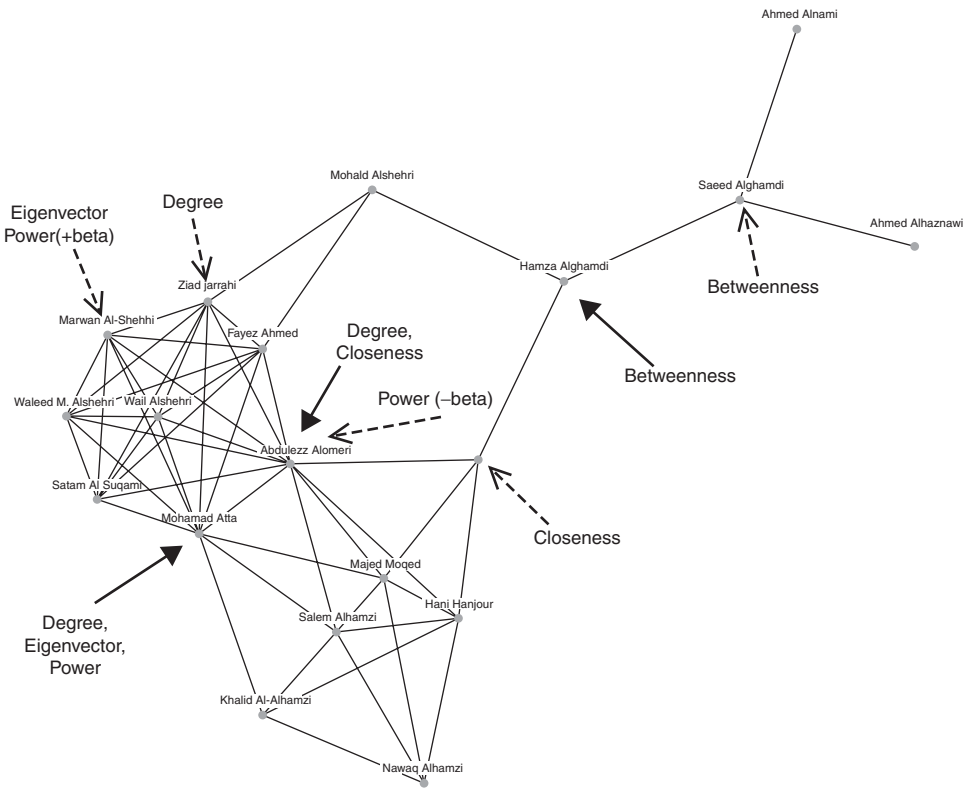

Figure XI.7. Summary diagram of centrality measures (solid arrows point to highest value; dashed arrows point to second largest (done using Netminer (Cyram 2004)).

the runners-up. We can see for instance that Atta has the highest value for degree centrality, eigenvector centrality, and power centrality, whereas Alomari has the highest value for degree centrality (tied with Atta) and closeness centrality and is the runner-up for power centrality (with a negative beta).

On the basis of our experience the most important centrality measures are power and eigenvector (which are typically in agreement). Closeness and, even more so, betweeness centrality signal the people who are crucial in securing fast communication between the different parts of the network.

XI.5 PARTITIONING OF NETWORKS

Often we obtain networks that contain hundreds and even thousands of vertices. To analyze the network effectively it is crucial to partition it into smaller subgraphs.

We present three methods below for taking a network and partitioning it into clusters. The first method is based on core computation, the second on classic graph algorithms for finding strong and weak components and biconnected components, and the third on block modeling.

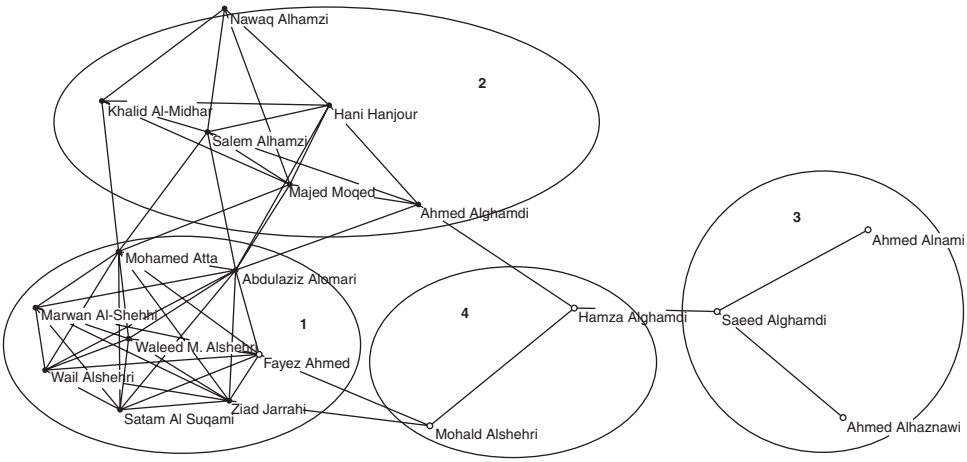

Figure XI.8. Core partitioning of the hijackers' graph.

XI.5.1 Cores

Definition: Let $G = (V, E)$ be a graph. A subgraph $S = (W, E \mid W)$ induced by the vertex set W is a *k-core* or a *core* of order k iff $\forall n \in W : \deg_S(n) \geq k$ and S is a maximal with respect to this property. The *main core* is the core of highest order. The *core number* of vertex n is the highest order of a core that contains this vertex.

Algorithm for finding the main core

Given a graph $G = (V, E)$, delete all vertices n and edges attached to them such that $\deg_S(n) < k$ and repeat until no vertices or edges can be deleted. The subgraph that remains after the iterative deletion is a core of order k. If an empty graph results, we know that no core of order k exists. We can perform a simple log $|V|$ search for the order of the main core. After the main core is discovered, we can remove these vertices and the associated edges from the graph and search again for the next core in the reduced graph. The process will terminate when an empty graph is reached. In Figure XI.8, we can see the cores that were discovered in the hijacker's graph. When a core was discovered, it was deleted from the graph and the search for the biggest core in the remaining graph started again.

We can see that four cores were found. The main core contains eight nodes and is of order seven (each vertex is connected to all other seven vertices), the second largest core has six vertices in it and an order of 3, the third core has three vertices and an order of 1, and the fourth one has two vertices and an order of 1.

We then used the shrinking option of Pajek (Operations | Shrink Network | Partition) to obtain a schematic view of the network based on the core partition. Each core is reduced to the name of its first member. For instance, the first member in the core marked 1 is Mohammed Atta, and hence the core is reduced to him. If there is at least one edge between the vertices of any two cores, then we will have an edge between the associated vertices in the reduced graph. The reduced graph,

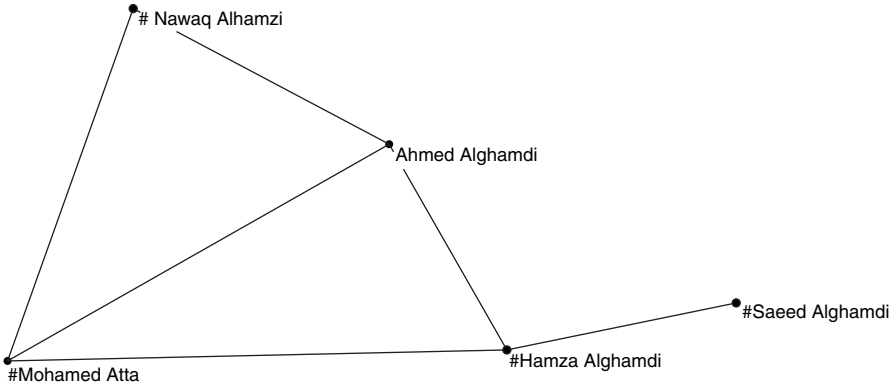

Figure XI.9. Shrinking the hijackers' graphs based on the core partition.

which is based on the shrinking of the core partitioning, is shown in Figure XI.9. A layered display of the cores is shown in Figure XI.10.

Alternatively, we can use a layered display of the network to see the different cores and the relations between them better. Each core is shown in a different *y*-level. This representation mainly enables us to focus on the intraconnections between the cores.

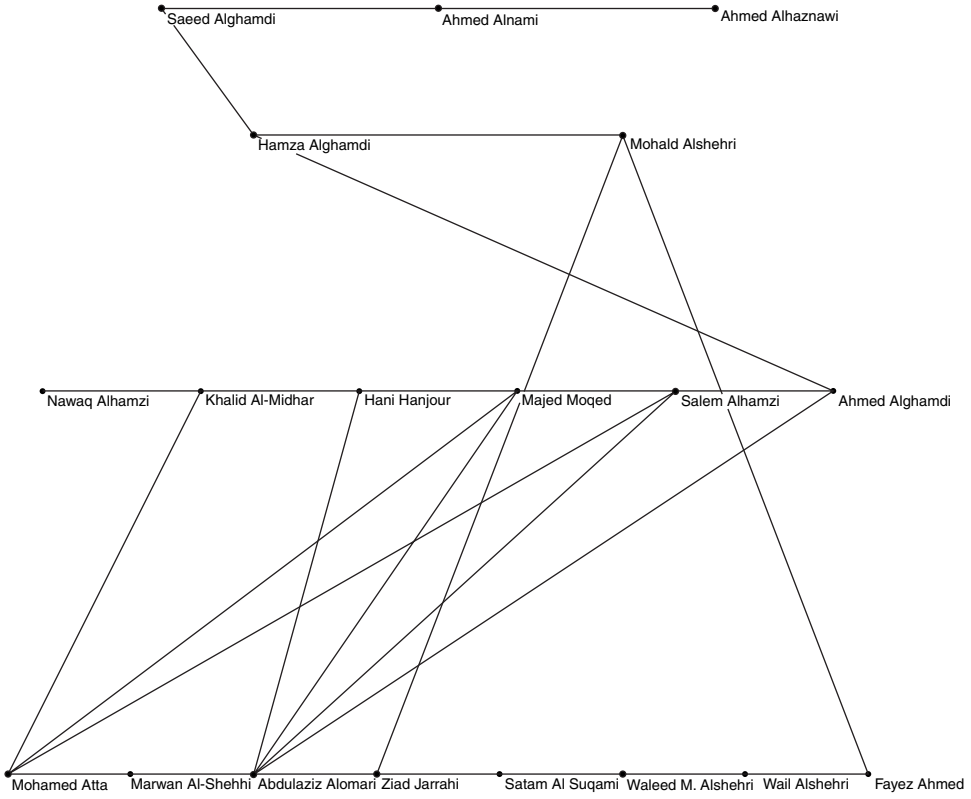

Figure XI.10. Layered display of the cores.

XI.5.2 Classic Graph Analysis Algorithms

Another way of partitioning a network is to use classic graph algorithms such as weak and strong component analysis and identification of bidirectional components.

Strong and Weak Components

Whether the network is directed or undirected is crucial to the component analysis of the network. A subset of vertices is called a strongly connected component if there is at least one walk from any vertex to any other vertex in the subset. A subset of vertices is called a weakly connected component if there exists at least one chain from any vertex to any other vertex in the subset.

A subset of vertices is called a biconnected component if there exist at least two chains from any vertex to any other vertex in the subset, where the chains share no common vertex.

Biconnected Components and Articulation Points

A vertex d of the network is an articulation point of the network if there exist two additional vertices b and c so that every chain between b and c also includes d. It follows that vertex d is an articulation point if the removal of d from the network disconnects it. A network is termed biconnected if, for every triple of vertices d, b, and c, there is a chain between b and c that does not include d. This means that a biconnected network remain connected even after any vertex from it is removed. There are no articulation points in a biconnected network. Articulation points expose weaknesses of networks, and elimination of articulation points will cause the network to be fragmented. The articulation points of the hijackers' graph are shown in Figure XI.11.

XI.5.3 Equivalence between Entities

Given a network of entities, we are often interested in measuring the similarity between the entities based on their interaction with other entities in the network. This

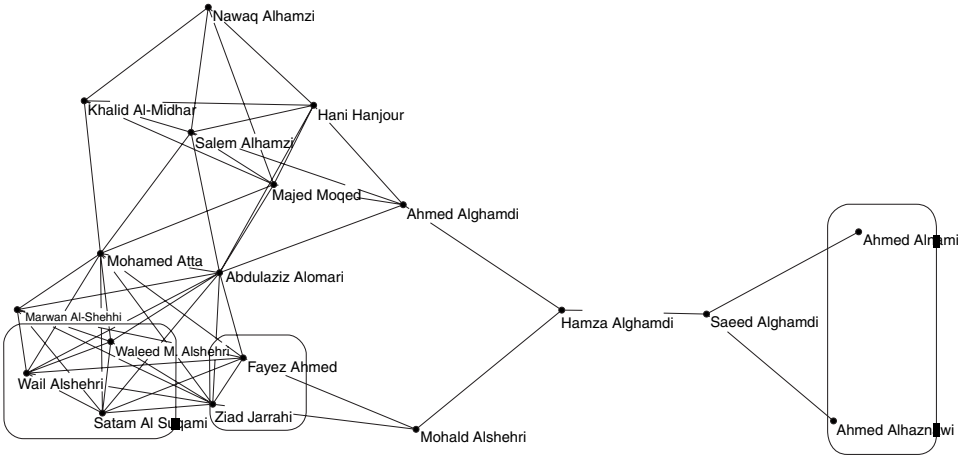

Figure XI.11. Articulation points of the hijackers' network (the number above the arrow signals the number of components that will result after removing the articulation point).

section formalizes this notion of similarity between entities and provides examples of how to find similar entities and how to use the similarity measure to cluster the entities.

Structural Equivalence

Two entities are said to be exactly structurally equivalent if they have the same relationships to all other entities. If A is "structurally equivalent" to B, then these two entities are "substitutable." Typically, we will not be able to find entities that are exactly structurally equivalent; hence, we are interested in calculating the degree of structural equivalence between entities. This measure of distance makes it possible to perform hierarchical clustering of the entities in our network.

We present two formal definitions for structural equivalence. Both are based on the connection vectors of each of the entities. The first definition is based on the Euclidian distance between the connection vectors and other on the number of exact matches between the elements of the vectors.

$$\text{EDis}(V_i, V_j) = \sqrt{\sum_k (M_{ik} - M_{jk})^2}$$

$$\text{Match}(V_i, V_j) = \frac{\sum_{k=1}^{n} \text{eq}(M_{ik}, M_{jk})}{n}, \quad \text{where} \quad \text{eq}(a, b) = \begin{cases} 1 & a = b \\ 0 & \text{otherwise} \end{cases}$$

Regular Equivalence

Two entities are said to be regularly equivalent if they have an identical profile of connections with other entities that are also regularly equivalent. In order to establish regular equivalence, we need to classify the entities into semantic sets such that each set contains entities with a common role. An example would be the sets of surgeons, nurses, and anesthesiologists. Let us assume that each surgeon is related to a set of three nurses and one anesthesiologist. We say that two such surgeons are regularly equivalent (and so are the nurses and the anesthesiologist) – that is, they perform the same function in the network.

Entities that are "structurally equivalent" are also "regularly equivalent." However, entities that are "regularly equivalent" do not have to be "structurally equivalent." It is much easier to examine if two entities are structurally equivalent because there is a simple algorithm for finding EDis and Match. It is much harder to establish if two entities are regularly equivalent because we need to create a taxonomy of semantic categories on top of the entities. In Figure XI.12 we can see two pairs of people and one triplet that are structurally equivalent. In Table XI.7 we can see the EDis computed for each pair of entities. Entities that are structurally equivalent will have an EDis of 0. For instance, Waleed M. Alshehri and Wail Alshehri are structurally equivalent, and hence their EDis is 0. Based on this table, we were able to use a hierarchical clustering algorithm (via the UCINET software package; see Section XI.7.2) and generate the dendogram shown in Figure XI.13. People who are very close in the dendogram are similar structurally (i.e, they have low EDis), whereas people who are far away in the dendogram are different structurally.

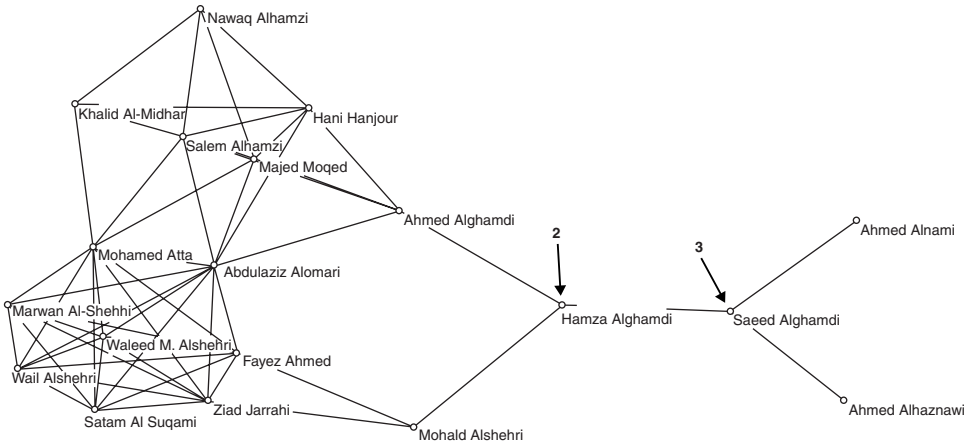

Figure XI.12. Structural equivalences in the hijackers' graph.

XI.5.4 Block Modeling

Block modeling is an analysis technique for finding clusters of vertices that behave in a similar way. Block modeling is based on the notions of structural and regular equivalence between vertices and as such is far more sensitive to the interconnections between vertices than the standard clustering techniques introduced before. Block modeling was introduced by Borgatti and Everett (1993). The technique is fairly general and can use a variety of equivalence relations between the vertices. The

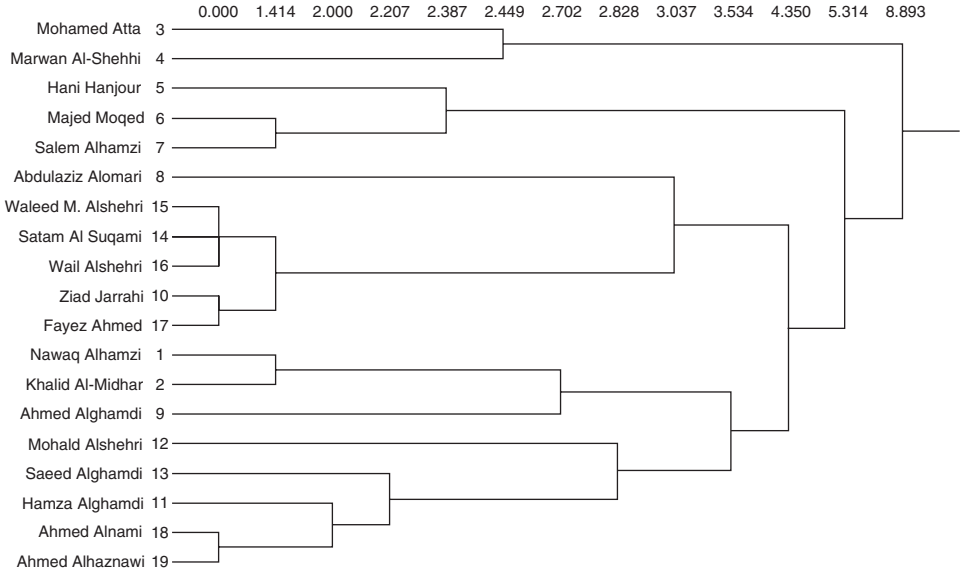

Figure XI.13. Clustering-based structural equivalence between the hijackers (we can see that {15,14,16} as well as {10,17} and {18,19} are structural equivalence classes).

Table XI.7. Euclidian Distance (Edis) between Each Pair of Entities

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1 Nawaq Alhamzi	0.0	1.4	9.3	9.6	3.7	2.8	3.7	4.2	2.4	4.9	3.7	3.7	3.7	4.7	4.7	4.7	4.9	3.2	3.2
2 Khalid Al-Midhar	1.4	0.0	9.4	8.4	4.0	2.4	3.5	4.0	2.8	4.7	4.0	4.0	4.0	4.5	4.5	4.5	4.7	3.5	3.5
3 Mohamed Atta	9.3	9.4	0.0	2.4	9.8	9.7	10.2	7.5	9.4	7.6	9.8	9.4	9.8	7.5	7.5	7.5	7.6	9.6	9.6
4 Marwan Al-Shehhi	9.6	8.4	2.4	0.0	10.7	8.7	9.3	7.6	9.5	7.2	9.5	9.1	9.5	7.1	7.1	7.1	7.2	9.3	9.3
5 Hani Hanjour	3.7	4.0	9.8	10.7	0.0	3.2	2.0	5.3	4.0	6.8	6.0	6.3	6.3	6.6	6.6	6.6	6.8	6.0	6.0
6 Majed Moqed	2.8	2.4	9.7	8.7	3.2	0.0	1.4	4.2	3.2	5.3	4.7	5.1	5.1	5.1	5.1	5.1	5.3	4.7	4.7
7 Salem Alhamzi	3.7	3.5	10.2	9.3	2.0	1.4	0.0	4.9	4.0	6.2	5.7	6.0	6.0	6.0	6.0	6.0	6.2	5.7	5.7
8 Abdulaziz Alomari	4.2	4.0	7.5	7.6	5.3	4.2	4.9	0.0	4.0	3.2	4.9	4.5	5.3	2.8	2.8	2.8	3.2	4.9	4.9
9 Ahmed Alghamdi	2.4	2.8	9.4	9.5	4.0	3.2	4.0	4.0	0.0	4.7	3.5	3.5	3.5	4.5	4.5	4.5	4.7	3.5	3.5
10 Ziad Jarrahi	4.9	4.7	7.6	7.2	6.8	5.3	6.2	3.2	4.7	0.0	4.2	3.7	4.7	1.4	1.4	1.4	0.0	4.2	4.2
11 Hamza Alghamdi	3.7	4.0	9.8	9.5	6.0	4.7	5.7	4.9	3.5	4.2	0.0	2.8	2.8	4.5	4.5	4.5	4.2	2.0	2.0
12 Mohald Alshehri	3.7	4.0	9.4	9.1	6.3	5.1	6.0	4.5	3.5	3.7	2.8	0.0	2.8	3.5	3.5	3.5	3.7	2.8	2.8
13 Saeed Alghamdi	3.7	4.0	9.8	9.5	6.3	5.1	6.0	5.3	3.5	4.7	2.8	2.8	0.0	4.5	4.5	4.5	4.7	2.0	2.0
14 Satam Al Suqami	4.7	4.5	7.5	7.1	6.6	5.1	6.0	2.8	4.5	1.4	4.5	3.5	4.5	0.0	0.0	0.0	1.4	4.0	4.0
15 Waleed M. Alshehri	4.7	4.5	7.5	7.1	6.6	5.1	6.0	2.8	4.5	1.4	4.5	3.5	4.5	0.0	0.0	0.0	1.4	4.0	4.0
16 Wail Alshehri	4.7	4.5	7.5	7.1	6.6	5.1	6.0	2.8	4.5	1.4	4.5	3.5	4.5	0.0	0.0	0.0	1.4	4.0	4.0
17 Fayez Ahmed	4.9	4.7	7.6	7.2	6.8	5.3	6.2	3.2	4.7	0.0	4.2	3.7	4.7	1.4	1.4	1.4	0.0	4.2	4.2
18 Ahmed Alnami	3.2	3.5	9.6	9.3	6.0	4.7	5.7	4.9	3.5	4.2	2.0	2.8	2.0	4.0	4.0	4.0	4.2	0.0	0.0
19 Ahmed Alhaznawi	3.2	3.5	9.6	9.3	6.0	4.7	5.7	4.9	3.5	4.2	2.0	2.8	2.0	4.0	4.0	4.0	4.2	0.0	0.0

general block modeling problem is composed of two subproblems:

1. Performing clustering of the vertices; each cluster serves as a block.
2. Calculating the links (and their associated value) between the blocks.

Formal Notations

Given two clusters C_1 and C_2, $L(C_1, C_2)$ is the set of edges that connect vertices in C_1 to vertices in C_2. Formally, $L(C_1, C_2) = \{(x, y) | (x, y) \in E, x \in C_1, y \in C_2\}$.

Because there are many ways to partition our vertices into clusters, we will introduce an optimization criterion that will help pick the optimal clustering scheme.

Before defining the problem formally, we will introduce a few predicates on the connections between two clusters. Visualizations of some of these predicates are shown in Figure XI.14

Predicate name	Formula and Acronym	Explanation
Null	$Null(C_1, C_2) \equiv \forall x \in C_1, \forall y \in C_2, (x, y) \notin E$	No connection at all between the clusters
Com (Complete)	$Com(C_1, C_2) \equiv \forall x \in C_1, \forall y (y \neq x) \in C_2, (x, y) \in E$	Full connection between the clusters
Row Regular	$Rreg(C_1, C_2) \equiv \forall x \in C_1, \exists y \in C_2, (x, y) \in E$	Each vertex in the first cluster is connected to at least one vertex in the second cluster.
Column Regular	$Creg(C_1, C_2) \equiv \forall y \in C_2, \exists x \in C_1, (x, y) \in E$	Each vertex in the second cluster is connected to at least one vertex in the first cluster.
Regular	$Reg(C_1, C_2) \equiv Rreg(C_1, C_2) \wedge Creg(C_1, C_2)$	All vertices in both clusters must have at least one vertex in the other cluster to which they are connected.
Row Dominant	$Rdom(C_1, C_2) \equiv \exists x \in C_1, \forall y (y \neq x) \in C_2, (x, y) \in E$	There is at least one vertex in the first cluster that is connected to all the vertices in the second cluster.
Column Dominant	$Cdom(C_1, C_2) \equiv \exists y \in C_2, \forall x (x \neq y) \in C_1, (x, y) \in E$	There is at least one vertex in the second cluster that is connected to all the vertices in the first cluster.
Row Functional	$Rfun(C_1, C_2) \equiv \forall y \in C_2, \exists \text{ single } x \in C_1, (x, y) \in E$	All vertices in the second cluster are connected to exactly one vertex in the first cluster.
Column Functional	$Cfun(C_1, C_2) \equiv \forall x \in C_1, \exists \text{ single } y \in C_2, (x, y) \in E$	All vertices in the first cluster are connected to exactly one vertex in the second cluster.

Formally, a block model of graph $G = (V, E)$ is a tuple $M = (U, K, T, Q, \pi, \alpha)$, where

- U is the set of clusters that we get by partitioning V.
- K is the set of connections between elements of U, $K \subseteq U \times U$.

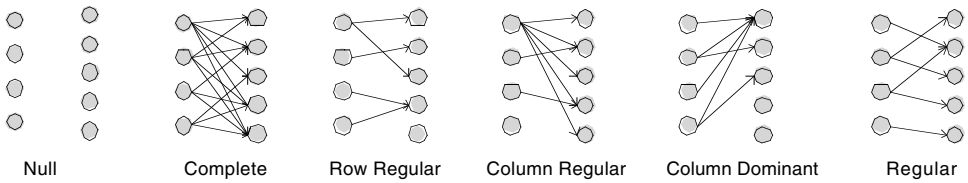

Figure XI.14. Visualization of some of the predicates on the connections between clusters.

- T is a set of predicates that describe the connections between the clusters.
- π is a mapping function between the cluster's connections and the predicates – $\pi : K \rightarrow T\backslash\{Null\}$.
- Q is a set of averaging rules enabling us to compute the strength of the connection between any two clusters.
- α is a mapping function from the connection between the clusters to the averaging rules – $\alpha : K \rightarrow Q$

Averaging rules (Q)

Listed below are a few options for giving a value to a connection between two clusters C_1 and C_2 based on the weights assigned to edges in $L(C_1,C_2)$.

$$Ave(C_1, C_2) = \frac{\sum_{e \in L(C_1,C_2)} w(e)}{|L(C_1, C_2)|}$$

$$Max(C_1, C_2) = \max_{e \in L(C_1,C_2)} w(e)$$

$$Med(C_1, C_2) = \text{median}_{e \in L(C_1,C_2)} w(e)$$

$$Ave - row(C_1, C_2) = \frac{\sum_{e \in L(C_1,C_2)} w(e)}{|C_1|}$$

$$Ave - col(C_1, C_2) = \frac{\sum_{e \in L(C_1,C_2)} w(e)}{|C_2|}$$

Finding the Best Block Model

We can define a quality measure for any clustering and on the basis of that measure seek the clustering that will yield the ultimate block model of the network. First, we compute the quality of any clustering of the vertices.

We start with a fundamental problem. Given two clusters C_1 and C_2 and a predicate $t \in T$, how can we find the deviation of $L(C_1, C_2)$ that satisfies t? This deviation will be denoted by $\delta(C_1, C_2, t)$. The approach here is to measure the number of 1's missing in the matrix $C_1 \times C_2$ from a perfect matrix that satisfies t. Clearly, $\delta(C_1, C_2, t) = 0$ iff $t(C_1, C_2)$ is true.

For example, if the matrix that represents $L(C_1, C_2)$ is

$$\begin{matrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1, \end{matrix}$$

then, because there are four 0's in the matrix, $\delta(C_1, C_2, \text{Com}) = 4$.

If we assign some weight to each predicate t, we can introduce the notion of error with respect to two clusters and a predicate $\varepsilon(C_1, C_2, t) = w(t) \cdot \delta(C_1, C_2, t)$. This notion can now be extended to the error over a set of predicates. We seek the minimal error from all individual errors on the members of the predicate set. This will also determine which predicate should selected to be the value of $\pi(C_1, C_2)$.

$$\varepsilon(C_1, C_2, T) = \min_{t \in T} \varepsilon(C_1, C_2, t)$$

$$\pi(C_1, C_2, T) = \arg\min_{t \in T} \varepsilon(C_1, C_2, t)$$

Now that the error for a pair of clusters has been defined, we can define the total error for the complete clustering. Basically, it will be the sum of the errors on all pairs of clusters as expressed by

$$P(U, T) = \sum_{C_1 \in U, C_2 \in U} \varepsilon(C_1, C_2, T).$$

If, for a given U, $P(U, T) = 0$, we can say that U is a perfect block model of the graph $G = (V, E)$ with respect to T. In most cases, it will not be possible to find a perfect block model; hence, we will try to find the clustering U' that minimizes the total error over all possible clustering of V.

If $T = \{\text{Null, Com}\}$ we are seeking a structural block model (Lorrain and White 1971), and if $T = \{\text{Null, Reg}\}$ we are seeking a regular block model (White and Reitz 1983).

Block Modeling of the Hijacker Network

We present two experiments with the hijacker network. In both experiments we seek a structural block model. The objective of the first experiment is to obtain four blocks (mainly because there were four flights). Using Pajek to do the modeling, we obtain the following connection matrix between the blocks (shown in Figure XI.15):

Final predicate matrix for the block modeling of Figure XI.15

	1	2	3	4
1	Com	–	–	–
2	null	com	–	–
3	Com	com	null	–
4	null	null	null	Null

We can see that only four almost complete connections were identified (after removing the symmetric entries). Two of them are the clique of cluster 2 and the almost clique of cluster 1. In addition, we have almost a complete connection between clusters 1 and 3 and between clusters 2 and 3. All other connections between clusters are closer to satisfying the null predicate than they are to satisfying the com predicate.

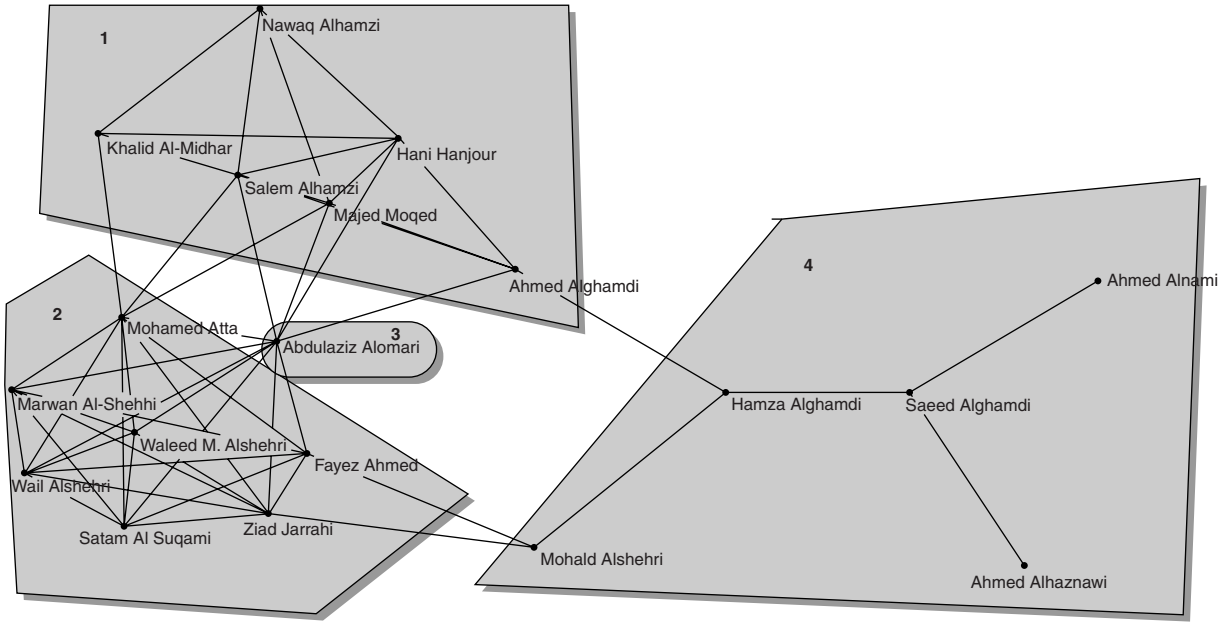

Figure XI.15. Block modeling with four blocks.

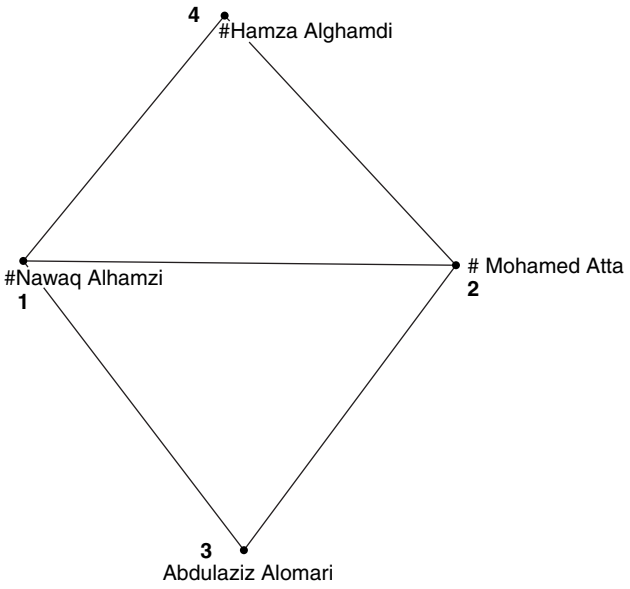

Figure XI.16. Shrinking the network based on the 4 blocks of 15.

The final error matrix is shown below; we can see that cluster 2 is a complete clique because its error is 0, whereas we can see that the connection between clusters 3 and 1 is not complete because three connections are missing – namely, between Abdulaziz Alomari and any of {Khalid Al-Midhar, Majed Moqed, and Nawaq Alhamzi}. The total error is 16. In order to see a schematic view of the network, we shrank the clusters into single nodes. If there was at least one connection between the clusters, we will see a line between the cluster's representatives. The name selected for each cluster is the name of the first member of the cluster (alphabetically based on last name, first name). The shrunk network is shown in Figure XI.16.

Final error matrix for the block modeling of Figure XI.16

	1	**2**	**3**	**4**
1	4	–	–	–
2	3	0	–	–
3	2	0	0	–
4	1	2	0	4

The objective of the second experiment is to see how the clustering and associated error cost changes when we set a higher number of target clusters. We run the block modeling of Pajek again specifying that we want to obtain six blocks or clusters. In this case the total error dropped to 9. The six blocks are shown in Figure XI.14 and then we show the predicate matrix of the block modeling and the final error matrix. We can see that five of the six blocks are close to a complete block (clique), whereas there are only three connections between the blocks.

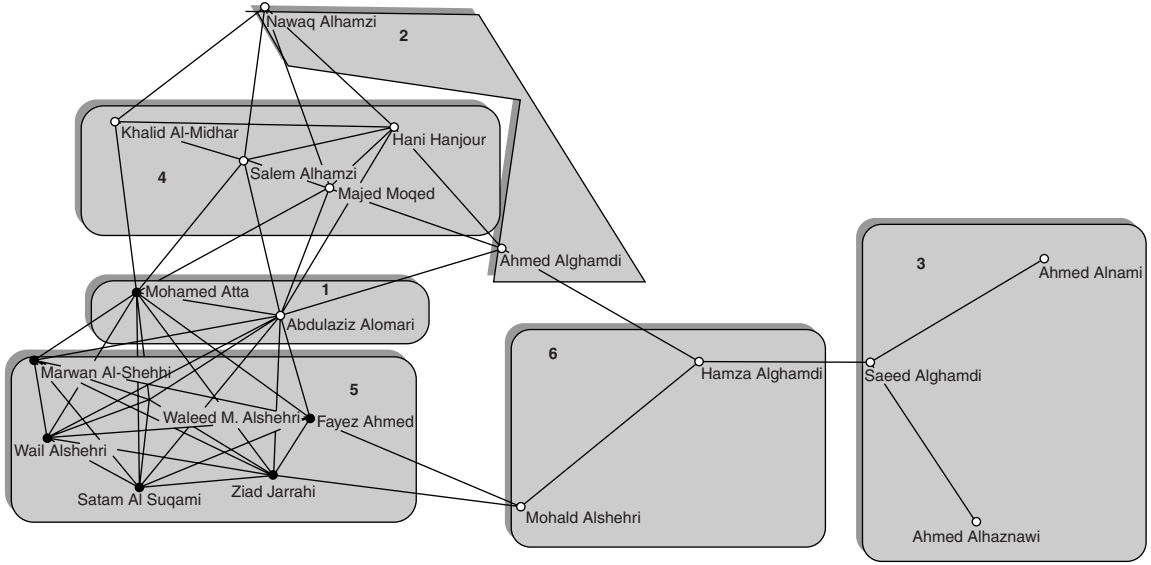

Figure XI.17. Block modeling with six blocks.

Here are the final predicate matrix and error matrix for the block modeling of Figure XI.17

	1	2	3	4	5	6
1	com	–	–	–	–	–
2	null	Null	–	–	–	–
3	null	Null	com	–	–	–
4	com	Com	Null	com	–	–
5	com	Null	Null	null	Com	–
6	null	Null	Null	null	Null	Com

	1	2	3	4	5	6
1	0	–	–	–	–	–
2	0	0	–	–	–	–
3	1	0	1	–	–	–
4	0	0	0	0	–	–
5	2	0	1	0	0	–
6	0	1	1	2	0	0

XI.6 PATTERN MATCHING IN NETWORKS

Often we have a pattern expressed as a small graph P and we want to see if it is possible to find a subgraph of G that will match P. This problem may arise, for instance, when we want to see if an instance of a given scenario can be found in a large network. The scenario would be expressed as a small graph containing a small number of vertices with specific relations that connect them. We then want to see if instances of the scenario can be found within our network. An example of such a pattern is shown in Figure XI.18. We have specified a pattern of one person who is connected *only* to three other people who have no connections between themselves. We can find three subgraphs within the hijackers' graph that contain a vertex connected to only three other vertices (marked 1, 2, and 3 in the figure); however, only 1 and 2 fully match the pattern. Subgraph 3 does not match the pattern because Fayez Ahmed and Ziad Jarrahi are connected. The naïve algorithm for finding exact matches of the pattern is based on simple backtracking – that is, if a mismatch is found the algorithm backtracks to the most recent junction in the graph visited before the failure. We can also search for approximate matches using techniques such as edit distances to find subgraphs that are similar to the pattern at hand. One of the most common patterns to be searched in a graph is some form of a directed graph that involves three vertices and some arcs connecting the vertices. This form of pattern is called a triad, and there are 16 different types of triads. One of them is the empty triad, in which there are no arcs at all, and another one is the full triad in which six arcs connect every possible pair of vertices in the triad.

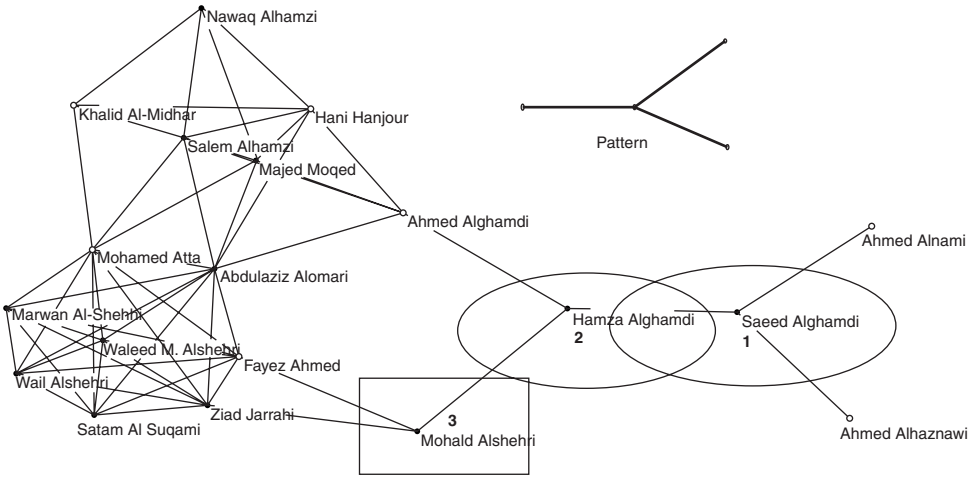

Figure XI.18. Pattern matching in the hijackers' graph.

XI.7 SOFTWARE PACKAGES FOR LINK ANALYSIS

There are several packages for performing link analysis in networks. Some are fairly expensive and hence are probably out of reach for the causal user. We describe here three packages that are either totally free or relatively inexpensive.

XI.7.1 Pajek

Pajek is a freeware developed by the University of Ljubljana that can handle networks containing hundreds of thousands of vertices. Pajek expects to get the input networks in a proprietary format, which includes the list of vertices and then lists of arcs (directed) and edges (undirected) between the vertices. There are programs that enable converting a simple set of binary connections to the Pajek (.net) format. Pajek supports a very large number of operations on networks, including centrality computations, path finding, component analysis, clustering, block modeling, and many other operations. In addition it includes a built-in drawing module that incorporates most the layout algorithms described in this chapter.

Pajek can be downloaded from
<http://vlado.fmf.uni-lj.si/pub/networks/pajek/>.
The converters can be downloaded from

<http://vlado.fmf.uni-lj.si/pub/networks/pajek/howto/text2pajek.htm> and
<http://vlado.fmf.uni-lj.si/pub/networks/pajek/howto/excel2Pajek.htm>.

XI.7.2 UCINET

UCINET is a fairly robust network analysis package. It is not free, but even for nonacademics it costs less than 300 dollars. It covers all the operations described in this chapter, including centrality measures (with a larger variety of options than

Pajek), clustering, path finding, and component analysis. UCINET can export and import Pajek files. Netdraw is the visualization package of UCINET.

UCINET and Netdraw can be downloaded from <http://www.analytictech.com/download_products.htm>.

XI.7.3 NetMiner

NetMiner is the most comprehensive package of the three, but it is also the most expensive. The professional version costs a little less than 1,000 dollars for commercial use. The package offers all the operations included in UCINET and Pajek and is fairly intuitive to use.

NetMiner can be downloaded from <http://www.netminer.com/NetMiner>.

XI.8 CITATIONS AND NOTES

Section XI.1

For a great introduction to graph algorithms, please refer to Aho, Hopcroft, and Ullman (1983). For in-depth coverage of the area of social network analysis, see Wasserman and Faust (1994) and Scott (2000).

Section XI.2

Force-based graph drawing algorithms are described in Kamada and Kawai (1989) and Fruchterman and Reingold (1991). Algorithms for drawing large graphs are addressed in Davidson and Harel (1996), Harel and Koren (2000), and Hadany and Harel (2001).

Section XI.4

The degree centrality was introduced in Freeman (1979). The betweenness centrality measure is due to Freeman (1977, 1979). The closeness centrality measure was introduced in Sabidussi (1966). The power centrality is due to Bonacich (1987). The eigenvector centrality originates from Bonacich (1972). Good descriptions of basic graph algorithms can be found in Aho et al. (1983). Cores have been introduced in Seidman (1983).

Section XI.5

The notions of structural equivalence and regular equivalence were introduced in Lorrain and White (1971) and further expanded in Batagalj, Doreian, and Ferligoi (1992) and Borgatti and Everett (1993). Block modeling was introduced in Borgatti and Everett (1992) and Hummon and Carley (1993). The implementation of block modeling in Pajek is described in Batagelj (1997) and De Nooy, Mrvar, and Batageli (2004).

Section XI.6

The notion of edit distance between graphs as vehicles for finding patterns in graphs is described in Zhang, Wang, and Shasha (1995). Finding approximate matches in undirected graphs is discussed in Wang et al. (2002).