„Schreiben Sie doch am besten alles auf
damit andere Ihnen folgen können.“
(Lothar Krempel)

Ulrich Eumann:
Manual zur Netzwerkvisualisierung

Stand: 9. Juni 2011

Vorbemerkung

Dieses Manual entstand ursprünglich im Frühjahr 2009 im Rahmen meiner Bemühungen darum, mit UCINet, Pajek und SVG eine ansprechende Visualisierung zu erhalten Inzwischen hat sich Softwarelandschaft so weiter entwickelt, dass der hier eingeschlagene umständliche Weg nicht mehr nötig ist. Die Programme **VISONE** und **Gephi** bieten zahlreiche Ansätze und sind von der Dateneingabe an sehr viel einfacher zu handhaben als die Klassiker.

Aus Zeitmangel habe ich nicht den ganzen Text überarbeitet, sondern die neuen Abschnitte ergänzt. Wer mag, kann über die Lesezeichen der pdf-Datei direkt darauf zusteuern.

Erfassung der Daten

Erstellen des Netzwerks über eine Tabellendatei mit 2 Arbeitsmappen (einer von vielen Wegen nach Rom):

- *Verbindungen zwischen Personen* mit Anweisungen in DL (Data Language, das **UCINet** Datenformat) im Dateikopf (in unserem Fall eine Edgelist mit embedded labels), Spalten z.B. Person1, Person2, Intensität der Verbindung plus weitere zu unserer Information (zum Beispiel der Quellenbeleg oder eine Einschätzung der Validität der Aussage über die Verbindung) oder zum Herausfiltern bestimmter Teilnetzwerke (zum Beispiel der Zeitpunkt einer Verbindung oder der Aussage über eine Verbindung). Die Zahl der Knoten (n) im DL-Bereich muss nicht korrekt sein, sie muss aber höher als die Zahl der realen Knoten sein, sonst werden die über n hinausgehenden Knoten nicht eingelesen. Sie sollte aber auch nicht viel größer sein, weil von ihr letztlich die Dauer der Berechnung abhängt.
- *Attribute der verbundenen Personen* mit der Überschrift *Node Data. (als Dateikopf für das VNA-Format von **NetDraw**, dem kostenlosen 2D-Visualisierungsprogramm von **UCINet**), Spalten z.B. ID, Geschlecht, Alterskohorte plus weitere zur eindeutigen Identifikation der Person sowie alles was wir später grafisch als qualitative oder quantitative Attribute dargestellt haben möchten (Organisationszugehörigkeit usw.).

Wir erstellen für das Widerstands-Projekt des NS-Dokumentationszentrums eine Tabellendatei pro Archivakte (besser wäre vielleicht eine Datei pro Gerichtsverfahren, aber die Option haben wir noch, weil die Akten eine Untermenge der Verfahren sind), damit wir hinterher die jeweiligen ‚Aktennetzwerke' (oder ‚Prozessnetzwerke') auswerten können, und fusionieren die einzelnen Netzwerke dann jeweils nach Auswertung einer Akte in die Datei mit dem Gesamtnetzwerk. Das hat seine Vorteile in der Erfassung, der Übersichtlichkeit und in den Auswertungsoptionen. Die Fusion der ‚Aktennetzwerke' führt aber ab einer bestimmten Größe auch zu Problemen (doppelte Einträge zu einer Person in den Attributen, unterschiedliche Schreibweisen eines Namens in verschiedenen Akten usw.). Eine grundsätzliche Lösung dafür gibt es vermutlich nicht.

Die Bezeichnungen der einzelnen Attributgruppen eines Netzwerks (männlich, weiblich usw.) werden von **UCINet** intern in Zahlen verwandelt und so auch später nach **Pajek** exportiert. Sie sind dann nur noch schwer wieder zu rekonstruieren. Die Zuordnung von Zahlen und Bezeichnungen als Legende kann man am besten im Auge behalten, wenn man die Datei *OUTPUT.LOG2* von **UCINet**, die beim Importieren der Attributdatei erstellt wird, unter einer eindeutigen Bezeichnung abspeichert (z.B. *Attributgruppen.txt*).

Sowohl **UCINet** als auch **Pajek** produzieren manchmal Fehlermeldungen aufgrund von Dateinamen, die unkonventionelle Zeichen enthalten (mehrere Punkte, Leerzeichen usw.). Deswegen sollte man sich strikt an die Regeln von ISO 9660 bzw. ISO 9660-Level 2 zur Benennung von Dateien halten.

Einlesen der Daten in UCINet

Zunächst speichern wir die beiden einzelnen Arbeitsmappen der Tabellendatei im CSV-Format (Comma Separated Values, ein Exportformat) mit Leerzeichen als Feldtrennzeichen und Anführungszeichen als Textmarkierungszeichen, die erste unter *Dateiname.csv,* die zweite unter *Dateiname_att.csv* (der Versuch, sie unter *Dateiname.att.csv* zu verwenden, führte auf Grund einer Fehlinterpretation des Punktes zu Fehlermeldungen von **UCINet**).

Im Text-Editor werden aus der ersten CSV-Datei (*Dateiname.csv*) die überflüssigen Anführungszeichen im Dateikopf sowie die Spaltenüberschriften entfernt und die Datei wird als txt-Datei gespeichert (*Dateiname.txt*)

Im Text-Editor werden aus der zweiten CSV-Datei (*Dateiname_att.csv*) die überflüssigen Anführungszeichen im Dateikopf entfernt und die Datei wird als Datei mit der Endung VNA gespeichert (*Dateiname_att.vna*).

In **NetDraw** öffnen wir zur Kontrolle der korrekten Erfassung der Daten das Netzwerk (*Dateiname.txt*) und die dazugehörigen Attribute (*Dateiname_att.vna*). Falls die Zahl der Personen im Netzwerk und die Zahl der Einträge bei den Attributen nicht gleich sind, erhalten wir eine Fehlermeldung (The labels in the attribute data do not wholly match up with network. Data may be wrong. Danger!). **NetDraw** zeigt dort, wo Daten fehlen oder Schreibungen von Namen unterschiedlich sind, Fragezeichen an. Wir können dann die fehlenden Daten ergänzen oder widersprüchliche Schreibungen von Namen korrigieren. Wenn die Daten passen, sehen wir eine Tabelle mit den Node Attributes. Bei größeren Netzwerken ist diese Tabelle schnell sehr unübersichtlich. Es empfiehlt sich dann, die Tabelle in die Tabellenkalkulation zu kopieren und dort mit dem Auto-Filter die fehlerhaften Einträge mit dem Fragezeichen als Feldinhalt herauszufiltern und diese dann in der Tabellenkalkulation oder in **NetDraw** zu korrigieren.

Wenn Netzwerk und Attribute zusammenpassen, öffnen wir **UCINet** und lesen die erste Datei *Dateiname.txt* Data → Import Text File → DL und die zweiten Datei *Dateiname_att.vna* Data → Import Text File → VNA ein. Das Programm wandelt sie automatisch in das UCINet-Format (*Dateiname.##d* und *Dateiname.##h* bzw. *Dateiname_att.##d* und *Dateiname_att.##h*) um (aus der Netzwerk-Datei wird dabei eine *Matrix*, die **UCINet** als Grundlage für alle Berechnungen verwendet).

Visualisierungen haben zwei grundlegende Funktionen:

- *Heuristik*: Erkennen von Strukturen des Netzwerks – dazu reicht **NetDraw** mit seinen vielfältigen Funktionen wie Anzeige von Attributen durch Farben oder Formen und dem sehr gelungenen Ego Network Viewer (**NetDraw** kann aber nur Pixelgrafiken exportieren, die sich nur sehr begrenzt durch Grafikprogramme weiter bearbeiten lassen).
- *Publikation*: um eine ausreichende Qualität der Grafiken für eine Veröffentlichung zu erreichen, müssen die Daten nach **Pajek** übertragen werden, das intern mehr Bearbeitungsmöglichkeiten und extern mehr Exportmöglichkeiten (Vektorgrafik) hat.

Am Beginn jeder Arbeit an Visualisierungen von Netzwerken stehen die folgenden Fragen:

- *Für welchen Zweck will ich visualisieren?*
- *Was soll dargestellt werden?*
- *Welche spezifischen Informationen soll die Grafik vermitteln?*

Erst die Antworten auf diese Fragen erlauben eine methodisch gesteuerte Auswahl von Teilnetzwerken und Darstellungsoptionen aus der Vielzahl an Möglichkeiten, die die Visualisierung bietet. Da man die Grafiken leicht mit Informationen überfrachten kann, ist nämlich Zurückhaltung geboten. Der Vorteil der Netzwerkvisualisierung liegt in der *Zeitersparnis* bei der Informationsaufnahme gegenüber Texten oder Tabellen. Wenn diese durch eine Überkomplexität, die ausführlicher Benutzungsanweisungen bedarf, ins Negative verkehrt wird, macht die ganze hier beschriebene Prozedur keinen Sinn mehr. Eine Auswahl

aus den zahlreichen Aspekten des Netzwerks für eine bestimmte Informationsgrafik ist also empfehlenswert. Diese Auswahl legt fest, welche (bereits vorhandenen oder noch zu berechnenden) Daten, Werte, Parameter von **UCINet** an **Pajek** übergeben werden müssen.

Die Netzwerkgrafik kann über die reine Struktur des Netzwerkes hinaus grundsätzlich auf sechs verschiedenen Wegen zusätzliche Informationen anbieten:

1. über die Größe der Knoten (z.B. Stellenwert einer Person)
2. über die Farbe der Knoten (z.B. Geschlecht einer Person)
3. über die Farbe der Umrandung der Knoten
4. über die Stärke der Kanten (z.B. Intensität einer Verbindung)
5. über die Farbe der Kanten (z.B. Art einer Verbindung)
6. über die Größe der Beschriftung und
7. über die Farbe der Beschriftung.

Für die meisten Informationstypen, die verwendet werden sollen, muss ein Parameter (Attribut der Person) bei der Auswertung der Quellen erfasst werden!

In **UCINet** kann man nun die Netzwerkdateien nach allen Regeln der Social Network Analysis (SNA) bearbeiten, die Dateien manipulieren (Teilnetzwerke extrahieren, die Matrix symmetrisieren usw.) oder Formeln über die Struktur des Netzwerks oder die Eigenschaften von Knoten (Menüs Tools und Network) für die Ermittlung von bestimmten Strukturzusammenhängen berechnen.

Auch **Pajek** kann viele der klassischen Formeln der SNA berechnen, ist aber in diesem Bereich überkomplex strukturiert. Da für die historische Forschung ohnehin einige wenige Algorithmen wie zum Beispiel Density, Geodesic Distance, Centrality Degree, Betweenness, Connectedness oder Cliques reichen, lohnt sich die aufwändige Einarbeitung in diesen Bereich von **Pajek** meines Erachtens nicht. Die Kenntnis der grundlegenden Konzepte ist allerdings von großem Nutzen für die Verwendung von **Draw**.

Unter bestimmten Fragestellungen kann es interessant sein, Ego-Netzwerke verschiedener geodätischer Distanzen aus dem Gesamtnetzwerk herauszuziehen. Das Extrahieren von Ego-Netzwerken in **UCINet** Data → Extract → Ego Network führte vor Version 6.240 unter bestimmten Bedingungen zu einer Speicherzugriffsverletzung (Access violation). Nun kann man als Input Dataset das Gesamtnetzwerk auswählen, einen Focal Node aus der alphabetisch sortierten Liste der Knoten (L) auswählen, entscheiden, ob der gewählte Knoten mit eingezeichnet werden soll, und den Namen der Ausgabedatei (standardmäßig mit der Endung –ego) festlegen. Nun ist es möglich, alle möglichen Werte auf der Basis *ausschließlich* der Alteri des ausgewählten Ego-Netzwerks berechnen zu lassen.

Leider lassen sich die geodätischen Distanzen des Ego-Netzwerks bei dieser Prozedur nicht gesondert bestimmen. Das geht besser, wenn man unter **NetDraw** aus dem Gesamtnetzwerk das Ego-Netzwerk und die Distanz Layout → Ego Networks (new) auswählt und dann die Daten in das UCINet-Format (*Dateiname.##h*) abspeichert File → Save Data As → UCINet → Binary Network. Das Extrahieren von Ego-Netzwerken über **NetDraw** hat außerdem den Vorteil, dass die Attribute der Knoten des Ego-Netzwerks ebenfalls exportiert werden können File → Save Data As → UCINet → Attributes.

Ein anderes Teilnetzwerk, das hier extrahiert werden kann, ist der Hauptbestandteil des Netzwerks (Main Component) Data → Extract → Main Component. Allerdings wird bei dieser Prozedur nur das Netzwerk extrahiert und scheint es keinen Weg zu geben, die Attribute der Knoten aus dem Main Component mitzuextrahieren. Die Visualisierung und

Analyse des Main Components beschränkt sich daher auf ganz spezifische Informationen bzw. Fragen.

Darüber hinaus bietet **UCINet** eine ganze Reihe von Verfahren, um Netzwerke direkt miteinander zu vergleichen. Über Data → Match Multiple Datasets kann man die Knoten von zwei Netzwerken (oder mehreren) miteinander in Beziehung setzen. Man kann 1. mit dem Button Primary festlegen, dass man die Knoten des primären Datensatzes im sekundären Datensatz suchen möchte, 2. mit dem Button Intersection die Schnittmenge an Knoten zwischen beiden Datensätzen ermitteln und 3. mit dem Button Union die Knoten anzeigen, die insgesamt in den angegebenen Datensätzen vorhanden sind. **UCINet** erstellt für jedes Verfahren zwei Dateien mit dem Suffix -matched, über **NetDraw** kann man sich beide dann anschauen. Sie enthalten jeweils über eines der drei angegebenen Verfahren ausgewählten Knoten und *alle* ihre Verbindungen in dem *jeweiligen* Netzwerk.

Bei den Isolates, die bei dem Verfahren Union in beiden Grafiken erscheinen, handelt es sich exakt um die Differenz zwischen den beiden Datensätzen. So kann man auf etwas umständliche Weise auch ein Netzwerk vom anderen subtrahieren. Allerdings gibt es kein Verfahren, um eine Liste der Isolates zu generieren und ihre Auflistung in NetDraw endet je nach Anzahl an einer bestimmten Stelle des Alphabets, nämlich an der, mit der die Unterkante des Bildschirmfensters erreicht ist. Über die Berechnung des Freeman Degrees ist es dann möglich, die Liste der Isolates zu erhalten, das sind genau die Personen, die einen Degree von 0,000 aufweisen.

Für *jede* Information, die eine Grafik über die Struktur des Netzwerks und die Intensität der Verbindungen hinaus vermitteln soll – z.B. bestimmte Eigenschaften der Knoten (Geschlecht, Organisation, Centrality Degree, Betweenness) als ihre Farbe (Partition) oder ihre Größe (Cluster) -, muss eine Partition- oder Cluster-Datei aus **UCINet** an **Pajek** übergeben werden.

NetDraw verfügt im Unterschied zu **Draw** von **Pajek** über eine Druckoption File → Print.

Ausgabe der Daten nach Pajek

Export der Daten aus **UCINet** nach **Pajek** in das Pajek-Format (*Dateiname.net*) Data → Export → Pajek → Network, dabei kann sowohl das Netzwerk als auch die Datei mit den Attributen angegeben werden.

Die Daten werden von **UCINet** beim Export geteilt (in der NET-Datei untereinander geschrieben):

- „Vertices“: für jeden Knoten (Vertex) eine Zeile mit fünf Spalten, 1) eine fortlaufende Nummer, 2) das Label, 3) die X-Koordinate, 4) die Y-Koordinate und 5) die Z-Koordinate für die erste grafische Darstellung in Pajek (xy = 0 ist links oben und xy = 1 ist rechts unten)
- „Arcs“: für jede Verbindung (Arc oder Edge) eine Zeile mit mindestens drei Spalten: 1) Person1, 2) Person2, 3) Intensität. Als vierte Spalte kann man z.B. bestimmte Farben für bestimmte Verbindungen festlegen (siehe unten).

Was die xy-Koordinaten betrifft, ist zu beachten, dass die Fläche, in der die Knoten platziert werden, zunächst einmal grundsätzlich quadratisch ist, d.h. der Abstand zwischen x = 0 und x = 1 ist genauso groß wie der zwischen y = 0 und y = 1.

Die Kanten werden von **Pajek** automatisch entsprechend dem Seitenverhältnis des verwendeten Monitors bzw. Bildschirmfensters gestreckt oder gestaucht. Es ist zu empfehlen, ein Seitenverhältnis zu wählen, das dem Seitenverhältnis der späteren Druckseite entspricht

(DIN-Formate 1:1,41), weil sonst der Platz links und rechts vom Graphen schlecht ausgenutzt wird. Natürlich kann man die Grafik in **Inkscape** mit der Maus auf das ganze Blatt vergrößern, aber dann werden die kreisförmigen Knoten zu Ellipsen.

Die einzelnen Attribute der Personen können als Partitions (*Dateiname.clu*) Data → Export → Pajek → Categorical Attribute oder als Vectors (*Dateiname.vec*) Data → Export → Pajek → Quantitative Attribute nach **Pajek** exportiert werden. Bei den Partitions ist der Export über **NetDraw** bequemer als der über **UCINet**, weil man die einzelnen Attribute über eine Liste auswählen kann und die CLU-Dateien den Namen des Attributs zugewiesen bekommen (z.B. *Geschlecht.clu*).

Für manche Fragestellungen ist es sinnvoll, wenn die Knoten in der Grafik nicht alle gleich groß sind, sondern ihre Größe von Werten abhängt, die **UCINet** für den Stellenwert einzelner Personen berechnet, wie z.B. Betweenness oder Centrality Degree; diese Werte müssen dann aus der erstellten UCINet-Datei (z.B. *Betweenness.##d*) Data → Export → Pajek → Quantitative Attribute als quantitative Attribute (*Dateiname.vec*) ausgegeben werden. Gegebenenfalls muss über die VEC-Datei, die eine reine Textdatei ist, ein höherer ‚Kontrast' eingestellt werden, indem man das Attribut (via **Excel**) mit einem Faktor X multipliziert, oder es nach seiner Relation zum Mittelwert mit einem Faktor X multipliziert bzw. dividiert. Hierbei ist zu beachten, dass deutsche Umlaute am Anfang eines Namens dazu führen können, dass die Listen von Namen und den verschiedenen Werten, die Pajek führt, nicht mehr übereinstimmen.

Wir öffnen nun die von **UCINet** erstellte Datei mit dem Gesamtnetzwerk (*Dateiname.net*) und den eventuell zu betrachtenden Partitions (*Dateiname.clu*) oder Vectors (*Dateiname.vec*) in **Pajek** File → Network → read, File → Partition → read, File → Vector → read. Die Anzahl der Knoten in allen drei möglichen Dateien muss identisch sein, sonst gibt es eine Fehlermeldung (Network, Partition and Vector of equal size needed). Es ist möglich, die unterschiedlichen Daten (Netzwerk, Partitions, Vectors) in eine Gesamtdatei (*Dateiname.paj*) abzuspeichern File → Pajek Project File→ Save, die **Pajek** dann auch komplett wieder laden kann File → Pajek Project File→ Read. Allerdings wird die Reihenfolge der Partitions dabei nicht korrekt wieder geladen. Die verwendeten Partitions werden aber alle geladen und lassen sich über die Auswahlfelder wieder in die Reihenfolge bringen.

Transfer in Grafik

Um die Grafik zu betrachten, klickt man in **Pajek** auf Draw und wählt die zu betrachtenden Daten aus: das ganze Netzwerk (Draw), das Netzwerk und bestimmte kategoriale Attribute (Draw-Partition), das Netzwerk und bestimmte quantitative Attribute (Draw-Vector) oder Kombinationen daraus (Draw-Partition-Vector) und einige mehr.

Die wichtigsten Layouts in **Pajek**:

- Circular → Using Partition: interessant vor allem im Hinblick auf die einzelnen Teil-Netzwerke (die jeweils kreisförmig angeordnet werden und ihre Verbindung untereinander)
- Energy („Spring Embedder": die Platzierung eines Knoten erfolgt durch Anziehung und Abstoßung durch andere Knoten in mehreren Durchgängen). Es gibt zwei verschiedene Algorithmen dafür, die ihre jeweiligen Vor- und Nachteile haben: → Kamada-Kawai (mit 6 verschiedenen Einstellungen) und → Fruchterman-Reingold (2D, 3D und Einstellung des Abstands zwischen Knoten).

Layouts können kombiniert werden, indem man z.B. Teile eines Layouts (Markieren durch Ziehen eines Rechtecks über die Grafik mit der rechten Maustaste) neu darstellen lässt. Auch die Reihenfolge, in der bestimmte Layout-Operationen durchgeführt werden, beeinflusst auf komplexe Weise die Grafik.

Ein Grundproblem von **Draw** besteht darin, dass es die Knoten zwar nach den Angaben eines Vectors (siehe unten) unterschiedlich groß darstellen kann, aber die unterschiedlichen Radien beim Berechnen der Positionen der Knoten durch den Spring Embedder nicht berücksichtigt werden. Der Layout-Algorithmus bezieht sich *ausschließlich auf die Kräfte zwischen den Knoten*! Das führt unter anderem dazu, dass sehr dichte Komponenten sehr kompakt dargestellt werden und Komponeten mit sehr geringer Dichte sehr extensiv.

Normalerweise werden bei jedem Aufruf einer Layout-Prozedur alle Knotenpositionen neu berechnet. Man kann jedoch (in Grenzen!) bestimmte Knoten an bestimmten Punkten fixieren. Dazu erstellt man aus dem Gesamtnetzwerk eine neue Partition Partition → Create Constant Partition. Dazu braucht **Pajek** zwei Parameter: die Dimension of Partition, wo die Anzahl der Knoten im Netzwerk angeboten wird, die man am besten per Klick bestätigt, und Constant, wo man sinnvollerweise 0 (null) auswählt, weil Null der Cluster (Netzwerksektor, Gruppe) sein soll, dessen Knoten in ihrer Position veränderbar bleiben sollen („restrict the automatic relocation to the vertices in class 0"). Diese Datei kann man speichern File → Partition → save. Wenn man sie zum Bearbeiten öffnet File → Partition → edit, sieht man, dass sie zwei Spalten enthält (Val und Label). Nun wählt man anhand des Labels die Knoten aus, die fixiert werden sollen, und trägt eine andere Zahl als 0 (null) in das Feld Val ein. Dann ruft man die Prozedur Layout → Energy → Kamada Kawai → Fix Selected Vertices auf. Der Spring Embedder wird nun ausschließlich auf die Knoten in Cluster Null angewendet. Allerdings werden dabei auch die angeblich fixierten Knoten um ein paar Pixel bewegt, um den Platz auf dem Bildschirm besser auszunutzen.

Um die Graphen von zwei oder mehreren verschiedenen Netzwerken um einen Kern von identischen Akteuren vergleichbar zu halten, kann man für diese Akteure schon in den zu Grunde liegenden Tabellen oder in den NET-Dateien von **Pajek** bestimmte Positionen über ihre XY-Koordinaten festlegen. Diese Akteure stellt man in einer neu erstellten Partition in Cluster $\neq 0$ (ungleich null) und führt für alle Netzwerke die Prozedur Fix Selected Vertices durch. Die fixierte Position ist aber nur eine relative. Ein Versuch mit zwei Dummy-Netzwerken von je 8 Knoten (von denen drei in beiden vorhanden waren) ergab eine Abweichung in der Platzierung der drei fixierten Knoten von bis zu 15 Prozent von der Ursprungsposition ($x = 0.5728$ statt $x = 0.5000$), was wohl nicht zuletzt auch von der Bildschirmdiagonale abhängt. Da **Draw** ohnehin den Graphen zur besseren Platzausnutzung staucht oder dehnt, ist die Position der fixierten Knoten, wenn auch nur in geringem Maße, *auch* von der Position der anderen Knoten abhängig. Eine im Ausdruck auf den Millimeter genau identische Position bestimmter Knoten in mehreren Graphen ist also allein mit **Pajek** nicht zu erreichen. Man kann natürlich hinterher den drei oder 30 Knoten, die in allen Graphen am selben Punkt stehen sollen, in der NET-Datei wieder manuell den Ausgangswert zuweisen.

Bei zu vielen kleinen Komponenten im Netzwerk, die auf Grund der langen Knoten viel Platz wegnehmen, gibt es in **Pajek** die Möglichkeit, eine Mindestgröße für darzustellende Komponenten anzugeben. Über die Prozedur Net → Components → Weak wird mit Hilfe eines Schwellenwerts eine Partition (z.B. *extrakt.clu*).erstellt, in der Komponenten, die kleiner als der Schwellenwert sind, in den Cluster 0 (null) geschrieben werden. Im nächsten Schritt Operations → Extract from Network → Partition kann man mit dem Parameter 1-* (alle Cluster von 1 bis zum letzten) die Knoten in Cluster null aus dem Netzwerk entfernen. Leider bleiben davon alle schon während der **Pajek**-Sitzung geöffneten Partitions und Vectors unberührt. Aus der Partition mit den Attributen unseres Netzwerks kann man nun aber ebenfalls

die Knoten in Cluster null extrahieren. Dazu muss man die zu reduzierende Partition im ersten Feld und die Partition *extrakt.clu* im zweiten Feld unter Partitions auswählen. Nun kann man mit Partitions → Extract Second from First und dem Parameter 1-* auch hier die kleinen Komponenten entfernen. Aus dem Vector, der die Größe der Knoten bestimmt, kann man des weiteren, wenn unsere Partition *extrakt.clu* als erste Partition ausgewählt ist, über Vector → Extract Subvector und den Parameter 1-* auch den Vector um die Knoten in Cluster null verkürzen.
Wenn für Knoten und Kanten in eckigen Klammern Zeitangaben über das Auftauchen und das Verschwinden eines Elements existieren – z.B. [29-35] -, kann **Pajek** Netzwerke für bestimmte Zeitintervalle extrahieren. Über Net → Transform → Generate in Time lässt sich zwischen drei verschiedenen Arten von Generierung wählen: 1. All: unter Angabe von Startzeitpunkt, Endzeitpunkt und Intervalllänge erstellt das Programm Netzwerke für jedes vorhandene Zeitintervall. 2. Only different: wie oben, aber neue Netzwerke werden nur für bestimmte Intervalle erzeugt. 3. Intervall: hier wird das Netzwerk nur für einen bestimmten Intervall, also zum Beispiel 10-17 erzeugt. Die erzeugten Netzwerke lassen sich dann als NET-Dateien oder Grafiken abspeichern.

Bearbeitung der Grafik in Pajek

Im Grafikmodus bei **Pajek** kann man die Position einzelner Knoten nachjustieren, man kann den einzelnen Knoten anfassen (Mauszeiger auf den Knoten, linke Taste drücken, ziehen). Es ist aber auch möglich, Gruppen von Knoten, die einen Attributwert gemeinsam haben, zu verschieben, wenn man den Mauszeiger knapp neben einen Knoten setzt, der zur Gruppe gehört. Vorsicht ist bei großen Netzwerken damit geboten! Der Ausgangszustand ist nämlich bei 300 Knoten nicht mehr leicht wiederzufinden!

Bei der Darstellung in **Pajek** wird die ursprünglich quadratische Grundfläche unseres Netzes je nach Bildschirmdiagonale auf eine rechteckige Fläche gedehnt. Über Options → Layout → Real xy proportions kann das Bildschirmfenster von **Draw** wieder auf das quadratische Original umgestellt werden.

Auf einzelne Bereiche der Grafik lassen sich, nachdem der entsprechende Bereich markiert wurde (durch Ziehen eines Rechtecks über die Grafik mit der rechten Maustaste) alle Funktionen anwenden, so lange man nicht wieder ZoomOut anklickt

Die Einstellung von Labels, Werten, Größe, Farbe der Knoten und Kanten erfolgt über das Menü Options (Lines, Sizes, Colors usw.) von **Pajek**, bei der Schriftgröße Options → Size → Font gibt es aber anscheinend die Mindestgröße 8. Alle Werte gelten *ausschließlich* für die Darstellung in **Draw**!

Die Farben der Knoten (zur Auswahl von Farben siehe Lothar Krempel: Visualisierung komplexer Strukturen, Frankfurt 2005) lassen sich generell festlegen: Options → Colors → Partition Colors. In einer Übersicht lassen sich die Vorzugsfarben für 40 verschiedene Partitions auswählen. Nach Experimenten mit harmonisch abgestuften Farben (Dunkelrot, Rot, Orange, helles Orange) zeigte sich, dass ein größerer Kontrast zwischen den Farben (die Komplementärfarben Rot, Blau, Gelb, Grün usw.) den Graphen zwar vielleicht weniger ästhetisch, dafür aber leichter lesbar macht. Auch die Farbe der Umrandung der Knoten lässt sich durch eine Partition festlegen.

Die Strichstärke der Kanten kann entsprechend der Intensität der Verbindung angezeigt werden Options → Lines → Mark Lines → Different Widths, sind die Kanten zu schmal oder zu fett, kann über Options → Size → Lines ein Faktor eingestellt werden, mit dem der Wert der Intensität multipliziert wird (Standard = 1); die Intensität der Verbindung kann aber

auch über unterschiedliche Graustufen abgebildet werden Options → Lines → Mark Lines → GreyScale (die aber teilweise auf Ausdrucken von Schwarzweißdruckern gar nicht zu erkennen sind, da müsste man eventuell über die Hintergrundfarbe nachhelfen).

Der Sinn einer Einfärbung von Kanten Options → Colors → Edges/Arcs → As Defined on Input File könnte darin bestehen, zum Beispiel besondere Arten (nicht Intensitäten) von Beziehungen zu kennzeichnen. Bei uns wären das vorzugsweise freundschaftliche oder verwandtschaftliche Beziehungen, von denen wir aber nicht wissen, ob und wie sie die Intensität der Widerstandsbeziehungen beeinflusst haben. Dazu kann man in der NET-Datei eine Spalte einfügen, in der man mit dem Präfix „c “ eine der in **Pajek** festgelegten Farben für bestimmte Verbindungstypen festlegt (siehe zu den Farbennamen S. 84 des Manuals pajekman.pdf), wobei man den Farbennamen *exakt* so schreiben muss wie im Manual (z.B. c LightMagenta).

Der Versuch, den Farbennamen in der Tabellendatei hinter die Namen der beiden verbundenen Personen zu schreiben, funktioniert nicht. **UCINet** erwartet in der dritten Spalte eine Zahl, die die Intensität der Verbindung ausdrückt und ersetzt die Farbennamen durch 0.000. Man muss also die festgelegten Farbennamen durch Suchen und Ersetzen in eine Zahl umwandeln und diese Umwandlung später in der NET-Datei wieder rückgängig machen, wo man allerdings eine neue Spalte mit der Intensität als dritte Spalte einfügen mussen.

Man kann aber auch in **Draw** einen Knoten anklicken, dann in der Verbindungsübersicht eine Verbindung auswählen und dann eine New Relation Number vergeben, der dann automatisch aus der Liste der Relation Colors eine andere Farbe als den anderen zugewiesen wird.

Die Farbe der Beschriftung der Labels lässt sich über das zweite Attribut Second Partition beeinflussen, das ordinal skaliert sein kann (Geschlecht, Organisation usw.).

Die Größe der Labels kann in **Pajek** ebenfalls über einen Wert beeinflusst werden, den wir als drittes Attribut Third Partition übergeben (erscheint im 3. Feld der Rubrik Partition oder hat die Nummer 3 als Präfix bei **Pajek**). Dieses Attribut ist am besten ein *metrisch skaliertes numerisches Attribut*, damit die unterschiedlichen Größen auch unterschiedliche Einheiten des Attributs repräsentieren. Ich habe dabei mit dem Wert des genormten Centrality-Degree nach Freeman (NrmDegree) die besten Erfahrungen gemacht, den ich in eine ‚quantitative' Partition umwandeln lasse Vector → Make Partition → By Truncating (abs). Allerdings empfielt es sich, alle Werte vor der Verwendung in der neuen Partition um eins zu erhöhen, da in dem Verfahren die Nachkommastellen abgeschnitten werden, Labels mit dem Wert null also nicht mehr angezeigt werden.

Bei größeren Netzwerken (mehr als 200 Knoten) bietet es sich an, die Labels nur noch *ab einem bestimmten Einfluss* des Knotens anzuzeigen, weil sonst je nach Dichte einzelner Netzwerk-Komponenten viele Knoten nicht mehr sichtbar sind. Man kann wie oben einen Vector in eine Third Partition umwandeln, die über die Textgröße entscheidet. Je nachdem wie groß die Labels der ‚unwichtigen' Knoten sind, kann man hinterher noch über den XML-Editor mit Suchen und Ersetzen allen Texten mit einer bestimmten (niedrigen) Textgröße (< font-size:15px) eine Textgröße null (font-size:0px) zuweisen.

Anscheinend stehen alle geladenen Partitions unter **Draw** zur Verfügung in Bezug auf die Einfärbung der Knoten. Über Previous und Next lassen sie sich der Reihe nach ansteuern. Über die Titelleiste des Fensters (ganz oben) hat man immer die Kontrolle darüber, welches Netz / welche Partition / welcher Vector aktuell dargestellt sind.

Die drei Felder im Bereich Partition und die je zwei Felder im Bereich Net und Vector bedeuten keine Beschränkung auf Graphen mit maximal acht verschiedenen Daten. Man kann zum Beispiel im ersten Schritt eine Partition erstellen, die nur für die Fixierung bestimmter

Knoten da ist. Sofern man mit der Darstellung zufrieden ist, speichert man die Positionen in der Net-Datei und ersetzt die Partition im nächsten Schritt durch eine andere Partition, die vielleicht die Schriftfarbe bestimmt. Möchte man dann wieder das Layout ändern, muss man wieder den Schritt zurück machen.

Es ist auch möglich Knoten, die einen Wert von 0 (null) haben, mitsamt den dazu gehörenden Kanten auszublenden Options → Layout → Size of Vertex 0

Die Datei pajek.ini enthält die jeweils aktuellen Einstellungen der Grafik des aktuellen Visualisierungsprojekts in **Pajek** (Farben, Größen, Linienstärken, Formen usw.) und wird beim nächsten Öffnen des Programms wieder aufgemacht. Man kann aber parallel mit verschiedenen Einstellungs-Dateien arbeiten, wenn man verschiedene Visualisierungsprojekte parallel verfolgen möchte, in dem man sie unter verschiedenen Namen (z.B. gesamtnetz.ini, egonetz.ini) abspeichert Options → Ini File → Save und vor dem Wechsel zu einem anderen Projekt wieder lädt Options → Ini File → Read.

Man kann die Grafik nicht über **Pajek** ausdrucken. Der einfachste Weg geht über den Export einer Bitmap-Datei, die dann über den **Microsoft Office Picture Manager** oder ein ähnliches Programm ausgedruckt werden kann. Der Ausdruck entspricht dabei recht genau dem Bild auf dem Monitor. Die Export-Einstellungen (siehe unten) werden dabei aber ignoriert, da sie nur für bestimmte Exportformate (EPS, SVG usw.) gelten. Man sollte also farbige Hintergründe vor dem Export in weiße ändern, um Tinte oder Toner zu sparen.

Graphen mit einigen hundert Knoten, die sich auf wenige große, ein paar mittlere und einige kleine Komponenten aufteilen, verlangen dem Algorithmus des Kamada-Kawai Spring Embedders von **Pajek** nicht nur in Bezug auf die Rechenzeit, sondern vor allem auf die Darstellung alles ab. Relativ leere Komponenten werden auseinander gezogen, relativ volle und verdichtete zusammengedrückt. Außerdem arbeitet der Algorithmus offenbar zeilenorientiert: die größte Komponente kommt links in die erste Zeile, rechts davon die zweitgrößte bis zum rechten Rand, dann geht es links in der zweiten Zeile weiter usw. usf.

Der einfachste denkbare Weg, einzelne Komponenten von Hand an passende Positionen zu schieben, ist nicht gangbar, weil man in **Draw** nur einzelne Knoten oder Cluster mit der Maus positionieren kann. Die Fixierung zentraler Akteure an bestimmten Positionen auf Grund des Augenscheins bringt uns dem Ziel ein bisschen näher. Man muss aber mit mehreren Durchgängen rechnen, vor jedem die NET-Datei mit den geänderten Koordinaten neu laden (desgleichen bei verunglückten Neuberechnungen). Danach kann man einzelne, zu sehr gequetschte Bereiche mit der rechten Maustaste auswählen und über Options → Transform → Resize gesondert in X-, Y- und/oder Z-Richtung ausdehnen. Ab ca. 400 Knoten kann die Grafik-Engine von **Pajek** aber nicht mehr seriös empfohlen werden, der Zeitaufwand bei der manuellen Bearbeitung steigt exponentiell. Bei Netzwerken dieser Größe lässt sich nur noch im Zusammenspiel von **VISONE** und **Pajek** ein ansprechendes Ergebnis erzielen.

Wenn man an der Grafik in **Draw** ausgiebige manuelle Veränderungen vorgenommen hat, möchte man das natürlich beim nächsten Aufruf von **Pajek** nicht wieder tun müssen. Da die Net-Datei ohnehin schon beim Aufruf die xy-Koordinaten der Knoten enthält, kann man seine Änderungen abspeichern, indem man auf das Diskettensymbol links neben dem ausgewählten Netzwerk klickt.

Visualisierung mit VISONE

Eine Alternative bzw. Erweiterung zu Pajek ist **VISONE** (http://visone.info/), ein auf JAVA aufsetzendes kostenloses Tool zur Analyse und Visualisierung von sozialen Netzwerken.

VISONE kann NET-Dateien von **Pajek**, DL-Dateien von **UCINet** sowie Matrizen im CSV-Format einlesen und Dateien u.a. als Bitmap-Grafiken oder im NET-, DL-, SVG- und im GraphML-Format ausgeben, einem „comprehensive and easy-to-use file format for graphs“. Mit **VISONE** kann man auf Grund einer Vielzahl von zum Teil ungewöhnlichen Layout-Algorithmen bei der Visualisierung von großen Netzwerken bessere Ergebnisse erzielen als mit **Pajek**. Da **VISONE** – anders als **Pajek** - auch die Möglichkeit bietet, größere Gruppen von Knoten zu markieren und zu verschieben, eignet sich das Programm hervorragend für den weiteren Feinschliff des Graphen nach der Erstellung durch **Pajek** und vor der letzten Bearbeitung in **Inkscape** oder dem **XML Editor**.

Vorsicht! Falls man als Plattform statt des empfohlenen **JAVA Web Start** das normale **JAVA** benutzt, handelt man sich eine Menge Probleme ein, von regelmäßigen Abstürzen bis hin zu fehlerhaft exportierten NET-Dateien.

Knoten-Attribute lassen sich in **VISONE** leicht hinzuladen. Dazu muss allerdings das Feld mit dem Namen der Person die Überschrift "id" tragen. Wir speichern die Datei – allerdings ohne den bisher benutzten Dateikopf für den VNA-Import nach **UCINet** – und öffnen in VISONE, wo wir das Netzwerk bereits über eine Adjazenzmatrix geladen haben, den attribute manager, klicken auf nodes und import & export, dort wählen wir unter import die Datei aus, die wir importieren wollen und die Spalte "id", über die wir Netzwerk und Attribute verbinden (join by) können. Ein klick auf apply und die Daten sind importiert.

Wenn es geklappt hat, sehen wir die Attribute bei den node properties im dritten Reiter attributes. **Open Office Calc** und neuere Versionen von **Excel** bieten beim Speichern einer Tabelle als CSV-Datei an, Text- und Feldtrennzeichen selbst festzulegen.

Man kann die Knoten dann nach Attributen gruppieren, beispielsweise nach Geschlecht, und bekommt dann zwei übereinanderliegende Layer von Männern und Frauen, in denen sich der einzelne Knoten immer noch an seinem Platz in der Struktur des Gesamtnetzwerks befindet, die ich aber auch auseinander schieben kann. Ich kann über den Reiter selection ein oder mehrere Attribute auswählen und mir zum Beispiel der Reihe nach die Knoten zu den verschiedenen Attributwerten (Geschlecht, Alterskohorte) anzeigen lassen. Sie sind dann markiert und können gleichförmig gestaltet werden (Farbe, Form usw.). Wenn man dann die Netzwerk-Datei wieder abspeichert, bleiben die Attribute erhalten, müssen also nicht jedes Mal wieder neu geladen werden.

Wenn man das Gesamtnetzwerk in verschiedene ‚Eigenschafts-.Netzwerke‘ aufteilen will, kann man vorerst nur die unerwünschten Eigenschaften löschen und den übrig bleibenden Graphen unter anderem Namen abspeichern. Man muss dann, so lange keine undo- und redo-Funktionen vorhanden sind (die aber geplant sind), das Gesamtnetzwerk leider neu laden

Auch die Verbindungen lassen sich in **VISONE** aus einer CSV- oder TXT-Datei laden, statt sie umständlich über **UCINet** und **Pajek** zu bearbeiten und dann erst in **VISONE** zu laden. **VISONE** benötigt allerdings eine Adjazenz-Matrix, kann also nicht mit unseren untereinander geschriebenen Verbindungseinträgen arbeiten. Um diese zu erstellen, öffne ich in **UCINet** den DL-Editor, kopiere die Verbindungen aus der Tabellenkalkulation hinein und wähle Edgelist (1-mode) als Format. **UCINet** erstellt dann die Matrix, die ich als NET-Datei exportieren und in **VISONE** einlesen kann.

Dabei gehen allerdings unsere Verbindungs-Attribute, unter anderem der Zeitpunkt der Verbindung, verloren, nur die dritte Spalte mit den Verbindungs-Intensitäten bleibt erhalten.

Die anderen Attribute der Verbindung (bei uns u.a. Quelle, Zeitpunkt, Bemerkung) lassen sich im Prinzip in **VISONE** als Attribute laden, dazu brauchen wir aber die „id“ der Kanten.

Im Prinzip ist die Sache einfach, allerdings nicht wenn ich den hier bisher üblichen Weg über eine CSV-Datei und **UCINet** einschlage, weil ich dort keinen Einfluss auf die Sortierung der Matrix-Zeilen habe. Wenn ich meine Verbindungen aber in den DL-Editor von **UCINet** kopiere, kann ich festlegen, dass die Einträge alphabetisch sortiert werden sollen (sort alphabetically). Für den unwahrscheinlichen Fall, dass ich nur je eine Verbindung zwischen zwei Personen habe, brauche ich nur in der Tabellenkalkulation die beiden Spalten nach Person1 und Person2 zu sortieren, eine laufende Nummer pro Zeile zu addieren und schon habe ich meine „id“, die mit der internen „id“ in **VISONE** identisch ist und mit der ich dann die Verbindungs-Attribute zuweisen könnte.

Leider sind die meisten Netzwerke komplizierter. In unserem Beispiel haben 64 Knoten 225 Verbindungen, wir haben also 225 Kanten-„id“'s. In unserer ursprünglichen Excel-Tabelle habe ich aber 366 Zeilen mit Verbindungen. Da in einer Matrix jede Person nur einmal auftreten kann, werden sämtliche Verbindungen von zwei Personen in eine Richtung addiert. Meine 141 doppelten Verbindungen fallen also der Prinzipien der Matrixbildung zum Opfer. Wenn man sich allerdings eine Visualisierung mit fünf gleichgerichteten Linien von Person1 zu Person2 vorstellt, erkennt man, dass dieser Graph nicht mehr übersichtlich und nicht mehr selbst erklärend wäre und damit nicht mehr die Kriterien für eine sinnvolle Visualisierung erfüllen würde.

Selbstverständlich kann man auch in VISONE Messwerte wie Zentralität über Formen oder Farben visualisieren. Die Werte berechnet man mit den Menüs, die bei klick auf den Reiter analysis bei Wahl des tasks indexing zur Verfügung stehen: also etwa node density, node centrality oder node distance. Sie werden dann als Attribute des Knotens gespeichert, also beispielsweise der centrality degree im Feld degree unter node properties und dem Reiter attributes. Über das Menü visualization lassen sie sich dann grafisch darstellen. Man markiert dazu alle zu verändernden Knoten, wählt im Menü mapping, darunter size als type, node area als property, als node value degree (%) und klickt auf visualize! Mit color als type, node color als property und als node value degree (%) lassen sich über verschiedene Methoden Farben zuweisen, zum Beispiel über die Methode brightness, mit der Knoten mit höheren Werte dunklere Farbtöne zugewiesen erhalten.

Unter der Vielzahl von Layouts, die **VISONE** anbietet, ist auch das Status Layout. Dieses kann man unter anderem dazu "zweckentfremden", um einen Zeitablauf darzustellen. Wir haben zum Beispiel die Tage der Vernehmungen unserer Widerständler, rechnen diese in ganze Zahlen um und können nun mit dem Layout, die Tage auf der y-Achse visualisieren: der 1. Zeitpunkt liegt dann am unteren, der letzte am oberen Rand der Grafik. Den Vernehmungstag als Attribut der ersten Person laden wir über den attribute manager hinzu, nachdem wir vorher unser Netz aus einer NET-Datei erstellt haben, das keine Attribute enthält.

Wie auch bei den Klassikern lassen sich in **VISONE** Werte und Attribute über Eigenschaften der Knoten und Kanten wie Größe oder Farbe darstellen. Dazu wählen wir im Menü visualization unter category nicht layout, sondern mapping. Wir haben dann die Optionen, die Farbe, die Größe, das Label, die Koordinanten oder den Z-Layer entsprechend bestimmter Werte darstellen zu lassen. Wenn wir zum Beispiel das Zentralitätsmaß (Degree) berechnet haben (siehe Reiter analysis), können wir die Knotengröße entsprechend gestalten, indem wir im mapping Menü size wählen, als property node area und als node value den gerade berechneten degree. Leider lässt sich auch durch das größer Skalieren des Wertes kein besserer Kontrast zwischen Knoten mit starkem Degree und mit geringem Degree erreichen,

da **VISONE** immer mit relativen Werten arbeitet. Wenn wir zum Beispiel die verschiedenen Organisationen, denen unsere vernetzten Personen angehören, über die Farbe des Knotens darstellen wollen, wählen wir unter mapping den Unterpunkt color. Dort können wir die Farbe des Knotens, des Knotenrandes und der Kante beeinflussen. Es gibt 4 verschiedene mehrere Farbmodelle, interpolation, saturation, brightness und color table. Wir können unter dem letzten Punkt einzelne Farben für Attributwerte festlegen, aber die Sättigung oder Helligkeit an die Werte knüpfen.

VISONE bietet anders als die Klassiker **UCINet** und **Pajek** nicht nur die Möglichkeit, Knotenpositionen anhand von ausgetüftelten Layout-Algorithmen berechnen zu lassen. Ist diese erst einmal festgelegt, können wir unter visualization und layout ein link routing layout auswählen, mit dem sich die Platzierung der Kanten optimieren lässt. Auch ein label placement layout mit verschiedenen Optionen steht bereit, um eine optisch ansprechende Positionierung der Knotenbeschriftungen zu erreichen.

Export der Grafik nach SVG

Der Export der Grafik in das SVG-Format erfolgt über Export → 2D → SVG → General oder Labels/Arcs/Edges oder Line Values – jede Methode hat ihre Vorzüge, zum Beispiel in Bezug auf die Manipulierbarkeit von Teilen der Grafik im XML-Editor oder **Inkscape**.

Welche Funktion haben die Exporttypen?

- General: im Browser ist nur das Netzwerk zu sehen. In **Inkscape** kann man jeden einzelnen Knoten, jede Kante, jedes Label einzeln zur Bearbeitung auswählen oder löschen. Man kann aber auch mit dem Auswahlwerkzeug ein Rechteck um bestimmte Elemente ziehen und diese dann bearbeiten oder löschen.
- Labels/Arcs/Edges: im Browser finden sich oben links zwei rote Punkte beschriftet Arcs und VerticesLabels, unten rechts finden sich zwei Checkboxen, mit denen die Kanten bzw. Labels ein- oder ausgeblendet werden können. In **Inkscape** kann man mit einem Klick auf ein Label oder einen Knoten alle Labels oder alle Knoten zur Bearbeitung auswählen oder löschen.
- Partition → Classes: im Browser sieht man oben links einen roten Punkt beschriftet mit Lines among clusters und eine Art Legende der Clusters in den Farben der jeweiligen Knoten (die Labels geben aber nur die laufende Nummer und nicht die Bezeichnungen der dargestellten Attribute wider). Unten rechts finden sich drei Checkboxen, mit denen die einzelnen Cluster ein- oder ausgeblendet werden können. In **Inkscape** kann man mit einem Klick auf ein Element eines Clusters den ganzen Cluster zur Bearbeitung auswählen oder löschen.
- Partition → Classes with semi-lines: im Browser sieht man oben links dieselbe Legende der Clusters in den Farben der jeweiligen Knoten (siehe oben). Unten rechts finden sich zwei Checkboxen, mit denen die einzelnen Cluster ein- oder ausgeblendet werden können, wobei die Linien zwischen Knoten, die verschiedenen Clustern angehören, halbiert werden. In **Inkscape** kann man mit einem Klick auf einen Knoten den gesamten Cluster zur Bearbeitung auswählen oder löschen, wobei ebenfalls die Kanten zu Knoten des anderen Cluster halbiert werden.

- Partition → nested Classes: im Browser sieht man unten rechts zwei Checkboxen, mit denen die einzelnen Cluster ein- oder ausgeblendet werden können, wobei einer als mit dem anderen verschachtelt (nested) dargestellt wird. In **Inkscape** kann man mit einem Klick auf ein Element eines Clusters den ganzen Cluster zur Bearbeitung auswählen oder löschen.
- Line Values → Classes: im Browser sieht man rechts mehrere Checkboxen (die Varianz zwischen dem niedrigsten und dem höchsten Linienwert wird in Einserschritte zerlegt), mit denen sich bestimmte Linienwerte ein- oder ausschalten lassen)
- Line Values → nested Classes: im Browser sieht man dasselbe wie oben, nur dass die Cluster ineinander verschachtelt sind.

Über Line Values → Options kann man dabei einstellen, wie die Linienwerte dargestellt werden sollen: mit verschiedenen Farben, Graustufen oder Strichstärken.

Es gibt grundsätzlich *kein WYSIWYG* **in Draw**. **Pajek** übergibt kein 1:1-Abbild der Grafik nach SVG. Ein Großteil der Einstellungen, die man in **Draw** an der Grafik vornimmt (Hintergrundfarbe, Faktoren der Knoten- oder Label-Größe, Linienstärke usw.), müssen in den Exporteinstellungen Export → Options erneut vorgenommen werden (siehe unten)

Ein erster Test des Layouts der SVG-Datei kann in einem SVG-fähigen Browser (siehe http://de.wikipedia.org/wiki/Scalable_Vector_Graphics) durchgeführt werden, indem in **Draw** die Exportoption HTML with SVG gewählt wird und die HTML-Datei zum Betrachten geöffnet wird.

Die Exporteinstellungen in Pajek Export → Options umfassen fünf Bereiche:

- EPS/SVG Vertex Default: Farben der Knoten, Umrandung und Labels, Form der Knoten, Label-Position, Schriftgröße der Labels. Wenn man die Knotengröße von dem Stellenwert eines Knotens abhängig macht, muss man den Standardwert (Radius des Startpunktes des Textes vom Mittelpunkt des Knoten aus berechnet) erhöhen. Bei unterschiedlich großen Knoten (nach Degree oder ähnlichem) wird die Festlegung des Radius etwas knifflig. Dies lässt sich dadurch umgehen, dass man Labels nur ab einer bestimmten Größe anzeigen lässt (siehe oben). Ansonsten nimmt man einen Kompromisswert zwischen dem Radius der kleinsten und der mittelgroßen Knoten. Die Labels der wichtigsten Knoten muss man dann eben in **Inkscape** von Hand etwas weiter nach außen platzieren.
- EPS/SVG Line Default: Farbe und Dicke von Linien, Größe und Position der Pfeile, Linienlabels. Bei einer Edge Width von 0.25 ließ sich die subjektiv beste Übereinstimmung der Strichstärken in **Draw**, im Browser und in **Inkscape** erzielen – sofern man nicht die Linienstärken in Graustufen ausgewählt hat. Dann muss der Wert auf 3 erhöht werden. Wer verhindern möchte, dass **Pajek** bei zweiseitigen Verbindungen je zwei Pfeile an SVG übergibt - was die Darstellung in **Inkscape** überfrachtet und die Bearbeitung erschwert – muss in das Feld Straight Lines ein Häkchen für das Ausschalten von reziproken Linien setzen (es sei denn, man möchte ein- und ausgehende Verbindungen gesondert angezeigt bekommen).
- Verschiedene Einstellungen: EPS, SVG, X3D, VRML Faktor Node Size, 3D-Effekte bei den Knoten in SVG (sehr schön!)
- Background Colors: drei verschiedene Hintergrundfarben können aus einem Popup-Menü ausgewählt werden. Standardmäßig sind alle drei auf „No“

eingestellt, was im Browser zu einem dunkelgrünen und in Inkscape zu einem weißen Hintergrund führt.

- EPS Border: Einstellung der Ränder unter EPS

Die Auflösung der SVG-Datei, die **Pajek** erstellt entspricht in etwa dem Bildschirm-Format. Das mag für kleine Netzwerke mit einer zweistelligen Zahl an Knoten reichen. Für große Netze mit mehreren hundert Knoten ist die Auflösung zu gering. Man muss in dem Fall einen kontraintuitiven Kunstgriff anwenden, nämlich über Options → Transform → Resize die Größe etwa um fünf multiplizieren. Man sieht dann zwar nur noch ein Fünftel des Netzwerks bei **Draw** auf dem Bildschirm, bekommt aber eine SVG-Datei mit einer brauchbaren Auflösung, mit der sich gute Poster für eine Präsentation erstellen lassen.

Bearbeitung der SVG-Datei im XML-Editor

Die von Pajek exportierte SVG-Datei kann nun *als Text* - als Reihe von Anweisungen an den Computer wie er die grafischen Elemente darstellen soll - mit dem **Open XML Editor** bearbeitet werden.

Das **SVG**-Format (Scalable Vector Graphics) ist ein Standard des World Wide Web Consortium zur Beschreibung zweidimensionaler Vektorgrafiken in der XML-Syntax. Die Extensible Markup Language, abgekürzt XML, ist eine erweiterbare Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten. Die Dokumentation der Elemente und Eigenschaften von SVG findet sich unter http://www.w3.org/TR/2003/REC-SVG11-20030114/

Öffnen der SVG-Datei im **Open XML Editor** über den File system explorer (linke Spalte) oder File → open. Die Datei beginnt mit drei Zeilen Kommentar (in <!-- ->). Das erste grafische Element in unserem SVG-Code beschreibt das Wort path, nämlich *den Umriss einer Form*. Zwischen den auf das Wort path folgenden Anführungszeichen werden bestimmte Punkte sowie die Linien zwischen diesen Punkten beschrieben – je komplexer die Form, desto umfangreicher wird der path! - in diesem Fall die kleine Spinne des Pajek-Logos.

Unsere Knoten verbergen sich in dem Element ellipse, unsere Kanten im Element line und unsere Labels in dem Element text. Jedem Element ist ein Kommentar in Klartext vorangestellt, damit wir wissen, um was es sicht handelt (z.B. Arc: 1 33 – also die Linie zwischen den Knoten 1 und 33 - oder Vertex: 14 – der Knoten mit der Nummer 14). Diese Elemente weisen jeweils verschiedene Eigenschaften auf: z.B. xy-Koordinaten, xy-Radien, Farbfüllungen, Randstärken oder Schriftarten. Jede dieser Eigenschaften kann individuell oder pauschal über bestimmte Prozeduren verändert werden.

Im XML-Editor finden wir den in den Exportoptionen eingestellten Radius für das Label (gemessen vom Mittelpunkt des Knotens her) wieder: Bei einem Ellipsenmittelpunkt von cx = 0 und cy = 0, befindet sich der untere linke Punkt des Labels bei x = 6,47 und y = 24,15 (bei einem Radius von 25 und einem Winkel von 285° für das Label).

Pauschale Änderungen an Elementen lassen sich mit den Werkzeugen der Transformation in SVG realisieren. Man kann Elemente verschieben (Translation), vergrößern oder verkleinern (Skalierung), drehen (Rotation) oder kippen (Neigung). Man muss dazu allerdings den Programmcode der ausgewählten oder aller Knoten, Kanten oder Labels durch Suchen und

Ersetzen um den Transformationsausdruck ergänzen. Dadurch kann natürlich die ganze Grafik durcheinandergeraten.

Vor der endgültigen Bearbeitung zu publizierender Grafiken müssen die Umlaute in der Text-Datei im normalen Text **Editor** entsprechend der ISO-8859-Codetabelle per Suchen und Ersetzen umgewandelt werden (Ä = Ä Ö = Ö Ü = Ü ä = ä ö = ö ü = ü ß = ß), weil sie sonst im SVG-Code und in der Grafik durch Fragezeichen ersetzt werden (das Semikolon gehört zum Code). Falls nur wenige Knoten vorliegen, kann man die Fragezeichen auch in **Inkscape** manuell umändern.

Bei einer großen Zahl von Labels mit deutschen Sonderzeichen bietet sich *eigentlich* an, die Änderung entsprechend der obigen Liste durch Suchen und Ersetzen im XML-Editor vorzunehmen. Leider transformiert **Pajek** schon bei der Ausgabe nach SVG die Umlaute pauschal in Fragezeichen. Man muss also die Sonderzeichen *vor der Öffnung der Net-Datei* durch **Pajek** ändern. Dann allerdings wandelt Pajek das kaufmännische „und“ (&) in & um, so dass aus einem kleinen „ä“ nicht der Ausdruck ä, sondern der Ausdruck ä wird. Wenn man hinterher im XML Editor das #38; wieder pauschal löscht, kommt man ans Ziel der Operation! Im **Open XML Editor** ruft man dazu durch strg-r den Modus Replace Text auf.

Bearbeitung der SVG-Datei in Inkscape

Öffnen der SVG-Datei mit **Inkscape** Datei → Öffnen.

Da die SVG-Datei von **Pajek** im Querformat gespeichert wurde, bietet es sich an, auch das **Inkscape**-Dokument auf Querformat umzustellen Datei → Dokumenteneinstellungen → Querformat

Um in die Grafik hinein- oder herauszuzoomen werden die Plus- und Minustaste des Ziffernblocks benutzt.

Änderungen an einzelnen Knoten werden von **Inkscape** nicht in den ursprünglichen SVG-Code geschrieben, sondern **Inkscape** fügt Anweisungscode ein, durch den die Veränderungen am ursprünglichen Code beim Ausführen der Datei vorgenommen werden.

Das Verschieben von Knoten sollte man im Grafikmodus bei **Pajek** vornehmen, weil in **Inkscape** die Linie, der Pfeil, und das Label nicht mitgezogen werden. Diese lassen sich zwar gruppieren und dann gemeinsam verändern, aber dabei handelt es sich um eine Vergrößerung oder Verkleinerung der gruppierten Elemente und nicht um eine Verschiebung des Knotens. Mit ein wenig Übung hat man allerdings schnell heraus, wo man anfassen muss, um ein Ensemble aus Knoten, Kante und Label – nacheinander! - zu verschieben. Dieses Verfahren bietet sich aber wirklich nur bei sehr wenigen zu korrigierenden Knoten an.

Um einzelne Teile der Grafik in **Inkscape** zu bearbeiten, markiert man die gesamte Grafik, öffnet mit Strg-F ein Menü Suchen, klickt auf Auswahl durchsuchen und wählt den zu bearbeitenden Typ aus, z.B. Alle Formen, und klickt auf Suchen. Man kann so die Farbe aller Knoten, Kanten oder Labels ändern, sie vertikal oder horizontal (neu) ausrichten, Teile von ihnen löschen usw.

Dasselbe kann man selbstverständlich auch mit einzeln ausgewählten Knoten, Kanten oder Labels tun, indem man sie zur Bearbeitung anklickt.

In **Inkscape** ist es zum Beispiel möglich, alle Knoten mit einem Farbverlauf zu versehen. Dadurch ist es je nach Auswahl der zwei oder mehr Farben möglich, dass die Kanten, die

jeweils bis zum Mittelpunkt des Knotens gezogen werden, teilweise sichtbar werden. Man sollte also auf transparente Farben verzichten.

Wenn man in **Inkscape** für eine Grafik Titel oder eine Legende mit Erläuterungen zu den verschiedenen Farben erstellt, und diese auch für andere Grafiken über dasselbe Netzwerk verwenden will, geht das anscheinend nur, indem man alles bis auf den Titel oder die Legende löscht und diese dann als neue SVG-Datei abspeichert Datei → Speichern unter und dann von der nächsten zu bearbeitenden Grafik-Datei aus importiert Datei → Importieren. Ein Austausch von Elementen zwischen verschiedenen SVG-Dateien über Copy & Paste ist offenbar nicht möglich.

Eine weitere Möglichkeit, die erst ein grafischer Editor wie **Inkscape** bietet, ist die Verwendung von Bitmap-Dateien (ein historisches Foto) als Hintergrundbild für das historische Netzwerk. Das lässt sich auf verschiedenen Wegen realisieren: Man kann zum Beispiel eine Bitmap-Datei in das Arbeitsblatt kopieren bearbeiten → einfügen, dann auf einer neuen Ebene Ebene → Ebene hinzufügen den Graphen importieren (und dabei den weißen Hintergrund löschen). Nachdem man beide Ebenen übereinander ausgerichtet hat, lässt sich die Transparenz der Hintergrundebene über die kleine Schaltfläche unten links (rechts neben den Einstellungen für Füllung und Konturlinie) oder im Ebenen-Menü in Prozent festlegen.

Das Arbeiten in Ebenen ist in **Inkscape** allerdings etwas knifflig, vor allem das Treffen der richtigen Ebene mit dem Auswahlwerkzeug. Von daher empfiehlt es sich vielleicht zunächst die Ebene zu präparieren, in der man am meisten zu ändern hat (vorzugsweise die Datei mit dem Netzwerk, in der zum Beispiel Labels zu verschieben sind). Diese speichert man dann ab und lädt sie wieder gemeinsam als entsprechende Ebene einer zu erstellenden Datei mit Hintergrund-Foto, Netzwerk-Ebene und der Ebene mit Titel und Legende.

Ausgabemöglichkeiten der SVG-Datei von Inkscape

Drucken der SVG-Datei über Datei → Drucken

Ausgabe als Bitmap-Datei (PNG) über Datei → Bitmap Exportieren mit einem Menü, in dem man Maße, Auflösung und Dateinamen festlegen kann

Ausgabe als PDF-Datei über Datei → Speichern unter und dann legt man unten rechts in dem Auswahlmenü als Format PDF via Cairo fest. Wie sich zeigt, wird der Wert für die Transparenz eines eventuellen Hintergrundbilds nicht eins zu eins übernommen (subjektiv war das Bild im PDF doppelt so hell wie eingestellt). Man wird da experimentieren müssen.

Weitere Formate, in denen **Inkscape** die Datei abspeichern kann, sind unter anderem PostScript und Encapsulated PostScript, Enhanced Metafile, das OpenDocument-Format von **Open Office Org** oder LaTeX.

Statische Darstellung dynamischer Netzwerke

Netzwerkgrafiken sind normalerweise statisch, Momentaufnahmen zu einem bestimmten Zeitpunkt. Manchmal sind wir aber mehr an der *Entwicklung* eines Netzwerks interessiert als an einem bestimmten *Zustand*; in unserem Fall zum Beispiel an der Entwicklung der Kenntnis der Kölner Gestapo von bestimmten Widerstandsgruppierungen. Um das darzustellen, müssen wir leider das von einem Layout-Algorithmus erzeugte Netzwerk zerstören, da wir die X-Achse der Visualisierung zweckentfremden müssen, um über sie den Zeitablauf zu

visualisieren. Dieses Verfahren eignet sich nur für ganz bestimmte Netzwerkstrukturen und ganz bestimmte Fragestellungen.

Der wohl einfachste Weg besteht darin, in Excel den Abstand zwischen dem ersten und dem letzten Tag einer Zeitreihe zu bestimmen. Durch diese Zahl der Tage teilen wir dann den Abstand zwischen dem ersten und dem letzten X-Punkt. Da Pajeks Fenster in X-Richtung genau eine Einheit breit ist, setzen wir den ersten Punkt bei 0,02 und den letzten bei 0,98, so dass unser Abstand zwischen x1 und xn 0,96 Einheiten beträgt. Bei einer Entwicklung über 100 Tage hätten wir dann einen Platz von 0,0098 Einheiten pro Tag. Nun können wir die Werte für die jeweiligen Abschnitte berechnen (Zahl der jeweiligen Tage multipliziert mit 0,0098) und erhalten damit eine proportionale Verteilung der Knoten über die Zeit. Diese können wir dann in der Net-Datei mit den dort berechneten X-Koordinaten austauschen (Vorsicht, die X-Koordinaten stehen in der zweiten Spalte).

Der Nutzen dieser Darstellung ist begrenzt. Für Personen, die an Spring-Embedder-Layouts gewöhnt sind, ist die Grafik zunächst vielleicht eher verwirrend als erhellend. Intuitiv erschließt sich die Visualisierung nicht so leicht wie ein Netzwerk-Graph. Sie hat dennoch ihre Vorzüge, jedenfalls so lange, wie wir in der Geschichtswissenschaft noch vorrangig über bedrucktes Papier kommunizieren.

Animation dynamischer Netzwerke mit Pajek und SVG

Ein anderer Weg wäre die Animation von dynamischen Netzwerken über einen Zeitfaktor, der dafür sorgt, dass Knoten und Kanten zum richtigen Zeitpunkt auftauchen oder verschwinden, wenn die dazu gehörigen Personen im historischen Netzwerk auftauchen oder verschwinden. Um das technisch umzusetzen, kann man den Graphen mit dem Gesamtnetz, in dem eine Partition die einzelnen Zeiteinheiten (z.B. Tag der Vernehmung) enthält, über Partition → Classes nach SVG exportieren. Man exportiert die Datei dann mit **Inkscape** als *Gesamtnetz(n)* in das PNG-Format. Danach löscht man die in der zeitlich letzten Class befindlichen Knoten und die Kanten zu diesen Knoten und speichert den Rest unter *Gesamtnetz(n-1).png*. So verfährt man weiter bis zur letzten, also chronologisch ersten, Zeiteinheit. Die einzelnen PNG-Dateien lassen sich dann schön in eine Präsentation mit **PowerPoint** oder **Open Office Impress** einbinden. Es ist aber auch möglich mit dem freien Open-Source Animationsprogramm **Pencil** eine Animation zu erstellen, indem in jeden Container für eine Zeiteinheit eine der PNG-Grafiken platziert wird. Diese kann dann in das verbreitete Flash-Format (*Animation.swf*) exportiert werden.

Diese Vorgehensweise hat allerdings den Nachteil, dass die Zeitpunkte *als Attribute der Verbindung* (Tag der ersten Aussage über eine Verbindung) unter den Tisch fallen, da wir sie der **Pajek**-Logik folgend als Attribute der Person (Tag der ersten Vernehmung) dargestellt haben. Wir verlieren also recht viel an Komplexität. Wir haben dann, um die Zeitpunkte der Erwähnung einer Verbindung transparent zu machen, manuell in der von **Pajek** generierten NET-Datei die Kanten eingefärbt, für jeden weiteren Tag, an dem eine Verbindung in der Vernehmung genannt wurde, ein neuer Farbton oder eine neue Farbe. Da wir keine Partition zur Bestimmung der Knotenfarbe brauchen, können wir den Graphen allerdings nicht über Partition → Classes nach SVG exportieren und dann in **Inkscape** auch nicht ganze Classes auf einmal löschen.

Im Bereich der Netzwerkanimation macht ästhetisch übrigens durchaus Sinn, was sich bei statischen Grafiken als verwirrend erwiesen hat: die Farben für die Knoten der einzelnen Zeitpunkte harmonisch abgestuft auszuwählen. Es gibt allerdings das Problem, dass in Pajek ca. doppelt so viele Farbennamen vorhanden sind wie auswählbare Farben für die Partitionen.

Wer seine Knoten so einfärben will wie die Kanten, muss also improvisieren bzw. das Einfärben erst im SVG-Code vornehmen.

Ebenfalls knifflig ist das Problem doppelter Verbindungen: Will man konsequent sein, darf das dynamische Netz nur je eine Verbindung zwischen zwei Personen enthalten. Sonst legt **Pajek** mehrere Linien in verschiedenen Farben übereinander, wobei nicht immer die jeweils passende zu sehen ist. Auch das Löschen der Linien in **Inkscape** wird dadurch aufwändiger.

Die Programmierer von **Pajek** haben mit **PajekToSVGAnim** (http://vlado.fmf.uni-lj.si/pub/networks/pajek/SVGanim/default.htm) ein Programm entwickelt, mit dem dynamische Netzwerke animiert werden können. Das Programm verlangt aber eine sehr spezielle Struktur der zu Grunde liegenden Datei, die genauso überkomplex und schlecht erklärt ist, wie **Pajek** selbst.

Die weitaus elegantere Lösung ist die *Animation mit SVG*. Sie hat aber, das sei gleich vorausgeschickt, einen gravierenden Nachteil: Programme, die die Animationen darstellen können, sind dünn gesät. Nur mit dem Nischen-Browser **Opera** (http://www.opera.com/) lässt sich die Netzwerk-Animation ohne weitere Zusatzprogramme problemlos betrachten (und überrascht feststellen, dass die Pajek-Spinne am unteren Bildrand entlang krabbelt)! Nach Installation des **Adobe SVG Viewers** (http://www.adobe.com/svg/viewer/install/) als Browser-Plugin ließ sich die Animation auch im **Internet Explorer** betrachten. **Firefox** hat zwar eine gute eigene SVG-Engine, Animationen laufen aber nur mit der Beta-Version 6.0 des Viewers (http://www.adobe.com/svg/viewer/install/beta.html) stabil. Dazu muss vorher das Browser-eigene SVG-Tool deaktiviert werden (man tippt about:config in die Adresszeile und ändert dann den Wert von svg.enabled auf false). Ein standalone-Programm, das SVG anzeigen kann, ist mir nicht bekannt. Die **Inkscape**-Entwickler planen für die Programmversion 0.54 den „Full Animation Support“, d.h. Animationen im Programm betrachten, bearbeiten und exportieren zu können (gegenwärtig läuft der Start von Version 0.48). Konverter von SVG-Animationen in das Flash-, AVI- oder GIF-Format habe ich gleichfalls noch keine entdecken können. Wie es scheint, ist die Animation mit SVG so lange ein relativer Holzweg, wie *Inkscape* keine Ressourcen zur Bearbeitung und zum Export zur Verfügung stellt.

Wie bei jeder Animation muss man auch bei der Animation von Netzwerken mit SVG eine Art Drehbuch entwickeln, in dem steht, was zu welchem Zeitpunkt passieren soll. Hier ein Beispiel:

Zeitpunkt	Datum	neu	Knoten		Kanten		Bemerkung	Sekunde
			Name	Nr	von	zu		
1	30.01.1935	Vertex	Arendt, Alfred	4			sagt aus	5
2	30.01.1935	Arc	Arendt, Alfred		4	2		10
2	30.01.1935	Arc	Arendt, Alfred		4	3		10
2	30.01.1935	Arc	Arendt, Alfred		4	5		10
3	30.01.1935	Vertex	Alfred	2			wird genannt	15
3	30.01.1935	Vertex	Apfel, -	3			wird genannt	15
3	30.01.1935	Vertex	Baer, Gustav	5			wird genannt	15

Unser Drehbuch enthält für jeden Zeitpunkt der Animation bestimmte Anweisungen dahingehend, welche Knoten (Nummer nach Pajek) und Kanten (von Nummer zu Nummer) darzustellen sind. Unter Zeitpunkt verstehen wir die einzelnen Sequenzen, in denen Knoten

oder Kanten auf dem Bildschirm erscheinen. Als Abstand zwischen den Sequenzen haben wir 5 Sekunden festgelegt.

Die Animation mit SVG ist recht aufwändig. Jedes Element muss *einzeln* im **XML Editor** animiert werden, das heißt jede Zeiteinstellung eines Elements per Hand eingetragen werden. Es gibt gegenwärtig eine handvoll von Projekten für die Entwicklung eines SVG-Animations-Editors, aber es scheint bisher keines so weit gediehen zu sein, dass es diese Arbeit automatisieren könnte. Wie bei allen anderen Darstellungsformen von Netzwerken gilt es auch hier, die spezifischen Stärken dieser Darstellungsform zu ermitteln und zu nutzen. Das Animieren von großen Netzwerken erfordert recht viel Zeit. Das Medium der Animation sollte also auf Fälle beschränkt werden, in denen die spezifischen Vorteile der Animation gegenüber der statischen Darstellung zum Tragen kommen. Hinzu kommt, dass Animationen nicht zu den üblichen Medien historiografischer Wissensproduktion passen Eine *gedruckte* ‚Animation' ist ein Widerspruch in sich! Als Haupteinsatzgebiet von Animationen kommen daher vor allem Vorträge in Frage. In Texten abgedruckte Links, die auf HTML-Seiten mit eingebetteten SVG-Animationen verweisen, sind zwar auch denkbar, aber doch die weniger elegante Lösung.

Die SVG-Befehle animate und set umfassen eine Vielzahl von Anweisungen, zum Beispiel den Startzeitpunkt einer Animation, ihre Dauer oder auslösende Ereignisse (z.B. Mausklick). Er bezieht sich auf bestimmte Eigenschaften von Elementen, etwa auf ry, den Radius unseres Knotens in vertikaler Richtung, und ändert die darin festgehaltenen Werte über die Attribute from und to, die den Zustand des Elements am Beginn und am Ende der Animation beschreiben. SVG berechnet dabei die Zwischenstufen für eine flüssige Animation während der Durchführung selbst.

Für die Animation der Kanten benutzen wir den Kunstgriff, das Element zunächst unsichtbar zu machen und erst zum entsprechenden Zeitpunkt in der Animation einzublenden. Das geschieht dann allerdings etwas abrupt. Der Versuch, die Kanten zunächst in der Hintergrundfarbe darzustellen und dann in die korrekte Farbe zu animieren, führte zu Lücken in durchkreuzten anderen Kanten. Der eleganteste Weg besteht darin, die Kanten sanft ausfahren zu lassen. Dazu animiert man die Zielkoordinaten des Knotens - zunächst überschreibt man sie mit den Startkoordinaten, so dass die Linie nicht zu sehen ist, dann fügt man die korrekten Zielkoordination über die Animation wieder ein -, was allerdings den Aufwand sehr erhöht. Die Linie wird dann, je nach Dauer der Animation, stufenlos ausgefahren. Auch die Labels werden zunächst unsichtbar gestellt und dann in die Sichtbarkeit animiert.

Auch wenn bei Animationen mit SVG vieles von Hand gemacht werden muss, lässt sich doch einiges automatisieren. Man kann etwa über ein paar Suchen- und Ersetzen-Vorgänge im normalen **Editor** – der längere Suchen- und Ersetzen-Ausdrücke bearbeiten kann als der **Open XML Editor** - samtliche Kanten zunächst identisch, d.h. mit den selben Zeitangaben, animieren. *Vorsicht*: auch der **Editor** schneidet überlange Such-Ausdrücke ab! Man muss dann eventuell im zweiten Schritt das Abgetrennte ebenfalls über Suchen und Ersetzen noch anhängen. Außerdem sollte man aufpassen, dass sich keine Zeilenschaltung in den Ersetzen-Ausdruck einschmuggelt, weil sonst die darauf folgenden Zeichen gelöscht werden.

Im Fall der Knoten werden nun im ersten Schritt die Radien rx und ry auf null gesetzt. In Schritt zwei werden zwei animate-Ausdrücke eingefügt, die den Inhalt der Elemente rx und ry auf den Wert des ursprünglichen Radius setzt. Dazu kann man z.B. nach dem Ausdruck </ellipse> suchen (beendet die Ellipsen-Anweisung) und ihn durch <animate attributeName="rx" attributeType="XML" begin="#s" dur="#s" fill="freeze" from="0" to="#"/></ellipse> ersetzen, und dann noch einmal durch <animate attributeName="ry" attributeType="XML" begin="#s" dur="#s" fill="freeze" from="0" to="#"/></ellipse> ersetzen (# steht für eine beliebige Zahl).

Bei den Kanten ergänzen wir im line-Element den Ausdruck visibility="hidden" als letztes Attribut. Dann animieren wir das Unsichtbarkeits-Attribut, indem wir das Element </line> durch <set attributeName="visibility" attributeType="CSS" to="visible" begin="#s" dur="#s" fill="freeze" /></line> ersetzen.

Die Labels lassen sich analog zu den Kanten vor der Animation durch Einfügung des Ausdrucks visibility="hidden" im Element text ausblenden. Dann sucht man den Ausdruck #px;fill:rgb(#,#,#)" und ersetzt ihn durch #px;fill:rgb(#,#,#)" visibility="hidden". Im zweiten Schritt ersetze ich den Ausdruck </text> durch <set attributeName="visibility" attributeType="CSS" to="visible" begin="#s" dur="#s" fill="freeze" /></text>. Auf eine ähnliche Weise lässt sich auch ein Untertitel des Graphen animieren, der die jeweiligen historischen Zeitpunkte für jeden Schritt der Animation anzeigt.

Ästhetisch gibt es viele Optionen. Ihre Verwendung hängt natürlich vom jeweiligen Erkenntnisinteresse ab. Auch wollen der Aufwand und der mögliche Ertrag gut abgewägt werden. Wenn man zum Beispiel jeden einzelnen aus einer Gruppe von Knoten, die zum selben Zeitpunkt erscheinen, zwei Zehntelsekunden später als den nächsten ins Bild bringt, entwickelt sich die Animation harmonischer und die einzelnen Verbindungen gehen nicht im Gesamtbild unter. Man verliert dann aber die Möglichkeit, die Zeitpunkte dieser Knoten pauschal noch einmal durch Suchen und Ersetzen zu verändern.

Animation dynamischer Netzwerke mit SoNIA und NEVADA

Ein weiterer Weg zur Animation dynamischer Netzwerke ist das auf JAVA aufsetzende Package **SoNIA** (Social Network Image Animator) (http://www.stanford.edu/group/sonia/). SoNIA kann mit den NET-Dateien arbeiten, die **Pajek** produziert. Die Einträge zu den Vertices und Arcs müssen allerdings in eckigen Klammern um Angaben über den Zeitraum ergänzt werden. Der hinzugefügte Ausdruck [21-39] bedeutet in diesem Fall zum Beispiel, dass ein Knoten in der 21. Zeiteinheit in der Animation auftaucht und bis zur 39. Zeiteinheit zu sehen bleibt. Auch hier ist vor allem Animieren die Erstellung eines Drehbuchs nützlich.

So weit, so schnell, so einfach. Die Tücke liegt darin, dem Programm abzugewöhnen, die Position aller Knoten zu jeder Sekunde völlig neu zu berechnen, was die Entwicklung des Netzwerks mehr verwirrt als erhellt. In **SoNIA** öffnet man über Load Files die zu animierende Datei. Im Menü Create Layout legt man den Animationstyp und den Layouttyp fest. Da ich meine Koordinaten aus **Pajek** behalten wollte, wählte ich als Layout coordinates from original file. Dann geht man in dem aufpoppenden Fenster mit dem Netzwerk auf Timeline und wählt den letzten slice. Über Layout → Apply Layout legt man nun als Starting Coordinates coordinates from original file fest, klickt auf Apply to current, und wählt im Menu Rescaling rescale layout to fit window. Nun sieht man das Netzwerk im von Pajek berechneten Endzustand. Dann wählt man in der Timeline den vorletzten slice und geht wieder auf Layout → Apply Layout. Dort lauten die Parameter nun bei den Starting Coordinates from next slice und bei Rescaling none und wendet dieses Layout auf alle früheren slices an Apply in reverse. Nun lässt sich in der Timeline mit dem dritten Button die ganze Animation anschauen. Von den mühsam aus der Pajek-Anleitung ausgewählten Farben bleibt dabei aber nicht viel übrig. Mehr als vier verschiedene Farben für Kanten scheint Sonia nicht zu kennen.

Über Export lassen sich die Einzelbilder der einzelnen slices und über Export Network die ganze Animation als Video exportieren, entweder als **Flash**- oder – nach Installation entsprechender Software – als **QuickTime**-Movie. Während der Exportalgorithmus für

Quicktime auch mit größeren Netzwerken mühelos klarkommt, geht der für das Flash-Format ab einer bestimmten Zahl von Knoten und Kanten in die Knie.

Die Netzwerke, die **SoNIA** animiert sind allerdings recht filigran, die Elemente werden nicht sanft animiert, sondern abrupt eingeblendet. Außerdem lässt sich kein Titel einfügen, statt dessen muss die laufende Uhr oben links hingenommen werden. Ästhetisch hat die Animation mit SVG eine Menge mehr zu bieten, verlangt dafür aber auch eine Menge mehr Arbeit. Für welchen Weg man sich entscheidet, dürfte also davon abhängen, was mit der Animation geschehen soll, welchem Publikum sie präsentiert werden soll.

NEVADA (Network Visualization and Analysis for Dynamic Networks) setzt ebenfalls auf JAVA auf und ist eine Entwicklung einer Informatiker-Arbeitsgruppe am Lehrstuhl für Software-Technik der Uni Trier (http://www.st.uni-trier.de/gd/). Durch sieben verschiedene Menüs lassen sich die Eigenschaften von Knoten, Kanten, Layouts festlegen und die Analyse bestimmter Werte durchführen. Das Wort „Animation“ kommt, trotz des Anspruchs, ein Werkzeug für ein „Dynamic Graph Drawing“ entwickelt zu haben, im einführenden Text des Manuals nicht vor!

Die Visualisierung dynamischer Netzwerke lässt sich lässt sich nur über den Import einer NET-Datei von **Pajek** starten, weil nur dann überhaupt ein Button gezeigt wird, der das Hinzuladen weiterer Sequenzen erlaubt, den späteren Frames unserer Animation. Meine ursprüngliche Vorstellung, ein Netzwerk in Nevada erst aufzubauen und dann, Frame für Frame, weiterzuentwickeln, lässt sich jedenfalls nicht realisieren, da die einzelnen Frames nicht als NET-Datei gespeichert werden können und das Programm beim Laden eines Frames mit einer Datei im programmeigenen GAML-Format abstürzt. Anscheinend muss jede Sequenz einzeln in **Pajek** erstellt werden und dann nach **NEVADA** geladen werden.

Die fertige Animation kann dann in das SVG-Format exportiert werden. Sowohl der **Open XML-Editor**, **Inkscape**, **Firefox** und der **Internet Explorer** haben sich geweigert, die SVG-Datei zu öffnen, da sie nicht den Regeln der „Wohlgeformtheit“ bei XML-Dokumenten entspricht. Erst wenn man die Datei in einen **Text-Editor** lädt, den Inhalt in ein leeres Dokument im **Open XML Editor** kopiert und abspeichert, erhält man eine reguläre SVG-Animation. Im Browser sieht man dann oben links zwei graue Dreiecke, mit denen die Animation schrittweise vorwärts und rückwärts gespielt werden kann. Da es keine Exporteinstellungen gibt, kann man z.B. die Labels der Knoten nicht exportieren. Außerdem wird auch nicht korrekt exportiert. In Fall meines Testnetzwerks wurden dabei Knoten aus dem ersten Frame zu Isolates im zweiten und dritten Frame, die in den **NEVADA**-Fenstern im dritten Frame nicht zu sehen sind.

Angesichts gravierender Bugs im Programm und der letzten Beta-Version vom 24. März 2009 entsteht der Eindruck, dass das Programm nicht weiterentwickelt wird. In der vorliegenden Form ist eher von einer Verwendung abzuraten.

Außerdem gibt es eine „pilot version“ von **dyNet**, einem neuen Software-Projekt zur „dynamic network visualization“ (http://www.markowetzlab.org/software/networks.html). Leider wird hier wieder ein neues XML-basiertes Datenformat geführt (XGMML). Die Informationen auf der Website sind sehr knapp, das Programm nicht intuitiv bedienbar. Es soll aber demnächst ein Aufsatz des Programmiers erscheinen. Vielleicht sieht man dann, in welche Richtung es geht.

Animation dynamischer Netzwerke mit VISONE

Seit der Version 2.6 von **VISONE** gibt es einen weiteren Layout-Algorithmus: *Dynamic Layout* und die Möglichkeit der Animation dynamischer Netzwerke. Grundlage für die Animation sind selbstverständlich auch hier Frames mit verschiedenen Stadien eines Netzwerks. Dazu verwenden wir einzelne GraphML-Dateien, die wir über das zweitrechte Icon in der Menüleiste („opens the network collection manager“) zu einer Network Collection verknüpfen. In dem Menü Network Collections vergeben wir einen Namen und wählen aus einer Liste von Dateien diejenigen aus, die gemeinsam animiert werden sollen.

An diesem Punkt stellt sich die grundsätzliche Frage, ob die Position der Knoten über die Animation hinweg eher stabil sein soll oder ob jedes Frame-Netzwerk sein eigenes Layout behalten soll. Wenn das entschieden ist, wählt man für die geladenen Frame-Netzwerke unter dem Reiter visualization das Node Layout *Dynamic Layout*. Hier lässt sich unter stability die Stabilität der Knotenposition zwischen null und hundert Prozent festlegen, und hier kann entschieden werden, welcher Referenztyp das Layout der einzelnen Frame-Netzwerke bestimmt, also zum Beispiel das Frame-Netzwerk mit den meisten Knoten oder das vereinte Gesamtnetzwerk (*aggregation network*). Dann klickt man auf layout! und alle Frame-Networks werden gleichförmig angeordnet.

Wenn alle Frames geladen und verknüpft sind, kann man mit dem Icon ganz rechts („animate between successive networks“) die Animation erstellen lassen, die in einem gesonderten Fenster abläuft. Das Animationsfenster zeigt oben links die Bezeichnung der Collection und bietet die Möglichkeit die Geschwindigkeit des Ablaufs zu verändern, außerdem wird angezeigt, welcher Frame gerade zu sehen ist. Man klickt auf den Button mit dem üblichen nach rechts zeigenden Dreieck und lässt die Animation ablaufen. Zurück geht es offenbar nur schrittweise.

Die Animation lässt sich leider nicht speichern oder exportieren. Man braucht vorerst noch ein Programm mit Screencast-Option (Bildschirm-Aufnahme), um die Animation abzufilmen. Dafür hat sich **CamStudio**, das relativ einfache und daher auf vielen Rechnern gut arbeitende Open-Source-Screencast-Programm als brauchbar erwiesen (http://camstudio.org/). Man klickt auf Region und wählt den ganzen Bildschirm oder eine in Pixel-Koordinaten (oben links = 0/0) anzugebende Region, die abgefilmt werden soll, klickt auf den großen roten Punkt und lässt dann in **VISONE** die Animation ablaufen. Am Ende der Animation klickt man auf das große blaue Quadrat und stoppt die Animation. **CamStudio** erstellt dann eine AVI-Datei, die in jedem Media-Player laufen sollte, kann aber das Video auch in das Flash-Format exportieren. Mit Programmen wie dem **AVI-Splitter** (http://www.brizsoft.com/avisplit/) kann man das Video hinterher schneiden. Der Export von Animationen vermutlich unter anderem in das SVG-Format steht bei den Programmierern von **VISONE** bereits auf der Liste der *planned features*.

Weitere Ausgabemöglichkeiten

Eine Slideshow aus mehreren Grafiken eines Netzwerks (z.B. ein Graph für jeden Zeitpunkt einer Entwicklung) lässt sich mit den Bitmap-Dateien, die **Draw** oder **Inkscape** ausgibt, in **MS Powerpoint** oder in **Open Office Impress** erstellen. Auch diese kann wichtige heuristische Informationen über das Netzwerk und die Akteure liefern. In unserem Projekt lässt sich so etwa das Schneeballsystem erhellen, mit dem die Kölner Gestapo die Widerstandsgruppen immer wieder aufgerollt hat. Man kann die Folien als Präsentation

ausgeben aber auch von **Impress** in verschiedene andere Formate exportieren, zum Beispiel in das Videoformat Flash (Internet-Standard, der von jedem aktuellen Browser dargestellt werden kann).

Verknüpfung geografischer und relationaler Daten

Wir geben aus einer Datenbank unter diversen anderen Knotenattributen Wohnort und Straße aus und erstellen daraus eine Tabelle, die zumindest die Spalten Label (Name), Ort und Straße enthält. Die Spaltenbezeichnung „Label“ braucht *Gephi*, um automatisch das richtige Label zuzuweisen.

Wir haben zwei Optionen: entweder lassen wir 1.) durch http://www.batchgeo.com/de/ unsere Tabelle in eine Google-Maps-Karte verwandeln, die wir aber nicht abspeichern können. Wir erhalten aber per Email eine URL unter der diese Karte auch später zu finden sein soll. Auf dieser Site finden wir dann auch einen Button zum download der zu Grunde liegenden KML-Datei. Oder wir geben 2.) unsere Tabelle auf http://stevemorse.org/jcal/latlonbatch.html?direction=forward in das linke Feld „Adresses“ ein und wählen rechts „Latitude, Longitude“. Es erfolgt dann eine „Batch Conversion of Address to Latitude/Longitude“, also eine Stapelverarbeitung von einer Reihe von Adressen in eine Liste von Geokoordinaten (rechter Kasten).

Für 500 Adressen benötigen beide Websites kaum fünf Minuten Rechenzeit. Die Liste von Steve Morse kopieren wir in unsere Tabelle, durch klick auf Daten → Text in Spalten und Angabe des Trennzeichens erhalten wir zwei Spalten, die wir „Latitude“ und „Longitude“ nennen. Leider sind die Daten nicht zu hundert Prozent zuverlässig, wir müssen also stichprobenartig unsere Daten überprüfen. Dazu können die Geokoordinaten helfen, die wir in der HTML-Datei finden, die batchgeo zur Verfügung stellt. Sie lassen sich über die zum download zur Verfügung gestellte kml-Datei mit Excel über XML-Import wieder in Tabellenform bringen. Wie es aussieht, gibt es aber durchaus grundsätzliche Abweichungen zwischen den Koordinaten, je nachdem von wem sie bezogen wurden. Erste Stichproben zeigen, dass die Daten von batchgeo verlässlicher sind als die von Steve Morse, was vermutlich mit Rundungsoperationen zusammenhängt.

Diese Tabelle können wir dann nach dem Abspeichern als csv-Datei aus *Gephi* über Data Laboratory und Import Spreadsheet laden. Sie sollte tunlichst keine leere erste Zeile haben, sonst erhalten wir das Trennzeichen als Überschrift über jede einzelne Spalte und eine Fehlermeldung. Die vorgeschlagenen Datenformate beim csv-Import behalten wir bei mit Ausnahme von „Latitude“ und „Longitude“, wo wir als Datenformat „float“ (Fließkommazahl) wählen.

In *Gephi* wählen wir im Modus Overview das GeoLayout, ein Plugin, das wir vorher installieren müssen. Wir legen den Maßstab (scale) fest – je geringer die Abstände unserer Knoten desto höher muss er sein – sagen dem Programm, in welchen Feldern wir „Latitude“ und „Longitude“ gespeichert habe, den gleichnamigen nämlich. Mit einem klick auf run wird nun eine maßstabsgerechte Karte erstellt, die wir später als pdf- oder svg-Datei ausgeben können. Die svg-Datei können wir in *Inkscape* über eine zweite Ebene mit einer hochauflösenden Karte möglichst ebenfalls im svg-Format legen, so dass wir die Daten ansprechend präsentieren können.

Schluss

Man sieht oft erst hinterher in einer vergrößerten PDF-Datei wo noch Positionen von Knoten optimiert werden können. Man wird in dem Fall nicht vermeiden können, die letzten Schritte zu wiederholen, also zunächst in **Pajek** die entsprechenden Knoten zu verschieben, dann den von **Pajek** eingefügten Code & im **Editor** zu ersetzen, dann den letzten Feinschliff in **Inkscape** oder im **Open XML-Editor** vorzunehmen und dann wieder ein PDF zur Veröffentlichung zu erstellen. Nur durch große Sorgfalt beim Arbeiten in **Pajek** kann dieses umständliche Nachbessern vermieden werden.

Die verschiedenen Programme zur Erstellung, Visualisierung und Weiterverarbeitung von Netzwerken haben alle ihre sehr spezifischen Stärken und Schwächen und es ist keineswegs grundsätzlich so, dass die einen Programme die Schwächen der anderen ausgleichen. Bei manchen Funktionen hat man also die Wahl zwischen verschiedenen Wegen, bei anderen ist man mangels Alternativen gezwungen, mit einer unvollkommenen Software Vorlieb zu nehmen. Für jede Anwendung der SNA-Software will also gut überlegt sein, worin das Ziel besteht, und welches Programm mit seinen besonderen Stärken dafür am besten geeignet ist.

Dr. Ulrich Eumann
NS-Dokumentationszentrum der Stadt Köln
ulrich.eumann@stadt-koeln.de