VISUALISIERUNG VON STATUS IN SOZIALEN NETZWERKEN

Diplomarbeit
von

Frank Müller

Fachbereich Mathematik und Statistik
Universität Konstanz

Februar 2000

Meiner Mutter und meinen Großeltern gewidmet

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Konstanz, den 24. Februar 2000 Frank Müller

Inhaltsverzeichnis

1 Einleitung **1**

2 Einführung in die Graphentheorie **4**

3 Soziale Netzwerke **7**
3.1 Ein kurzer, geschichtlicher Überblick 7
3.2 Grundlagen 8
3.3 Prominenzmaße 11

4 Visualisierung von Status **40**
4.1 Entwicklung und Ziel der Visualisierung 40
4.2 Netzwerksubstanz 46
4.3 Design 47
4.4 Algorithmus 51

5 Geschichtete Darstellung von Netzwerken **54**
5.1 Layerzuweisung 55
5.2 Kreuzungsminimierung 70
5.3 Zuweisung der Koordinaten 91

6 Schlußbemerkung **95**

A Implementation **98**
A.1 Softwarepakete 98
A.2 Module und Funktionen 100
A.3 Benutzerführung 107

B Graphische Beispiele **111**

Kapitel 1

Einleitung

Wir verbringen einen Großteil unserer Zeit mit anderen Menschen, angefangen bei der Familie, unserem Arbeits- oder Ausbildungsplatz bis hin zu Freizeitaktivitäten in Vereinen oder mit Freunden. Jede dieser Personengruppen kann als *soziales Netzwerk* beschrieben werden. Sowohl die graphische Darstellung sozialer Netzwerke, als auch die Betrachtung der sozialen Verflechtungen spielen in dieser Arbeit eine zentrale Rolle.
Bestandteile eines solchen Netzwerkes bilden die sozialen Einheiten (auch *Akteure* genannt) und die Beziehungen zwischen ihnen. Akteure müssen dabei nicht notwendigerweise Personen verkörpern, sondern können auch ganze Gruppen (z.B. Abteilungen einer Firma) oder auch geographische Einheiten (z.B. Städte) darstellen. Beziehungen wie "stehen in geschäftlicher Verbindung" oder "ist befreundet mit" sind typisch für soziale Netzwerke.
In der Mathematik bildet die *Graphentheorie* die Grundlage für diese Strukturen. Akteure können als *Knoten*, Beziehungen als *Kanten* eines *Graphen* dargestellt werden. Knoten werden dabei zumeist als Punkte in der graphischen Darstellung gezeichnet, Kanten als Linien oder Pfeile. Kapitel 2 beschäftigt sich mit der allgemeinen Definition von *Graphen* und den Eigenschaften, die für die Untersuchung eines *sozialen Netzwerkes* von Wichtigkeit sind.
Die graphische Darstellung von Netzwerkdaten wird auch als *Visualisierung* bezeichnet. Das Kapitel 4 befaßt sich mit der genauen Klärung des Begriffes *Visualisierung* und dessen Eigenschaften.
Ein *Layout*, also die räumliche Anordnung der graphischen Objekte, muß dabei bestimmten Bedingungen genügen, wie z.B. dem Kriterium der guten Lesbarkeit oder der Datentreue. Die Aussagen, die mittels dieses Layouts gemacht werden, sollten möglichst einfach und schnell zu erkennen sein. Daten dürfen durch die Umsetzung in ein Bild nicht verfälscht werden. Dies spricht auf der einen Seite für die Verwendung von Statistiken spricht. Andererseits

kann dabei die Übersicht über die Gesamtstruktur des Netzwerkes verloren gehen, was durch die hier benutzte Methode verhindert wird.
Betrachten wir die Struktur eines Netzwerkes etwas genauer. Es ist möglich durch die Lage der Knoten und Kanten ein Maß anzusetzen, durch das den Knoten innerhalb der vorliegenden Struktur ein Wert zugeordnet wird. Übertragen wir diese Werte wieder auf das soziale Netzwerk, so sprechen wir von der *Wichtigkeit* der einzelnen Akteure, die durch die auf der Struktur gemessenen Werte bestimmt ist. Es können dabei verschiedene Arten von Wichtigkeit unterschieden und berechnet werde. Damit befaßt sich das Kapitel 3.
Ein zentraler Gegenstand dieser Arbeit ist ein spezielles Maß, genannt *Status*. Die Größe der Statuswerte bestimmt die Position eines Akteurs im Layout. Neu ist, daß die Wichtigkeit oder in diesem Fall der Status nicht etwa durch Größe der Knoten, sondern durch die zugewiesenen y-Koordinaten im zweidimensionalen Layout visualisiert wird.
Ein Hauptanliegen dieser Arbeit ist es, die Verbindung zwischen den berechneten Daten, in unserem Fall den Statuswerten und dem Netzwerk herzustellen und graphisch sichtbar zu machen. Dafür werden Verfahren benutzt, die noch nicht in dieser Kombination zusammengestellt wurden.
Die hier verwendete Art der Visualisierung entsteht durch den *Algorithmus von Sugiyama*, der die Positionierung von *Knoten* und *Kanten* eines *Graphen* in drei Schritte unterteilt.

1. Die *Layerzuweisung*: In der ursprünglichen Form des Algorithmus werden die Knoten einzelnen Schichten, sogenannten *Layern* zugewiesen. Sie bestimmen die y-Koordinaten der Knoten und werden üblicherweise äquidistant gesetzt. Dies ist in der hier zu beschreibenden Vorgehensweise nicht der Fall, da die Höhe der Layer durch die berechneten Statuswerte bestimmt werden. Zusätzlich wird eine *Clusterung* vorgestellt, die mehrere Layer, die dicht beieinander liegen, verschmelzen kann.

2. Die *Kreuzungsminimierung* (Vermeidung möglichst vieler Kantenüberschneidungen): Bisherige Verfahren versuchen zumeist, Schicht für Schicht die Anzahl der Kreuzungen zwischen einem Layer und dem nächsten zu minimieren. Eine davon verschiedene Methode wird benutzt, um durch Kombination eines herkömmlichen Verfahrens mit dieser Methode bessere Ergebnisse zu erhalten. Die Effektivität dieser Kombination wird anhand von Testergebnissen dargestellt.

3. Die *Zuweisung der horizontalen Koordinaten*: Der letzte Schritt beschäftigt sich mit der horizontalen Positionierung der Akteure und

der zusätzlich eingefügten Knickpunkte der Kanten. Durch die oben beschriebene Verschmelzung der Layer, ist es nötig, weitere Knickpunkte zu integrieren und geeignet zu positionieren.

Neben den oben erwähnten Methoden wird in Kapitel 5 eine Vielzahl von verschiedenen Lösungsmöglichkeiten für jedes der drei Teilprobleme beschrieben. Im Anhang wird die Umsetzung des Themas durch eine Implementation in **C++** dargelegt. Die Klassen und Funktionen werden aufgeführt und erklärt, ebenso die Benutzung des Programmes und Möglichkeiten, die sich durch neu eingeführte Funktionen bieten.Die erhaltenen Ergebnisse werden am Ende der Arbeit durch einige graphische Beispiele veranschaulicht.

Kapitel 2

Einführung in die Graphentheorie

Die Struktur, die einem Sozialen Netzwerk zugrunde liegt, ist die eines *Graphen*. Um das Verständnis zu erleichtern und mit den Begriffen vertraut zu werden, wird hier eine kurze Einführung in das Thema gegeben.

Ungerichteter Graph:
Ein ungerichteter Graph $G = (V, E)$ besteht aus einer endlichen Knotenmenge V und einer Kantenmenge E, für die gilt:

$$E \subseteq \{\{u, v\} : u, v \in V\}$$

Gerichteter Graph:
Ein gerichteter Graph $D = (V, A)$ besteht aus einer endlichen Knotenmenge V und einer Menge A von *gerichteten* Kanten, für die gilt:

$$A \subseteq \{(u, v) : u, v \in V\}$$

Gerichtet bedeutet, daß Anfangs- und Endknoten einer Kante genau definiert sind, was bei ungerichteten Kanten nicht der Fall ist.
Eine Kante in einem Graphen verbindet im Normalfall zwei Knoten. Die Knoten bezeichnet man dann mit *adjazent* oder *benachbart*, Knoten und Kante sind *inzident*. Besitzt eine Kante nur einen inzidenten Knoten, spricht man von einer *Schleife* oder einem *loop*. Ein Knoten ohne inzidente Kanten heißt *isoliert*. Es wird der Fall auftreten, daß wir die Knoten eines Graphen in zwei Hälften unterteilen, zwischen denen Kanten existieren, innerhalb der Hälften nicht. Man spricht dabei von einem *bipartiten Graphen*. Ein bipariter Graph ist ein ungerichteter Graph $G = (V, E)$, in dem die Menge der Knoten V in zwei Mengen V_1, V_2 eingeteilt werden kann, so daß für eine Kante $\{u, v\} \in E$

gilt, daß entweder $u \in V_1$ und $v \in V_2$ oder $u \in V_2$ und $v \in V_1$. Da wir uns bei der Netzwerkanalyse stark mit den zu einem Knoten inzidenten Kanten beschäftigen und gerichtete und ungerichtete Kanten unterscheiden müssen, wird der *Grad* eines Knotens eine wichtige Rolle spielen. Der *Grad* eines Knotens u in einem ungerichteten Graphen ist die Anzahl der zu u inzidenten Kanten. Er wird mit $deg(u)$ bezeichnet. In einem gerichteten Graphen ist der *Eingangsgrad* eines Knoten u die Anzahl der in u einlaufenden Kanten, der *Ausgangsgrad* die Anzahl der aus u auslaufenden Kanten. Die Notation für einen Knoteneingangsgrad ist $deg_{in}(u)$, die für einen Knotenausgangsgrad $deg_{out}(u)$. Wir wollen uns aber nicht nur mit den *direkten* Verbindungen zwischen Knoten, also adjazenten Knoten beschäftigen, sondern insbesondere mit den *indirekten*, d.h. Verbindungen, die Knoten über mehrere Kanten und Zwischenknoten miteinander verbinden. In diesem Zusammenhang tritt der Begriff des *Weges* auf. Ein *Weg der Länge k* (bzw. *gerichteter Weg der Länge k*) in G (bzw. D) ist eine Knotenfolge $< v_0, v_1, \ldots, v_k >$ mit $\{v_{i-1}, v_i\} \in E$ (bzw. $(v_{i-1}, v_i) \in A$) für alle $1 \leq i \leq k$. v_k heißt von v_0 aus *erreichbar*, falls ein Weg zwischen v_0 und v_k bzw. von v_0 nach v_k existiert. Falls $v_i \neq v_j$ für alle $0 \leq i, j \leq k$, heißt der Weg *einfach*. Gilt für einen Weg $< v_0, \ldots, v_k >$, daß $v_0 = v_k$, so nennt man ihn *Zykel*. Ist zusätzlich die Bedingung $v_i \neq v_j$ für alle $0 < i, j < k$ mit $i \neq j$ erfüllt, wird er als *einfacher Zykel* bezeichnet. Besitzt ein Graph keinen Zykel, nennt man ihn *azyklisch*.
Einige der zum Messen der Wichtigkeit benutzten Maße können nur unter der Bedingung angewandt werden, daß der Graph *zusammenhängend* ist. Im ungerichteten Fall bedeutet dies, es gitl für je zwei Knoten $u, v \in V$,daß v von u aus erreichbar ist. Läßt man in einem gerichteten Graphen die Richtungen der Kanten weg, entsteht der *zugeordnete ungerichtete* Graph. Ist dieser *zugeordnete ungerichtete* Graph zusammenhängend, gilt dasselbe für den gerichteten Graphen. Man spricht von einem *stark zusammenhängenden* Digraphen, falls es zu je zwei Knoten $u, v \in V$ einen gerichteten Weg von u nach v und von v nach u gibt. Ein ungerichteter, zusammenhängender Graph ohne Zykel heißt *Baum*. Wir werden auf diesen Begriff in Verbindung mit der *Closeness*-Zentralität noch zu sprechen kommen.
Unter Umständen wollen wir uns bei der Analyse nur auf einen Teil des Netzwerkes beschränken. $G' = (V', E')$ heißt *Teilgraph* (oder *Subgraph*) von $G = (V, E)$, falls $V' \subseteq V$ und $E' \subseteq E$ und alle Inzidenzen in G' den Inzidenzen in G entsprechen. Gilt dabei, $V' \subseteq V$ und $E' = \{e \in E :$ Endknoten von e sind in $V'\}$, bezeichnet man G' als *knoteninduzierten Teilgraphen* von G. $G' = (V', E')$ heißt *kanteninduzierter Teilgraph* von G, falls $E' \subseteq E$ und $V' = \{v \in V :$ v ist Endknoten von mindestens einem $e \in E'\}$. Die bezüglich der Knotenmenge inklusionsmaximalen, zusammenhängenden Teilgraphen von G heißen *Zusammenhangskomponenten*. Der Graph $G =$

(V, E) mit $E = \{\{u, v\} : u, v \in V, u \neq v\}$ heißt vollständiger Graph (über V). Ein knoteninduzierter Teilgraph von G, der vollständig ist, heißt *Clique* in G.
Zwei Begriffe, die in engem Zusammenhang mit der Wichtigkeit innerhalb eines Netzwerkes stehen, sind *Broker* und *Brücke*. Ein *Broker*, zu deutsch *Makler*, ist eine Person, die eine Verbindung zwischen mehreren Parteien bildet, z.B. zwischen dem *Käufer* und dem *Verkäufer*. In der Graphentheorie entspricht dies der Struktur eines *cutpoint*. Ein *cutpoint* ist ein Knoten n eines Graphen G, der bei Entnahme den Graphen in mindestens eine Zusammenhangskomponente mehr als vorher zerlegt, d.h. hat der Graph das Aussehen $G = (V, E)$ und entspricht k der Anzahl der Zusammenhangskomponenten, dann zerfällt G bei Entnahme von n in einen Graphen $G' = (V', E')$ mit mindestens $k + 1$ Zusammenhangskomponenten, wobei $V' = V \setminus \{n\}$ und $E' = E \setminus \{\{u, v\} : u = noder v = n\}$. Eine *Brücke* kann analog definiert werden, mit dem Unterschied, daß es sich um eine Kante, keinen Knoten handelt. Ist eine Kante e des Graphen $G = (V, E)$ mit k Zusammenhangskomponenten eine *Brücke*, so zerfällt der Graph bei Entnahme von e in einen Graphen $G' = (V, E')$ mit mindestens $k + 1$ Komponenten, wobei $E' = E \setminus \{e\}$.

Kapitel 3

Soziale Netzwerke

Das Kapitel "Soziale Netzwerke" teilt sich in mehrere Abschnitte. Es wird eingeleitet mit einem geschichtlichen Überblick zur Entstehung und Verwendung der sozialen Netzwerke. Dann folgt eine allgemeine Einführung in das Themengebiet, um Begriffe, die zur Darstellung nötig sind, zu erläutern und einen allgemeinen Überblick zu schaffen. Anschließend werden die im Mittelpunkt stehenden Maße für die Wichtigkeit innerhalb eines Netzwerkes, nämlich *Zentralität* und *Prestige*, zuerst allgemein, dann einzeln beschrieben. Die Darstellung der Themen des Kapitels folgt im wesentlichen dem Buch [WF94].

3.1 Ein kurzer, geschichtlicher Überblick

Die ersten wichtigen Untersuchungen zu sozialen Netzwerken finden vor über 60 Jahren statt, eingeleitet durch *Jakob L. Moreno*. Im April 1933 stellt Moreno das "Soziogramm" auf einer Medizinertagung vor. Auch die Öffentlichkeit erfährt von dieser Entwicklung durch einen Bericht der *New York Times* vom 3. April 1933. Andere Zeitungen greifen das Thema auf und die Forschungsergebnisse werden in ganz Amerika bekannt.

Dieser Zeitpunkt markiert den Anfang der *Soziometrie*, dem Vorläufer der sozialen Netzwerkanalyse und vieler Gebiete der Sozialpsychologie. Mit *Soziometrie* bezeichnet man das Messen (griech. metron = das Maß) von Beziehungen in kleineren sozialen Einheiten. Diese Einheiten können aus einzelnen Personen oder kollektiven sozialen Einheiten bestehen. Eine genauere Erläuterung folgt im Abschnitt *Grundlagen*. Moreno veranschaulichte diese Beziehungen von Einheiten durch die graphische Darstellung des Netzwerkes in Form des *Soziogrammes*, der ersten *Visualisierung* (siehe Kapitel 4) von sozialen Netzwerken [Mor53]. Das Soziogramm stellt die zu betrachtenden

sozialen Einheiten eines Netzwerkes (z.B. Menschen, Firmen, Vereine, ...) als Punkte im zweidimensionalen Raum dar und die Beziehungen zwischen Paaren dieser Einheiten durch Linien, welche die entsprechenden Punkte verbinden. Diese Form wird bis heute in ihrer Grundidee beibehalten, auch wenn die Einheiten nicht notwendigerweise als einfache Punkte dargestellt werden. Weiteres zur Geschichte des Soziogrammes findet sich in [HNC65]. Forsyth und Katz [FK46] waren unter den Ersten, die Matrizen zur Untersuchung von sozialen Netzwerken benutzten. Insbesondere die Arbeit von Katz [Kat53] macht es möglich, den Status sozialer Einheiten zu messen und ihn dann in der hier dargestellten Form sichtbar zu machen.

3.2 Grundlagen

Die Basisstruktur eines Netzwerkes ist ein *Graph*, wie in Kapitel 2 beschrieben. Da es sich bei der Analyse von sozialen Netzwerken in erster Linie nicht um Graphentheorie, sondern um die Untersuchung sozialer Eigenschaften handelt, werden wir einige Bezeichnungen einführen, die in das Gebiet der Sozialwissenschaften gehören. Die Struktur ist und bleibt nach wie vor die eines Graphen, es werden Methoden der Graphentheorie angewandt wie "Suchen von kürzesten Wegen" u.ä., die Notation wird sich jedoch in manchen Teilen ändern. Dies liegt an der Interpretation der einzelnen graphischen Komponenten, die je nach Art der Anwendung eine andere Bedeutung erhalten (s.u.). In einem sozialen Netzwerk bezeichnet man die *Knoten* des darunterliegenden Graphen als *Akteure* und die *Kanten* als *Beziehungen* zwischen diesen Akteuren.

Akteure Die Akteure eines sozialen Netzwerkes sind die *individuellen, sozialen Einheiten*, zwischen denen Beziehungen bestehen. Es kann sich dabei um *Personen*, aber auch um *Personengruppen* oder *Institutionen* handeln. Alle Einheiten zusammen bilden ein *Gruppe*, die klar nach außen hin abgrenzbar ist. Betrachten wir den Fall einer Menge von Studenten, die gemeinsam ein Vorlesung besuchen. Jeder Student bildet eine individuelle, soziale Einheit, die Menge aller Studenten, die diese Vorlesung besucht, bildet die Gruppe. Sie ist klar zu trennen von anderen Gruppen, da nur Studenten enthalten sind, die diese eine, spezielle Vorlesung besuchen. Weitere Beispiele aus der Praxis wären

- Kinder in einer Klasse
- Bewohner eines Stadtviertels

- Abteilungen einer Firma
- öffentliche Ämter in einer Stadt
- Staaten der Erde

In den meisten Fällen konzentrieren sich soziale Netzwerke auf eine Ansammlung von Akteuren desselben Typs, also z.B. Menschen in einer Arbeitsgruppe. Dies wird als *One Mode Network* bezeichnet. Der Inhalt dieser Arbeit beschränkt sich auf die Beschreibung von Netzwerken dieser Art. Es sei jedoch auf die Existenz anderer Typen von Netzwerken hingewiesen. *Two Mode Networks* konzentrieren sich auf entweder zwei Mengen von Akteuren, oder auf eine Menge von Akteuren und eine Menge von Ereignissen. Schenkungen von profitablen Firmen an caritative Einrichtungen beispielsweise beschreiben ein *Two Mode Network*, das sich auf zwei Gruppen von Akteuren bezieht [Gal85], [GW89]. Eine Gruppe von Frauen, die eine Vielzahl von sozialen Funktionen übernimmt, ist ein *Two Mode Network*, das die Beziehungen einer Gruppe von Akteuren mit einer Gruppe von Ereignissen beschreibt und deshalb auch als *affiliation network* (*affiliation* = Teilnahme) bezeichnet wird.
Es ist möglich, Netzwerke mit einer höheren Anzahl von *modes* zu betrachten, in der Praxis aber nicht üblich. Daher wollen wir uns auf die Beschreibung der ersten Möglichkeiten beschränken.

Beziehungen Akteure in einem sozialen Netzwerk sind durch soziale Beziehungen miteinander verknüpft. Diese können unterschiedlichste Formen annehmnen. Die Besonderheit einer solchen Beziehung ist, daß sie eine Beziehung zwischen einem *Paar* von Akteuren begründet und niemals nur *einem* Akteur zugeschrieben werden kann. Beispiele solcher Beziehungen sind

- Bewertung einer Person durch eine andere, z.B. durch Freundschaft, Liebe, Respekt
- Transfer von Materialressourcen, z.B. Geschäftstransaktionen, verleihen oder borgen von Materialien
- Gesellschaft oder Zugehörigkeit, z.B. Teilnahme am selben sozialen Ereignis, Mitglied im gleichen Club
- Kontakte untereinander, z.B. miteinander reden, Briefe schreiben
- Bewegung zwischen Plätzen oder gesellschaftlichen Leveln, soziale (Aufstieg von der "Arbeiterklasse" in die "gehobene Gesellschaft") oder physikalische Bewegungen (Wanderungen)

- Physikalische Beziehungen, z.B. Straßen, Flüsse, Brücken
- Formale Beziehung, z.B. Autorität
- Biologische Beziehung, z.B. Verwandtschaft

Beziehungen zwischen Akteuren können auf zwei verschiedene Arten auftreten, *gerichtet* oder *ungerichtet*. Wie sich herausstellen wird, ist die Trennung dieser beiden Typen von Beziehungen sehr wichtig bei der Berechnung von Maßen für die Akteure. Für die Betrachtung einiger Maße ist es nötig, *Sender* und *Empfänger* einer Beziehung zu unterscheiden, also Ursprung und Ziel einer Kante.

Dyade, Triade, Untergruppe Die Daten eines soziales Netzwerk können in verschiedene Level eingeteilt werden. Auf dem untersten Level sind die Akteure selbst angesiedelt. Sobald zwischen zwei Akteuren eine Beziehung besteht, erreichen wir das nächste Level, die *Dyade*, gefolgt von der *Triade* und der allgemeinen *Untergruppe*. Diese Begriffe werden im folgenden definiert.

Definition 1 (Dyade, Triade, Untergruppe) *Eine* Dyade *besteht aus einem Paar von Akteuren und den (möglichen) Beziehungen zwischen diesen Akteuren.*
Eine Triade *ist eine Menge von drei Akteuren und den (möglichen) Beziehungen zwischen diesen.*
Eine Untergruppe *ist eine Menge von beliebig vielen Akteuren und deren Beziehungen untereinander*

Die Dyade ist häufig die Basiseinheit für die Analyse von sozialen Netzwerken. Für bestimmte Anwendungen reichen sie jedoch nicht mehr aus, die Berechnung von Status beispielsweise ist allein mit dieser Betrachtungsweise nicht möglich. Erst die Erforschung von Untergruppen höherer Kardinalität oder des gesamten Netzwerkes liefert ein verwertbares Ergebnis. Interessant ist die Untersuchung der Transitivität bei Triaden oder größeren Untergruppen. Sie führt in einem späteren Abschnitt zur Berechnung verschiedener Zentralitäts- und Prestigewerte, insbesondere zur Bestimmung des Status eines Akteurs.

Gruppe Die Netzwerkanalyse beschäftigt sich nicht nur mit Dyaden, Triaden oder allgemeinen Untergruppen, sondern auch mit den Beziehungen zwischen Systemen von Akteuren. Dabei besteht ein System aus Beziehungen zwischen Mitgliedern einer (mehr oder weniger) begrenzten *Gruppe*. *Gruppe* bezeichnet hier die Ansammlung aller Akteure, zwischen denen Beziehungen gemessen werden können. Folglich muß es möglich sein, die Akteure durch bestimmte Kriterien zu einer begrenzten Menge zusammenzufassen.

Definition 2 (Gruppe) *Eine* Gruppe *ist eine endliche Menge von Akteuren, die durch konzeptionelle, theoretische oder empirische Gründe als endliche Menge von Individuen behandelt werden. Auf diese Menge von Akteuren können Netzwerkmaße angewandt werden.*

Wir beschränken uns zur Vereinfachung auf den endlichen Fall.

Relation Eine Beziehung besteht immer zwischen zwei Akteuren, einem einzelnen Paar. Die Gesamtheit der Beziehungen einer bestimmten Art zwischen Mitgliedern einer Gruppe nennt man *Relation*. Die Menge von Freundschaften zwischen je zwei Kindern einer Klasse oder die Menge der formalen Beziehungen zwischen Nationen dieser Erde bilden beispielsweise eine solche *Relation*.

Soziales Netzwerk Das *Soziale Netzwerk* umfaßt alle bisher eingeführten Begriffe und bildet den Abschluß der einführenden Definitionen.

Definition 3 (Soziales Netzwerk) *Ein Soziales Netzwerk besteht aus einer endlichen Menge (oder Mengen) von Akteuren und einer Relation zwischen diesen.*

Die Besonderheit des Sozialen Netzwerkes ist die Präsenz einer solchen Relation.

3.3 Prominenzmaße

Zentralität und Prestige

Einer der wichtigsten Punkte bei der Analyse von Netzwerken ist die Identifikation des *wichtigsten* Akteurs. Wichtigkeit ist dabei nicht eindeutig. Aus diesem Grund existieren verschiedene Definitionen für Wichtigkeit bzw. Prominenz. Eines haben alle Maße gemeinsam - sie machen die Wichtigkeit von der Lage des Akteurs im Netzwerk abhängig. Das bedeutet, daß sich die wichtigsten Akteure an strategisch wichtigen Positionen im Netzwerk befinden. Diese *wichtigen* Positionen können sich je nach Art des Maßes unterscheiden. An dieser Stelle sei schon ein kleiner Einblick in die im folgenden noch genau zu erklärenden Begriffe gewährt, um diese Unterschiede an einem Beispiel deutlich zu machen. Ein besonders zentraler Knoten muss nicht notwendigerweise einen hohen *Status* besitzen. Da bei der Berechnung von *Status* insbesondere die indirekten Beziehungen benutzt werden, ist es

möglich, daß der Akteur mit dem höchsten *Status* nur eine einzige einlaufende Kante besitzt, was beim Messen von *Degree*-Zentralität im gerichteten Fall einen geringen Zentralitätswert zur Folge hätte. Es wird sich zeigen, daß sich die Bedingungen für Wichtigkeit in den Bereichen *Zentralität* und *Prestige* stark unterscheiden, zum Teil sogar entgegengesetzt sind. Werden beim Berechnen der *Degree*-Zentralität im Fall von gerichteten Beziehungen die *Knotenausgangsgrade* betrachtet, verwendet man beim *Degree*-Prestige nur die *Knoteneingangsgrade*. Schon an dieser Stelle soll darauf hingewiesen werden, daß Prestige, anders als Zentralität, nur anhand von gerichteten Beziehungen berechnet werden kann. Der *Sender* und der *Empfänger* einer Beziehung müssen ganz klar voneinander zu unterscheiden sein. Zentralität kann dagegen sowohl in gerichteten, als auch in ungerichteten Beziehungen berechnet werden.
Die Maße für Wichtigkeit basieren auf folgenden Eigenschaften, die noch genau erläutert werden

- Degree (Knotengrad)
- Closeness (Distanz)
- Betweenness
- Information
- Proximity (Nähe) und
- Status (Rang)

Durch diese Definitionen können die Akteure mit Werten versehen werden, welche die Wichtigkeit eines Akteurs innerhalb eines Netzwerkes widerspiegeln sollen. Diese Werte werden wir mit *Index* bzw. *Indizes* bezeichnen. Ferner existiert die Möglichkeit, verschiedene Netzwerke untereinander mittels *Gruppenindizes* zu vergleichen.

Maße für Wichtigkeit in gerichteten und ungerichteten Beziehungen Wie wir bereits gesehen haben, gibt es zwei Arten von Beziehungen, *gerichtete* und *ungerichtete*. Unterscheidet sich die Berechnung der Maße in diesen Fällen? Kann es sein, daß manche Maße bei der Verwendung von gerichteten bzw. ungerichteten Beziehungen nicht berechnet werden können? In beiden Fällen kann die Frage mit *JA* beantwortet werden. Da die genaue Berechnung der Maße noch beschrieben wird, soll auf diese Fragen an späterer Stelle eingegangen werden.

Im folgenden werden zuerst Maße für die einzelnen Akteure definiert und dann, sofern möglich, auf Gruppenmaße ausgeweitet. Zum besseren Verständnis und um die Unterschiede verdeutlichen zu können, werden die Zentralitätsmaße zuerst auf Graphen angewandt, die sich nicht in der Anzahl ihrer Knoten, sondern in der Form ihrer Beziehungen unterscheiden. Im Anschluß daran wird ein komplexeres Netzwerk betrachtet, dessen Layout je nach Art der Berechnung variiert.
Es werden die gleichen Beispielgraphen wie in [WF94] bzw. [Fre80] benutzt, um die Ideen, die hinter den Maßen stecken, zu verdeutlichen.

- *Stern*
- *Kreis*
- *Pfad*

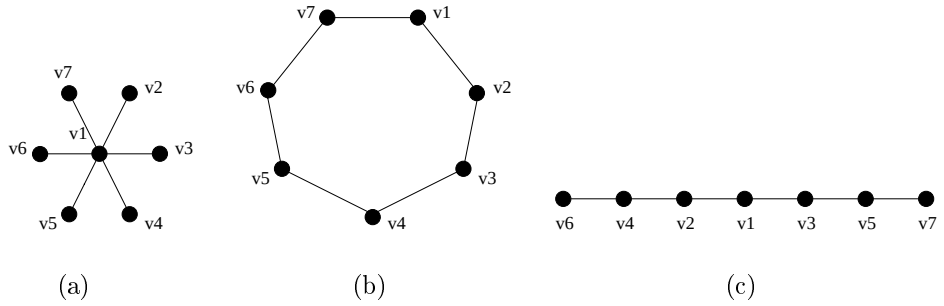


Abbildung 3.1: 3 Spezialgraphen mit jeweils 7 Knoten

Die Besonderheiten der Graphen liegen in der Struktur ihrer Beziehungen. Die Indizes können daher je nach Graphstruktur stark variieren.
Der *Stern* besitzt einen Knoten, der zentral erscheint, obwohl wir Zentralität noch nicht genau definiert haben. Bei Entfernung dieses Knotens existieren keine Beziehungen mehr und der Graph zerfällt in einzelne Knoten. Allein dieser Aspekt gibt Hinweise darauf, wie zentral der Knoten v_1 sein muß, was sich in der Berechnung der Zentralitätsmaße bestätigen wird.
Der *Kreis* besitzt keinen Knoten, der in irgendeiner Weise hervorsticht. Alle Knoten haben dieselbe Anzahl von Beziehungen, egal welcher Knoten entfernt wird, es entsteht immer dasselbe Bild.
Beim *Pfad* dagegen verhält es sich anders. Es existiert weder nur *eine* Klasse von Knoten, wie es im *Kreis* der Fall ist, noch *zwei* Klassen, in die der

Stern bei genauerer Betrachtung aufgeteilt werden kann. Die Zentralität der Knoten im *Pfad* nimmt ab, je weiter ein Knoten von der Mitte entfernt ist. Einen ersten Eindruck davon erhält man bei der Entfernung eines Knotens. Ein Endknoten (v_6, v_7) besitzt nur eine einzige Beziehung, die bei Löschung des Knotens entfällt, den noch verbleibenden Graphen aber nicht in verschiedene Zusammenhangskomponenten teilt. Im Gegensatz dazu hinterläßt das Fehlen eines Knotens mit zwei Beziehungen zwei Zusammenhangskomponenten.

Zu Beginn der 80er Jahre fing man an, Prominenz mit Sichtbarkeit gleichzusetzen [KB83]. Ein Akteur gilt als prominent, falls ihn seine Beziehungen für andere Akteure sichtbar machen. Man unterscheidet zwei Typen von Sichtbarkeit, *Zentralität* und *Prestige*. Zentrale Akteure sind sichtbar, da sie in viele Beziehungen verwickelt sind, prestigreiche Akteure sind sichtbar, da viele Beziehungen auf sie gerichtet sind. Beide Arten von Wichtigkeit basieren auf den Zeilen- und Spalteneinträgen der Soziomatrix. Diese Matrix besteht im einfachsten, nicht-gewichteten Fall aus den Einträgen 0 und 1. Die 1 zeigt an, daß zwischen den Akteuren eine Beziehung besteht. Falls keine solche Beziehung existiert, wird die Stelle mit 0 markiert.
Im ungerichteten Fall ist die Matrix symmetrisch, d.h. die Beziehungen eines Akteurs können durch Ablesen der zugehörigen Zeilen- *oder* Spalteneinträge bestimmt werden, da Zeile und Spalte für einen Akteur identisch sind. Die $n-1$ Einträge der $n \times n$-Matrix bestimmen die Wichtigkeit eines Akteurs.
Im gerichteten Fall unterscheiden sich die Zeilen- und Spalteneinträge in der Regel. Somit sind $2(n-1)$ Einträge für die Wichtigkeit der Akteure verantwortlich. Da bei der Berechnung des Status insbesondere die indirekten Beziehungen einfließen, beschränkt sich die Berechnung nicht auf die $2(n-1)$ Einträge. Alle Akteure, von denen ein gerichteter Pfad zu dem betrachteten Akteur existiert, werden berücksichtigt.

3.3.1 Akteur- und Gruppenmaße

Der folgende Abschnitt enthält eine kurze, allgemeine Beschreibung der Prominenzmaße *Zentralität* und *Prestige*. Da Wichtigkeit auf verschiedenen Ebenen gemessen werden kann, wird diese Trennung hier mit der getrennten Darstellung von Akteurmaßen und Gruppenmaßen aufrecht erhalten.

Akteurzentralität und Akteurprestige Zentralität bedeutet für einen Akteur soviel wie "verwickelt in viele Beziehungen". Da Zentralität auch im

ungerichteten Fall berechnet werden kann, hängt die Wichtigkeit nicht immer von der Sender- oder Empfängerrolle eines Akteures ab. Dies ändert sich bei gerichteten Beziehungen. Ist ein Akteur im Bereich "Senden" aktiv, spricht das für einen hohen Zentralitätswert. Im Gegensatz dazu führt eine hohe Menge von "empfangenen" Beziehungen zu einem hohen Prestigewert. Es wird sich zeigen, daß die Knotenein- und -auslaufgrade einfache Maße zur Berechnung von *Prestige* und *Zentralität* sind.
Der Ausdruck *Prestige* sollte an dieser Stelle nicht mißverstanden werden. Ein hoher Prestigewert in einem Netzwerk, das *Verachtung in einer Gruppe von Personen* darstellen soll, bedeutet keinesfalls, daß der Akteur hoch angesehen ist. Bedeutet die Beziehung "gibt einen Rat an", ändert sich die Berechnung des Prestigemaßes nicht. Auch wenn vom allgemeinen Verständnis her Akteure, die einen Rat geben, wichtig sein sollten, wird dennoch *der* Akteur als prestigereich erachtet, der viele Beziehungen "empfängt". Die Idee des *Prestige* muß daher immer im Zusammenhang mit der Art der Relation betrachtet werden.

Gruppenzentralität und Gruppenprestige Es besteht die Möglichkeit, einige Akteurmaße auf Gruppenmaße auszuweiten, um Netzwerke miteinander vergleichen zu können. Dabei ist noch nicht klar, was genau ein Gruppenmaß mißt. Die Einführung eines allgemeinen Gruppen-Zentralitätsindex gibt einen ersten Aufschluß über diese Frage.
Sei dazu $C_A(v_i)$ der Zentralitätsindex eines Akteurs v_i, gemessen durch ein Zentralitätsmaß A in einem Netzwerk mit n Akteuren. Sei ferner $C_A(v^*)$ definiert als $max_i C_A(v_i)$, das Maximum über alle berechneten Werte der n Akteure. Der Ausdruck

$$\sum_{i=1}^{n}[C_A(v^*) - C_A(v_i)]$$

beschreibt die tatsächliche Summe der Differenzen zwischen dem größten Wert v^* und allen anderen. Er wird mit *Numerator* bezeichnet [Fre79]. Neben diesem für ein bestimmtes Netzwerk berechneten Wert, wird ein theoretischer Wert zur Definition des allgemeinen Gruppen-Zentralitätsindex benutzt, der *Denominator*. Er beschreibt die theoretisch maximal mögliche Summe von Differenzen eines Netzwerkes mit n Akteuren, d.h.

$$max \sum_{i=1}^{n}[C_A(v^*) - C_A(v_i)]$$

Man geht also nicht von einem bestimmten Netzwerk aus, sondern von allen möglichen Netzwerken mit n Akteuren. Durch Vergleich der beiden Gößen

Numerator und *Denominator* ergibt sich der allgemeine Gruppen-Zentralitätsindex als

$$C_A = \frac{\sum_{i=1}^{n}[C_A(v^*) - C_A(v_i)]}{max \sum_{i=1}^{n}[C_A(v^*) - C_A(v_i)]}$$

Die Werte für diesen Index liegen aufgrund der Konstruktion zwischen 0 und 1. Extremwerte werden für Graphen mit spezieller Struktur angenommen, beispielsweise von dem bereits erwähnten *Stern*. Ein Akteur besitzt Beziehungen zu allen anderen, wobei diese keine direkten Beziehungen untereinander haben. Bei Löschen des zentralen Knotens zerfällt das Netzwerk in $n - 1$ Zusammenhangskomponenten. Die maximal mögliche Summe von Differenzen wird angenommen, weshalb *Numerator* und *Denominator* für diesen Spezialgraphen identisch sind und $C_A(Stern)$ den Wert 1 annimmt. Ein anderer Spezialfall ist die Gruppe von Netzwerken, in denen die Zentralitätswerte für alle Akteure identisch sind, beispielsweise im *Kreis*, einem Netzwerk, das nur aus isolierten Knoten oder einem vollständigen Graphen besteht. Die Summe der Differenzen der Zentralitätswerte ergibt 0, was zu einem Gruppenmaß von 0 führt.
Das Gruppenmaß ist also ein Indikator dafür, wie stark die Werte eines Netzwerkes variieren. Existieren Akteure, die im Gegensatz zu den anderen Akteuren des Netzwerkes einen hohen Zentralitätswert besitzen, so ist das Gruppenmaß entsprechend hoch, je nach Höhe der Differenzwerte. Je ähnlicher die Indizes für die einzelnen Akteure sind, desto näher liegt der Wert des Gruppenmaßes bei 0. Das Gruppenmaß kann somit als Maß für die Variabilität, Zerstreuung oder Ausdehnung angesehen werden.
Im Fall des allgemeinen Gruppen-Prestigemaßes ist die Konstruktion des Denominators jedoch nur in Spezialfällen möglich. Daher werden oft einfacher zu berechnende Größen wie *Varianz* benutzt, um einen Gruppen-Prestigeindex zu berechnen.

In den folgenden Abschnitten werden zunächst jeweils die Maße für die einzelnen Akteure definiert und soweit möglich, auf Gruppenmaße ausgeweitet. Dabei werden zuerst die ungerichteten, dann die gerichteten Beziehungen untersucht.

3.3.2 Degree-Zentralität

Nach der einfachsten Definition für Akteur-Zentralität ist *der* Akteur der zentralste, der die meisten Beziehungen besitzt. Im Fall der *Degree*-Zentralität wird dies anhand der adjazenten Knoten bzw. inzidenten Kanten, gemessen, mittels des *Knotengrades*. Vergleichen wir die Graphen *Stern* und *Kreis* mit-

einander, wird die Wichtigkeit (oder Unwichtigkeit) eines Knotens deutlich.

Stern: Ein Akteur hat Beziehungen zu allen $n-1$ anderen. Diese $n-1$ anderen sind untereinander nicht direkt verbunden, nur über den zentralen Akteur in der Mitte. Fehlt er, fehlen alle Beziehungen, der Graph zerfällt in nicht-zusammenhängende Komponenten. Fehlt ein nicht-zentraler, ein *Randknoten*, ändert sich nichts an der Struktur, allein die Kardinalität der Knoten und Kanten wurde um 1 verringert.

Kreis: Alle Akteure haben den gleichen Zentralitätsindex, da sie beliebig austauschbar sind, keiner ist *aktiver* als der andere. Egal welcher Knoten aus dem Graphen entfernt wird, die Struktur ist danach immer dieselbe.
Es sollte bemerkt werden, daß nur direkte, d.h. adjazente Beziehungen betrachtet werden, indirekte gehen nicht mit in die Betrachtung ein. Prominenz wird an dieser Stelle also gleichgesetzt mit *Aktivität* bzw. *Knotengrad.*

Akteur Degree-Zentralität Ein *Degree*-Zentralitätsindex für einen einzelnen Akteur läßt sich durch den Grad eines Knotens $deg(v_i)$ im darunterliegenden Graphen berechnen. Daraus ergibt sich folgende Definition für den Index $C_D(v_i)$

$$C_D(v_i) = deg(v_i) = deg_{out}(v_i) = deg_{in}(v_i)$$

$deg_{out}(v_i)$ und $deg_{in}(v_i)$ bezeichnen jeweils den Knotenaus- bzw. den Knoteneingangsgrad. Da wir uns im ungerichteten Fall befinden, unterscheiden sich deg_{in} und deg_{out} nicht. Der maximal annehmbare Wert liegt bei $n-1$. Somit kann der Index mittels Division durch $n-1$ normiert werden

$$C'_D(v_i) = \frac{deg(v_i)}{n-1}$$

Diese Normierung bietet den Vorteil einer Unabhängigkeit von n, die Werte können mit Indizes aus anderen Netzwerken verglichen werden. Im folgenden sind alle normierten Maße mit einem Apostroph gekennzeichnet, um sie von den nicht-normierten Maßen zu unterscheiden.
Betrachte dazu die Werte für *Degree*-Zentralität für die Spezialgraphen *Stern*, *Kreis* und *Pfad.*
Stern:
$deg(v_1) = 6, deg(v_i) = 1$ für i=2, ... ,7
Die normierten Werte sind somit $\{1.0, 0.167, \ldots, 0.167\}$.
Kreis:
$deg(v_i) = 2$ für alle i, also $C'_D(v_i) = 0.333$

Pfad:
$deg(v_i) = 2, C'_D(v_i) = 0.333$ für i=1, ... ,5
$deg(v_i) = 1, C'_D(v_i) = 0.167$ für i=6,7.

Diese Art von Zentralitätsindizes läßt sich durch Programme wie UCINET u.a. berechnen, muß also für große Netzwerke nicht von Hand untersucht werden. Ein hoher Knotengrad spricht somit für eine hohe Zentralität, eine *zentrale Position* im Netzwerk. Akteure mit niedrigem Knotengrad können als *Randpunkte* betrachtet werden. Falls sie komplett isoliert sind, besitzen sie den Knotengrad $deg(v_i) = 0$.

Beispiel Als Vorlage für dieses Beispiel dient Padgett's *Netzwerk der Florentinischen Familien* und den Beziehungen, die durch Heirat entstanden sind. Die Akteure dieses Netzwerkes sind somit ganze Familien, keine einzelnen Personen. Es handelt sich um ein Beispiel aus der Praxis, die Familien und Beziehungen haben im Italien des 15. Jahrhunderts wirklich existiert. 16 Familien wurden aufgrund ihrer historischen Wichtigkeit aus einer Menge von 116 Familien ausgewählt. Die gesammelten Daten wurden aus [Ken78] entnommen und in [PA93] und anderen analysiert.
Die berechneten *Degree*indizes finden sich in Tabelle 3.1.
Figur 3.2 zeigt eine graphische Darstellung der berechneten Indizes.

Gruppen-Degree-Zentralisierung Wie wir bereits wissen mißt ein Gruppen-Zentralitätsmaß die Variabilität der einzelnen Akteur-Indizes. Benutzen wir zur Berechnung den allgemeinen Gruppen-Zentralitätsindex, so erhalten wir den Denominator durch die Berechnung der Akteurindizes. Da für den *Stern* das Maximum angenommen wird, besitzt der Denominator den Wert $(n-1)(n-2)$. Diese Zahl setzt sich aus der Differenz der Werte des zentralen Knotens und der anderen Knoten $((n-1)-1) = (n-2)$ und aus der Anzahl der Differenzen $(n-1)$ zusammen. Insgesamt ergibt sich somit der Index

$$C_D = \frac{\sum_{i=1}^{n}[C_D(v^*) - C_D(v_i)]}{[(n-1)(n-2)]}$$

Berechnen wir diese Indizes für unsere drei Spezialgraphen, erhalten wir folgende Werte:
$C_D(Stern) = 1$, der Index wird maximal, da ein Akteur alle anderen überschattet.
$C_D(Kreis) = 0$, der Index wird minimal, da sich die Werte der einzelnen Akteure nicht unterscheiden.
$C_D(Pfad) = 0.277$.

Tabelle 3.1: Degreeindizes für Padgett's Florentinische Familien Netzwerk

Akteurnummer	Familienname	Degreeindex
1	Acciaiuoli	0.0714286
2	Albizzi	0.214286
3	Barbadori	0.142857
4	Bischeri	0.214286
5	Castellani	0.214286
6	Ginori	0.0714286
7	Guadagni	0.285714
8	Lamberteschi	0.0714286
9	Medici	0.428571
10	Pazzi	0.0714286
11	Peruzzi	0.214286
12	Pucci	0
13	Ridolfi	0.214286
14	Salvati	0.142857
15	Strozzi	0.285714
16	Tornabuoni	0.214286

Verwenden wir anstelle des allgemeinen Gruppen-Zentralitätsmaßes die Varianz der Knotengrade zur Berechnung eines Gruppenindex, erhalten wir

$$S_D^2 = [\sum_{i=1}^{n} (C_D(v_i) - \bar{C}_D)^2]$$

wobei $\bar{C}$ den *Degree*index des Hauptakteurs, d.h. den maximalen Index, beschreibt. Dieser Index nimmt das Minimum von 0 an, wenn alle Knotengrade gleich sind bzw. der Graph regulär ist. Das Maximum hängt von n und der gesamten Menge der Knotengrade ab.
Auch der Durchschnitt der Knotengrade läßt sich zur Berechnung eines Gruppen-*Degree*maßes verwenden. Als Formel ergibt sich

$$C_D = \sum C_D(v_i)/n = \sum deg_{out}(v_i)/n$$

Die Werte liegen im Bereich zwischen 0 und $n-1$. Daher kann eine Normierung mittels Division durch $n-1$ erhalten werden. Dieser normierte Knotengraddurchschnitt entspricht exakt der Dichte des Graphen

$$\sum C_D(v_i)/n(n-1) = \sum C'_D(v_i)/n = \Delta$$

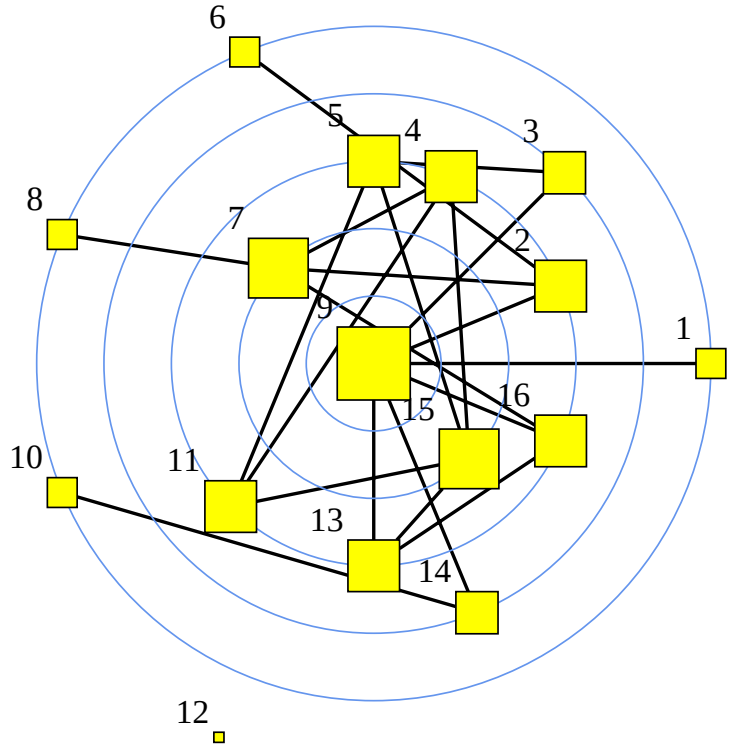


Abbildung 3.2: Degreeindex für Padgett's Florentinische Familien

Werte der Dichte variieren zwischen 0 bei einem leeren Graphen bis 1 bei einem vollständigen Graphen.
Da keiner der drei hier betrachteten Spezialgraphen diesen Anforderungen entspricht, ergeben sich keine Extremwerte, sondern alle Werte liegen relativ nahe beieinander.
Stern: 0.286
Kreis: 0.333
Pfad : 0.286

Die Dichte ist der vielleicht am weitesten verbreitete Index für Gruppenmaße, was sich in der Häufigkeit der Verwendung widerspiegelt (siehe [Bla77], [Bot57], [Bar69]).
Weitere Gruppen-*Degree*maße finden sich z.B. bei [Sni81a], [Sni81b].

Degree-Zentralität bei gerichteten Beziehungen Da es bei den gerichteten Beziehungen für die Berechnung der Zentralität um die Eigenschaft "senden" geht, wählt man anstelle des allgemeinen Knotengrades den *Knotenausgangsgrad*. Dadurch erhalten wir für den Akteurindex

$$C_D(v_i) = deg_{out}(v_i).$$

Da $n-1$ den maximalen Wert darstellt, kann eine Normierung wieder mittels Division durch diesen maximalen Wert erhalten werden

$$C_D(v_i) = deg_{out}(v_i)/(n-1).$$

Der allgemeine Gruppenindex für *Degree*-Zentralität berechnet sich wie im ungerichteten Fall durch Numerator und Denominator. Da der Denominator

$$max \sum_{i=1}^{n} [C_D(v^*) - C_D(v_i)]$$

den Wert $(n-1)^2$ annimmt, schreibt sich der Index als

$$C_D = \frac{\sum_{i=1}^{n} [C_D(v^*) - C_D(v_i)]}{(n-1)^2}.$$

Eine weitere Möglichkeit besteht in der Berechnung der Varianz. Sie setzt sich zusammen aus der Summe der quadrierten Differenzen jedes Knotenausgangsgrades und des maximalen Knotenausgangsgrades

$$S_D^2 = [\sum_{i=1}^{n} (deg_{out}(v_i) - max_i(deg_{out}(v_i)))^2].$$

3.3.3 Closeness-Zentralität

Das Prinzip dieser Art von Zentralität beruht auf der *Distanz*, nicht mehr direkt auf den Knotengraden. Man mißt, wie nahe ein Akteur allen anderen Akteuren im Netzwerk ist. Die Idee für diese Zentralität ist, daß ein Akteur genau dann zentral ist, wenn er schnell mit allen anderen "interagieren" kann. Anders ausgedrückt benötigen zentrale Knoten in einem Netzwerk "eine minimale Anzahl von Schritten", um andere Knoten zu erreichen. Die *kürzesten Wege*, die zentrale Knoten und andere verbinden, müssen so kurz wie möglich sein. *Closeness* kann also mit "minimaler Distanz" gleichgesetzt werden. Zentralität steht durch diese Erklärung in inverser Beziehung zu Distanz. Die Zentralität eines Knotens nimmt bei größerer Entfernung zu den

anderen Knoten ab, da mehr Kanten auf den kürzesten Pfaden existieren, die diesen Knoten mit den anderen verbinden.
Ein Beispiel für maximale *Closeness*-Zentralität ist der *Stern*. Es existiert *ein* Knoten, der adjazent zu allen anderen ist, d.h. er besitzt die kürzest möglichen Pfade zu allen andern. Da er der einzige Akteur ist, der alle anderen in einer minimalen Anzahl von Schritten erreichen kann, erhält er einen maximalen Wert für *Closeness*. Er benötigt keine anderen Akteure für eine Interaktion, da er direkt mit jedem einzelnen verbunden ist.

Akteur-Closeness-Zentralität Eine Möglichkeit, um zu messen, wie nahe ein Akteur allen anderen ist, ist die Verwendung von *kürzesten Wegen*. Damit sind Pfade von einem Knoten zu einem anderen gemeint, die eine minimale Anzahl von Kanten durchlaufen. Werden die Kanten gewichtet, spielt nicht mehr nur die Anzahl der Kanten, sondern auch die Gewichtung eine relevante Rolle. Somit gehen bei dieser Art der Zentralitätsmessung nicht nur die direkten, sondern insbesondere die indirekten Beziehungen in die Berechnung mit ein.

Betrachte dazu folgende Situation. Sei $< v_i = u_0, u_1, \ldots, u_k = v_j >$ eine Knotenfolge, die einen Weg von Akteur i zu Akteur j beschreibt. Die Länge des Weges berechnet sich durch die Anzahl k der (in diesem Fall ungewichteten) Kanten. Existiert kein l mit $l < k$, so daß ein Pfad $< v_i = u_0, u_1, \ldots, u_l = v_j >$ zu finden ist, der die Akteure i und j durch eine geringere Anzahl von Kanten verbindet, dann ist $< v_i = u_0, u_1, \ldots, u_k = v_j >$ ein kürzester Weg von Akteur i zu Akteur j. Kürzeste Wege müssen nicht eindeutig sein, was gerade bei ungerichteten Graphen häufig der Fall ist. Sei $d(*,*)$ eine Distanzfunktion, d.h.

$d(v_i, v_j) :=$ Anzahl Kanten auf dem kürzesten Weg von Akteur i zu Akteur j.

Die Gesamtdistanz von Akteur i zu allen anderen berechnet sich durch $\sum_{j=1}^{n} d(v_i, v_j)$. Da die Zentralität bei steigender Distanz abnimmt, kann der *Closeness*index durch

$$C_C(v_i) = [\sum_{j=1}^{n} d(v_i, v_j)]^{-1}$$

beschrieben werden. Ist ein Akteur zu allen anderen adjazent, bestehen die kürzesten Wege aus jeweils nur einer Kante. Dies ist beim *Stern* der Fall. Der Index nimmt dann den maximalen Wert von $(n-1)^{-1}$ an. Sind einer oder mehrere Akteure von dem betrachteten Akteur i aus nicht erreichbar, d.h.

existiert kein verbindender Pfad zwischen den beiden Akteuren, so erhält der Index den Wert 0. Daher ist es sinnvoll, diesen Index nur auf zusammenhängende Graphen anzuwenden.
Um dieser Forderung einen Sinn zu geben, nehmen wir an, der Graph sei nicht zusammenhängend. Insbesondere existiere ein isolierter Knoten v_k, der damit Knotengrad 0 hat. Dann gilt für alle Knoten $v_i \neq v_k$

$$d(v_i, v_k) = \infty.$$

Damit ist die Distanzsumme für jeden Knoten ∞ und die berechneten Indizes sind alle gleich, der Wert ist 0.
Um Akteure verschiedener Netzwerke vergleichen zu können, muß wieder eine Normierung durchgeführt werden. Da der maximale Wert für einen Akteur in einem Netzwerk mit n Knoten $n-1$ beträgt, lassen sich alle Werte durch Multiplikation mit $n-1$ normieren. Somit erhalten wir

$$C'_C(v_i) = \frac{n-1}{[\sum_{j=1}^{n} d(v_i, v_j)]} = (n-1)C_C(v_i).$$

Die normierten Werte liegen wieder im Intervall $[0, 1]$. Der normierte Index ist also nichts anderes als die umgekehrte, durchschnittliche Distanz zwischen Akteur i und allen anderen. Graphentheoretiker reden in diesem Zusammenhang vom *Center* des Graphen. Ein spezielles Center ist das *Jordan Center* [Jor69]. Es bezeichnet *die* Menge von Knoten, welche die kleinste maximale Distanz zu allen anderen Knoten hat. Die Vorgehensweise, um ein solches Center zu finden, sei hier kurz beschrieben. Sei A eine $n \times n$-Matrix der kürzesten Wege, wobei die Einträge die Länge der kürzesten Wege zwischen allen Paaren von Knoten beinhalten. Finde den jeweils größten Eintrag in jeder Zeile. Diese Distanzen (auch *eccentricities* [Bra99]) sind die maximalen Distanzen von jedem Akteur zu den anderen. Suche nun den kleinsten dieser maximalen Abstände. Alle Knoten, die diese kleinste, maximale Distanz besitzen, sind Teil des *Center* des Graphen. Ein weiterer Begriff, der in diesem Zusammenhang genannt werden sollte, ist der des *Centroid* [Syl82]. Dieser *Centroid* ist abhängig von den Knotengraden und wird hauptsächlich bei Bäumen angewandt. Um den *Centroid* zu finden, betrachte die Äste von jedem Knoten aus. Definiere das Gewicht des Astes als Anzahl der Kanten. Das Gewicht des Knotens ist das maximale Gewicht jedes Astes am Knoten. Der *Centroid* ist dann *die* Teilmenge von Knoten mit dem kleinsten, maximalen Gewicht.
Zur Veranschaulichung des *Closeness*index werden die Werte für die drei Spezialgraphen *Stern*, *Kreis* und *Pfad* berechnet. Alle drei sind zusammenhängend, d.h. die kürzesten Wege sind endlich und der normierte *Closeness*index kann

berechnet werden.
Stern: $C'_C(v_1) = 1.0, C'_C(v_i) = 0.545$ für i=2, ... ,7
Kreis: $C'_C(v_i) = 0.5$ für alle i
Pfad : $C'_C(v_1) = 0.5$, absteigend bis zu $C'_C(v_6) = C'_C(v_7) = 0.286$

Um kürzeste Wege zu finden und deren Länge zu berechnen existieren gute Algorithmen, z.B. durch n-maliges Anwenden einer Breitensuche oder den Algorithmus von Dijkstra. Einige davon sind bereits implementiert in Programmen wie UCINET oder SNAPS.

Beispiel Wir wollen uns den *Closeness*index wieder am Beispiel von *Padgett's Florentinischem Familien Netzwerk* verdeutlichen. Die Struktur hat sich an einigen Stellen verändert. Akteure, die zuvor den gleichen *Degree*index hatten, unterscheiden sich im *Closeness*index. Gut zu sehen ist diese Veränderung bei den Familien *Albizzi, Bischeri, Castellani, Peruzzi, Ridolfi* und *Tornabuoni*. Bei exakt gleichem *Degree*index zeigt sich hier ein relativ breiter Wertebereich. Im Gegenzug dazu besitzen die Familien *Castellani* und *Salvati* denselben *Closeness*index, unterscheiden sich jedoch deutlich bei den Werten des *Degree*index. Die Tabelle 3.2 gibt Aufschluß über die genauen Werte, die dann in Figur 3.3 graphisch dargestellt werden.

Gruppen-Closeness-Zentralisierung Aufgrund der verschiedenen Möglichkeiten zur Berechnung eines Gruppen-*Closeness*index wollen wir an dieser Stelle wieder zuerst vom allgemeinen Gruppenindex ausgehen. Der Numerator schreibt sich durch Einsetzen der normierten Akteur-*Closeness*-Zentralität als

$$\sum_{i=1}^{n}[C'_C(v^*) - C'_C(v_i))]$$

Die größtmögliche Differenz ergibt sich bei einem Stern. Daher berechnet sich der Denominator durch

$$\begin{aligned} max \sum_{i=1}^{n}[C'_C(v^*) - C'_C(v_i))] &= (n-1)[\frac{n-1}{n-1} - \frac{n-1}{1+(n-2)2}] \\ &= \frac{(n^2-3n+2)}{(2n-3)} \\ &= \frac{(n-2)(n-1)}{(2n-3)} \end{aligned} \tag{3.1}$$

Der Index läßt sich somit schreiben als

$$C_C = \frac{\sum_{i=1}^{n}[C'_C(v^*) - C'_C(v_i))]}{[(n-2)(n-1)]/(2n-3)}.$$

Tabelle 3.2: Closenessindizes für Padgett's Florentinische Familien Netzwerk

Akteurnummer	Familienname	Closenessindex	Degreeindex
1	Acciaiuoli	0.368421	0.0714286
2	Albizzi	0.482759	0.214286
3	Barbadori	0.4375	0.142857
4	Bischeri	0.4	0.214286
5	Castellani	0.388889	0.214286
6	Ginori	0.333333	0.0714286
7	Guadagni	0.466667	0.285714
8	Lamberteschi	0.325581	0.0714286
9	Medici	0.56	0.428571
10	Pazzi	0.285714	0.0714286
11	Peruzzi	0.368421	0.214286
12	Pucci	0	0
13	Ridolfi	0.5	0.214286
14	Salvati	0.388889	0.142857
15	Strozzi	0.4375	0.285714
16	Tornabuoni	0.482759	0.214286

Das Maximum von 1 wird hier erreicht, falls ein Akteur zu allen anderen die kürzesten Wege der Länge 1 hat und die anderen Akteure kürzeste Wege der Länge 2 zu den verbleibenden $n-2$ Akteuren haben. Dies trifft beim *Stern*-Graphen zu.
Das Minimum von 0 wird beim *Kreis* bzw. bei einem vollständigen Graphen erreicht, wenn die Länge der kürzesten Wege immer gleich ist.
Für den dritten Spezialgraph, den *Pfad*, liegt der Wert des Index bei 0.277.
Wie beim Gruppen-*Degree*index kann man auch hier die Varianz anwenden, um einen Gruppen-Closenessindex zu berechnen. Dabei werden die Ungleichheiten zwischen den $C'_C(v_i)$ aufsummiert. Zur Vereinfachung definiere $\bar{C}_C$ als den Durchschnitt der normierten *Closeness*

$$\bar{C}_C := \sum C'_C(v_i)/n.$$

Als Index erhalten wir dann

$$S_C^2 = [\sum_{i=1}^{n}(C'_C(v_i) - \bar{C}_C)^2]/n$$

Das Minimum von 0 wird angenommen, falls alle Knoten den gleichen Ab-

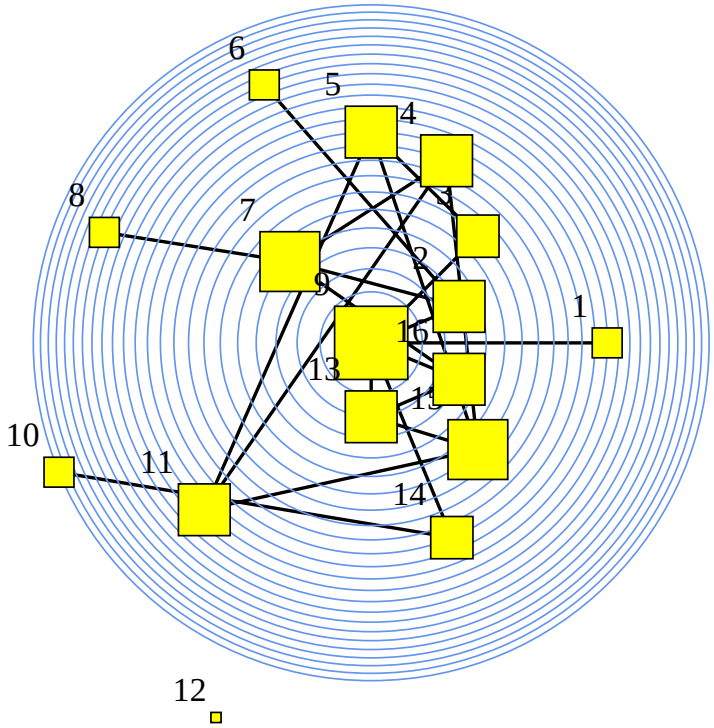


Abbildung 3.3: Closenessindex für Padgett's Florentinische Familien

stand voneinander, also gleichen Akteur-Index haben, was nicht notwendigerweise bedeutet, daß der Graph vollständig ist.

Closeness-Zentralität bei gerichteten Beziehungen Definiere die Distanzen zwischen zwei Akteuren i,j als die Länge der kürzesten Wege von v_i nach v_j. Der Unterschied zum ungerichteten Fall ist der, daß die kürzesten Wege von v_i nach v_j nicht notwendigerweise dieselben sind wie die von v_j nach v_i, d.h. es ist möglich, daß $d(v_i, v_j) \neq d(v_j, v_i)$. Seien die Distanzen $d(v_i, v_j)$ die Elemente einer $n \times n$ Distanzmatrix. Zur Berechnung des Akteur-*Closeness*index benötigt man die minimal mögliche Gesamtdistanz $n-1$ und die Summe der Zeile i, was der totalen Distanz von v_i zu allen anderen Akteuren entspricht. Der normierte Akteurindex hat dann das Aussehen

$$C'_C(v_i) := \frac{(n-1)}{[\sum_{j=1}^{n} d(v_i, v_j)]}$$

Dieser Index unterscheidet sich in keiner Eigenschaft vom Index im ungerichteten Fall, ausser der Tatsache, daß die kürzesten Wege von i nach j nicht dieselben sein müssen wie die von j nach i. Zur Berechnung des Gruppenindex kann man analog zum ungerichteten Fall vorgehen, indem man die normierten Indizes benutzt. Bisher ist jedoch keine Quelle bekannt, in der der Denominator dieses Index für gerichtete Beziehungen berechnet wurde.

Für den Akteur-*Closeness*index tritt wie im ungerichteten Fall ein Problem auf, falls der Graph nicht streng zusammenhängend ist. Streng zusammenhängend bedeutet, daß zwischen je zwei Akteuren ein gerichteter Pfad existiert. Ist dies nicht der Fall, gibt es mindestens ein Paar von Akteuren i, j , so daß die Distanz $d(v_i, v_j) = \infty$ ist. Das Problem läßt sich lösen, indem nur die von i aus erreichbaren Akteure betrachtet werden, die anderen nicht.
Ein weiterer Akteurindex kann durch Verallgemeinerung des einfachen Index $C'_C(v_i)$ erhalten werden. Hierzu wird der *Einflußbereich* von v_i benötigt,der aus *der* Menge von Knoten besteht, die in endlichen Schritten von v_i aus erreichbar sind (vgl. Graphentheorie). Sei J_i die Anzahl der Akteure im Einflußbereich von v_i. Der erweiterte Akteur-*Closeness*index mißt, wie nahe v_i den Akteuren im Einflußbereich ist. *Closeness* wird somit über die Distanzen von Akteur i zu den Akteuren im Einflußbereich definiert. Betrachte dazu den Quotienten $\sum d(v_i, v_j)/J_i$, d.h. den durchschnittlichen Abstand. Da nur erreichbare Knoten in Betracht gezogen werden, kann dieses Maß auch für nicht-zusammenhängende Graphen gebraucht werden und wird definiert als

$$C^*_C(v_i) := \frac{J_i/(n-1)}{\sum d(v_i, v_j)/J_i}.$$

Dieser Index gibt das Verhältnis der Menge der erreichbaren Akteure in der Gruppe zu den durchschnittlichen Distanzen dieser Akteure zu Akteur i an. Er ähnelt einem Prestigeindex, der zu einem späteren Zeitpunkt betrachtet wird.

3.3.4 Betweenness-Zentralität

Die *Betweenness*-Zentralität mißt den Anteil von kürzesten Verbindungen zwischen anderen Akteuren, auf denen ein Akteur i liegt. Zur Verdeutlichung betrachte folgende Situation:
Gegeben seien zwei Knoten u und v, die nicht adjazent sind. Seien a und b die zwei Knoten auf dem kürzesten Weg zwischen u und v, d.h. es existiert ein Weg $< u, a, b, v >$. Dann kann die Kommunikation zwischen u und v durch a und b beeinflußt werden (beispielsweise die Übertragung einer mündlichen

Nachricht beim Kinderspiel "Stille Post"). Dies wird besonders deutlich am Spezialfall *Stern*.
Der mittlere Knoten liegt auf allen kürzesten Wegen zwischen den anderen sechs Knoten. Dieser eine Knoten in der Mitte, der eine *zwischen* den anderen, hat die Kontrolle über die Wege im Graphen. Beim Spezialgraph *Pfad* haben die inneren Knoten mehr Kontrolle über einige Wege als die äußeren, die überhaupt keine Kontrolle haben.
Man sagt, die Akteure in der Mitte haben "mehr interpersonellen Einfluß" auf die anderen. Um zentral zu sein muß hier ein Akteur also auf kürzesten Wegen liegen. Diese Vermutung wurde schon früh angestellt [Bav48], [Shi53], [Sha54], aber erst ca. 20 Jahre später konnte diese Art von Zentralität gemessen werden [Ant71], [Fre79], [Pit79].

Akteur-Betweenness-Zentralität Die Vorgehensweise beim Messen von *Betweenness*-Zentralität ist relativ einfach. Man zähle die Anzahl der kürzesten Wege zwischen den Akteuren j und k - alle mit derselben Länge $d(v_j, v_k)$. Dann entscheide man, auf wievielen Wegen Akteur i enthalten ist.
Betrachten wir nun die Wahrscheinlichkeit, daß eine Kommunikation einen bestimmten Weg nimmt.
Annahme: Alle Kanten besitzen das gleiche Gewicht und die Kommunikation nimmt einen kürzesten Weg. Dann ergibt sich die folgende Wahrscheinlichkeit: Sei g_jk gleich der Anzahl der kürzesten Wege (engl. *geodesics*) zwischen j und k. Somit liegt die Wahrscheinlichkeit für die Wahl eines solchen Weges bei $1/g_{jk}$. Sei ferner $g_{jk}(v_i)$ die Anzahl der kürzesten Wege, die Akteur i enthalten. Ist die Wahrscheinlichkeit für die Wahl eines beliebigen Weges immer gleich hoch, so erhalten wir eine Wahrscheinlichkeit von $g_{jk}(v_i)/g_{jk}$ für die Wahl eines Weges, auf dem i liegt. Der Akteur-*Betweenness*index berechnet sich dann für v_i durch die Summe der Wahrscheinlichkeiten

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

für $i \neq j, k$. Das Minimum von 0 wird erreicht, falls v_i auf keinem der kürzesten Wege liegt.
Das Maximum hat den Wert $(n-1)(n-2)/2$, was der Anzahl der Paare von Akteuren entspricht, die v_i nicht enthalten. Das Maximum wird erreicht, falls v_i auf allen kürzesten Wegen liegt, z.B. beim *Stern*. Eine Normierung des Index kann mittels Division durch den maximalen Wert, also
$(n-1)(n-2)/2$ erhalten werden.

$$C'_B(v_i) := \frac{C_B(v_i)}{(n-1)(n-2)/2}$$

Der Wertebereich ist das Intervall [0,1].
Der Vorteil zum normierten *Closeness*index im ungerichteten Fall ist der, daß der *Betweenness*index auch für nicht zusammenhängende Graphen berechnet werden kann. Das Finden von kürzesten Wegen und der Anzahl von Wegen, auf denen v_i liegt, ist bereits implementiert in Programmen wie UCINET u.a. .

Als *Betweenness*-Werte für unsere Spezialgraphen ergeben sich:
Stern: $C'_B(v_1) = 1, C'_B(v_i) = 0$ für i=2, ... ,7
Kreis: $C'_B(v_i) = 0,2$für alle i
Pfad: Die Werte liegen hier im Bereich von $C'_B(v_6) = C'_B(v_7) = 0$ über $C'_B(v_2) = C'_B(v_3) = 0,533$ bis $C'_B(v_1) = 0,6$.

Beispiel Wieder zurück zum Netzwerk der Florentinischen Familien. Auffällig ist hier, daß im Gegensatz zu den bisher betrachteten Fällen die Indizes vieler Akteure den Wert 0 annehmen. Es existieren also einige Akteure, die nicht auf kürzesten Wegen liegen. Deutlich wird dies bei Betrachtung der Figur 3.4. Die Tabelle 3.3 gibt die berechneten Indizes wieder.

Gruppen-Betweenness-Zentralisierung Beginnen wir wieder mit der Berechnung mittels des allgemeinen Gruppenindex. Der Numerator läßt sich schreiben als $\sum_{i=1}^{n}[C_B(v^*) - C_B(v_i)]$. Da der maximale Wert für den *Stern* angenommen wird, erhalten wir für den Denominator den Wert $(n-1)^2(n-2)/2$. Daraus ergibt sich

$$C_B := \frac{2\sum_{i=1}^{n}[C_B(v^*) - C_B(v_i)]}{[(n-1)^2(n-2)]}$$

als Gruppen-*Betweenness*index. Analog kann der Index dargestellt werden durch

$$C_B := \frac{\sum_{i=1}^{n}[C'_B(v^*) - C'_B(v_i)]}{(n-1)},$$

indem die normierten Indizes verwendet werden. Das Minimum von 0 wird angenommen, falls alle Akteure den gleichen Akteur-*Betweenness*index besitzen, z. B. beim *Kreis*. Für den *Stern* wird das Maximum von 1 erreicht, der *Pfad* erhält einen Wert von 0,311.
Eine weitere Möglichkeit, den Gruppen-*Betweenness*index zu berechnen, wäre wiederum die Anwendung der Varianz der Akteur-*Betweenness*indizes. Da dies analog zum *Closeness*index geschieht, sei auf die genaue Beschreibung verzichtet.

Tabelle 3.3: Betweenness-Indizes für Padgett's Florentinische Familien Netzwerk

Akteurnummer	Familienname	Betweennessindex
1	Acciaiuoli	0.0
2	Albizzi	0.212454
3	Barbadori	0.0934066
4	Bischeri	0.104396
5	Castellani	0.0549451
6	Ginori	0.0
7	Guadagni	0.254579
8	Lamberteschi	0.0
9	Medici	0.521978
10	Pazzi	0.0
11	Peruzzi	0.0
12	Pucci	0.0
13	Ridolfi	0.113553
14	Salvati	0.142857
15	Strozzi	0.102564
16	Tornabuoni	0.0915751

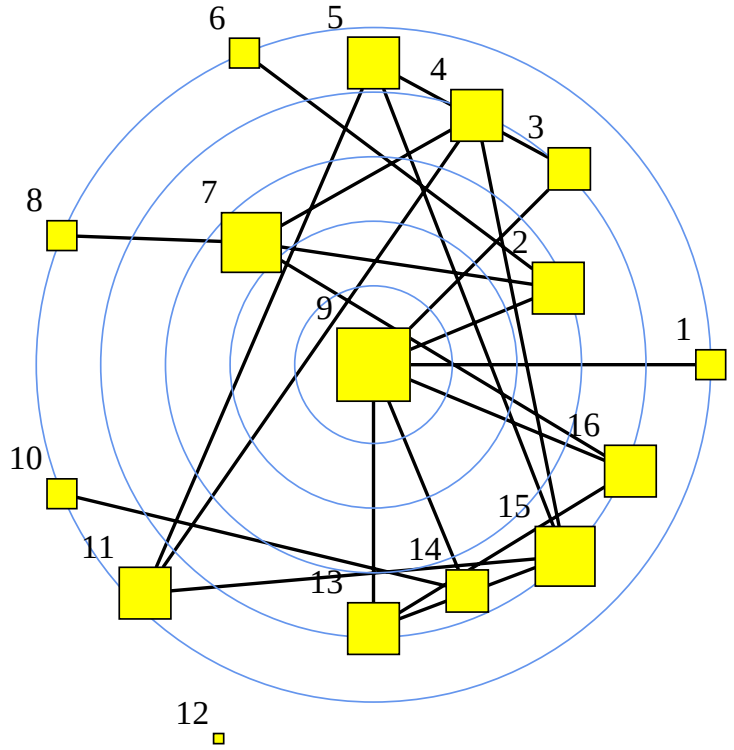


Abbildung 3.4: Betweennessindex für Padgett's Florentinische Familien

Betweenness-Zentralität für gerichtete Beziehungen Diese Indizes wurden unter Benutzung spezieller Algorithmen und Theorien für ungerichtete Beziehungen entwickelt. In [Gou87] wird der *Betweenness*index auf gerichtete Beziehungen durch Betrachtung der kürzesten Wege zwischen je zwei Akteuren erweitert. Ferner wird gezeigt, daß sich der Algorithmus für ungerichtete Beziehungen auch auf gerichtete anwenden läßt, da er automatisch geordnete Paare von Akteuren benutzt. Die Indizes $C_B(v_i)$ werden also für gerichtete und ungerichtete Beziehungen korrekt berechnet. Die normierten Indizes $C'_B(v_i)$ jedoch müssen mit 2 multipliziert werden, da der maximale Wert für den Index $(n-1)(n-2)$ ist. Diese Aussage basiert auf der Annahme, daß eine gerichtete Beziehung durch Verändern von *gegenseitigen* Beziehungen zu Kanten und Ignoranz der asymmetrischen Beziehungen in eine ungerichtete umgewandelt werden kann. Im ungerichteten Graphen existiert somit genau dann eine Kante, wenn beide Akteure sich im Originaldigraph gewählt haben.

3.3.5 Information-Zentralität

Bei der Berechnung des *Betweennessindex* sind wir davon ausgegangen, daß alle kürzesten Wege mit der gleichen Wahrscheinlichkeit gewählt werden. Betrachten wir nun die Akteure genauer, die auf diesen kürzesten Wege liegen. Pfade, die über Knoten mit hohem Knotengrad verlaufen, werden eher gewählt als andere. Der Grund dafür ist die Tatsache, daß ein Knoten mit z.B. Grad 10 auf wahrscheinlich mehr kürzesten Wegen liegt als ein Knoten mit einem geringeren Grad. Dies erklärt sich durch die hohe Zahl von inzidenten Kanten. Freeman's Annahme, daß alle Pfade mit derselben Wahrscheinlichkeit gewählt werden wäre also korrekt, wenn alle Knoten denselben Grad besitzen.
Ferner fällt auf, daß nur kürzeste Wege betrachtet werden. Dies muß aber nicht unbedingt realistisch sein. Denn es kann durchaus sinnvoll sein, längere Wege zu wählen, beispielweise um eine Information zu schützen oder zu verstecken. Im Fall des *Information*index werden daher *alle* Wege zwischen zwei Akteuren betrachtet. Ausserdem erhalten die Wege Gewichte, die abhängig von der Länge der Wege sind. Eine mögliche Gewichtung versieht jeden Weg mit dem Inversen der Länge der untersuchten Wege. Multiplizieren wir diese Gewichte mit der Länge eines kürzesten Weges, beschränken sich die Werte auf den Bereich zwischen 0 und 1.

Akteur-Information-Zentralität Die Berechnung der Information-Zentralität setzt eine Reihe von Matrixoperationen voraus. Ausgangspunkt ist eine $n \times n$ Matrix bei n Akteuren. Initialisiert wird sie mit folgenden Werten

$$A = \begin{cases} a_{ii} = 1 + \text{ Summe der Werte aller zu } v_i \text{ inzidenten Kanten,} \\ a_{ij} = 1, \text{ falls } v_i \text{ und } v_j \text{ nicht adjazent, } i \neq j \\ a_{ij} = 1 - x_{ij} \text{ sonst} \end{cases} \tag{3.2}$$

x_{ij} bezeichnet dabei den Wert der Beziehung zwischen Akteur i und j. Die Elemente der Matrix können also aus den Einträgen der Soziomatrix berechnet werden. Im nächsten Schritt wird die inverse Matrix zu A berechnet, $C := A^{-1}$, mit den Elementen c_{ij}. An dieser Stelle entscheidet sich, ob der Akteur-*Information*index berechnet werden kann oder nicht, denn nicht jede Matrix ist invertierbar. Besitzt die Soziomatrix eine oder mehrere Spalten oder Zeilen, die nur aus 0 bestehen, ist C nicht definiert, der Akteur-*Information*index kann nicht berechnet werden. Dies ist bei isolierten Knoten der Fall. Bestehen keine Beziehungen, sind die gesamte Spalte und Zeile des

Akteurs mit 0 belegt. Löschen wir isolierte Knoten aus der Menge der Akteure, können wir die Indizes für die verbleibenden, nicht-isolierten Knoten berechnen. Zur endgültigen Berechnung des Index benötigen wir noch zwei weitere Maße
T = $\sum_{i=1}^{n} c_{ii}$, die Summe der Diagonalelemente und
R = $\sum_{j=1}^{n} c_{ij}$, die Zeilensumme einer beliebigen Zeile,
da alle Zeilensummen gleich sind.
Der Akteur-*Information*index hat dann die Form

$$C_I(n_i) = \frac{1}{(c_{ii} + (T - 2R)/n)}$$

Er mißt, wieviel Information in den Wegen enthalten ist, die bei einem speziellen Akteur beginnen oder enden. Das Minimum liegt bei 0, Maximum existiert keines. Falls $T = 2R$ und $c_{ii} = 0$, wird durch 0 dividiert, der Bruch wird unendlich.
Der *relative Informationindex*

$$C'_I(v_i) = \frac{C_I(v_i)}{\sum_i C_I(v_i)}$$

stellt eine Normierung dar, indem jeder Index durch die Gesamtsumme der Indizes geteilt wird. Die Werte beschränken sich auf das Intervall [0,1] und summieren sich zu 1 auf. Somit können die Werte als Verhältnis von "totalem Informationsfluß" eines Graphen, kontrolliert von einem einzigen Akteur, aufgefaßt werden.
Verdeutlichen wir uns die Situation anhand der Spezialgraphen *Stern*, *Kreis* und *Pfad*.
Stern: $C'_I(v_1) = 0,234, C'_I(v_i) = 0,1277$ für i = 2, ... ,7
Bemerkung:
Obwohl v_1 auf allen kürzesten Wegen liegt, ist $C'_I(v_i) \neq 0$ für i=2, ... ,7
Kreis: $C'_I(v_i) = 0,1429$ für alle i
Pfad : Hier bewegen sich die Werte zwischen $C'_I(v_1) = 0,1822$ und $C'_I(v_7) = 0,1041$.

Gruppen-Informations-Zentralisierung Die Anwendung des allgemeinen Gruppenmaßes ist eine Möglichkeit, den *Information*index auf dem Gruppenlevel zu berechnen. Unglücklicherweise ist bisher keine Quelle bekannt, in welcher der Denominator dafür berechnet ist.
Berechnen wir den Gruppen-*Information*index mittels Durchschnittsinfor-

mation über alle Akteure

$$\bar{C}_I = \sum_i^n C_I(v_i),$$

wird ein Vergleich mit anderen Netzwerken schwierig. Die Grenzen hängen von der Anzahl der Akteure n ab. Da die Berechnung eines Gruppenmaßes dem Vergleich von Netzwerken miteinander dient, wollen wir diesen Durchschnitt nicht als echten Zentralitätsindex betrachten.
Die Berechnung der Varianz der Akteurindizes wiederum beschreibt einen echten Gruppenindex

$$S_I^2 = [\sum_{i=1}^n (C_I'(v_i) - \bar{C}_I)^2]/n.$$

Für die drei Spezialgraphen ergeben sich die folgenden Varianzen:
Stern: 0,001614
Kreis: 0,0
Pfad : 0,000986
was bedeutet, daß der *Stern* der heterogenste und der *Kreis* der am wenigsten heterogene Graph ist. Dieser *Information*index ist der einzige (dem Autor bekannte) Index, der auf gewichtete Relationen angewandt werden kann. Durch die Betrachtung aller Wege zwischen zwei Akteuren verallgemeinert er den von Freeman beschriebenen *Betweenness*index, der nur die kürzesten Wege zur Berechnung berücksichtigt.

Information-Zentralität für gerichtete Beziehungen Für den *Information*index müssen wir alle gerichteten Wege zwischen zwei Akteuren betrachten. Bisher ist jedoch nicht bekannt, wie die Theorie von Stephenson und Zelen [SZ89], die zur Beschreibung des *Information*index benutzt wurde, auf gerichtete Beziehungen erweitert werden kann. Daher empfiehlt sich im Fall von gerichteten Beziehungen der Gebrauch der Indizes $C_D'(v_i)$ und $C_C'(v_i)$ oder $C_C^*(v_i)$.

3.3.6 Akteur-Degree-Prestige

Das einfachste Akteur-Prestigemaß ist der Knoteneingangsgrad eines Akteurs, ähnlich der *Degree*-Zentralität, bei welcher der Knotenausgangsgrad zur Berechnung benutzt wird. Der Index hat das Aussehen

$$P_D(v_i) = deg_{in}(v_i)$$

Wie beim *Degree*-Zentralitätsmaß und den verwendeten Knotenausgangsgraden hängt das Maß von der Anzahl der Akteure, n, ab. Eine Normierung ist daher mittels Division durch den maximalen Knoteneingangsgrad möglich.

$$P'_D(v_i) = deg_{in}(v_i)/n - 1$$

Dieses Maß wird auch als "relativer Knoteneingangsgrad" bezeichnet. Je größer der Index, desto wichtiger ist der Akteur. Die Werte liegen durch die Normierung im Intervall [0,1]. Das Maximum von 1 wird erreicht, wenn ein Akteur von allen anderen "gewählt" wird, d.h. gerichtete Verbindungen von allen Knoten *zu* diesem einen Knoten existieren.

Gruppen-Degree-Prestige Die Berechnung des *Degree*-Prestige für Gruppen kann mit Hilfe der Varianz durchgeführt werden. Sie setzt sich zusammen aus der Summe der quadrierten Differenzen jedes Knoteneingangsgrades und des maximalen Knoteneingangsgrades

$$S_D^2 = [\sum_{i=1}^{n}(deg_{in}(v_i) - max_i(deg_{in}(v_i)))^2]$$

Die Berechnung erfolgt analog zur Berechnung der *Degree*-Zentralität von Gruppen im gerichteten Fall. Der Unterschied besteht nur in der Verwendung des Knoteneingangsgrades im Unterschied zum Knotenausgangsgrad.

3.3.7 Akteur-Proximity-Prestige

In der Berechnung des *Degree*-Prestige wurden nur die direkten Beziehungen betrachtet. Das *Proximity*-Prestige erweitert diese Anschauungsweise durch Hinzunahme des Einflußbereiches eines Akteurs i. Der Einflußbereich eines Akteurs i beschreibt in diesem Fall *die* Menge von Akteuren, von denen aus Akteur i erreichbar ist. Formal gesehen beinhaltet eine solche *Domain* alle Akteure, deren Einträge in der i-ten Spalte der Distanzmatrix endlich sind.
Alle Graphen, die hier bei der Messung des *Proximity*-Prestigemaßes betrachtet werden, müssen gerichtet sein, da Prestige sonst nicht gemessen werden kann. Das bedeutet insbesondere, daß sich die Distanzen von i *zu* den anderen Akteuren vermutlich von der Distanz *von* den anderen Akteuren unterscheiden. Die *Nähe* oder *Wichtigkeit* eines Akteurs i hängt also von der Distanz *zu* diesem Akteur ab, nicht von der Distanz, die *von* diesem Akteur zu allen anderen berechnet werden kann.
Definiere I_i als die Anzahl der Akteure im Einflußbereich von i. Dann erhalten wir ein grobes Maß für die Nähe durch die Berechnung der durchschnittlichen

Distanz $\sum d(v_i, v_j)/I_i$. Es sollte beachtet werden, daß hier nicht alle Akteure, sondern nur diejenigen aus dem Einflußbereich von i betrachtet werden. Das bietet den Vorteil, daß sich das Maß auch auf nicht-zusammenhängende Netzwerke anwenden läßt. Da der Index von der Größe des Netzwerkes bzw. von der Größe des Einflußbereiches abhängt, ist es schwer, die Werte mit denen anderer Netzwerke zu vergleichen.
Ein besseres Maß für Proximity erhält man, indem das Verhältnis der Akteure, die i erreichen können zur durchschnittlichen Distanz der Akteure zu i betrachtet wird.

$$P_P(v_i) = \frac{I_i/(n-1)}{\sum d(v_j, v_i)/I_i}$$

Auch hier wird nur über die Akteure im Einflußbereich aufsummiert. Das Verhältnis wird größer, falls die Akteure, die i erreichen können, im Durchschnitt näher sind.
Die Eigenschaften dieses Maßes entsprechen denen des *Closeness*-Zentralitätsindex. Das Maximum von 1 wird erreicht, falls alle Akteure adjazent zu n_i sind, also gerichtete Beziehungen von allen Akteuren zu Akteur i bestehen und die Länge 1 besitzen. Ist ein Akteur i nicht erreichbar, also $I_i = 0$, dann wird das Minimum von $P_P(v_i) = 0$ angenommen. Die Größe des Index spiegelt wider, wie nah ein Akteur der Menge der Akteure insgesamt ist.

Gruppen-Proximity-Prestige Gruppenmaße lassen sich einfach anhand des Durchschnitts oder der Varianz der Akteur-*Proximity*indizes definieren.

1. $\bar{P}_P = \sum_{i=1}^{n} P_P(v_i)/n$
2. $S_P^2 = \sum_{i=1}^{n} (P_P(v_i) - \bar{P}_P)^2/n$

Die Werte für $\bar{P}_P$ liegen im Intervall [0,1]. Das Minimum von 0 wird für einen leeren Graphen, das Maximum von 1 für einen vollständigen Graphen angenommen.
Die Varianz ist positiv und mißt, wie heterogen eine Menge von Akteuren im Bezug auf Nähe ist.
Ein weiteres Maß für Proximity wurde in [Har59] vorgestellt. Er benutzt neben dem Begriff des *Status*, den er als Gesamtdistanz von Akteur i zu allen anderen Akteuren definiert, den Begriff *Contrastatus*, der die Gesamtdistanz von *allen* Akteuren zu Akteur i beschreibt. Die Maße würden sich also in der hier verwendeten Schreibweise darstellen als
$\sum_j d(v_i, v_j)$ und
$\sum_j d(v_j, v_i)$.

Status wird somit als Synonym für Akteur-*Closeness*-Zentralität und *Contrastatus* für Akteur-*Proximity*-Prestige benutzt. Der *Netzstatus* eines Akteurs beschreibt die Differenz zwischen diesen beiden Summen.

3.3.8 Status (Rang)

Bisher wurde die Betrachtungsweise für Prestige immer um einen Schritt erweitert. Der erste Index betrachtet nur die direkten Beziehungen zwischen Akteuren, während der zweite alle Akteure aus dem Einflußbereich einschließt, also Akteure, die direkt oder indirekt in Beziehung stehen. Im nächsten Schritt werden diese Beziehungen mit folgendem Ziel gewichtet: Akteure, die von wichtigen Akteuren gewählt wurden, erhalten einen höheren Prestigewert als andere, die von unwichtigeren Akteuren gewählt wurden. Der Sinn dieser Wertung wird an einem Beispiel deutlich. Nehmen wir an, die Beziehung in dem betrachteten Netzwerk bedeutet "um Rat fragen". Nehmen wir weiter an, daß viele Akteure existieren, die den Akteur mit Namen "Weise" um Rat fragen. Dadurch erhält "Weise" folgerichtig einen hohen Prestigewert. Richtet dieser Akteur "Weise" wiederum seine Fragen an den Akteur mit dem Namen "Noch-Weiser", so muß dieser einen noch höheren Prestigewert erhalten als "Weise", da er in diesem Fall die "letzte Instanz" ist. Die Berechnung des Statusindex erweist sich als aufwendiger als die Berechnung der bisherigen Maße. Daher müssen zuerst einige Konstrukte eingeführt werden.
Definiere $P_R(v_i)$ als *Status*- oder *Rang*-Prestigemaß für Akteur i. Die Idee ist, daß der Rang eines Akteurs als Funktion der Ränge der anderen Akteure angesehen wird [Kat53]. Betrachte dazu die *i-te* Spalte der Soziomatrix, die beinhaltet, welche Akteure i wählen. Durch Multiplikation der Einträge x_{ij} mit dem Rang P_R der Akteure erhalten wir eine Linearkombination, die den Rang von i mißt.

$$P_R(v_i) = x_{1i} \cdot P_R(v_1) + x_{2i} \cdot P_R(v_2) + \ldots + x_{ni} \cdot P_R(v_n)$$

Nehmen wir an, daß v_2 von v_5 und v_7 gewählt wird, x_{52} und x_{72} seien 1 und alle anderen Einträge in der zweiten Zeile der Soziomatrix seien 0. Der Rang von v_2 würde sich dann berechnen als

$$P_R(v_2) = P_R(v_5) + P_R(v_7)$$

Hier wird deutlich, daß bei hohem Rang von v_5 und v_7 Akteur 2 ebenfalls einen hohen Rang erhält, durch die Addition sogar einen noch höheren.
Beschreiben wir dieses Problem auf mathematischer Ebene. Es existieren n

Gleichungen, die alle von den Indizes $P_R(v_i)$ abhängen, also n Gleichungen mit n Unbekannten. Definiere p als den Vektor der Rangindizes

$$p = (P_R(v_1), P_R(v_2), \ldots, P_R(v_n))$$

und X als die gesamte Soziomatrix. Daraus ergibt sich

$$p = X'p \text{ oder analog}$$

$$(I - X')p = 0.$$

I beschreibt hier die Einheitsmatrix der Dimension n und p und 0 Vektoren der Länge n. Diese Gleichung ist identisch zu einer charakteristischen Gleichung, um das Eigensystem einer Matrix zu finden, in dem p der Eigenvektor von X' zum Eigenwert 1 ist. Da diese Gleichung nicht notwendigerweise endliche Lösungen produziert, muß ein Weg gefunden werden, der das Problem löst und unendliche Lösungen vermeidet.
Im ersten Schritt wird die Soziomatrix so normiert, daß die Spaltensumme 1 ergibt. Dies hat zur Folge, daß eine charakteristische Matrixgleichung mit einer wohlbekannten Lösung entsteht. Insbesondere wird bei diesem Vorgehen der Eigenvektor zum größten Eigenwert der normierten Matrix X' gesucht. Da die Summen der Spalten der Soziomatrix 1 ergeben, wird der erste Eigenwert 1 sein und der dazugehörige Eigenvektor p_1 der Vektor der Rangindizes, d.h.

$$p_1 = (P_R(v_1), \ldots, P_R(v_n))'$$

Diese Methode kann noch ein wenig verändert werden, um die Lösung zu erhalten, die in der Implementation umgesetzt wurde. Dazu muss die Matrix nicht normiert werden. Im Gegenzug fließt der Vektor der Spaltensumme in die Berechnung ein. Ein weiterer Faktor a wird eingeführt. Die Ausgangssituation an dieser Stelle ist die Matrix $aX + a^2X^2 + \ldots + a^kX^k + \ldots$. was bedeutet, daß man eine Matrix mit einer verminderten Anzahl von Wegen zwischen je zwei Knoten betrachtet. Da der Parameter a unbekannt ist und nicht genau bestimmt werden kann, muß er geschätzt werden. Katz gibt als Schranken für a den größten Eigenwert λ_1 und $2\lambda_1$ an. Als einfache, obere Schranke kann die größte Spalten- oder Zeilensumme definiert werden. Um eine Lösung für das modifizierte System von Gleichungen zu finden, muß ein Vektor p berechnet werden, der das System

$$[(1/a)I - X']p = x$$

löst. x ist der Vektor der Knoteneingangsgrade der nichtnormierten Matrix, entspricht also dem Vektor der Spaltensummen. Unterschiede zum ursprünglichen Gleichungssystem sind somit zum einen das Erscheinen des Faktors a

und zum anderen die Gleichsetzung mit Vektor x, der aus der Summe der "direct choices", der *direkten Wahlen* besteht und nicht der Nullvektor ist. Ein Beispiel zur Berechnung der Statusprestigewerte eines gerichteten Netzwerkes findet sich in Kapitel 5, Abschnitt 5.1.

Kapitel 4

Visualisierung von Status

Visualisierung ist ein wichtiger Faktor bei der Analyse von Netzwerken. Durch sie kann auf einfache Weise die Struktur eines Netzwerkes veranschaulicht und erkannt werden. Insbesondere lassen sich die wichtigen Akteure und Beziehungen dazwischen hervorheben. Der folgende Teil besteht aus zwei Hauptabschnitten, der Untersuchung von Wichtigkeit und Geschichte der Visualisierung einerseits und der Beschreibung der drei Aspekte der Visualisierung von Information, *Substanz, Design* und *Algorithmus*, andererseits. Das Ziel dieser Visualisierung von Status ist kein Balkendiagramm, an dem die Werte der einzelnen Akteurindizes abgelesen werden können, sondern die kompakte Darstellung von Statuswerten, Akteuren und Beziehungen zwischen ihnen. Eine zentrale Rolle spielen dabei die Beziehungen, die zwischen den Akteuren bestehen und durch ein Balkendiagramm allein nicht dargestellt werden können. Der Zusammenhang zwischen den Akteuren geht verloren. Eine genaue Erklärung des Zieles wird im Abschnitt *Substanz* gegeben.

4.1 Entwicklung und Ziel der Visualisierung

Geschichtliche Entwicklung der Visualisierung von Netzwerken Aufgrund der Fülle von zur Verfügung stehenden Daten, sei hier nur ein kleiner Einblick gegeben. Für ausführlichere Informationen sei auf [Klo81] oder [BKR+99] verwiesen.
Eine der ersten Visualisierungen von Netzwerken ist das Soziogramm von Moreno (1933), das schon in Kapitel 3 beschrieben wurde. Es hatte große Auswirkungen auf Teile der Sozialwissenschaften, der Sozialpsychologie, Sozialanthropologie und andere Bereiche der Forschung. Ein Beispiel für die Anwendung der Soziogramme ist die *Hawthorne-Studie*, in der soziale Bezie-

hungen durch Soziogramme beschrieben werden [RD39]. Die Akteure werden dabei durch Kreise dargestellt, Beziehungen durch Pfeile. Um den Status der Akteure anzuzeigen, werden sie auf horizontalen Linien plaziert. Diese Darstellung erinnert stark an die implementierte Form der Visualisierung von Status in sozialen Netzwerken. Die Knoten werden zwar nach anderen Maßstäben als in der Hawthorne-Studie auf die verschiedenen Level verteilt, die Grundstruktur ist jedoch dieselbe.
Durch das Aufkommen von Computern verbessern sich die Möglichkeiten zur graphischen Darstellung erheblich, beispielsweise durch interaktives Bewegen von Akteuren oder die Nutzung von Animationen. Ein Beispiel, bei dem der Einsatz von Computern hilfreich gewesen wäre, ist das *Lady Bountiful* Diagramm [LS38]. Lundberg und Steel untersuchten ein Dorf mit etwa 1000 Einwohnern und erhielten dabei Daten von 200 Familien (94%). Es entstanden aufgrund der Fülle der Daten 8 Soziogramme, von denen eines als das *Lady Bountiful* Diagramm bekannt wurde. In dieser Situation wurden *Cluster* verwendet, um Individuen zu einer Einheit zusammenzuschließen. Auf eine ähnliche Vorgehensweise bei der Bestimmung der Layer im später beschriebenen Algorithmus sei an dieser Stelle schon einmal hingewiesen. Weiteres zum *Lady Bountiful* Diagramm findet sich in [Klo81].
Zu Beginn der Entwicklung entstanden zwei besondere Arten von Präsentationen für Soziogramme. Die erste bestand darin, die zentralsten Akteure in die Mitte des Blattes zu zeichnen, um die restlichen Akteure um sie herum zu plazieren. Um dabei eine gute Übersicht zu erhalten, wurde darauf geachtet, so wenig Kantenkreuzungen wie möglich zu verursachen. Diese Idee wurde 1940 von Northway [Nor40] ausgearbeitet, der die Akteure nach ihrer Zentralität auf konzentrischen Kreisen anordnete, so daß die zentralsten Akteure in der Mitte plaziert wurden. Je weniger zentral ein Akteur war, desto weiter wurde er nach außen versetzt.
Bei der zweiten Methode wurden die Knoten auf einer Kreislinie angebracht, so daß die Kanten zwischen ihnen alle innerhalb der Kreislinie gezeichnet werden konnten. Dadurch wurde versucht, die Anzahl der Kantenüberschneidungen zu minimieren. Man bezeichnet diese Darstellung auch als *Kreisdiagramm*. Auch in den heute verwendeten Programmen, z.B. dem in der Implementation verwendeten `GraphWin`, werden Funktionen zur Verfügung gestellt, die diese Form von Layout erzeugen können.
Um die Knoten mit Attributen zu versehen, wurden Formen und Beschriftungen benutzt, durch welche die Akteure zu unterscheiden waren. Diese Vorgehensweise wurde bis heute beibehalten. Das erste dreidimensionale Bild erschien 1966 in [LG66].
In den 70er Jahren konnten mit Hilfe von Computerprogrammen Daten visualisiert werden, auch in größeren Mengen und in relativ kurzer Zeit im

Vergleich zur manuellen Ausarbeitung. Techniken und Ideen zur Visualisierung, die gleichzeitig mit der Einführung von Hilfsmitteln der relationalen Analyse in der Sozialwissenschaft entwickelt wurden, fasste man in den 70er und 80er Jahren unter dem Begriff "Soziale Netzwerkanalyse" zusammen. Es wurden Programme entwickelt, die mittels mathematischer und statistischer Techniken Indizes für die Akteure und komplette Netzwerkstrukturen berechnen, mit Methoden wie "multi-dimensional scaling" (MDS) oder Clusteranalyse. Seit den 70er Jahren wurde MDS von vielen Autoren benutzt, um *Proximity* oder *Zentralität* von Akteuren zu visualisieren [LP76], [LK87].

Ziel der Visualisierung Programme für die Netzwerkanalyse dienen in erster Linie der Berechnung von Maßen wie Zentralität, Prestige oder der Dichte eines Netzwerkes. Einige bieten jedoch auch die Möglichkeit der graphischen Darstellung von Daten, was für das Verständnis sehr wichtig sein kann, da man auf einen Blick wichtige Akteure und Beziehungen erkennt. In der Literatur finden sich deutlich mehr Artikel und Bücher, die sich mit der Berechnung von Maßen in Netzwerken beschäftigen, als mit der Visualisierung derselben. Doch ganz ohne Visualisierung kann es sehr schwer fallen, sich mit dem Thema zu beschäftigen. Die graphische Darstellung erleichtert es uns, Probleme zu sehen, und diese zu lösen. Aus diesem Grund wurden in Kapitel 3 einfache Beispielgraphen, die sich nicht in der Anzahl ihrer Knoten, aber in der Art, wie diese miteinander verbunden sind, unterscheiden, eingeführt, um sofort ein Gefühl entwickeln zu können, was die einzelnen, berechneten Maße wirklich wiedergeben sollen.
Die Techniken zur Darstellung haben sich seit dem Beginn der Netzwerkvisualisierung nicht wesentlich geändert, es wurden lediglich einige Verfeinerungen, oder die Hinzunahme von Hilfsmitteln vorgenommen. Gerade die Zuhilfenahme von Computern macht es heutzutage möglich, die Darstellung dem gewünschten Ideal von *Schönheit* anzupassen. Es können Veränderungen vorgenommen werden durch direktes Bewegen von Objekten mittels Mouse oder Keyboard, was früher bei Zeichnungen von Hand nicht möglich war. Manche Forscher benutzten Pokerchips, um die Akteure darzustellen und diese beliebig verschieben zu können, bevor sie eine Zeichnung anfertigten [Nor40]. Eine optimale Darstellung eines Netzwerkes mit einer Knotenzahl von etwa 100 war vor der Benutzung von Rechnern kaum denkbar und extrem zeitaufwendig. Zu große Netzwerke wurden auf mehrere einzelne Diagramme aufgeteilt, was z.B. beim *Lady Bountiful*- Diagramm zu beobachten ist.
Wir sehen also, daß die Visualisierung einen wichtigen Platz in der Analyse von Netzwerken einnimmt und daher nicht vernachlässigt werden darf. Zusätzlich hilft die Betrachtung des Aussehens von visualisierten Netzwerken

Tabelle 4.1: Adjazenzmatrix eines gerichteten Graphen

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

bei der Bewertung von bisherigen Arten der Visualisierung ebenso wie bei der Entwicklung objektiver Standards von *graphischer Schönheit*.
Die Daten eines Netzwerkes lassen sich auf verschiedene Arten darstellen. Ein Netzwerk kann beispielsweise durch eine Adjazenzmatrix $A = (a_{ij})$ beschrieben werden. Die Akteure werden durch Zahlen oder Buchstaben repräsentiert.
Besteht eine Beziehung zwischen Akteur i und Akteur j, enthält die Matrix an den Positionen a_{ij} und a_{ji} den Eintrag 1, ansonsten den Eintrag 0. Dies gilt natürlich nur bei ungerichteten Graphen, ansonsten erhält die Matrix nur an einer der beiden Positionen den Eintrag 1 (s. Abbildung 4.1). Da wir uns in diesem Abschnitt auf die Visualisierung von Status konzentrieren wollen, sollten wir, wie in Kapitel 3 beschrieben, von gerichteten Beziehungen, also gerichteten Graphen ausgehen.
Die graphische Darstellung derselben Daten durch Abbildung 4.1 ist erheblich einfacher zu lesen, als die oben beschriebene Adjazenzmatrix. Dies bezieht sich nicht darauf, daß direkte Beziehungen leicht zu erkennen sind (welche in der Matrix auch einfach zu finden sind), sondern insbesondere auf das Erkennen der indirekten Beziehungen. Solche Beziehungen können in der Matrixdarstellung nur sehr schlecht und mit deutlich höherem Zeitaufwand erkannt werden, z.B. der Weg von Knoten 2 über Knoten 3 und 4 bis zum Knoten 8. Daher bietet es sich an, sich bei der Netzwerkanalyse intensiv mit der Visualisierung, insbesondere der graphischen Darstellung, zu befassen. Bei der bloßen Analyse von Tabellen oder Matrizen kann es sehr leicht passieren, daß der Überblick verloren geht, einfache Zusammenhänge, die in einem Bild sofort erkannt werden können, werden erst nach langwierigen Untersuchungen deutlich.
Die graphische Visualisierung bietet ausserdem die Möglichkeit, auf einfache

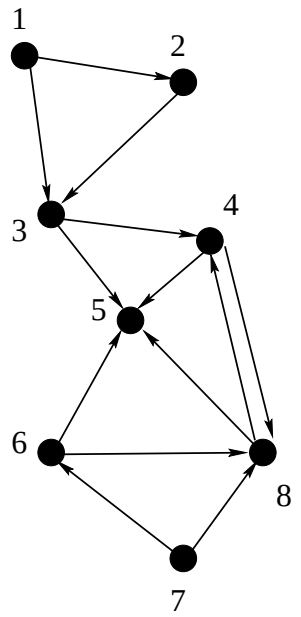


Abbildung 4.1: Graphische Darstellung des Graphen aus Tabelle 4.1

Weise zusätzliche Informationen sichtbar zu machen. Diese Informationen lassen sich im wesentlichen in zwei Arten unterteilen, Attribute *semantischer* und *syntaktischer* Art. Darauf soll an einem späteren Zeitpunkt noch genauer eingegangen werden. Es sei nur soviel vorweg genommen, daß sich die Möglichkeiten zur Sichtbarmachung von Information nicht nur auf die Formen, Farben und Größen der Knoten und Kanten beziehen, sondern auch, was in unserem Fall sehr wichtig ist, auf die Positionierung derselben. Dadurch, daß Knoten oder Kanten an bestimmte Positionen gesetzt werden, lassen sich Eigenschaften wie der oben beschriebene *Status* sehr leicht erkennen.
Isolierte Knoten, also Akteure, die in keiner Beziehung zu den anderen Akteuren stehen, sind auch in Matrizen schnell zu finden, da sowohl Zeile als auch Spalte nur Nullen enthalten. Schwieriger wird es beim Zerfall des Graphen in mehrere Zusammenhangskomponenten, wenn diese nicht nur aus isolierten Knoten bestehen. Gibt es also Gruppen von Akteuren in einem Netzwerk, die zwar innerhalb der Gruppe miteinander agieren, mit den anderen Gruppen aber nicht in Beziehung stehen? Oder existieren Beziehungen, die nur in eine Richtung fließen, also eine eindeutige Hierarchie bilden? Fragen solcher Art können durch eine Visualisierung schneller und einfacher beantwortet werden.

Es ist klar, daß es für dieselben Daten verschiedene Darstellungen gibt. Dabei

können sich die visualisierten Daten nicht nur in der Art der Wiedergabe (z.B. "Werden Akteure nach Zentralitätswerten auf konzentrischen Kreisen angeordnet oder nach Prestigewerten auf verschiedenen Schichten?"), sondern auch in der Qualität der Darstellung unterscheiden. Diese Qualität hängt aber nicht unbedingt von den benutzten Hilfsmitteln ab, die im Laufe der Zeit immer vielseitiger geworden sind. So ist es heute möglich, mittels spezieller Software, neuen Algorithmen, Verwendung von 3D und Animationen die Gestalt eines Netzwerkes anders aussehen zu lassen, als das noch vor ein paar Jahren der Fall war. Durch farbige Darstellungen können wichtige Eigenschaften oder Informationen hervorgehoben werden, was jedoch nicht immer ein Garant für eine bessere Qualität ist. Tufte [Tuf83] stellt einige Prinzipien zur Bewertung der Qualität einer Visualisierung auf. Hervorragende Grafiken

- sind die gut dargestellte Präsentation von interessanten Daten - eine Sache von Substanz, von Statistiken und von Design
- bestehen aus komplexen Ideen in Verbindung mit Klarheit, Präzision und Effizienz
- geben dem Betrachter die größte Anzahl von Ideen in der kürzesten Zeit mit der wenigsten Farbe auf kleinstem Raum
- sind beinahe immer vielseitig
- erfordern es, die Wahrheit über die Daten zu sagen (*Datentreue*)

Diese Grundsätze bilden kein absolutes Muß für ein gelungenes Layout, können jedoch als ein Anhaltspunkt dienen, der unter Umständen angepaßt oder erweitert werden kann. Oft ist es nicht möglich, alle erwähnten Grundsätze zu beachten. Einerseits existieren Probleme technischer Art, bei denen die geforderten Prinzipien nicht umgesetzt werden können. Ein Beispiel dafür wäre ein gewichtetes Layout, bei dem die Beziehungen zwischen allen Akteuren mit dem Wert 1 gewichtet sind. Daher müssten alle Kanten im Netzwerk nach Tufte's Prinzip der Datentreue die gleiche Länge besitzen. Wir wissen, aber, daß dies nicht immer möglich ist oder die Darstellung unübersichtlich macht. Betrachte dazu als Beispiel den K_4, den vollständigen Graphen mit 4 Knoten. Im dreidimensionalen Fall kann die angesprochene Bedingung durch die Darstellung in einem gleichseitigen Tetraeder erfüllt werden. Im zweidimensionalen Fall jedoch ist dies unmöglich.
Auf der anderen Seite kann eine schlechtere Darstellung gewollt und durchaus von Nutzen sein. Bei den Tests von Algorithmen werden Graphen produziert,

die eine besonders schlechte Darstellung besitzen, um die Güte des Algorithmus zu testen. Es gibt also keine Techniken, die hohe Qualität garantieren, auch wenn in manchen Fällen einige Präsentationen nützlicher und effektiver sind.

Die drei Aspekte der Visualisierung von Information Die Visualisierung von Information besteht darin, gegebene Daten in eine graphische Darstellung umzuwandeln. Dies muß durch bestimmte Methoden bewerkstelligt werden, welche die in den Daten enthaltenen Informationen offenlegen. Eine akzeptable Visualisierungsmethode muß somit die wichtigen Informationen herausstellen, eine passende Abbildung definieren und ein entsprechendes Bild erzeugen. Wir wollen diese Aspekte mit *Substanz*, *Design* und *Algorithmus* bezeichnen und im folgenden genauer darauf eingehen.

4.2 Netzwerksubstanz

Die *Substanz* der Daten eines sozialen Netzwerkes ist es, Informationen über die sozialen Strukturen zu geben. Die typische *Substanz* eines sozialen Netzwerkes sind die *Netzwerkmaße*, also Indizes, die für die Akteure berechnet werden, um sie in ihrer Wichtigkeit messen und zu unterscheiden. Jede Netzwerkvisualisierung muß im Bezug auf die Information, die betrachtet wird, präzise sein, da ansonsten vielfältige Interpretationen möglich sind, die auch zu falscher Auslegung der Information führen kann.
Um die Wichtigkeit eines Akteurs innerhalb eines Netzwerkes zu messen, gibt es viele Möglichkeiten, die von den direkten und indirekten Beziehungen der Akteure abhängen. Auf der Ebene der Graphentheorie bedeutet dies z.B. messen von Ereichbarkeit, Länge von Wegen, Knotengrad usw. . Eine genaue Beschreibung befindet sich in Kapitel 3, in dem verschiedene Maße beschrieben und berechnet werden.
Aufgabe einer Visualisierung ist es, Eigenschaften des Netzwerkes darzustellen und hervorzuheben. Diese Eigenschaften können in zwei Kategorien eingeteilt werden, die *syntaktischen* und die *semantischen* Attribute, die in Tabelle 4.2 genauer beschrieben werden.

Maße für die *syntaktischen* Attribute sind nur abhängig von der Netzwerkstruktur, d.h. von bestehenden und nichtbestehenden Beziehungen zwischen den Akteuren. Diese Attribute spiegeln sich in der Visualisierung durch die

Tabelle 4.2: Kategorien der Netzwerksubstanz

Syntaktische Attribute	**Semantische Attribute**
abgeleitete Akteurattribute	**Akteurattribute z.B.**
Zentralität	Größe einer sozialen Einheit
Prestige	Alter einer sozialen Einheit
Strukturelle Aufteilung	**Aufteilung der Attribute z.B.**
Cohesive Untergruppen	Organisatorische Untereinheiten
Strukturell gleiche Akteure	
Abgeleitete Netzwerkattribute	**Netzwerkattribute z.B.**
Größe	Zeitraum der Datenerfassung
Dichte	Verläßlichkeit
Zentralisierung	
Ausgewählte strukturelle Rollen	**Ausgewählte Attribute z.B.**
Brücke	verschiedene institutionelle Rollen wie
Broker	Chef einer Firma
	Klassensprecher in einer Schulklasse

Position der Akteure wieder. Da wir uns auf *Status* beschränken wollen, spielen die vertikalen Koordinaten eine große Rolle. Sie beschreiben die Höhe des Levels, auf dem sich der Akteur befindet und geben gleichermaßen die berechneten Prestigewerte wieder.
Semantische Attribute beziehen sich auf die einzelnen Akteure und Beziehungen und hängen nur von diesen, nicht aber von der gesamten Netzwerkstruktur ab. Handelt es sich bei den Akteuren beispielsweise um Personen, so können sich diese im Aussehen unterscheiden durch Alter, Grösse, Geschlecht usw., was sich jedoch nicht auf ihre Position, sondern nur auf ihre Form, Farbe oder Größe auswirkt. Die ersten drei Unterkategorien der beiden Attribute *syntaktisch* und *semantisch*, befassen sich jeweils mit *Akteuren, Gruppen* und dem *gesamten Netzwerk*, also den Unterteilungen, die schon zuvor eingeführt wurden. Je nach der Art der Analyse, die gemacht werden soll, kann es sein, daß man sich für die eine oder andere Unterkategorie mehr interessiert oder alle Teile in ihrer Gesamtheit betrachten will, was es unmöglich macht, die Unterkategorien nach Wichtigkeit und Nutzen zu ordnen.

4.3 Design

Das *graphische Design* beschreibt die Art, wie die *Substanz* eines Netzwerkes in graphische Elemente umgesetzt wird. Da die Substanz eines sozialen Netzwerkes die Netzwerkmaße sind und uns in diesem Fall insbesondere der

Status von Akteuren interessiert, wollen wir uns mit der Frage beschäftigen, wie Akteure, Beziehungen und der Status der Akteure graphisch dargestellt werden.
Man kann bei der Netzwerkanalyse zwei Arten von strukturellen Perspektiven unterscheiden, die "Beziehungs-" und die "Profil-" Perspektive. Liegt das Ziel einer Netzwerkanalyse in der genauen Untersuchung der direkten und indirekten Beziehungen zwischen den Akteuren, so bezeichnet man diesen Standpunkt als *Beziehungsperspektive*. Bei der Betrachtung eines Netzwerkes durch die *Profilperspektive* interessieren die Ähnlichkeiten der verschiedenen Profile der einzelnen Akteure. Akteure mit ähnlichem Profil befinden sich an ähnlichen Positionen im Netzwerk. Die verschiedenen Level, auf denen soziale Netzwerke betrachtet werden können, also *Akteur, Gruppe* und *gesamtes Netzwerk*, werden in einem späteren Teil des Abschnittes ausführlich behandelt. Beschäftigen wir uns zuerst mit der Darstellung von Status in sozialen Netzwerken.
In [BKR+99] wird beschrieben, wie die Idee, ein Strukturmaß als Nebenbedingung in eine Netzwerkvisualisierung einzubauen, für Zentralität umgesetzt werden kann. Dieselbe Situation liegt hier vor, mit dem Unterschied, daß nicht Zentralität, sondern Prestige, insbesondere *Status*, als Strukturmaß dient. Ziel ist es, sowohl die *Rohdaten*, das Netzwerk, sowie das *Strukturmaß*, also Zentralität oder Prestige, gemeinsam darzustellen. Im Normalfall wird dies getrennt getan durch Verwendung einer Tabelle oder eines Balkendiagrammes zur genauen Auflistung der berechneten Werte und einer graphischen Darstellung durch beispielsweise einen Springembedder. Durch die alleinige Betrachtung der Zentralitäts- bzw. Prestigewerte ist es nicht möglich, die Struktur des Netzwerkes zu rekonstruieren. Der Gruppen-*Degree*index eines Kreises unterscheidet sich beispielsweise nicht von dem eines vollständigen Graphen. Beide Indizes sind 0, da die Länge der kürzesten Wege immer gleich ist. Daher ist es sinnvoll, sowohl die Struktur, als auch die Strukturmaße gleichzeitig in einer einzigen Zeichnung widerzugeben.
Beispiele, in denen ein Strukturmaß als Nebenbedingung in eine Zeichnung eingebaut wurde, finden sich in Kapitel 3, *Padgett's Netzwerk der Florentinischen Familien* oder in [BKR+99].
Bei der Verwendung der Zentralität als Strukturmaß bei einer Netzwerkvisualisierung, werden die Knoten, entsprechend ihren Werten, auf konzentrischen Kreisen angeordnet. Der Abstand der Knoten vom Mittelpunkt, d.h. der Radius der konzentrischen Kreise wird anhand der Zentralitätswerte bestimmt. Im Fall der Benutzung von Status als Strukturmaß geschieht dies auf ähnliche Weise. Die einzelnen Akteure werden in Schichten eingeteilt, die durch ihre Statuswerte bestimmt sind. Schichten werden aber nicht durch konzentrische Kreise kenntlich gemacht, sondern durch horizontal angeordnete Level, soge-

nannte *Layer*. Der oder die Knoten mit dem höchsten Statuswert werden in der hier benutzten Fassung auf dem obersten Level positioniert, die Knoten mit dem niedrigsten Index auf dem untersten.
Die graphische Darstellung kann in einigen Punkten variieren. Daher kann das graphische Design in drei Teile zerlegt werden, die
Layoutspezifikation (Positionierung), die
Renderingspezifikation(Aussehen, retinale Eigenschaften) und die
Animation (Zeitpunkt).

Eine *Positionierung* versieht die Knoten eines Netzwerkes nicht mit festen Koordinaten. Die Koordinaten müssen jedoch bestimmte Bedingungen erfüllen, die unabhängig von ihren exakten Positionen sind.
Der erste Schritt zur Positionierung teilt die Knoten in Schichten ein, so daß die Höhe jedes Knotens durch eine y-Koordinate gegeben ist. Die exakte Position ergibt sich erst am Ende des Algorithmus, wenn die Knoten in Schichten eingeteilt und auf diesen geordnet sind. Dann werden die genauen x-Koordinaten unter Berücksichtigung der Eigenschaften, die für die Kanten erfüllt sein müssen, berechnet. In der dieser Arbeit zugrunde liegenden Implementation ist ein Ziel, daß Kanten zwischen zwei Knoten maximal zwei Knicke besitzen und zwischen diesen Knicken vertikal verlaufen. Da bei der Bestimmung der Layer *Clusterung* verwendet werden kann, ist es nicht immer möglich, die maximale Grenze von zwei Knicken einzuhalten. Der Grund liegt in der Positionierung zusätzlicher *Dummyknoten*, was in Kapitel 5, Abschnitt 5.1 genau beschrieben wird. Da die Layer nicht äquidistant gesetzt werden, ist der Abstand unter Umständen sehr gering. Unterschreitet die Distanz einen festgelegten Schwellenwert, werden die Layer zu einem einzigen zusammengefaßt.
Das Aussehen, also die *retinalen Eigenschaften* der Objekte, ist bestimmt durch Größe, Form, Orientierung, Füllung, Farbe, Helligkeit und Transparenz. Die verschiedenen Darstellungsmöglichkeiten spiegeln die semantischen Eigenschaften eines Objektes wieder, haben also nichts mit Status- oder Zentralitätswerten zu tun. Eine Ausnahme dazu bildet die Kantenfärbung. Durch sie wird gekennzeichnet, ob eine Kante von oben nach unten (rot), von unten nach oben (schwarz) oder in beide Richtungen (grün) verläuft. Eine andere Möglichkeit, die Orientierung kenntlich zu machen, ist die Verwendung von Pfeilen am Ende einer Kante. Da Kanten sehr kurz sein können, wurde in der Implementation auf diese Methode verzichtet, um die Lesbarkeit des Layouts zu erhöhen. Um die Layer von Kanten des Netzwerkes unterscheiden zu können, wird für sie eine blaue Färbung gewählt. Die Farbe der Knoten ist nicht festgelegt und kann daher von Netzwerk zu Netzwerk variieren. Typischerweise werden Farben und Formen benutzt, um verschiedene Arten von Akteuren zu trennen, beispielsweise *männliche* und *weibliche* Akteure, *staat-*

liche und *ehrenamtliche* Behörden, einzelne Abteilungen u.ä. . Da sich die Formen der Akteure innerhalb eines Netzwerkes unterscheiden können (Kreise, Ellipsen, Rechtecke, Rauten ...), kann die Größe einer Institution durch den Flächeninhalt der jeweiligen geometrischen Objekte repräsentiert werden. Eine weitere Möglichkeit besteht darin, durch das Verhältnis zwischen Höhe und Breite das Verhältnis von Knoteneingangsgrad und Knotenausgangsgrad wiederzugeben. Wird ein Knoten v durch eine Ellipse repräsentiert, kann die Bestimmung der Radien in x- bzw. y-Richtung durch die Gleichungen

$$\frac{r_y(v)}{r_x(v)} = \frac{deg_{in}(v)}{deg_{out}(v)}$$

$$r_x(v) + r_y(v) = 2\pi \cdot deg_{in}(v) \cdot deg_{out}(v)$$

erfolgen. Ist die vertikale Länge größer als die horizontale, laufen mehr Kanten in den Knoten ein als aus, was für einen hohen Status spricht. Auch wenn diese Methode nicht den genauen Status widerspiegeln kann, so erhält man doch einen ersten Eindruck von der ungefähren Lage des Knotens in der Hierarchie.
Das graphisches Design spezifiziert also gewünschte Eigenschaften der Koordinaten für die zu zeichnenden Elemente wie Knoten, Kanten und Label. Dabei müssen bestimmte Designbedingungen eingehalten werden, soweit dies möglich ist. Es ist nicht immer möglich, alle Bedingungen, die gestellt sind, zu erfüllen. So läßt sich ein Layout, bei dem alle Kanten dieselbe Länge haben sollen, nicht immer zeichnen (siehe ersten Abschnitt des Kapitels). Diese Art von Bedingung nennt man *Designkriterium*. Es kann nicht immer vollständig erfüllt werden, sollte jedoch optimiert werden. Je besser die Designkriterien erfüllt werden, desto besser ist das Layout. Da zumeist mehr als eines zu erfüllen sind, ist es verständlich, daß die Kriterien im Konflikt miteinander stehen können. Somit muß ein gutes Layout nicht jedes einzelne Kriterium optimal erfüllen, sondern die Gesamtheit der Kriterien. Dabei können verschiedene Kriterien auch unterschiedlich gewichtet sein.

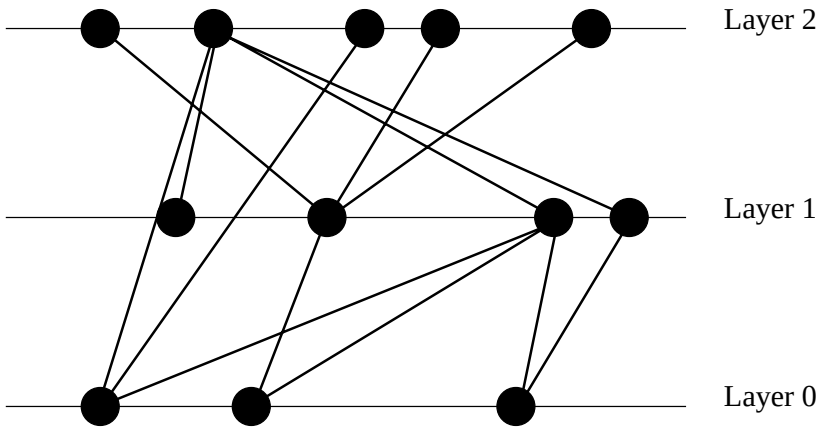
Wir erhalten als Resultat, daß *Design* in erster Linie auf eine effektive Umsetzung der *Substanz* Wert legt und erst dann auf die Schönheit der Bilder. Klar ist jedoch, daß auch das Verständnis von der Art der Darstellung abhängt. Wie schnell die *Substanz* des Netzwerkes in einer Visualisierung erkannt werden kann, liegt an der Effektivität des *Design*. Mathematisch gesehen steht eine solche Spezifikation in engem Zusammenhang mit einem Optimierungsproblem und muß daher die Bedingungen und die Zielfunktion für ein solches Problem näher bestimmen. Ein perfekter Algorithmus würde ein graphisches Layout erstellen, das alle Bedingungen erfüllt und für die Zielfunktion ein

Optimum erreicht.Wie wir bereits wissen, ist die Mathematische Grundlage eines Netzwerkes ein Graph $G = (V, E)$, der aus einer endlichen Menge V von Knoten und einer Menge E von Kanten besteht, welche die Knoten untereinander verbinden. Dabei stellen die Knoten Akteure dar und die Kanten die Beziehungen unter diesen. Semantische Attribute können durch Form und Farbe oder durch angebrachte Label angezeigt werden. Die Aufgabe des Designs ist es, die Art der Netzwerkelemente zu bestimmen, d.h. wie werden Akteure und ihre Beziehungen dargestellt. Üblicherweise werden Akteure durch Punkte, Beziehungen durch Linien repräsentiert [Ber83]. Diagramme, die dieses Design erfüllen, sind beispielsweise das *Soziogramm* und das *Scattergram* (bei dem die Kanten typischerweise weggelassen werden). Eine weitere Art der Darstellung wäre die Verwendung von Namen oder Logos für die Akteure, die Beziehungen könnten verschiedene Dicken haben, um anzuzeigen, wie intensiv eine Beziehung ist. Doch alle diese Möglichkeiten ändern nur etwas am Aussehen des Layouts, nicht die Positionen der Elemente oder die Präsentation selbst. Es werden nur *retinale Eigenschaften* verändert. Für weitere Informationen siehe z.B. [Tuf83]. Im Unterschied zu den *semantischen* Eigenschaften eines Netzwerkes (siehe 4.2), lassen sich die *syntaktischen* Eigenschaften am besten durch die Positionierung der Akteure und Beziehungen darstellen. Dies zeigt sich besonders gut bei Zentralität oder Prestige. Ein *Broker* oder eine *Brücke* sind durch ihren Standort leicht zu erkennen, ebenso wie Akteure, die von ihrer Wichtigkeit auf derselben Stufe stehen.

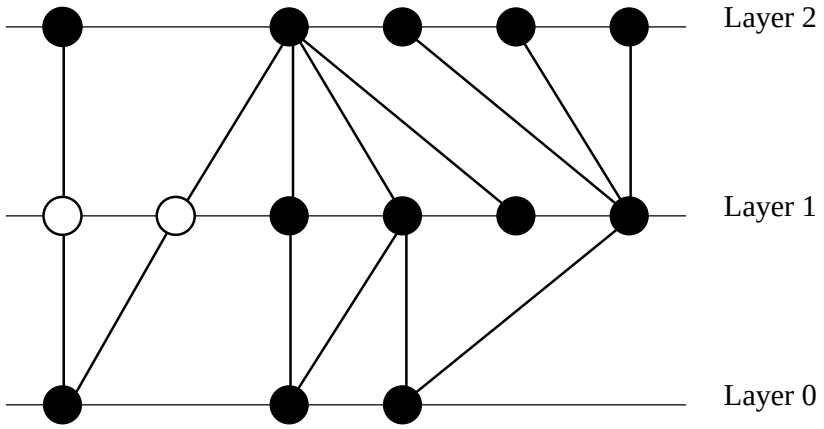
4.4 Algorithmus

Aufgabe des *Algorithmus* ist es, eine *Designspezifikation* für einen gegebene *Netzwerksubstanz* umzusetzen. Die durch das Design gestellten Forderungen, *Designkriterien*, müssen durch die Anwendung eines Algorithmus verwirklicht werden. In vielen Fällen ist es nicht möglich, eine perfekte Visualisierung zu schaffen, die den Anforderungen des Designs genügt. Und oft führt die - augenscheinlich - bestmögliche Realisierung zu großen Problemen, die durch Abschwächung eines Kriteriums erheblich besser gelöst werden können. Ein Beispiel findet sich in der hier beschriebenen Umsetzung des *Algorithmus von Sugiyama*. Eine Voraussetzung zur Layerzuweisung ist ursprünglich der Fakt, daß der Graph kreisfrei ist. Dazu müssen unter Umständen einige Kanten in ihrer Richtung umgekehrt werden. Es treten dabei mehrere Probleme auf, die durch die umgesetzte Art der Layerzuweisung umgangen werden.
Ein Punkt ist die Berechnung des Status. Da die einlaufenden Kanten für die Bestimmung des Ranges benutzt werden, verändert sich das Ergebnis durch

das Umkehren einer Kante. Außerdem muß die Menge der in der Richtung vertauschten Kanten nicht eindeutig sein, was wiederum zu verschiedenen Ergebnissen führen kann. Dazu kommt noch, daß das Problem, eine solche Menge zu finden, $\mathcal{NP}$-schwer ist [Kar72].
Gegebenenfalls ist es sogar unmöglich, ein Kriterium zu erfüllen (siehe Abschnitt *Ziel der Visualisierung*). Algorithmen können (oder müssen) also ein spezielles Design *definieren*, aber nicht unbedingt jedes Design *implementieren*. Um ein Layout zu erhalten, das zu einem gegebenen Netzwerk und einem graphischen Design alle geforderten Bedingungen erfüllt, benötigt man eine algorithmische Umsetzung. Diese berechnet ein *Layout*, das *Aussehen* und unter Umständen eine *Animation*. Da sich die Anforderungen in der *Renderingspezifikation* normalerweise nicht überschneiden, gibt es im Bezug auf die Umsetzung dieser Bedingungen für den Algorithmus keine Schwierigkeiten. Ein Algorithmus muß aber jede der gestellten Bedingungen erfüllen und die Kriterien so gut es möglich ist optimieren. Es liegt also ein Optimierungsproblem vor, das nicht immer in annehmbarer Zeit gelöst werden kann. In diesem Fall werden Näherungslösungen gesucht.
Ein Beispiel ist die *Kreuzungsminimierung*. Ein optimales Layout eines Netzwerkes, das den Status von Akteuren beschreibt, besitzt keine Kreuzung. Dies ist in vielen Fällen unmöglich. Auch die Verwendung eines Algorithmus, der eine optimale Lösung zur Kreuzungsminimierung berechnet, ist nicht immer ratsam, insbesondere bei großen Netzwerken. Daher werden Heuristiken benutzt, die zwar nicht immer die optimale Lösung finden, aber einen Kompromiß aus Güte und Laufzeit eingehen. In Figur 4.2 ist zu sehen, wie eine Kreuzungsminimierung gut und weniger gut gelöst werden kann. Die nicht gefüllten Knoten sind Dummyknoten, die durch die Konvertierung in ein vollständig geschichtetes Layout entstanden sind.
Die algorithmische Umsetzung des in dieser Arbeit zu behandelnden Problems ist aufgrund des Umfangs Thema des nächsten Kapitel und wird dort ausführlich behandelt.



(a)



(b)

Abbildung 4.2: Schlechtes (a) und gutes (b) Beispiel für Kreuzungsminimierung

Kapitel 5

Geschichtete Darstellung von Netzwerken

Wie wir bereits gesehen haben, gibt es verschiedene Möglichkeiten, Netzwerke darzustellen. Als Beispiel wären nur die Darstellung mittels eines Kreisdiagrammes oder nach der Bestimmung von Zentralitätswerten die Darstellung der Akteure auf konzentrischen Kreisen erwähnt. In diesem Abschnitt wollen wir uns mit der geschichteten Darstellung von Netzwerken beschäftigen. Da wir uns dem Problem der Darstellung auf graphentheoretischen Niveau nähern, wollen wir die für diesen Bereich üblichen Notationen verwenden. Wir sprechen im folgenden daher nicht mehr von *Akteuren, Verbindungen* und *Netzwerken*, sondern von *Knoten, Kanten* und *Graphen*.
Die Knoten werden aufgrund berechneter Prestigewerte in verschiedene Schichten, sogenannte *Layer* eingeteilt. Dabei entsteht eine Ordnung, die als *Hierarchie* bezeichnet wird. Im folgenden werden die Knoten auf ihren jeweiligen Schichten geordnet und positioniert. Diese Vorgehensweise wird mit Hilfe des *Algorithmus von Sugiyama* durchgeführt [DETT99], [STT81]. Er berechnet für einen gegebenen, azyklischen und gerichteten Graphen $G = (V, E)$ ein hierarchisches Layout. Der Algorithmus läßt sich in drei Schritte unterteilen, die unabhängig voneinander auf verschiedene Arten behandelt werden können. Diese Schritte sind

1. Zuweisung eines Layers für jeden Knoten im Graph
2. Minimierung der Kantenüberschneidungen
3. Zuweisung der Koordinaten für die Knoten und die Layer

Die drei Teile können völlig unabhängig voneinander behandelt werden. Die einzelnen Schritte liefern jeweils ein Ergebnis, das im nächsten Schritt verwendet wird, müssen aber nicht gezwungenermaßen in dieser Form bearbeitet

werden. So ist beispielsweise die Minimierung der Kantenüberschneidungen ein Problem rein kombinatorischer Natur. Die Anzahl der Kreuzungen hängt nur von der Reihenfolge der Knoten, nicht aber von der genauen Position ab. Auch bei der Zuweisung der Layer spielen die endgültigen Koordinaten keine Rolle, die Knoten werden nur in Schichten eingeteilt und zu Beginn beliebig geordnet. Die für das Problem bestmögliche Ordnung und genaue Positionierung wird in den Schritten 2 und 3 behandelt.
Um den Algorithmus von Sugiyama in seiner ursprünglichen Form anzuwenden, muß vor Beginn der Prozedur der Graph *azyklisch* gemacht werden, falls dies noch nicht der Fall ist. Dies geschieht mit Hilfe des *Cycle Removal*. Die Idee hierbei ist, so wenig Kanten wie möglich *umzudrehen*, um den Graphen *azyklisch*, d.h. kreisfrei zu machen. Diese Vorgehensweise erlaubt es uns, alle Kanten in eine Richtung zu zeichnen, was für die normalerweise benutzte Layerzuweisung von großer Wichtigkeit ist. Am Ende des Algorithmus werden die zuvor umgedrehten Kanten wieder mit ihrer originalen Orientierung versehen. Da die Layerzuweisung in der Implementation auch bei azyklischen Graphen funktioniert, wird auf die genauere Beschreibung des *Cycle Removal* verzichtet. Ein weiterer Grund ist die Tatsache, daß durch das Umdrehen einiger Kanten die Statuswerte verfälscht werden können. Zur Berechnung der Prestigewerte werden die einlaufenden Kanten betrachtet. Verändert man bei einer Kante die Richtung, ändern sich bei zwei Kanten der Eingangs- bzw. Ausgangsgrad der Knoten, was zu einer Veränderung der Werte führt (siehe auch Kapitel 4, Abschnitt 4.4).

5.1 Layerzuweisung

Das Ziel des ersten Schrittes des Algortihmus von Sugiyama ist es, jeden Knoten einer Schicht der Hierarchie zuzuordnen. Aufgrund der Vielzahl der Möglichkeiten, die zur Layerzuweisung existieren, soll hier nur ein kleiner Einblick gegeben werden.
Der erste Teil des Kapitels beschäftigt sich mit der Layerzuweisung allgemein. Definitionen und Bedingungen für das Bilden einer Hierarchie werden angesprochen. Danach werden verschiedene Methoden der Layerzuweisung erklärt, gefolgt von einer Beschreibung der implementierten Version, die unter anderem *Status* und *Clusterung* benutzt.

5.1.1 Allgemeines

Um eine sinnvolle Zuweisung der Layer gewährleisten zu können, müssen Bedingungen, die an ein gutes Layout gestellt werden, erwähnt und erläutert werden. Dazu gehört beispielsweise nicht die genaue *Positionierung* der Knoten oder Kanten, da dies Aufgabe der Koordinatenzuweisung ist, sondern etwa die Bestimmung der *Höhe* und *Breite* des Layouts.
Gegeben sei ein azyklischer, gerichteter Graph $G = (V, E)$.

Definition 4 *Mit einer* Schichtung *oder* Hierarchie von G *bezeichnet man die Aufteilung der Knotenmenge V in Untermengen $L_1, \ldots, L_n$, so daß gilt*

$$u \in L_i, v \in L_j, (u, v) \in E \Longrightarrow i > j$$

Dabei entspricht die Höhe H *des geschichteten Graphen der Anzahl* h *der Layer, die* Breite W *der Anzahl der Knoten im größten Layer, d.h. $max_{1 \leq i \leq n}|L_i|$. Die* Höhe *einer Kante $(u, v) \in E$ mit $u \in L_i, v \in L_j$ ist die Differenz der beiden Layer, also $i - j$. Man nennt den Graphen* einfach geschichtet *genau dann, wenn keine Kante eine Höhe größer* 1 *besitzt.*

Es ist möglich, daß die Hierarchie schon vorgegeben ist, beispielsweise bei einem Stammbaum, einer Laufbahn oder ähnlichem. Dann wird dieser erste Schritt des Algorithmus übersprungen. Bei der Umsetzung des Algorithmus in der Implementation wird die Hierarchie durch die Berechnung der Statuswerte der Knoten bestimmt. Breite und Höhe können somit nicht mehr in der hier beschriebenen Form beeinflußt werden. Das Ziel der Layerzuweisung ist es, einen azyklischen Digraphen in einen geschichteten Digraphen zu transformieren. Dabei sollten drei Bedingungen beachtet werden.

1. Das Layout sollte kompakt sein, d.h. die Höhe und die Breite sollten gering gehalten werden. Da die Layerdistanz in allgemeinen konstant ist, ergibt sich als untere Schranke für die Höhe die maximale Anzahl der Kanten eines Weges von einem Knoten mit *Knoteneingangsgrad* 0 (auch: *Source*) zu einem Knoten mit *Knotenausgangsgrad* 0 (auch: *Sink*).

2. Das Layout muß *einfach geschichtet* sein. Dazu werden *Dummyknoten* eingefügt. Jede Kante (u, v) mit einer Höhe größer 1 wird durch einen Weg $(u = v_1, \ldots, v_k = v)$ ersetzt, indem Dummyknoten $v_2, \ldots, v_{k-1}$ eingefügt werden. Dieser Vorgang wird in Abbildung 5.1 verdeutlicht. Das Hinzufügen von Dummyknoten ist notwendig, um die Kreuzungsminimierung optimal durchführen zu können, da es sehr schwierig ist, Kreuzungen für Kanten mit einer Höhe größer 1 zu behandeln.

3. Die Anzahl der Dummyknoten soll niedrig gehalten werden. Falls die Anzahl der Kanten im Bereich $O(n)$ liegt mit einer Höhe im Bereich $O(n)$, so ist die Anzahl der Dummies quadratisch.
   Daß die Bedingung, die Anzahl der Dummyknoten niedrig zu halten, sinnvoll ist, wird nach der Betrachtung der folgenden Argumente deutlich.

   - Die Laufzeit des Algorithmus hängt von der Anzahl der Knoten ab, wobei sowohl *echte* Knoten als auch *Dummyknoten* gerechnet werden.
   - Knicke in den Kanten machen das Layout unleserlicher. Da Knicke nur bei Dummyknoten auftreten, sollte auf eine erhöhte Anzahl von Dummies verzichtet werden. Obwohl das Problem durch die Setzung der horizontalen Koordinaten gemindert werden kann, ist es dennoch besser, von Anfang an weniger Dummyknoten zu setzen.
   - Die Anzahl der Dummies ist eine Anzeige für die Länge einer Kante, d.h. je mehr Dummies vorhanden sind, desto länger ist die Kante. Je länger eine Kante ist, desto unübersichtlicher wird das Layout, da es einfacher ist, einer kurzen Kante mit dem Auge zu folgen als einer langen.

In [Fri96] wird eine obere Schranke für die Anzahl der Dummyknoten angegeben. *Frick* faßt seine Ergebnisse in einem **Satz** zusammen.

Satz 1 *Eine k-Schicht-Hierarchie eines azyklischen Digraphen $D = (V, A)$ mit $n = |V|, m = |E|$ und Länge der längsten Kante $k - 1$ kann höchstens*

$$\begin{aligned} d(h,n,m) \leq & \sum_{i=1}^{l-1}(n-k+i)\cdot(k-i-1) + \\ & min\{n-k+l, m-\sum_{j=1}^{l-1}(n-k+j)\}\cdot(k-l-1) \end{aligned} \tag{5.1}$$

Dummyknoten besitzen, wobei $h \leq n, m \geq n-1$ und

$$l = min\{k-2, \llcorner\sqrt{(n-k)^2+n+2m-3k+9/4} - n + k + 1/2 \lrcorner\}.$$

Der Beweis, ebenso wie Beispiele, finden sich in [Fri96].

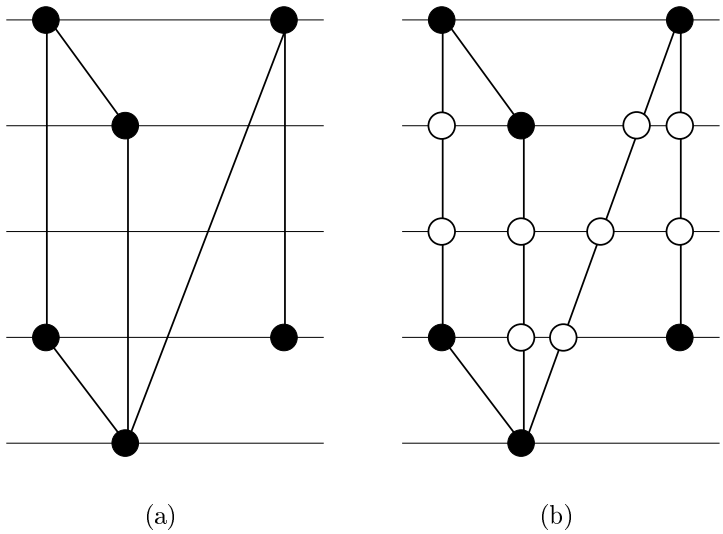


Abbildung 5.1: Hierarchy vor (a) und nach (b) dem Einfügen von Dummyknoten

5.1.2 Methoden der Layerzuweisung

Es werden nun einige Möglichkeiten der Schichtung beschrieben und an Beispielen verdeutlicht. Um eine möglichst eindeutige Beschreibung zu erhalten, wird an einigen Stellen der Pseudo-Code angegeben.

Longest-Path-Layering Das Longest Path Layering ist in zwei Schritte unterteilt.
1. Schritt:
Plaziere alle Knoten mit Knotenausgangsgrad 0 (*Sink*) in Layer L_1
2. Schritt:
Plaziere dann alle erreichbaren Knoten v in Layer L_{p+1}, falls der längste Weg von v zu einem *Sink* die Länge p hat.

Der Algorithmus besitzt zwei positive Eigenschaften.

- Da der Graph keine Zykel besitzt, kann die Schichtung in Linearzeit berechnet werden.
- Die Anzahl der Layer ist minimal, was zur Folge hat, daß die Höhe insgesamt minimal ist.

Ein Nachteil ist, daß das Layout zu breit sein kann. Folglich muß ein Layout mit Breitenminimierung gesucht werden. Dabei ist zu beachten, daß das Problem, ein Layout mit minimaler Höhe *und* minimaler Breite zu finden, $\mathcal{NP}$-vollständig ist. Zur Verdeutlichung betrachte folgende Situation: Jeder Knoten repräsentiere eine Aufgabe in einem Multiprozessorsystem, die eine Zeiteinheit zur Bearbeitung benötige. Diese Aufgaben werden auch als *Unit-Time-Task* bezeichnet.
Ferner existiere genau dann eine Kante im darstellenden Graphen, wenn Aufgabe u vor v erledigt wird und die Aufgaben voneinander abhängig sind. Eine graphische Darstellung wird durch Abbildung 5.2 gegeben. Das Ziel ist, jeder

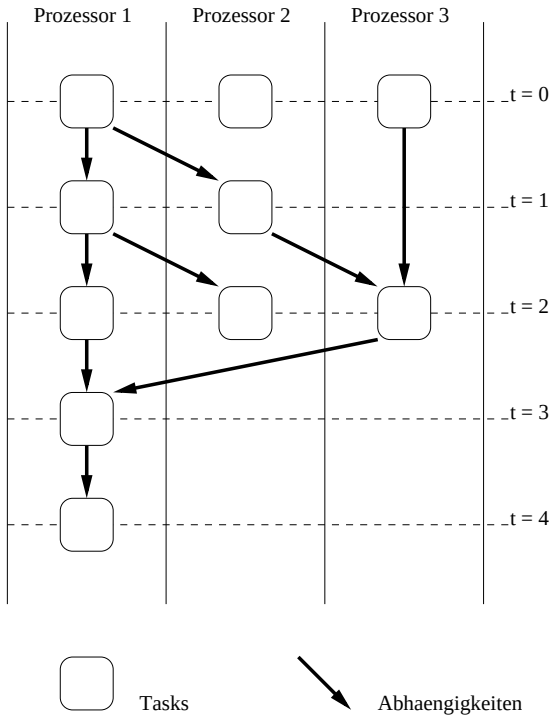


Abbildung 5.2: Multi-Prozessor-Scheduling

Aufgabe einen der W Prozessoren zuzuweisen, so daß alle Aufgaben in der Zeit H bearbeitet sind. Dieses Problem wird mit
Precedence-Constrained-Multiprocessor-Scheduling-Problem bezeichnet.
Wenn wir dieses Problem auf das unsere übertragen, erkennen wir, daß W der Breite und H der Höhe des Layouts entspricht. Da das *Multiprocessor-Scheduling-Problem* $\mathcal{NP}$-vollständig ist [GJ91], ergibt sich die $\mathcal{NP}$-Vollständigkeit für unser Problem. Daher werden wir Heuristiken betrachten, die das Problem möglichst effektiv lösen.

Das *Coffman-Graham-Layering* ist eine Heuristik zur Lösung unseres Problems, die aus dem Bereich des Multiprozessor-Scheduling kommt.
Gegeben sei ein gerichteter, Graph $G = (V, E)$ und eine Konstante $W \in \mathbb{N}$. Gesucht ist ein Layout von G mit einer Breite von höchstens W. Ziel der Heuristik ist es, die Höhe des Layouts möglichst klein zu halten [LS77].
Bisher sind wir bei der Bestimmung der Breite eines Layouts nur von der Anzahl der echten Knoten ausgegangen. Die Dummyknoten wurden nicht berücksichtigt. Diese Vorgehensweise kann beibehalten werden, solange die echten Knoten sichtbar größer sind als die Dummies. Meistens ist dies der Fall, da die echten Knoten der Darstellung von Personen, Aufgaben usw. dienen und damit sichtbar sein müssen, die Dummyknoten jedoch nur die Knickpunkte für die Kanten darstellen. Ist dies nicht der Fall, muß diese Tatsache beim *Coffman-Graham-Algorithmus* berücksichtigt werden.
Der Algorithmus besteht wiederum aus zwei Phasen. In der ersten Phase werden die Knoten geordnet, in der zweiten Phase werden sie einem Layer zugewiesen. Um die Knoten zu sortieren, muß zuerst eine Ordnung festgelegt werden, die eine Sortierung möglich macht. Sei S eine endliche Menge mit Elementen aus $\mathbb{N}$. Sei ferner das Maximum einer solchen Menge S definiert als

$$max(S) := \{s \in S : s > t \; \forall \; t \in S \setminus \{s\}\}$$

Dann gilt für zwei Mengen S, T, daß $S < T$, falls

- $S = \emptyset$ und $T \neq \emptyset$ oder
- $S \neq \emptyset, T \neq \emptyset$ und $max(S) < max(T)$ oder
- $S \neq \emptyset, T \neq \emptyset$, $max(S) = max(T)$ und
  $S - max(S) < T - max(T)$

Sei ferner $\pi(v)$ eine Funktion, die jedem Knoten $v \in G$ ein Label $\in \mathbb{N}$ zuordne.

Algorithmus: Coffman-Graham-Layering

Gegeben: Ein gerichteter Graph $G = (V, E)$, eine positive, ganze Zahl W
Gesucht: Ein Layout von G mit einer Breite von höchstens W

Zu Beginn seien die Label der Knoten nicht inititalisiert
Für $i = 1$ bis $|V|$ führe aus
    Wähle einen Knoten v, der noch kein Label besitzt, so daß die Menge $\{\pi(v) : (u, v) \in E\}$ bezüglich der oben definierten Ordnung minimal ist

Setze $\pi(v) = i$
Sei $k = 1; L_1 = \emptyset; U = \emptyset$
Solange $U \neq V$ führe aus
Wähle ein $u \in V \setminus U$, so daß jeder Knoten der Menge
$\{v : (u, v) \in E\}$ in U liegt und $\pi(u)$ maximal wird
Falls $|L_k| < W$ und für jede Kante $(u, w), w \in L_1 \cup L_2 \cup \ldots \cup L_{k-1}$
Füge u in L_k ein
Sonst setze $k = k + 1, L_k = u$
Füge u in U ein

Man erhält somit eine einfache, lexikographische Ordnung, bei welcher der größte Wert die wichtigste Rolle spielt. Folglich ist $\{1, 2, 3\} < \{3, 4\}$ und $\{1, 2, 4\} < \{1, 3, 4\}$. In der ersten Phase des Algorithmus werden den Knoten entsprechend der Ordnung Label zugewiesen. Nachdem die ersten $k-1$ Label zugewiesen wurden, weise das Label k einem Knoten v zu, der die folgenden Bedingungen erfüllt

1. v besitzt noch kein Label
2. alle Knoten u mit $(u, v) \in E$ besitzen schon Label
3. Bei den Knoten die (1) und (2) erfüllen ist die Menge der direkten Vorgänger von v, d.h. die Menge $\{\pi(u) : (u, v) \in E\}$, minimiert nach der obigen Definition von Ordnung.

Die Abbildung 5.3 zeigt die berechneten Label. Aufgrund der eben beschriebenen Ordnung wird Knoten a vor b und Knoten c vor d mit Labeln versehen, da $\{4\} < \{3, 4\}$ und ebenso $\{4, 5\} < \{3, 8\}$.
In der zweiten Phase des Algorithmus werden die Knoten den Layern zugewiesen. Dabei kommt die in der Voraussetzung geforderte Bedingung ins Spiel, daß kein Layer mehr als W Knoten enthalten darf. Beginne mit dem untersten Layer L_1 und arbeite sich dann Layer für Layer bis zum obersten Layer L_h vor. Dabei wird ein Knoten v einem Layer L_k zugewiesen, falls

- v noch keinem Layer zugewiesen wurde und
- gilt: $\forall u$ mit $(v, u) \in E$ ist $u \in L_1, \ldots, L_{k-1}$

Falls mehrere Knoten mit diesen Eigenschaften existieren, wähle den mit dem größten Label. Existieren keine derartige Knoten oder ist Layer L_k voll, d.h. $|L_k| = W$, so gehe weiter zu Layer L_{k+1}. Die Höhe h des Layouts unterliegt bei dieser Vorgehensweise der Schranke

$$h \leq (2 - \frac{2}{W}) \cdot h_{min},$$

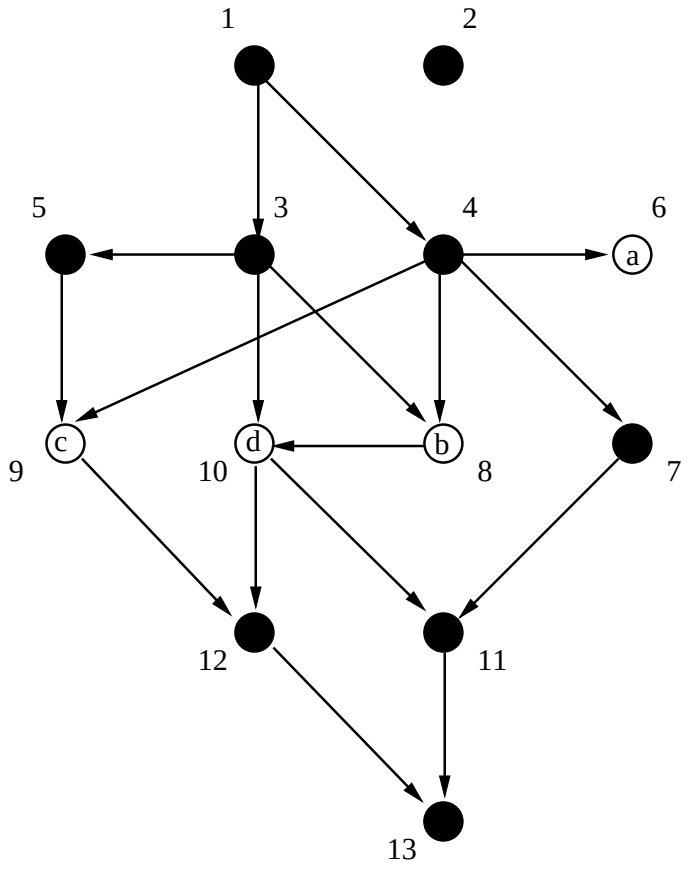


Abbildung 5.3: Durch das *Coffman-Graham-Layering* berechnete Label

wobei W für die Breite, h_{min} für die minimale Höhe bei Breite W steht [LS77].

Die eingefügten Dummyknoten, die in dieser Art der Berechnung für die Bestimmung der Breite nicht berücksichtigt wurden, verursachen trotz ihres minimalen Platzbedarfes Kosten. Die Gründe wurden am Anfang des Abschnittes beschrieben. Daher ist es sinnvoll, die Anzahl der Dummies zu minimieren. Es ist möglich, in polynomialer Zeit ein Layout zu berechnen, das eine minimale Anzahl von Dummyknoten besitzt [GKNV93], und zwar mit Hilfe der *Integer-Programmierung*.
Wir gehen dabei davon aus, daß jeder Knoten u eines azyklischen, gerichteten Graphen $G = (V, E)$ eine y-Koordinate $y(u)$ besitzt, die folgende Eigenschaften erfüllt:

1. $y(u) \in \mathbb{N} \; \forall \; u$

2. $y(u) \geq 1 \; \forall \; u$

3. $y(u) - y(v) \geq 1 \; \forall \; (u, v) \in E$

Durch diese Funktion y erhalten wir ein Layout mit Layern $L_m = \{u \in V : y(u) = m\}$ Definiere eine weitere Funktion f mit

$$f := \sum_{(u,v) \in V} (y(u) - y(v) - 1)$$

als Summe der vertikalen Höhe von Kanten in diesem Layout minus der Anzahl von Kanten. Das bedeutet, daß f die totale vertikale Ausdehnung aller Kanten mißt, d.h. die Anzahl der Dummyknoten eines durch die Funktion y gegebenen Layouts berechnet. Damit haben wir das Layerzuweisungsproblem reduziert auf die Auswahl von y-Koordinaten, um f zu minimieren unter Berücksichtigung der Bedingungen (1),(2) und (3). Das Problem besitzt nach [GKNV93] eine ganzzahlige Lösung.
Nehmen wir an, daß der Wert $y(u)$ einer optimalen Lösung nicht in $\mathbb{N}$ liegt. Das bedeutet, daß u zwischen den Linien $\ulcorner y(u) \urcorner$ und $\llcorner y(u) \lrcorner$ liegt mit $y_u := \llcorner y(u) \lrcorner$. Nach Bedingung (3) gilt aber, daß kein Nachbar von u zwischen y_u und y_{u+1} liegt. Falls der Knotenausgangsgrad von u größer als der Knoteneingangsgrad ist, so wird die der Wert von f nicht größer, falls wir u ein Stück nach unten verschieben. Solange keine Kanten (u, v) mit $y(v) > y_u - 1$ existieren, kann u bis y_u nach unten geschoben werden, ohne die Bedingungen (2) und (3) zu verletzen (siehe Figur 5.4). Also wird f nicht größer, falls

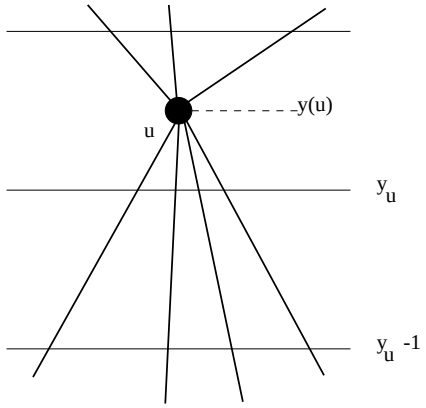


Abbildung 5.4: Verschieben eines Knotens nach unten

$y(u) \notin \mathbb{N}$ abgerundet wird. Achtet man auf die Reihenfolge, in der die Knoten bewegt werden, so können alle Knoten mit einem höheren Knotenaus-

als Knoteneingangsgrad nach unten auf eine y-Koordinate aus $\mathbb{N}$ verschoben werden. Analog kann man verfahren bei niedrigerem Knotenaus- als Knoteneingangsgrad. Folglich kann jedes $y \notin \mathbb{N}$ zu einem $y \in \mathbb{N}$ gemacht werden, ohne die Bedingungen (1) - (3) zu verletzen oder den Wert von f zu erhöhen. Verschiedene Möglichkeiten zur Lösung dieses Problems werden in [GKNV93] behandelt.

Oft eignen sich Multiprozessor-Scheduling-Methoden besser, um Kompaktheit für ein Layout zu erreichen. Da die Knoten meist horizontal beschriftet sind, sind die Zwischenräume in dieser Richtung größer als in der vertikalen. Es ist also sinnvoll, zu versuchen, die Breite klein zu halten, was bei Multiprozessor-Scheduling-Methoden bevorzugt getan wird. Will man eine geringe Breite des Layouts erreichen, bietet sich das *Longest-Path-Layering* an. In der Praxis dient die Minimierung der Anzahl von Dummyknoten nicht nur der kürzeren Kantenlänge und damit der besseren Lesbarkeit, sondern auch besserer Kompaktheit. Doch auch hier ergibt sich wieder das Problem der extremen Laufzeit. Die Kombination von Höhen- und Dummyminimierung führt zu einem $\mathcal{NP}$-vollständigen Problem. Genauere Beschreibungen finden sich in [Lin92].

Schichtung für Allgemeine Graphen Bis zu diesem Zeitpunkt sind wir davon ausgegangen, daß der Graph azyklisch ist, um die Existenz einer Schichtung zu garantieren. Es existieren jedoch auch Methoden, mit denen allgemeine Graphen ein hierarchisches Layout erhalten können. Die Vorgehensweise in diesem Fall ist relativ einfach. Die Kanten, die nach oben zeigen, werden einfach umgedreht. Einige Möglichkeiten zur Bestimmung einer Hierarchie seien hier kurz beschrieben.

1. Die *Tiefensuche* oder die *Breitensuche* ermöglichen die Schaffung einer Hierarchie in Linearzeit.

2. Durch das Berechnen eines *minimal aufspannenden Baumes* für ungerichtete Graphen ergibt sich ebenfalls ein Hierarchisches Layout. Diese Methode ist sehr nützlich, falls die Kanten gewichtet sind. Besitzen sie ein Gewicht $w(e)$, so beschreiben wir ihre Kosten mit $\frac{1}{w(e)}$ Damit erhalten Kanten mit hoher Priorität kleine Höhen.

3. Eine weitere Methode besteht in der Verwendung eines *Spring-Embedders*. Dabei werden die Koordinaten unter Verwendung von Anziehungs- und Abstoßungskräften zwischen den Knoten berechnet. Der *Spring-Embedder* produziert eine Schichtung mit beinahe gleich langen Kanten.

Die hier noch nicht erwähnte, für diese Arbeit aber wichtigste Form der Bestimmung einer Hierarchie ist die Layerzuweisung der Knoten durch Berechnung ihrer Statuswerte. An dieser Stelle kommt die *Clusterung* ins Spiel, die unter bestimmten Bedingungen Layer verschmelzen kann. Im kommenden Abschnitt werden wir genauer auf die Berechnung der Statuswerte und die Clusterung eingehen.

Statusberechnung und Clusterung Der Aufbau einer Hierarchie kann durch die Berechnung der Statusindizes für die einzelnen Akteure erfolgen. Dadurch erhalten die Layer keine ganzzahligen Werte, sondern, nach Normierung, Werte aus dem Intervall [0,1]. Bei Layern, die zu nahe aneinander liegen, wird eine *Clusterung* angewandt, die mehrere Layer miteinander verschmelzen läßt. Wir werden uns im folgenden zuerst mit der Berechnung der Statuswerte, dann mit der *Clusterung* beschäftigen. Beide Teile werden jeweils durch ein Beispiel ergänzt.

Hierarchieaufbau mittels Statusberechnung Durch die Berechnung des Statusmaßes, kann eine Hierarchie für den Graphen erstellt werden. Jeder Knoten erhält einen Prestigewert, der das Layer bestimmt. Anders als in den bereits beschriebenen Möglichkeiten der Layerzuweisung werden die Layer nicht in gleichen Abständen angeordnet, sondern erhalten einen normierten Wert, der im Intervall [0,1] liegt. Dem Layer mit dem höchsten Prestigewert wird das Label 1, dem Layer mit dem niedrigsten Wert das Label 0 zugeordnet. Da die Werte im Normalfall nicht automatisch innerhalb dieser Grenzen liegen, muß eine Normierung der berechneten Werte vorgenommen werden. Eine genauere Beschreibung erfolgt später. Nach dieser Normierung werden Layer zusammengefaßt, die einen Abstand zueinander besitzen, der einen individuell bestimmbaren Schwellenwert unterschreiten. Dadurch entstehen *Cluster*, der Vorgang an sich wird mit *Clusterung* bezeichnet. Dieser komplette Vorgang der *Layerzuweisung* und *Clusterung* soll nun im Detail beschrieben werden.

Layerzuweisung durch Berechnung der Prestigewerte Die Berechnung der Prestigewerte folgt der im Abschnitt 3.3.8 beschriebenen Vorgehensweise. Zur Lösung der Gleichung

$$[(1/a)I - A']p = x,$$

welche uns die Statuswerte für die einzelnen Knoten liefert, benötigen wir noch einige Voraussetzungen. Mit x bezeichnen wir den Vektor der Knoteneingangsgrade der nichtnormierten Matrix A, was einem Vektor aus den

Spaltensummen entspricht. Der Vektor b bezeichnet den Lösungsvektor, der die jeweiligen Prestigewerte für die Knoten enthält. A' beschreibt die Transponierte Adjazenzmatrix und I die Einheitsmatrix der Größe $n \times n$, wenn n die Anzahl der Knoten ist. Der noch fehlende Faktor a bzw. $1/a$ kann nicht berechnet und muß daher geschätzt werden. Der Wert für diesen Faktor sollte zwischen dem größten Eigenwert und dem doppelten des größten Eigenwertes liegen. Katz verwendet zur Bestimmung des endgültigen Statusvektor noch einen Faktor m, der sich näherungsweise durch

$$(n-1)!a^{n-1}e^{1/a}$$

berechnen läßt. Dies wird in der Implementation nicht getan, da die Werte normiert werden und durch das Intervall [0,1] beschränkt sind. Zum besseren Verständnis sei nun ein Beispiel angeführt.

Beispiel zur Berechnung der Prestigewerte Gegeben sei eine 6×6-Matrix mit folgenden Einträgen

	A	B	C	D	E	F
A	0	0	0	0	0	1
B	0	0	1	0	0	1
C	0	1	0	1	0	1
D	1	0	0	0	1	0
E	0	0	0	1	0	1
F	1	0	0	1	0	0
Spaltensumme	2	1	1	3	1	4

Der daraus resultierende Graph ist in Figur 5.5 dargestellt.
Der konventionelle Statusvektor würde sich berechnen durch Division der Spaltensummen durch $n-1=5$, d.h. der Vektor der Prestigewerte hätte das Aussehen

$$p = (0.4, 0.2, 0.2, 0.6, 0.2, 0.8).$$

Da bei der Berechnung von Status nicht nur direkte, sondern insbesondere indirekte Verbindungen zählen, erhalten wir ein anderes Ergebnis. Dabei variiert neben den Statuswerten auch die Reihenfolge der Akteure im Bezug auf ihre Wichtigkeit.
Nach der Berechnung des größten Eigenwertes $\lambda_1 = 1.68377$ wählen wir den Faktor a als $1/2$, was bedeutet $1/a = 2$. Dann erhalten wir als zu lösendes Gleichungssystem

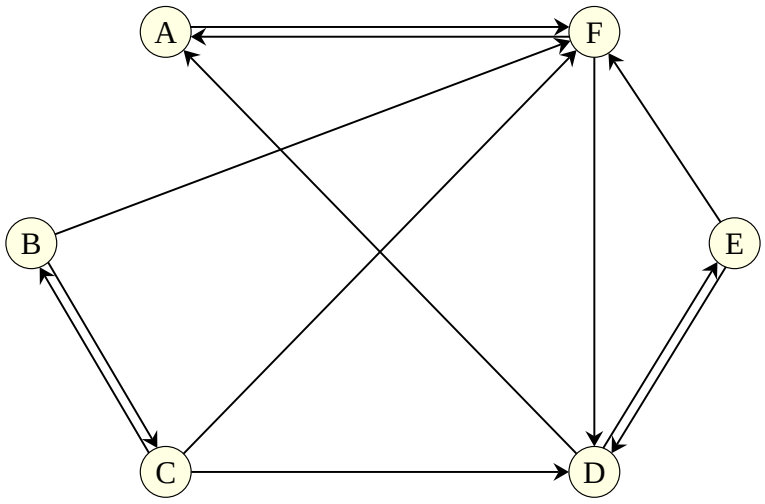


Abbildung 5.5: Ausgangsgraph zum Prestigelayout

$$
\begin{array}{cccccccc}
2p_1 & & & - p_4 & & - p_6 & = 2 \\
 & 2p_2 & - p_3 & & & & = 1 \\
 & -p_2 & + 2p_3 & & & & = 1 \\
 & & -p_3 & + 2p_4 & - p_5 & - p_6 & = 3 \\
 & & & - p_4 & + 2p_5 & & = 1 \\
- p_1 & - p_2 & - p_3 & & - p_5 & + 2p_6 & = 4
\end{array}
$$

Daraus resultieren die Werte $p_1 = 13.0, p_2 = 1.0, p_3 = 1.0, p_4 = 11.4, p_5 = 6.2, p_6 = 12.6$. Um eine Beschränkung auf das Intervall $[0, 1]$ zu erhalten, müssen zuerst die niedrigsten Werte auf 0 gesetzt werden. Dazu subtrahieren wir von allen Elementen des Vektors den Wert 1.0, was zum Zwischenergebnis $p = (12.0, 0.0, 0.0, 10.4, 5.2, 11.6)$ führt. Eine Normierung erhalten wir mittels Division durch 12 woraus sich der neue Statusvektor

$$p = (1.0, 0.0, 0.0, 0.8667, 0.4333, 0.9667)$$

ergibt.
Der aus dem Ursprungsgraph 5.5 entstandene Graph ist in Figur 5.6 zu sehen. Auffällig ist die Tatsache, daß sich die hier berechneten Statuswerte von denen des konventionellen Statusvektors unterscheiden, und zwar nicht nur in ihrem exakten Wert. Der wichtigste Akteur ist jetzt nicht mehr Akteur F, wie das bei der Berechnung des konventionellen Statusvektor der Fall war, sondern Akteur A.

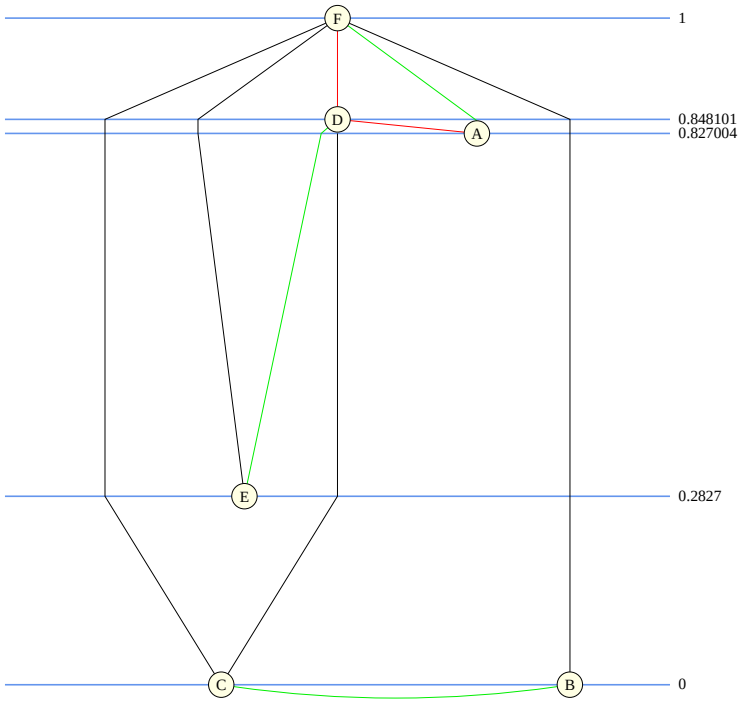


Abbildung 5.6: Graph nach berechnetem Prestigelayout

Clusterung Da die Layer nicht wie normalerweise üblich äquidistant gesetzt werden, sondern in ihren y-Koordinaten einem Vielfachen der Statuswerte entsprechen, kann es vorkommen, daß Kanten sehr flach zwischen zwei Layern verlaufen, so daß sie kaum von den horizontal laufenden Layern zu unterscheiden sind. Kreuzungen, die durch solche Kanten produziert werden, sind damit schlecht zu erkennen.
Aus diesen Gründen werden Layer, die sehr nahe beieinander liegen, d.h. deren Distanz einen selbst festzulegenden Schwellenwert unterschreiten, zu einem Layer zusammengefaßt. Dies kann auf verschiedene Arten geschehen. Die Vorgehensweise ist von der Idee her dieselbe. Man betrachtet die Hierarchie *Bottom-Up*, d.h. von unten nach oben. Begonnen wird mit Layer 0, bei Layer h endet die Betrachtung, wobei der *Anzahl der Layer - 1* entspricht. Nun wird überprüft, wie groß die vertikale Distanz zwischen Layer i und Layer $i+1$ ist. Wird hierbei die Schranke unterschritten, so werden die beiden Layer zusammengefasst, alle Knoten auf Layer $i+1$ werden dem Layer i zugewiesen. Alle weiter oben angesiedelten Layer, d.h. die Layer $i+1, \ldots, h$, mit h :=Anzahl der Layer, werden um eine Stufe herabgesetzt. Das bedeutet, daß aus Layer $i+2$ das Layer $i+1$ wird, aus Layer $i+3$ Layer $i+2$ usw. .

Fall 1:
Bei der *Clusterung* der Layer wird jeweils die y-Koordinate des als nächstes zu betrachtenden Layers mit der y-Koordinate des unteren der beiden bereits verschmolzenen Layer verglichen. Ist die Distanz immer noch unterhalb des festgelegten Wertes, wird das zu betrachtende Layer ebenfalls mit den beiden bereits vereinigten Layern zusammengefaßt, ansonsten nicht.
Fall 2:
Es werden jeweils die originalen y-Koordinaten der aufeinanderfolgenden Layern miteinander verglichen. Auch wenn zwei Layer geclustert werden, bleiben die ursprünglich berechneten originalen y-Koordinaten, die aus den Prestigewerten resultieren, gespeichert. Liegen diese Werte wiederum unterhalb einer Schranke, so wird das obere der beiden betrachteten Layer mit dem unteren zusammengefasst. Ist dieses auch schon verschmolzen, so bilden jetzt alle drei Layer eine Einheit. Diese Methode erscheint sinnvoll, da im endgültigen Layout die Layer wieder *ent-clustert* werden, d.h. sie werden jeweils auf der Höhe der ursprünglich berechneten y-Koordinaten gezeichnet. In der Implementation wurde daher diese Methode eingesetzt. Der Unterschied der beiden Clusterungen ist am besten an einem Beispiel zu sehen. Sei folgende Situation gegeben:
Die Hierarchie besitze fünf Layer mit den Werten:
Layer 4 : 1.0
Layer 3 : 0.7
Layer 2 : 0.5
Layer 1 : 0.3
Layer 0 : 0.0
Der Schwellenwert sei 0.25 . Im Fall 1 wird zuerst der Wert von Layer 0 mit dem von Layer 1 verglichen. Die Differenz ist 0.3, also größer als der Schwellenwert, die Layer werden nicht zusammengefaßt. Der Vergleich der Werte von Layer 1 und 2 ergibt einen Unterschied von 0.2, also unterhalb der Schranke von 0.25, daher werden die Layer 1 und 2 geclustert. Der Einfachheit halber werden in diesem Beispiel die Namen der Layer auch nach der Clusterung beibehalten und nicht wie oben beschrieben geändert. Die Differenz der Werte von Layer 3 und Layer 1 (da Layer 1 das untere der beiden zusammengefaßten Layer ist) liegt bei 0.4, also oberhalb des Schwellenwertes, was bei Layer 3 und Layer 4 mit einem Abstand von 0.3 auch der Fall ist. Es wird also keine weitere Clusterung durchgeführt.
Im Fall 2 ist der Anfang derselbe, Layer 0 und 1 werden nicht geclustert, Layer 1 und 2 werden wie in Fall 1 zusammengefasst. Jetzt werden jedoch nicht die Werte von Layer 1 und Layer 3 verglichen, sondern die von Layer 2 und Layer 3. Die Differenz ergibt 0.2, also einen Wert kleiner als 0.25 . Damit wird Layer 3 mit Layer 1 und 2 verschmolzen, also zu einem einzigen

Cluster, bestehend aus drei Layern, zusammengefaßt. Bei der Betrachtung von Layer 3 und 4 ergibt sich keine neue Clusterung, analog zu *Fall 1*. Das Ergebnis im ersten Fall ist also eine Hierarchie mit *vier* Layern, von denen *eines* geclustert ist und die Knoten von *zwei* Layern enthält. Im zweiten Fall entsteht eine Hierarchie mit *drei* Layern, *eines* davon geclustert und mit dem Inhalt von *drei* Layern.

5.2 Kreuzungsminimierung

Zum Beginn der Kreuzungsminimierung im Algorithmus von Sugiyama muß der erste Schritt, die *Layerzsuweisung*, bereits abgeschlossen sein, d.h. die Eingabe zum zweiten Schritt ist ein *einfach geschichteter Digraph*. Noch einmal zur Erinnerung, *einfach geschichtet* bedeutet, daß keine Kante über mehr als ein Layer geht. Dies geschieht durch Einfügen der Dummyknoten bei längeren Kanten. Beträgt die Höhe einer Kante mehr als 1, so wird auf jedem Layer, das von der Kante gekreuzt wird, ein Dummyknoten eingefügt. Bei der Kreuzungsminimierung sollte man beachten, daß die Anzahl der Kreuzungen nicht von der genauen Position der Knoten abhängt, sondern von ihrer Reihenfolge. Das Ziel dieses zweiten Schrittes bei Sugiyama ist somit, eine angemessene Reihenfolge der Knoten zu bestimmen und keine Zuweisung der horizontalen Koordinaten, was im dritten und letzten Schritt geschieht. Auch wenn das Problem damit vereinfacht klingt (und es auch ist), ist es immer noch $\mathcal{NP}$ -vollständig, sogar für den Fall von nur 2 Layern [GJ83] oder falls nur ein einziger Nicht-Dummyknoten in jedem Layer ist [MNF90]. Zur Lösung des Problems wurden deshalb Heuristiken entwickelt, von denen ein Auswahl im folgenden dargestellt wird.

Layer-by-layer-sweep Die Vorgehensweise besteht aus zwei Schritten.

1. Wähle eine Knotenordnung für Layer 1
2. Wähle eine Knotenordnung für Layer i, $i = 2, 3, \ldots, h$, wobei die Ordnung auf L_{i-1} fix bleibt und L_i so geordnet wird, daß die Kreuzungen minimal (bzw. reduziert) werden zwischen Kanten mit Endknoten in den Layern L_i und L_{i-1}.

Das Problem beschränkt sich damit auf die Reduktion von Kreuzungen zwischen Kanten mit Endknoten in benachbarten Layern, daher auch der Name *2-Layer-Crossing-Minimization* oder *One-Sided-Crossing-Minimization*. Für das *Layer-by-Layer-Sweep* gibt es verschiedene Varianten. Es besteht die

Möglichkeit, jeweils das obere und untere Layer zu fixieren und die Reihenfolge im Layer dazwischen zu ändern, d.h. fixiere L_{i-1} und L_{i+1}, suche eine Reihenfolge für L_i zur Minimierung der Kreuzungen.
Weitere Varianten unterscheiden sich in der Wahl des Weges, d.h. ob das *Layer-by-Layer-Sweep* von oben nach unten oder von unten nach oben durchgeführt wird. Auch eine Kombination von beidem ist möglich (und sinnvoll), d.h. führe in beide Richtungen so viele *Sweeps* aus, bis sich die Anzahl der Kreuzungen nicht mehr vermindert.
Wir werden uns nun genauer mit dem *2-Layer-Crossing-Problem* beschäftigen, da dies die gemeinsame Basis für die angesprochenen Möglichkeiten ist. Doch vorab müssen noch einige Begriffe definiert werden.

Definition 5 *Ein 2-Schicht Digraph ist ein bipartiter Digraph* $G = (L_1, L_2, E)$ *mit* $L_1 \cap L_2 = \emptyset$. L_1, L_2 *sind Mengen von Knoten und* $E \subseteq L_1 \times L_2$, *d.h. für* $\{u, v\} \in E$ *gilt :* $u \in L_1$ *und* $v \in L_2$ *oder* $u \in L_2$ *und* $v \in L_1$.

Eine Ordnung auf dem Layer L_i sei gegeben durch eine Permutation π_i. $\pi(u)$ beschreibe die Position von u in dieser Reihenfolge.

Definition 6 *Sei* $cross(G, \pi_1, \pi_2)$ *definiert als die Anzahl der Kreuzungen eines Layouts, das beschrieben wird durch* π_1 *und* π_2. *Ferner sei* $opt(G, \pi_1)$ *definiert als* $min_{\pi_2} cross(G, \pi_1, \pi_2)$, *d.h. als Minimale Anzahl von Kreuzungen im Bezug auf die Knoten von* L_1 *geordnet durch* π_1.

Damit läßt sich das *2-Layer-Crossing-Problem* wie folgt beschreiben.
Gegeben: ein 2-Schicht Digraph $G = (L_1, L_2, E)$, geordnet durch π_1 für L_1.
Gesucht: eine Reihenfolge π_2 für L_2, so daß

$$cross(G, \pi_1, \pi_2) = opt(G, \pi_1)$$

Es ist zu beobachten, daß die Anzahl der Kreuzungen zwischen Kanten die inzident zu Knoten u sind und Kanten, die inzident zu Knoten v sind, nur abhängig ist von der relativen Position von u und v, nicht von den Positionen anderen Knoten. In Figur 5.7 ist ein Beispiel für einen 2-Schicht-Graph zu sehen. Tabelle 5.1 zeigt die zugehörige Kreuzungsmatrix.
In diesem Zusammenhang tritt der Begriff der Kreuzungszahl auf, der für viele Heuristiken ein wichtige Rolle spielt.

Definition 7 *Die* Kreuzungszahl c_{uv} *ist definiert als die Anzahl der Kreuzungen zwischen zu* u *inzidenten Kanten und zu* v *inzidenten Kanten, wobei* $\pi_2(u) < \pi_2(v)$.

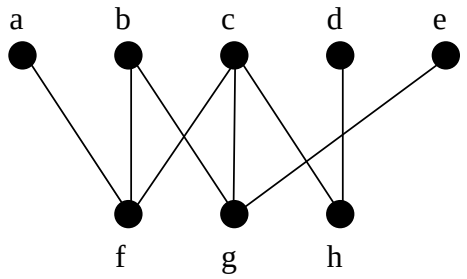


Abbildung 5.7: Ein 2-Schicht Graph mit 8 Knoten

Tabelle 5.1: Kreuzungsmatrix C zum Graph in Abbildung 5.7

C	f	g	h
f	0	1	0
g	6	0	2
h	5	3	0

Anders ausgedrückt: Für $u \neq v \in L_2$ ist c_{uv} die Anzahl der Paare $(u, w), (v, z)$ von Kanten mit $\pi_1(z) < \pi_1(w)$. Sei ferner $c_{uu} = 0$ für alle $u \in L_2$. Mittels der Anzahl der Kreuzungen lassen sich $\text{cross}(G, \pi_1, \pi_2)$ und eine untere Schranke für $\text{opt}(G, \pi_1)$ berechnen.

Lemma 1 *Falls $G = (L_1, L_2, E)$ ein 2-Schicht Digraph ist und π_1, π_2 zwei Ordnungen von L_1 bzw. L_2 sind, dann gilt:*
$cross(G, \pi_1, \pi_2) = \sum_{\pi_2(u) < \pi_2(v)} c_{uv}$ (1.1)
$opt(G, \pi_1) \geq \sum_{u,v} min(c_{u,v}, cv, u)$ (1.2),
wobei die Summe über alle ungeordneten Paare u, v von Knoten des oberen Layers sind.

Beweis: (1.1) ist klar nach Definition.
(1.2) Man beachte, daß bei jeder Reihenfolge von L_2 (inklusive der optimalen) gilt:

$$\pi_2(u) < \pi_2(v)$$

oder

$$\pi_2(v) < \pi_2(u)$$

woraus sofort die Behauptung folgt. □

Das Ziel der *2-Layer-Crossing-Minimization* ist es, bei gegebener Ordnung π_1 für L_1 eine Reihenfolge π_2 für L_2 zu finden, so daß die Anzahl der Kreuzungen minimiert wird. Das heißt, die Knoten auf L_2 müssen sortiert werden, um eine Lösung zu erhalten.
Einige Sortiermethoden werden im folgenden beschrieben.

Sortiermethoden
Adjacent-Exchange (auch: greedy-switching-heuristic)
Die erste Methode ähnelt dem Algorithmus *Bubblesort*. Hier werden adjazente Knotenpaare vertauscht unter Berücksichtigung der Kreuzungszahl [GKNV93].

Algorithmus: Adjacent-Exchange

Gegeben: Eine Ordnung π_1 für L_1
Gesucht: Eine Ordnung π_2 für L_2

Wähle eine Reihenfolge für L_2
Wiederhole folgenden Schritt
    Betrachte die Knoten von L_2 von links nach rechts,
    Vertausche u und v, falls $c_{uv} > c_{v,u}$
Bis die Anzahl der Kreuzungen nicht mehr kleiner wird.

Die Laufzeit dieser Methode liegt bei $\mathcal{O}(|L_2|^2)$ aufgrund der inneren Schleife. Der worst-case für Adjacent-Exchange ist in Abb. 5.8 dargestellt.

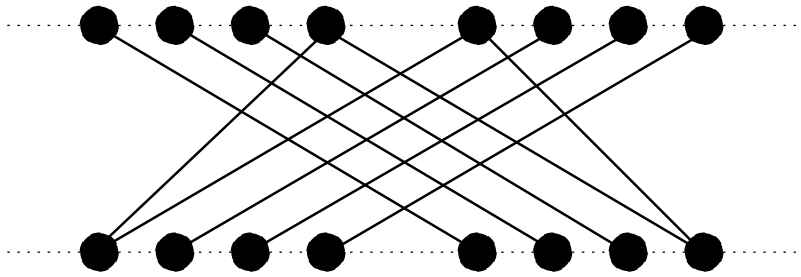


Abbildung 5.8: Worst-case für Adjacent-Exchange

Split
Die zweite Methode erinnert stark an den Algorithmus *Quicksort*. Auch hier wird ein Pivot-Element zur rekursiven Sortierung benutzt [EK86].

Algorithmus: Split

Gegeben: Eine Ordnung π_1 für L_1
Gesucht: Eine Ordnung π_2 für L_2

Falls L_2 nicht leer ist, führe aus
    Wähle ein Pivotelement $p \in L_2$
    $V_{left} = V_{right} = \emptyset$
    Für jedes $p \neq u \in L_2$ führe aus
        falls $c_{u,p} \leq c_{p,u}$
            plaziere u in V_{left}
            ansonsten in V_{right}
    Wende den Algorithmus rekursiv auf den durch V_{left} und V_{right} induzierten Digraphen an und gib die Konkatenation der Ausgaben dieser zwei Anwendungen aus.

Die Laufzeit dieses Algorithmus liegt wie bei Quicksort im worst-case bei $\mathcal{O}(|L_2|^2)$, in der Praxis bei $\mathcal{O}(|L_2| \cdot log|L_2|)$.
Die nun folgenden Methoden werden aufgrund ihrer positiven Eigenschaften, wie z.B. relativ gute Ergebnisse bei geringer Laufzeit, und ihrer Verwendung in der Implementation ausführlicher beschrieben.

Barycenter-Methode Die meistverbreitetsten Methoden zur Lösung des *2-Layer-Crossing-Problems* sind Variationen der Barycenter-Methode. Hier wird die x-Koordinate jedes Knotens $u \in L_2$ gewählt als *Barycenter* (Durchschnitt) der x-Koordinaten seiner Nachbarn, d.h.

$$\pi_2(u) := bary(u) \forall u \in L_2,$$

wobei

$$bary(u) := \frac{1}{deg(u)} \sum_{v \in N_u} \pi_1(v).$$

$deg(u)$ bezeichnet hier den Grad des Knotens u und $N_u = \{v : \{u, v\} \in E\}$ die Menge der Nachbarn von u.

Barycenter-Methode

Sortiere L_2 nach $bary(u)$
Falls $bary(u) = bary(v)$ trenne u und v beliebig durch einen kleinen Wert

Die Laufzeit beträgt hier nur $\mathcal{O}(|L_2|)$, d.h. das Problem kann in Linearzeit gelöst werden, da der Aufwand, $bary(u)$ zu finden, proportional zu $deg(u)$ ist. Wenn wir die Sortierung mit einbeziehen, erhalten wir eine Laufzeit von $|L_2|log|L_2|$. Die Anzahl der von der Barycenter-Methode ausgegebenen Kreuzungen wird mit $bary(G, \pi_1)$ bezeichnet.
Interessant ist die Tatsache, daß eine Lösung mit 0 Kreuzungen von der *Barycenter-Methode* gefunden wird, falls eine existiert.
Dasselbe gilt für die *Median-Methode*, die anstelle des Durchschnittes (bei der *Barycenter-Methode*) den Median zur Berechnung der Position benutzt.

Median - Methode Bei der Median-Methode wird die x-Koordinate für einen Knoten $u \in L_2$ berechnet durch den Median der x-Koordinaten der adjazenten Knoten. Seien dazu $v_1, \ldots, v_j$ die Nachbarn von u und $\pi_1(v_1) < \ldots < \pi_1(v_j)$. Dann gilt

$$med(u) = x(v_{\lfloor j/2 \rfloor}).$$

Falls $N_u = \emptyset$, dann wird der Median auf 0 gesetzt, also $med(u) := 0$. Um eine Ordnung für die Knoten im Layer L_2 herzustellen, geht man folgendermaßen vor:

Median-Methode

Sortiere L_2 nach $med(u)$
Falls $med(u) = med(v)$ führe aus
    Falls ein Knoten geraden Knotengrad, einer ungeraden besitzt,
        setze den Knoten mit ungeradem Knotengrad links vom Knoten mit geradem Knotengrad
    Falls die Knoten dieselbe Parität besitzen, wähle eine beliebige Reihenfolge.

Wir werden später noch einmal auf diese Art der Ordnung zurückkommen, wenn in Satz 1 eine Performance-Garantie gegeben wird.
Für jedes $u \in L_2$ kann $med(u)$ in einer Zeit proportional zum Knotengrad $deg(u)$ berechnet werden, analog zur *Barycenter-Methode*, womit die Laufzeit wieder in $|L_2|log|L_2|$ liegt [AHU83], [CLR90]. Das Layout kann somit in Linearzeit berechnet werden. Die Anzahl der Kreuzungen, die bei dieser Methode entstehen, werden mit $med(G, \pi_1)$ bezeichnet.

Auch wenn es möglich ist, ein Layout ohne Kreuzungen zu berechnen, falls es existiert, so gewährt keine der beiden Methoden eine optimale Lösung. Als Gegenbeispiele dienen die Abbildungen 5.9 und 5.10. Die Barycentermethode plaziert den Knoten u links von v und erzeugt $k-1$ Kreuzungen. Die optimale Lösung würde ein Layout mit nur einer Kreuzung produzieren.
Durch die Medianmethode werden $2k(k+1)+k^2$ Kreuzungen erzeugt, die optimale Lösung besitzt nur $(k+1)^2$ Kantenüberschneidungen.

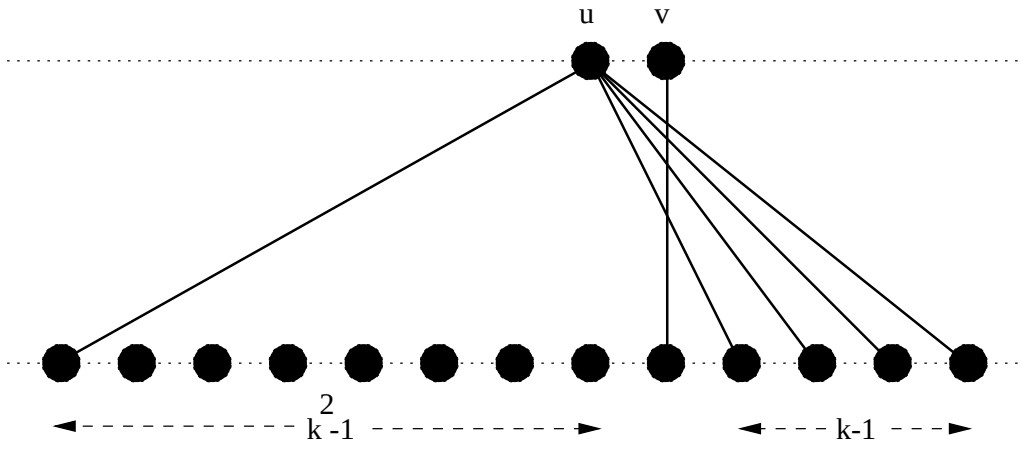


Abbildung 5.9: Worst case für die Barycentermethode

Dadurch ergibt sich das folgende Lemma.

Lemma 2 *1. Für jedes natürliche n existiert ein 2-Schicht Digraph $G = (L_1, L_2, E)$ mit $|L_1| = n, |L_2| = 2$ und eine Reihenfolge π_1 von L_1, für den gilt*

$$\frac{bary(G, \pi_1)}{opt(G, \pi_1)} \text{ liegt in } \Omega(\sqrt{n}).$$

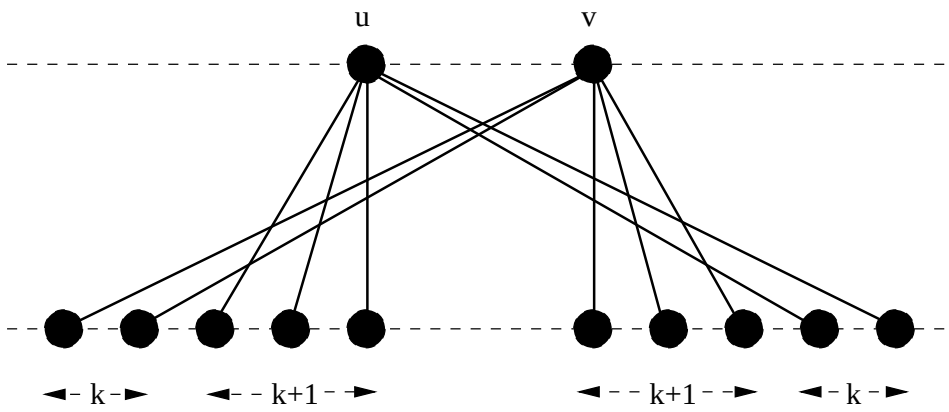


Abbildung 5.10: Worst case für die Medianmethode

2. *Für jedes natürliche n existiert ein 2-Schicht Digraph* $G' = (L'_1, L'_2, E')$ *mit* $|L'_1| = n, |L'_2| = 2$ *und eine Reihenfolge* π_1 *von* L'_1*, für den gilt*

$$\frac{med(G', \pi_1)}{opt(G', \pi_1)} \geq 3 - O(1/n).$$

Beweis: Als Ausgangssituation bei der Barycentermethode betrachte Figur 5.9, den Worst-Case. Seien $w_1, w_2, \ldots, w_n$ die Knoten von L_1, von links nach rechts geordnet mit einem Abstand von jeweils 1. Sei ferner n gegeben durch $n = k^2 + k - 1$. Sei v verbunden mit w_{k^2} und u mit w_1 und $w_{k^2+1}, w_{k^2+2}, \ldots, w_{k^2+k-1}$. Dann gilt

$$bary(u) = k^2 - k/2 - 1/2 < k^2 = bary(v)$$

Somit wird Knoten u links von Knoten v plaziert, wodurch $k-1$ Kreuzungen entstehen. Falls v links von u ist, so entsteht nur eine Kreuzung. Da k in $O(\sqrt{n})$, folgt die Ungleichung. Für die Medianmethode wird ein ähnliches Argument angewandt, indem Figur 5.10 als Worst-Case-Ausgangssituation betrachtet wird. Hier wird Knoten u links von Knoten v plaziert, was $2k(k+1) + k^2$ Kreuzungen verursacht. Das optimale Layout verursacht nur $(k+1)^2$ Kreuzungen. □

Der nun folgende Satz bietet eine Performance-Garantie für die Median-Methode. Im Beweis wird die zuvor beschriebene Ordnung benutzt.

Satz 2 *Für alle 2-Schicht Digraphen* $G = (L_1, L_2, E)$ *und alle Knotenreihenfolgen* π_1 *von* L_1 *gilt*

$$med(G, \pi_1) \leq 3 \cdot opt(G, \pi_1).$$

Beweis: Seien u und v Knoten in L_2. Die Medianmethode plaziere u links von v. Definiere 4 Sorten von Kanten, die zu u und v inzident sind (siehe auch 5.11 und 5.12)
$\alpha = (u, w) \in E : \pi_1(w) < med(u)$
$\beta = (v, w) \in E : \pi_1(w) > med(v)$
$\gamma = (v, w) \in E : \pi_1(w) < med(v)$
$\delta = (u, w) \in E : \pi_1(w) > med(u)$

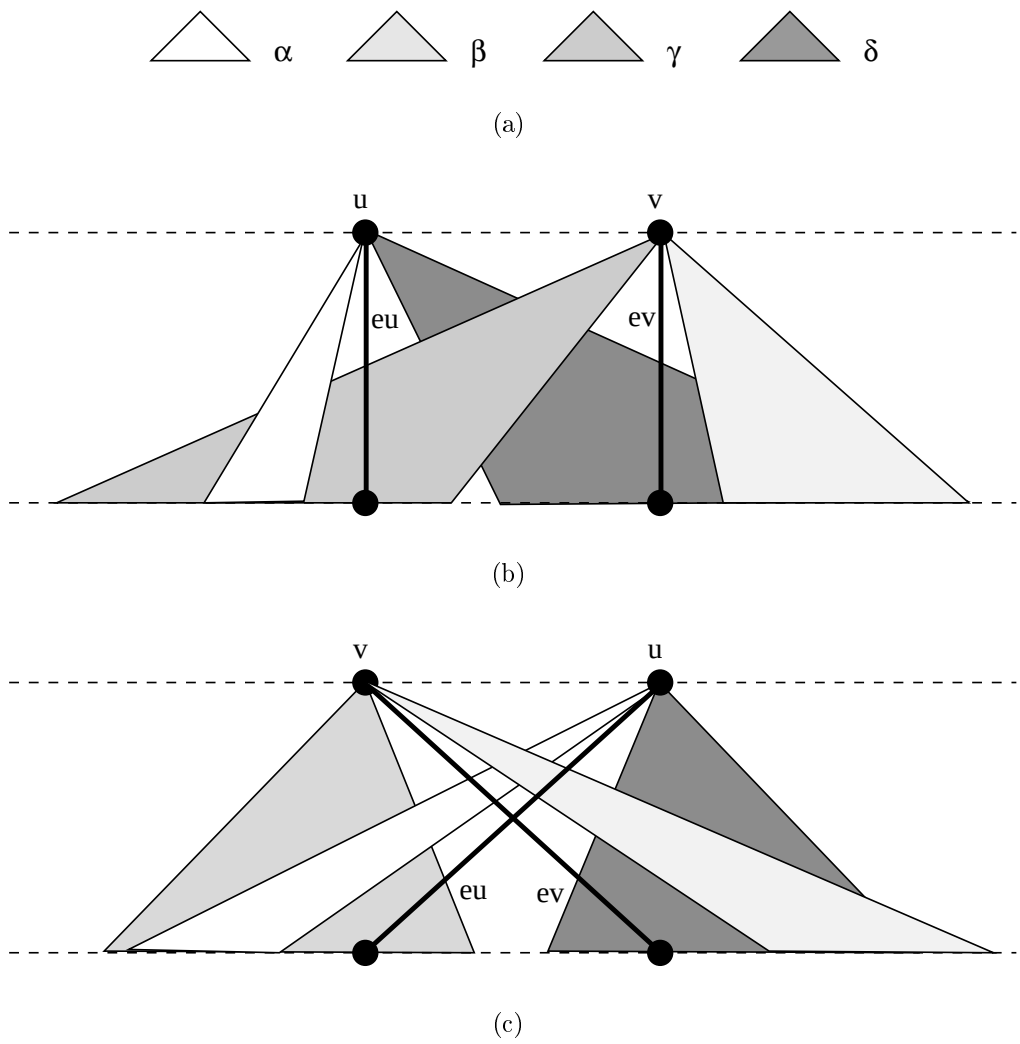


Abbildung 5.11: Schematische Darstellung der 4 Gruppen von Kanten $\alpha, \beta, \gamma, \delta$

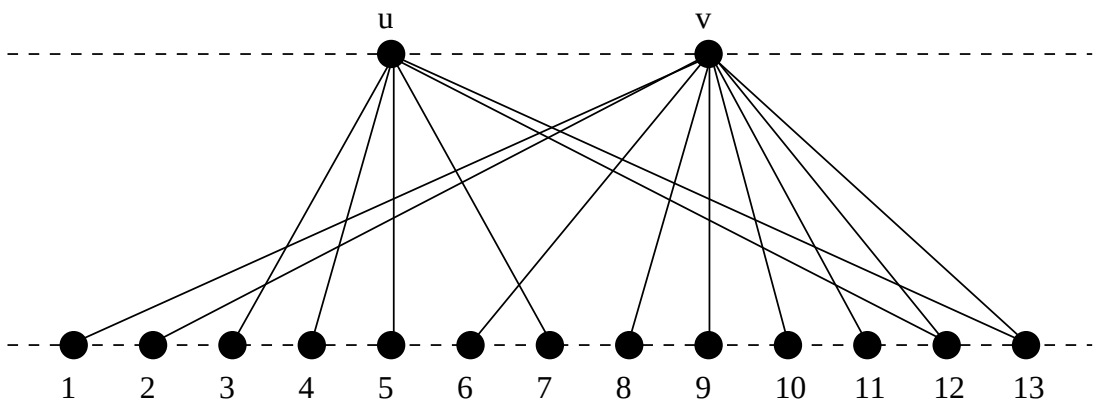


Abbildung 5.12: Beispielgraph mit den 4 Gruppen von Kanten $\alpha, \beta, \gamma, \delta$

Wichtig ist, daß u und v mit ihren jeweiligen Medianen durch Kanten verbunden sind, die wir mit e_u und e_v bezeichnen wollen. Diese zwei Kanten sind in keiner der oben angeführten Gruppen enthalten. Sei $a = |\alpha|, b = |\beta|, c = |\gamma|$ und $d = |\delta|$. Sei ferner $\epsilon = 0$ falls $med(u) = med(v)$ und $\epsilon = 1$ sonst.
Um die Behauptung des Satzes zu beweisen, werden zunächst einige Ungleichungen bewiesen, die uns Schritt für Schritt dem endgültigen Ergebnis näher bringen werden.
Behauptung:
1.)$c_{vu} \geq ab + a + b + \epsilon$
2.)$c_{uv} \leq ac + cd + bd + c + d$
3.)$c_{uv} \leq 3ab + a + 3b + 1$
4.)$c_{uv} \leq 3c_{vu}$

zu 1.) Nehmen wir an, daß v links von u plaziert wird (siehe Abbildung 5.11). Dann kreuzen alle Kanten aus α die Kante e_v, ebenso wie die aus β, die wiederum die Kante e_u treffen. Falls $med(u) = med(v)$, dann kreuzen sich e_u und e_v nicht und $\epsilon = 1$. Falls $med(u) \neq med(v)$, dann kreuzen sich e_v und e_u und $\epsilon = 1$. Daraus folgt die Behauptung 1.) .
zu 2.) Falls u links von v plaziert wird, dann können sich die Kanten in α nicht mit denen in β kreuzen und e_u und e_v ebenso nicht, woraus Behauptung 2.) folgt. zu 3.) Betrachten wir nun das Verhältnis der Größen der vier Gruppen untereinander. Falls u geraden Knotengrad hat, dann gilt $a = d$. Falls der Knotengrad ungerade ist, so ist $a+1 = d$. Analog gilt für geraden Knotengrad von v, daß $c = b$, und bei ungeradem Grad $c + 1 = b$. Ferner gilt immer, daß $d \leq a+1$ und $c \leq b$. Setzen wir diese Werte in 2.) ein, so erhalten wir 3.) . zu 4.) Um zu zeigen, daß $med(G, \pi_1) \leq 3opt(G, \pi_1)$, werden wir noch beweisen,

daß $c_{uv} \leq 3c_{vu}$. Dazu nehmen wir an, daß das Gegenteil der Fall ist und führen diese Annahme zum Widerspruch.
Sei also $c_{uv} - 3c_{vu} > 0$. Aus den Gleichungen 1.) und 3.) folgt $2a + 3\epsilon - 1 < 0$. Da a und ϵ nichtnegative Zahlen sind, müssen a und ϵ 0 sein. Das würde bedeuten, daß α leer ist und u einen Knotengrad von höchstens 2 hat. Aus $c_{uv} - 3c_{vu} > 0$ folgt, daß der Knotengrad *genau* 2 sein muß, da die Gleichung sonst nicht lösbar wäre. Folglich ist $d = 1$. Nach der Definition von ϵ und aus $a = \epsilon = 0$ folgt, $med(u) = med(v)$. Haben zwei Knoten denselben Median, so wird Knoten v links von u plaziert, falls der Knotengrad von u gerade, der von v ungerade ist. Daraus folgt für unsere Situtation, daß der Grad von v gerade ist. Folglich ist $c = b - 1$. Aus diesen Erkenntnissen und den 1.) und 2.) folgt, daß $c_{uv} = 3b - 1$ und $c_{vu} = b$. Dies führt zum Widerspruch, also gilt $c_{uv} \leq 3c_{vu}$, womit 4.) bewiesen wäre. Da dies für jedes Paar $u, v \in L_2$ gilt, ergibt sich

$$c_{uv} \leq 3min(c_{uv}, c_{vu}).$$

Wenn wir diese Ungleichung über alle Paare u, v mit $\pi_2(u) < \pi_2(v)$ aufsummieren und Lemma 1 verwenden, so erhalten wir

$$cross(G, \pi_1, \pi_2) \leq 3opt(G, \pi_1)$$

. □

Dieses Ergebnis kann unter gewissen Voraussetzungen, wie der Einschränkung des Knotengrades auf höchstens 3, noch verbessert werden zu

$$med(G, \pi_1) \leq 2opt(G, \pi_1),$$

worauf aber an dieser Stelle nicht näher eingegangen werden soll.
Im Fall eines geraden Knotengrades $deg(u)$ ergibt sich das Problem, daß der Median nicht eindeutig bestimmt ist. Daher existieren verschiedene Möglichkeiten, $med(u)$ festzusetzen.

Varianten zur Median-Methode *Average Median:*
Falls der Knotengrad von u $deg(u)$ gerade ist, so wähle das arithmetische Mittel der beiden Mediane.
Semi Median:
Falls der Knotengrad von u $deg(u)$ gerade ist, setze $s_{med}(u) = bary(u)$.

Für beide Varianten gilt, daß bei ungeradem Knotengrad $deg(u)$ der Median $med(u)$ zurückgegeben wird.
Tests belegen, daß diese Heuristiken die Ergebnisse des Originalalgorithmus verbessern.

Eine weitere Verbesserung der Average Median Methode ergibt sich durch die Einführung folgender Bedingungen [GKNV93]:

- Falls der Knotengrad $deg(u)$ ungerade ist, so setze $w_{med}(u) = med(u)$ (w_{med} = weighted median = gewichteter Median)
- Falls der Knotengrad $deg(u) = 2$, dann setze wmed als das arithmetische Mittel der Positionen der zwei Nachbarn.
- Falls der Knotengrad $deg(u) > 2$ und gerade ist, so setze

$$w_{med} = \frac{\pi_1(v_{j/2})right + \pi_1(v_{j/2+1})left}{left + right}$$

  wobei
  left $= \pi_1(v_{j/2}) - \pi_1(v_1)$ und
  right $= \pi_1(v_j) - \pi_1(v_{j/2+1})$

Durch diese Bedingung werden die Knoten auf *die* Seite gesetzt, auf der die Nachbarn enger beieinander sind [JM97].

Sifting Diese Methode geht zurück auf Rudell [Rud93] und wurde ursprünglich eingeführt, um die Anzahl von Knoten in *Reduced Ordered Binary Decision Diagrams - ROBDD's* - zu reduzieren. Durch den Aufbau des Algorithmus ist es möglich, ihn auch bei dem hier vorliegenden Problem zu verwenden.
Die Vorteile dieses Algorithmus liegen in der Güte der Ergebnisse, die besonders bei dünnen Graphen erzielt werden kann. Andererseits benoetigt er eine hohe Laufzeit, was gerade bei großen oder dichten Graphen ein Problem ist. Die Idee dabei ist, die optimale Position eines Knotens u zu bestimmen, während die anderen Knoten an ihren Positionen fixiert bleiben.

Algorithmus: Sifting

Für jedes $u \in V_2$ führe aus
    Verschiebe u an die linkeste Position
    $crossings := \sum_{\pi_2(u)<\pi_2(v)} c_{uv}$
    $min_crossings := crossings$
    Für $p = 1$ bis $|V_2| - 1$ führe aus
        $crossings := crossings - c_{p(p+1)} + c_{(p+1)p}$
        Vertausche die Knoten von Position p und $p + 1$

Falls $crossings < min_crossings$
Setze $min_crossings := crossings$
$best_position := p$
Verschiebe u an die Position $best_position$

Da jeder Knoten an jede Postition gesetzt wird, ist die Laufzeit quadratisch, d.h. in $\mathcal{O}(|L_2|^2)$

Greedy-Insertion-Algorithmus Bei diesem Algorithmus besteht die Idee darin, als nächsten Knoten immer den Knoten u zu wählen, der die Anzahl der Kreuzungen minimiert, die zu u inzidente Kanten mit Kanten inzident zu Knoten rechts von u haben. Dies gilt nur, falls wir die Knoten von links nach rechts betrachten, ansonsten werden die Knoten links von u betrachtet. Zur genaueren Betrachtung siehe [EK86]. Die Laufzeit des Algorithmus ist wie bei Sifting quadratisch, also in $\mathcal{O}(|L_2|^2)$.

Stochastische Heuristik Diese Heuristik kann auf unser Problem, die Ordnung auf einem Layer bei fixiertem zweiten Layer herzustellen, angewandt werden, obwohl sie ursprünglich dazu gedacht ist, die Ordnung auf beiden Layern zu schaffen. Bei der Originalmethode werden Knoten von L_1 und L_2 an freie Positionen nach ihren *assessment numbers* gesetzt. Diese werden mit Hilfe der *frequency matrix* $F = (f_{i,j})_{i,j}$ berechnet. Hier wird die Wahrscheinlichkeit für ein Kante geschätzt, eine Kreuzung im Layout zu verursachen. Genauer: Die *frequency number* $f_{i,j}$ einer Kante e ist *die* Anzahl von Kanten, die e kreuzen, falls der komplette, bipartite Graph gezeichnet wird. Die genaue Definition und der Pseudo-Code für diese Heuristik sind in [Dre95] zu finden.

Assignment Heuristik Zur Problemlösung mittels dieser Heuristik werden vorab einige Definitionen eingeführt. Seien $i, j = 1, 2, \ldots, |L_2|$. Definiere d_{ij} als die Anzahl der Kantenkreuzungen, die verursacht werden von Kanten inzident zu Knoten i, falls i an Postion j gesetzt wird und für den restlichen Graphen gilt, daß jeder andere Knoten in L_2 adjazent zu allen $v \in L_1$ ist. Somit ist d_{ij} eine obere Schranke für die tatsächliche Anzahl der Kreuzungen zwischen zu i inzidenten und allen anderen Kanten, unter der Voraussetzung, daß Knoten i an Position j gesetzt wird. Sei $D := ((d_{ij}))$ die Matrix, die aus den Einträgen d_{ij} bestehe. Dann wähle $|L_2|$ Elemente aus der Matrix, so daß jede Zeile und jede Spalte einmal benutzt wurde und die Summe der Elemente minimal ist [Cat95].

Optimale Kreuzungsminimierung Um die Güte einer heuristischen Lösung einzuschätzen, ist es natürlich nötig, die optimale Lösung zu kennen. Ein Beispielalgorithmus dafür wäre der *Branch-and-Cut*-Algorithmus, der bereits angesprochen wurde. Er liefert exakte Lösungen bei relativ geringer Laufzeit. In Zahlen ausgedrückt bedeutet dies, daß der Algorithmus bei bis zu 60 Knoten mindestens so schnell ist wie die anderen Heuristiken, mit Ausnahme der *Barycenter*- und *Median*-Methode.

Wenn wir das *2-Layer-Crossing-Problem* umschreiben zu einem *Integer-Program*, so können wir eine optimale Lösung angeben mit Hilfe der

Integer-Programing-Methode Definiere für $G = (L_1, L_2, E)$ einen Binärvektor $x \in 0,1^{\binom{(|L_2|}{2}}$ mit Einträgen x_{uv} für $u < v$, $x_{uv} = 1$, falls u links von v, $x_{uv} = 0$ sonst. Unter Benutzung von Lemma 1 erhalten wir

$$cross(G, \pi_1, \pi_2) = \sum_{u<v \in L_2} (c_{uv}x_{uv} + c_{vu}(1 - x_{uv})) =$$

$$= \sum_{u<v \in L_2} (c_{uv} - c_{vu})x_{uv} + \sum_{u<v \in L_2} c_{vu}$$

Da die hintere Summe konstant ist, läßt sich das Problem wie folgt beschreiben:
Minimiere

$$z = \sum_{u<v \in L_2} (c_{uv} - c_{vu})x_{uv}$$

mit

1. $0 \leq x_{uv} + x_{vw} - x_{uw} \leq 1 \forall u < v < w$ mit $u \neq v \neq w \in L_2$
2. $x_{uv} \in \{0, 1\} \forall u < v$ mit $u \neq v \in L_2$

Die erste Bedingung garantiert, daß der Vektor x wirklich eine totale Ordnung auf L_2 induziert.
Bemerkung: Der optimale Wert z^* muß nicht unbedingt der minimalen Anzahl der Kreuzungen entsprechen, da nach obiger Gleichung gilt:
minimale Anzahl Kreuzungen $= z^* + \sum_{u<v \in L_2} c_{vu}$.
Um eine optimale Lösung zu erhalten, wird ein *Branch-and-Cut*-Ansatz gemacht [JM97]. Dabei wird die Bedingung 2. etwas abgeschwächt zu

$$0 \leq x_{uv} \leq 1 \ \forall \ u < v, u \neq v \in L_2$$

. Da die zuerst aufgestellte Menge von Bedingungen in $\mathcal{O}(|L_2|^3)$ liegt, wird zur Lösung ein *Schnitt-Ebenen*-Ansatz benutzt, die folgendermaßen beschrieben werden kann:

- Beginne mit nur den oberen Ungleichungen
- Die Elemente der ersten Menge werden iterativ benutzt, um den Lösungsraum zu *schneiden*
- Ergibt dies eine ganzzahlige Lösung, beende den Algorithmus
- Ansonsten wähle Variable x_{uv} und behandle zwei Probleme, einmal mit $x_{uv} = 0$ und das andere mal $x_{uv} = 1$
- Löse dieses Problem rekursiv

Zur genaueren Untersuchung dieses Problems siehe [JM97]. Ein großer Vorteil dieses *Integer-Programing*-Ansatzes ist die Tatsache, daß die optimale Lösung garantiert gefunden wird. Der Nachteil wiederum ist, daß der Algorithmus nicht notwendigerweise in polynomialer Zeit endet. Daher eignet er sich gut für Graphen kleiner und mittlerer Größe.
Eine weitere Methode zur Lösung bietet die

Planarisation Hierbei wird kurzzeitig eine minimale Menge von Kanten gelöscht, so daß der Graph planar wird, also ohne Kreuzungen gezeichnet werden kann. Zum Schluß werden die entfernten Kanten wieder eingefügt. Diese Methode kann jedoch sehr viele Kreuzungen erzeugen, was in Abbildung 5.13 und 5.14 deutlich wird.

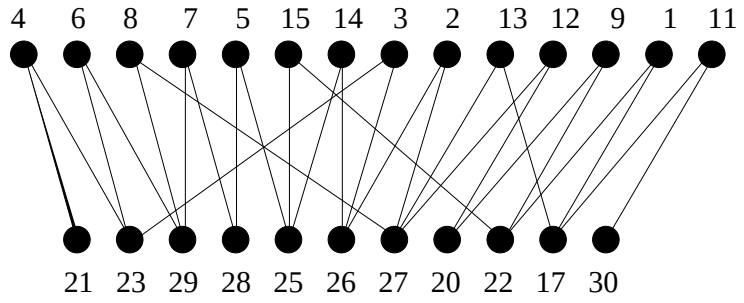


Abbildung 5.13: Mittels Planarisation gezeichneter Graph

Andererseits muß eine bessere Lesbarkeit nicht unbedingt weniger Kreuzungen bedeuten, was in Abbildung 5.13 zu sehen ist. Hier stechen die vier

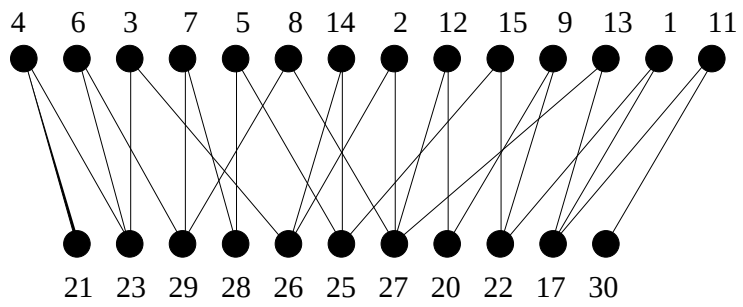


Abbildung 5.14: Graph mit minimaler Kreuzungszahl

kreuzenden Kanten sofort ins Auge und man kann allen Kanten gut folgen, was beim Layout mit weniger Kreuzungen in Abbildung 5.14 nicht unbedingt der Fall ist. Trotz der Nachteile, die diese Methode mit sich bringt, ist das Interesse an der Planarisation groß. Grund dafür ist die Möglichkeit, durch Lösung des *k-Layer-Planarization-Problem* eine bessere Lösung für das *k-Layer-Crossing-Problem* zu finden. Doch auch wenn das *k-Layer-Planarization-Problem* einfacher zu lösen erscheint, ist es Tatsache, daß auch das 2-Layer-Planarisations-Problem $\mathcal{NP}$-schwer ist. Zur genaueren Untersuchung und Betrachtung einer Lösung dieses Problems siehe [Mut96].

Bisher haben wir uns nur mit der *2-Layer-Crossing-Minimization* beschäftigt. Dies ist auch sinnvoll, da die Minimierung mit zunehmender Anzahl der Layer schwieriger wird. Dennoch sollte eine Methode zur Minimierung der Kreuzungen bei auch mehr als zwei Layern erwähnt werden.

k-Schicht-Kreuzungsminimierung Tutte's Algorithmus, der in [ES90] beschrieben wird, beschäftigt sich mit diesem Problem. Die Ergebnisse sind denen der Barycenter-Methode sehr ähnlich. Beim Positionieren der Knoten werden die Knoten des ersten und des letzten Layers fixiert. Dann werden in jedem anderen Layer die x-Koordinaten der Knoten berechnet durch den gewichteten Durchschnitt der x-Koordinaten der Nachbarn, d.h.

$$x(u) = \frac{1}{2 \cdot deg_{out}(u)} \cdot \sum_{v \in N_{out(u)}} x(v) + \frac{1}{2 \cdot deg_{in}(u)} \cdot \sum_{w \in N_{in(u)}} x(w).$$

Es muß nun also ein System von linearen Gleichungen gelöst werden, um $x(u)$ für alle u zu berechnen. Im Anschluß wird jedes Layer nach x sortiert.

Zusammenfassung der bisherigen Ergebnisse In der Theorie klingt die Median-Methode am geeignetsten, um das hier gestellte Problem zu lösen.

Es wird eine Reihenfolge konstruiert, die eine Anzahl von Kreuzungen verursacht, welche innerhalb eines Faktors 3 der minimalen Zahl von Kreuzungen liegt und außerdem in polynomialer Laufzeit berechnet werden kann.
In der Praxis wurden viele Tests durchgeführt, zum einen mit zufällig erzeugten Digraphen [EK86], [Mäk90], [JM97] oder mit konstruierten Digraphen [GKNV93]. Hierbei konnte jedoch keine Methode ermittelt werden, die eindeutig die besten Ergebnisse im Gesamtvergleich liefert. Somit empfiehlt sich eine Mischung aus mehreren Methoden, die in [GKNV93] getestet und bewertet wurde. Die Vorgehensweise ist hierbei die folgende:

1. Benutze die *Median*-Methode, um einen Initialreihenfolge zu bestimmen. Falls die Knoten u und v die gleichen Medianwerte besitzen, sortiere sie nach den *Barycenter*werten.

2. Um die Ausgabe des ersten Schrittes zu verbessern, benutze den *Adjacent-Exchange*-Algorithmus.

Eine weitere, sehr gute Methode ergibt sich bei der gemeinsamen Verwendung der Barycenter-Heuristik mit einer Heuristik, die sich *Global Sifting* nennt. Die Grundidee entspricht dem oben beschriebenen *Sifting*. Der Unterschied liegt in der Tatsache, daß nicht schichtweise die besten Positionen bestimmt werden, sondern global, d.h. in einer bestimmten Reihenfolge, welche die Knoten aus dem gesamten Netzwerk ordnet. Die Reihenfolge ergibt sich aus der Höhe der Knotengrade, die im Normalfall nicht schichtabhängig sind. Das *Global Sifting* eignet sich aufgrund der globalen Sichtweise gut für mehrschichtige Graphen, hat jedoch auch Nachteile. Bei schlechter Initialreihenfolge der Knoten ergeben sich viele Kreuzungen, die mit anderen Methoden verhindert werden können. Unzusammenhängende Teilgraphen können nicht gut entwirrt oder optimale Positionen nicht bestimmt werden. Dieses Verhalten kann durch das Vorwegschalten der *Barycenter*-Heuristik verbessert werden. Anhand einer Versuchsreihe wurde die Güte der jeweiligen Heuristik überprüft. Es wird sich zeigen, daß sich durch das Zusammenspiel zweier Heuristiken die besten Ergebnisse erzielen lassen. Die gesammelten Ergebnisse finden sich im Anschluß an die genaue Beschreibung von *Global-Sifting*

Global Sifting In der Implementation wird der erste Schritt zur Kreuzungsminimierung durch eine bereits umgesetzte *Barycenterheuristik* durchgeführt. Die Vorgehensweise ist bereits beschrieben, näheres zur Umsetzung findet sich in der AGD-Library, aus der die Heuristik importiert wird. Nach diesem ersten Schritt wird eine weitere Technik angewandt, die in dieser Form noch nicht beschrieben wurde, das *Global Sifting*. Das *Global Sifting* ist eine Variante des zuvor erläuterten *Sifting*. Der Unterschied besteht darin, daß

das Verschieben der Knoten nicht schichtweise geschieht, was beim Sifting der Fall ist. Im Falle des *Sifting* werden immer zwei Schichten betrachtet, von denen die Reihenfolge der einen fix ist, die Reihenfolge der anderen variabel.
Global Sifting weicht von diesem Konzept ab. Hier werden alle Knoten des Graphen $G = (V, E) = (V_1, \ldots, V_k, E)$ in einer Liste $l = (l_1, \ldots, l_n)$ mit $n = |V|$ gespeichert in absteigender Reihenfolge ihrer Knotengrade. Die l_i werden nun in ihren jeweiligen Layern in dieser Reihenfolge *gesiftet*, d.h. sie werden mittels Verschieben an die linkeste Position und dann durch Vertauschen mit dem jeweils rechten Nachbarn an *die* Position gesetzt, welche die geringste Anzahl von Kreuzungen verursacht. Falls keine Verbesserung erzielt werden kann, wird die Liste l invertiert und es wird ein Fehlversuch gezählt. Andernfalls wird die Reihenfolge für den nächsten Versuch beibehalten. Sind diese zwei Zyklen durchlaufen, wird die Reihenfolge der Listenelemente auf jeden Fall umgekehrt. Diese Vorgehensweise wird wiederholt, bis eine bestimmte Anzahl von Fehlversuchen erreicht worden ist. Die detaillierte Beschreibung für *Global-Sifting* wird im Anschluß gegeben.

Algorithmus: Global Sifting

Setze $failcount$ auf 0
Wiederhole
    Für alle Knoten l_i aus l führe $sift_node(l_i)$ aus
    Falls keine Verbesserung erzielt werden konnte
        Drehe die Reihenfolge in Liste l um
        $failcount++$

    Für alle Knoten l_i aus l führe $sift_node(l_i)$ aus
    Falls keine Verbesserung erzielt werden konnte
        $failcount++$

    Drehe die Reihenfolge in Liste l um

Bis $(failcount \geq MAXFAILS)$

Die Konstante MAXFAILS gibt die maximale Anzahl von Fehlversuchen an, die mittels der Variablen $failcount$ gezählt werden. Die Funktion $sift_node(l_i)$ setzt das bereits beschriebene Sifting um, d.h. jeder Knoten wird von seiner

Tabelle 5.2: Definition der in Tabelle 5.3 benutzten Abkürzungen

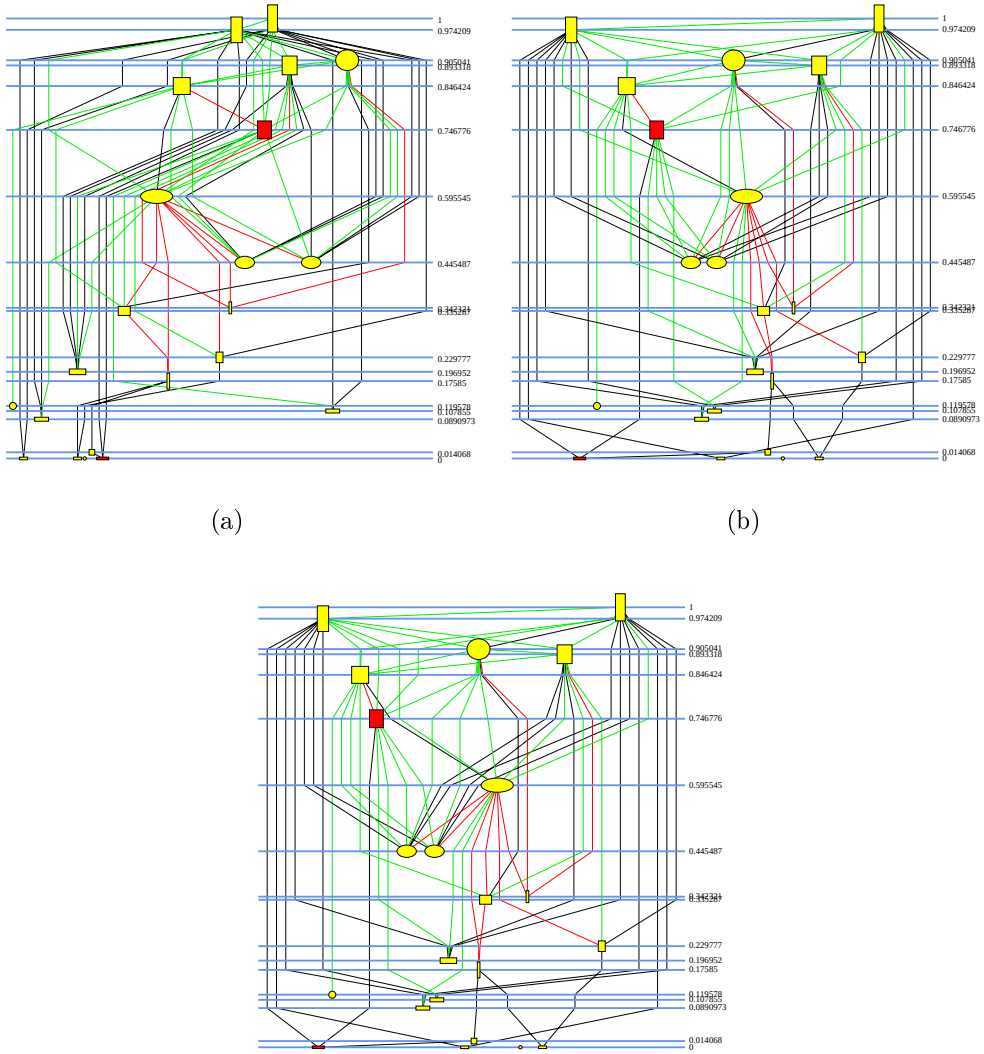
Bary	:=	Barycenter-Heuristik
Med	:=	Median-Heuristik
GS	:=	Global Sifting
Bary+GS	:=	Kombination aus Barycenter und Global Sifting
Med+GS	:=	Kombination aus Median und Global Sifting
EC	:=	Beidseitig bestätigte Verbindungen
EU	:=	Einseitig bestätigte Verbindungen

Tabelle 5.3: Anzahl der Kreuzungen bei verschiedenen Heuristiken

Stadt	Verbindung	Bary	Bary+GS	Med	Med+GS	GS
Ahlen	EU	33	31	34	31	68
Dresden	EC	1	1	0	0	1
Dresden	EU	215	165	209	163	279
Essen	EU	675	552	659	586	554
Köln	EU	884	813	1007	942	1051
Leipzig	EC	0	0	0	0	2
Leipzig	EU	1160	1021	1178	1092	1184
Plauen	EU	8	8	8	8	12
Ravensburg	EC	27	23	24	21	45
Ravensburg	EU	439	376	441	381	561
Singen	EU	54	48	49	43	92
Stuttgart	EU	233	213	221	212	483

aktuellen Position im Layer so weit links wie möglich geschoben und dann Position um Position nach rechts, um die Stelle im Layer zu finden, an der die wenigsten Kreuzungen produziert werden. Diese Methode hat sich in Tests als sehr erfolgreich erwiesen [MMS].
Durch das Zusammenspiel von *Barycentermethode* und *Global Sifting* konnte in Tests, die mittels der Implementation durchgeführt wurden, eine zum Teil recht hohe Verminderung der Kreuzungszahl erreicht werden. Die Ergebnisse sind in Tabelle 5.3 aufgeführt, die benutzten Abkürzungen werden in Tabelle 5.2 definiert.
Die Abkürzungen stehen für:
Wie wir sehen können, liefert die *Barycenter*-Heuristik nicht immer die optimale Lösung für ein *k-Schicht-Layout* für $k > 2$ (siehe *Dresden, EC*), auch

wenn dies für *2 Schichten* der Fall ist. Ebenfalls auffällig sind die relativ schlechten Ergebnisse des *Global Sifting* ohne Vorsortierung im Vergleich zur Kombination einer guten Heuristik wie *Median* oder *Barycenter* mit *Global Sifting*. Die Wahl der Heuristik zur Vorsortierung ist dabei nicht unbedingt von Bedeutung, je nach Graph erhalten wir mit *Barycenter* oder *Median* die besseren Ergebnisse. Existieren im Graphen mehrere dichter miteinander verbundene Teilgraphen, so können diese durch *Global Sifting* allein nicht gut entworren werden, was durch ein Vorwegschalten einer der beiden anderen Heuristiken verbessert werden kann.
Die Verminderung beträgt bei der Kombination von *Barycenter* und *Global Sifting* bis zu 23%, im Schnitt etwa $12,9\%$. Bei der Kombination von *Median* und *Global Sifting* erhalten wir eine Verbesserung um bis zu 22%, im Schnitt etwa $11,9\%$.
Die Abbildung 5.15 zeigt drei verschiedene Darstellungen des Graphen *Dresden, EU*, jeweils mit unterschiedlicher Kreuzungsminimierung. Es ist zu sehen, daß *Global Sifting* (Abb. 5.15(a)) alleine nicht den erwarteten Erfolg im Bezug auf Übersicht liefern kann. Die *Barycenter*-Methode (Abb. 5.15(b)) liefert ein recht übersichtliches Layout, daß jedoch an manchen Stellen (z.B. um die zwei Knoten in der linken oberen Ecke, betrachte die grünen Kanten) noch Mängel aufweist, die durch die Kombination mit *Global Sifting* (Abb. 5.15(c)) beseitigt werden können.
Auf das Anzeigen der Label wurde an dieser Stelle verzichtet, da das Hauptinteresse nicht in der Beschriftung, sondern in der besseren Übersicht liegt.



(a) (b)

(c)

Abbildung 5.15: Graph Dresden,EU erzeugt unter Verwendung von Global Sifting (a), Barycenter-Heuristik (b) und einer Kombination aus Barycenter-Heuristik und Global Sifting (c)

5.3 Zuweisung der Koordinaten

Bisher wurden die ersten zwei Schritte des Algorithmus von Sugiyama besprochen und Methoden zur Lösung dargestellt. Die Knoten wurden zum Aufbau einer Hierarchie in Layer eingeteilt, die Anzahl der Kreuzungen, die sich zwischen Kanten ergeben, wurde mittels Permutation der Knoten auf ihren Layern minimiert.
Der letzte Schritt weist die Knoten ihren genauen, horizontalen Positionen zu. Die Bedingungen für ein gutes Layout sind am Anfang dieses Kapitels beschrieben. Dazu gehört unter anderem die Forderung, daß Kanten möglichst gerade, d.h. vertikal, gezeichnet werden sollen.
Wie wir bereits wissen, werden Kanten mit einer Höhe größer 1 durch Dummyknoten unterteilt. Je nach Lage der Dummyknoten können Knicke entstehen. Wir suchen also Positionen für die Knoten, so daß die Winkel der Knicke relativ klein, die Kanten somit möglichst gerade sind, um eine bessere Lesbarkeit zu erreichen. Die Reihenfolge der Knoten darf hierbei nicht vertauscht werden.
Für die Lösung existieren exakte Algorithmen, die jedoch aufgrund ihrer Komplexität nur kleine Instanzen optimal lösen.

Exakte Algorithmen Das Problem kann wie folgt beschrieben werden : Sei $p = (v_1, v_2, \ldots v_k)$ ein gerichteter Pfad mit Dummyknoten $v_2, v_3, \ldots v_{k-1}$. Wäre der Weg gerade, so würde für $2 \leq i \leq k-1$ gelten:

$$x(v_i) - x(v_1) = \frac{i-1}{k-1} \cdot (x(v_k) - x(v_1))$$

wobei $x(v_i))$ die x-Koordinate von Knoten v_i beschreibt. Wichtig ist, daß diese Formel nur für ein Layout gilt, bei dem die Layer denselben Abstand haben. Die Formel kann jedoch auf Layer mit verschiedenem Abstand angepasst werden. Zur Vereinfachung definiere

$$\bar{x}(v_i) := \frac{i-1}{k-1}(x(v_k) - x(v_1)) + x(v_1).$$

Dies bezeichnet die x-Koordinate von v_i, falls der Knoten auf der direkten Verbindung von $x(v_1)$ und $x(v_k)$ liegt. Ein Maß für die Abweichung des Pfades von der geraden Strecke würde daher folgendermaßen berechnet werden können

$$dev(p) = \sum_{i=2}^{k-1} (x(v_i) - \bar{x}(v_i))^2$$

Um den Pfad so gerade wie möglich zu machen, minimiere die globale Summe

$$\sum_{\text{p ist ein Kantenpfad}} dev(p)$$

unter Berücksichtigung der Bedingung

$$x(w) - x(v) \geq \delta(w, v)$$

für alle Paare von Knoten w, v desselben Layers mit w rechts von v. Die Bedingung garaniert einen minimalen Abstand von δ zwischen zwei Knoten auf einem Layer. Außerdem bleibt damit die Reihenfolge, die durch den zweiten Schritt des Sugiyama-Algorithmus, die *Kreuzungsminimierung*, erzeugt wurde, erhalten. $\delta(w, v)$ wird berechnet durch die Größe der Knoten und die minimale Distanz, die zwischen zwei Knoten eingehalten werden soll.
Da die Zielfunktion dieses Problemes quadratisch ist, kann sie nur für kleine Instanzen optimal gelöst werden. Ausserdem würde eine optimale Lösung, falls diese nicht durch weitere Bedingungen eingeschränkt würde, eine exponentiale Breite zur Folge haben.

Ein anderes Ziel ist es, die Kanten so nahe wie möglich an senkrechte Linien zu zeichnen. Dieses Ziel wurde auch in der Implementation verfolgt. Die zugehörige Zielfunktion würde dann wie folgt aussehen

$$\sum_{(u,v) \in E} \Omega(u, v)\omega(u, v)|x(u) - x(v)|$$

wobei Ω ein Gewicht um lange Kanten zu begradigen bezeichnet und ω ein Maß für die Wichtigkeit einer Kante darstellt. Eine mögliche Gewichtung für die Kanten wäre die folgende:

$$\Omega(e) = \begin{cases} 8 & \text{, falls beide Endknoten Dummyknoten sind} \\ 2 & \text{, falls genau ein Endknoten ein Dummyknoten ist} \\ 1 & \text{, sonst} \end{cases} \tag{5.2}$$

Somit erlangen die *inneren* Kanten eines Pfades (die Kanten zwischen Dummyknoten) eine erheblich höhere Priotität als die anderen Kanten.
Zur Lösung dieses Problems wurde in [GKNV] eine völlig neue Methode entwickelt, die einen Hilfsgraphen erzeugt, mit dessen Hilfe das Problem in eine leichter zu lösendes umgewandelt werden kann.

Der Hilfsgraph $G_h = (V_h, E_h)$ enthält zusätzlich zu allen in G vorhandenen Knoten für jede Kante in G einen weiteren. V_h läßt sich damit schreiben $V_h =$

$V \cup \{[uv] | (u,v) \in E\}$. Im Hilfsgraphen G_h unterscheiden wir zwei Klassen von Kanten. Die eine Klasse beinhaltet die in G vorhandenen Originalkanten, die andere Klasse trennt die Knoten mit demselben Rang.
Für jede Kante $(u,v) \in E$ werden zwei neue Kanten $([uv],u)$ und $([uv],v)$ in G_h eingefügt. Wir definieren

$$\omega_h([uv],u) = \omega_h([uv],v) = \Omega(u,v)\omega(u,v)$$

und

$$\lambda_h([uv],u) = \lambda_h([uv],v) = 0.$$

Falls v ein linker Nachbar von w in G ist, so wird eine Kante (v,w) in E_h eingefügt. Ferner definieren wir $\omega_h(v,w) = 0$ und $\lambda_h(v,w) = \delta(v,w)$. Um den Zusammenhang zwischen den Problemen zu verstehen, betrachte die folgende Situation. Sei eine Lösung für das Positionierungsproblem auf G gegeben. Weise den Knoten $[uv]$ dem Minimum der beiden Layer, auf denen die Knoten u und v liegen, zu. Um eine optimale Layerzuweisung in G_h zu erhalten, muss $[uv]$ auf einem der beiden Layer liegen, also entweder auf dem Layer von v oder auf dem von u. Dann besitzt die eine, der beiden Kanten $([uv],u),([uv],v)$ die Länge 0, die andere die Länge $|x(u) - x(v)|$. Somit impliziert eine optimale Lösung in G_h ein Optimum in G und ein Layout für G_h ergibt eine Lösung für G.

Bestimmung des endgültigen Layout In der Implementation wurde das in der AGD-Library zur Verfügung stehende `FastHierarchyLayout` zur Bestimmung der endgültigen Koordinaten der Knoten benutzt, allerdings in einer veränderten Form. Da die umgesetzte Hierarchie nicht den für das `FastHierarchyLayout` geforderten Bedingungen entspricht, mußten einige Veränderungen vorgenommen werden, die in der Klasse `MyHierarchicalLayout` umgesetzt sind. Hierzu gehört die nicht äquidistante Positionierung der Layer und die Bestimmung des Mindestabstandes zwischen zwei Knoten auf einem Layer. Die Werte der Layer liegen im Intervall [0,1], das Layout soll auf die y-Koordinaten 0 bis 500 beschränkt sein. Daher ergibt sich für die vertikale Koordinate der Layer die Position *normierter Prestigewert*$\cdot$500. Zur Bestimmung des Mindestabstandes zwischen zwei Knoten werden alle benachbarten Knotenpaare betrachtet. Der Durchschnitt der Breiten wird mit einem Faktor (hier: 1.5) multipliziert. Die minimale Distanz wird als das Maximum der berechneten Werte gesetzt.
Die Positionierung der Dummyknoten in geclusterten Layern wird anschließend in einem nicht zum `MyHierarchicalLayout` gehörenden Teil des Programmes durchgeführt. Dabei werden die Dummyknoten so gesetzt, daß die zugehörigen Kanten, wenn möglich, senkrecht durch die verschmolzenen, d.h.

geclusterten Schichten verlaufen.
Die durch das `FastHierarchyLayout` generierten Layouts haben im optimalen Fall die Eigenschaft, daß die Kanten nicht mehr als zwei Knicke besitzen und die Teilstücke zwischen diesen Knicken vertikal gezeichnet werden. Eine genaue Beschreibung findet sich in [BJL99].

Kapitel 6

Schlußbemerkung

Wir haben unterschiedliche Möglichkeiten zur graphischen Darstellung von Wichtigkeit in einem Sozialen Netzwerk kennengelernt und untersucht. Die Unterschiede bestehen zum einen in der Wahl des Maßes für Wichtigkeit – Zentralität oder Prestige, werden Knotengrade oder kürzeste Wege zur Bestimmung des Wichtigkeitsmaßes verwendet, sind die Verbindungen gerichtet oder nicht – zum anderen in der Darstellung selbst.
Je nach Art der Beziehung erweist sich ein bestimmtes Maß als sinnvoll. Es gibt somit kein "generelles Maß" für Wichtigkeit, das sich auf alle auftretenden Beziehungen anwenden läßt, weshalb in dieser Arbeit auf die möglichst breitgefächerte Auswahl an Maßen Wert gelegt wurde. Zentraler Gegenstand war die Visualisierung von Status, d.h. die graphische Darstellung von Wichtigkeit, basierend auf berechneten Statuswerten.
Die Visualisierung an sich unterschied sich nicht wesentlich von den bisherigen Darstellungen von Netzwerken. Knoten wurden als geometrische Figuren wie Kreise, Rechtecke oder Rauten gezeichnet, Kanten als Linien. Die Färbung der Kanten diente einzig und allein der besseren Lesbarkeit. Sehr kurze Kanten konnten nicht mit Pfeilenden versehen werden; daher wurde die Farbe als Indikator für die Richtung der Kanten gewählt, schwarze Kanten verliefen von unten nach oben, rote von oben nach unten und grüne in beide Richtungen. Status wurde nicht durch die Größe eines Akteurs angezeigt, sondern durch verschiedene Höhen im Layout. Der Unterschied zu bisherigen Darstellungen bestand somit in der Art, wie die Positionen der Knoten und Kanten festgelegt wurden.
Die Positionen der einzelnen Akteure und Verbindungen bei der Visualisierung von Status in Sozialen Netzwerken wurden mit Hilfe des *Algorithmus von Sugiyama* bestimmt. Die Art der Bearbeitung der einzelnen Teilschritte unterschied sich jedoch von bisherigen Umsetzungen. Durch die strikte Trennung der verschiedenen Berechnungsschritte war es möglich, aus einer großen

Bandbreite von Variationen zur Bearbeitung der im Algorithmus auftretenden Teilprobleme zu wählen.
Neu war die Benutzung von Statuswerten, um die *Layerzuweisung* durchzuführen. Die Bedingung eines azyklischen Graphen als Eingabe für die Bestimmung der Schicht für jeden einzelnen Knoten konnte völlig vernachlässigt werden. Die normalerweise äquidistante Positionierung der Layer wurde dadurch ersetzt, daß die Distanz zwischen zwei Layern durch die berechneten Statuswerte festgelegt wurde. Ebenfalls neu war die *Clusterung* der Layer. Eine Verschmelzung ist zwar in manchen sozialen Netzwerken zu finden, jedoch nicht bei Layern, sondern eher bei Akteuren. Durch die *Clusterung* haben sich neue Probleme ergeben, z.B. das Einfügen neuer Dummyknoten. Um Kantenabschnitte, die durch verschmolzene Layer verlaufen, besser sichtbar zu machen, wurde ein möglichst vertikaler Verlauf der Kanten gewünscht und durch die Positionierung der neuen Dummyknoten verwirklicht. Dadurch konnte die Bedingung, daß eine Kante höchstens zwei Knickpunkte besitzen darf, nicht immer eingehalten werden, was aber nicht als großer Nachteil gesehen werden sollte.
Die *Kreuzungsminimierung* wurde durch die Kombination zweier Heuristiken im Vergleich zu bisher benutzten Versionen verbessert. Durch das Vorsortieren mittels *Barycenter*- oder *Median*heuristik und die anschließende Verwendung des *Global Sifting* konnte eine Verminderung der Kreuzungszahl um etwa 12% erreicht werden (siehe Kapitel 5, Abschnitt 5.2). Neben wiedergegebenen Gütegarantien wurden verschiedene Möglichkeiten zur Minimierung von Kantenüberschneidungen vorgestellt, von denen die *2-Schicht-Kreuzungsminimierung* die größte Anzahl an Variationen bot. Durch die problematische Durchführung einer *k-Schicht-Kreuzungsminimierung*, konzentrierten sich die meisten Heuristiken auf die Minimierung der Kreuzungen zwischen zwei Layern, was sich in der Fülle der in der Literatur zu findenden Möglichkeiten wiederspiegelte. Die Besonderheit im *Global Sifting* lag darin, daß zwar Kantenüberschneidungen zwischen zwei Layern vermindert wurden, die Vorgehensweise aber von der üblichen Reihenfolge abwich. Es wurde nicht schichtweise eine Kreuzungsminimierung durchgeführt, sondern *global*. Das bedeutet, die Knoten wurden aufgrund der Höhe ihres Knotengrades in eine Liste sortiert, die dann – unabhängig von den Schichten – durchlaufen wurde, um die Knoten in ihren jeweiligen Layern an die bestmögliche Position zu setzen.
Die *Zuweisung der Koordinaten* bot durchaus auch Varianten, die jedoch im Vergleich zur Auswahl an Heuristiken zur *Kreuzungsminimierung* eher in geringer Zahl zu finden waren. Ein Hauptanliegen war, die Kanten möglichst senkrecht verlaufen zu lassen, um eine gute Lesbarkeit zu gewährleisten. Ebenso wurden Überschneidungen von Knoten derselben Schicht durch einen

Mindestabstand vermieden. Durch die untypische Verschmelzung von Layern mußten zusätzliche Dummyknoten eingeführt werden, die nicht durch das Bilden einer für die Kreuzungsminimierung notwendigen vollständigen Hierarchie gesetzt wurden.
Die hier eingeführte Art der Visualisierung von Status in sozialen Netzwerken bietet mit Sicherheit noch Gelegenheit zur Variation oder Weiterentwicklung. *Clusterungen* können auf die unterschiedlichsten Weisen vollzogen werden, von denen eine hier sinnvoll erscheinende umgesetzt wurde. Die *Kreuzungsminimierung* ist noch nicht an ihre Grenzen gestoßen, da zum Teil mit bloßem Auge eine Kantenüberschneidung zu erkennen ist, die vermieden werden könnte. Das Problem liegt in der Komplexität, d.h. eine minimale Anzahl von Kreuzungen soll in möglichst geringer Zeit gefunden werden, was (im Moment noch) einen erheblichen Aufwand zur Folge hat. Bisher sind jedoch noch keine Algorithmen bekannt, die das Problem effektiver lösen. Beim Setzen der zusätzlich durch die *Clusterung* notwendig gewordenen Dummyknoten wurde eine Möglichkeit, die an dieser Stelle sinnvoll erschien, ausgewählt. Ändert sich die Art der *Clusterung* oder ein anderer Teil der Knotenpositionierung, kann eine andere Variation die Anforderungen besser erfüllen.

Im Anhang folgt nun die Beschreibung der durch die Implementation umgesetzten Methoden und eine Anleitung zur Anwendung des Programmes, ebenso wie einige graphische Beispiele.

Anhang A

Implementation

Die Beschreibung der Implementation wird in drei Abschnitte unterteilt. Der erste beschäftigt sich mit den verwendeten, zur Verfügung stehenden Softwarepaketen **LEDA**, **AGD** und **LAPACK**. Der zweite Abschnitt erläutert die selbst implementierten Module und Funktionen. Im letzten Teil wird dem Benutzer eine Einführung in den Umgang mit der Implementation gegeben.

A.1 Softwarepakete

Die Pakete **LEDA** und **AGD** hängen eng zusammen. **LEDA** steht für **L**ibrary of **E**fficient **D**ata types and **A**lgorithms, **AGD** kürzt **A**lgorithms for **G**raph **D**rawing ab. Die Pakete stehen für die Forschung und nicht kommerzielle Nutzung kostenlos zur Verfügung und können unter

http://www.mpi-sb.mpg.de/LEDA
bzw. **http://www.mpi-sb.mpg.de/AGD**

heruntergeladen werden.
LAPACK bietet die Möglichkeit, komplizierte Berechnung aus dem Bereich der Linearen Algebra durchzuführen. Die Abkürzung steht für **L**inear **A**lgebra **PACK**age. Ein kostenloses Download ist unter

http://www.netlib.org

erhältlich.
Alle drei Pakete sind in C++ implementiert.

LEDA Verwendet werden aus dieser Bibliothek in der Hauptsache zwei Datenstrukturen, *graph* und *GraphWin*. In diesem Zusammenhang werden Typen benutzt, die in enger Beziehung mit diesen beiden stehen, unter anderem *node* und *edge*, welche Knoten und Kanten repräsentieren. Jeder Knoten kann mit verschiedenen Eigenschaften versehen werden wie *Farbe, Form, Größe, Label* ... , ebenso die Kanten. Weitere Eigenschaften zu Knoten können mittels eines *node*- bzw. *edgearray* oder einer *node*- bzw. *edgemap* gespeichert werden.
GraphWin vereint die Eigenschaften der Datenstrukturen *graph* und *window*, so daß ein Graph bildlich dargestellt werden kann.

AGD Benutzt werden mehrere Strukturen und Module, die in erster Linie dem Layout dienen. Die Klasse `FastHierarchyLayout` wurde umgewandelt und ergänzt zu `MyHierarchicalLayout`. Der Hauptunterschied besteht in der Bestimmung der vertikalen Koordinaten, die durch die berechneten Statuswerte festgelegt sind. Auf diesen Punkt wird in der Beschreibung der Klasse `MyHierarchicalLayout` genauer eingegangen.
Die Verwendung eines hierarchischen Layout setzt eine Datenstruktur `Hierarchy` voraus. In ihr werden die Knoten in verschiedenen Schichten gespeichert, die den Namen `Layer` tragen. Ein `Layer` beinhaltet die Knoten und ihre Reihenfolge innerhalb der Schicht, nicht aber die genauen Positionen. Alle `Layer` zusammen bilden die `Hierarchy`. Sie ermöglicht die Benutzung der Heuristiken zur Minimierung der Kreuzungen.
Die Voraussetzung, eine Kreuzungsminimierung in der hier benutzten Form durchführen zu können, ist ein vollständig geschichteter Graph. Keine Kante darf eine Höhe größer 1 besitzen. Dies geschieht durch die Einführung der Dummyknoten. Da die Schichtung für die originalen Knoten schon gegeben ist, können die Dummyknoten unter Benutzung einer `GraphCopy` eingesetzt werden. Damit wird eine Kopie GC des Graphen G angelegt, wobei zusätzliche Informationen gespeichert werden. Zu jedem Knoten v_c in GC wird der korrespondierende Knoten v des Originalgraphen G gespeichert, ebenso wie zu jeder Kante e_c aus GC die Originalkante e in G. Dies ist natürlich nur der Fall, wenn Knoten und Kanten bereits in G existiert haben, also nicht nachträglich eingefügt wurden, wie das bei den *Dummyknoten* der Fall ist. Für jeden Knoten v aus G wird die Kopie v_c in GC gespeichert, für jede Kante in G eine Liste von *Dummykanten* in GC, die durch das Einfügen der Dummyknoten entsteht. Werden auf einer Kante $e = \{u, v\}$ aus G in GC zwei Dummyknoten d_1, d_2 eingefügt, wird in GC die Liste der Kanten $\{u, d_1\}, \{d_1, d_2\}, \{d_2, v\}$ gespeichert.

Zur Minimierung der Kreuzungen können verschiedene Heuristiken benutzt werden, von denen in der Implementation der `BarycenterHeuristic` der Vorzug gegeben wurde in Verbindung mit einer selbst implementierten Version von `GlobalSifting`. Gründe dafür finden sich in Kapitel 5, Abschnitt 5.2.

LAPACK Die Verwendung von LAPACK beschränkt sich auf zwei Berechnungen, die der Eigenwerte und Eigenvektoren der Adjazenzmatrix A zur Berechnung des Faktors $1/a$ und Lösung der linearen Gleichung $\{[1/a * I - A'] * p\} = x$ zur Berechnung der Statuswerte. Eine genauere Beschreibung wird in Abschnitt A.2 unter A.2.2 gegeben.

A.2 Module und Funktionen

Der Abschnitt A.2 beschreibt den Aufbau und die Handhabung der in der Implementation benutzten Klassen und Programmteile. Bei darüber hinausreichendem Interesse sei auf die jeweiligen Source- und Headercodes verwiesen, die hinreichend kommentiert sind.
Die Aufteilung entspricht im großen und ganzen dem in der Implementation gewählten zeitlichen Ablauf, d.h. es wird zunächst das Hauptprogramm beschrieben, dann die Layerzuweisung, gefolgt von der Kreuzungsminimierung und der endgültigen Positionierung der Knoten und Kanten.

A.2.1 Hauptprogramm

Durch die Bereitstellung des Menus *Sugiyama* (siehe Abbildung A.1) erhält der Benutzer die Möglichkeit, die Art des Layouts zu bestimmen. Die verschiedenen Einstellungen, die zur Verfügung stehen, werden im Abschnitt A.3 beschrieben und sollen deshalb an dieser Stellen nicht weiter ausgeführt werden.
Je nach Wahl der gewünschten Hierarchiebestimmung wird eine Funktion aufgerufen, welche die gewünschte Einteilung der Knoten in Level vornimmt. Die Verwendung der in Kapitel 5 beschriebenen *Clusterung* ist ebenfalls optional, und hat bei Benutzung an späterer Stelle das Einfügen und Positionieren neuer Dummyknoten zur Folge.
Ist dieser erste Teil des Algorithmus von Sugiyama, die *Layerzuweisung*, abgeschlossen, wird die *Minimierung der Kreuzungen* vorgenommen. Aufgrund der guten Ergebnisse in der Praxis wird hierzu eine Kombination einer bereits in der *AGD* zur Verfügung stehenden Heuristik, der *BaeycenterHeuristic*, und

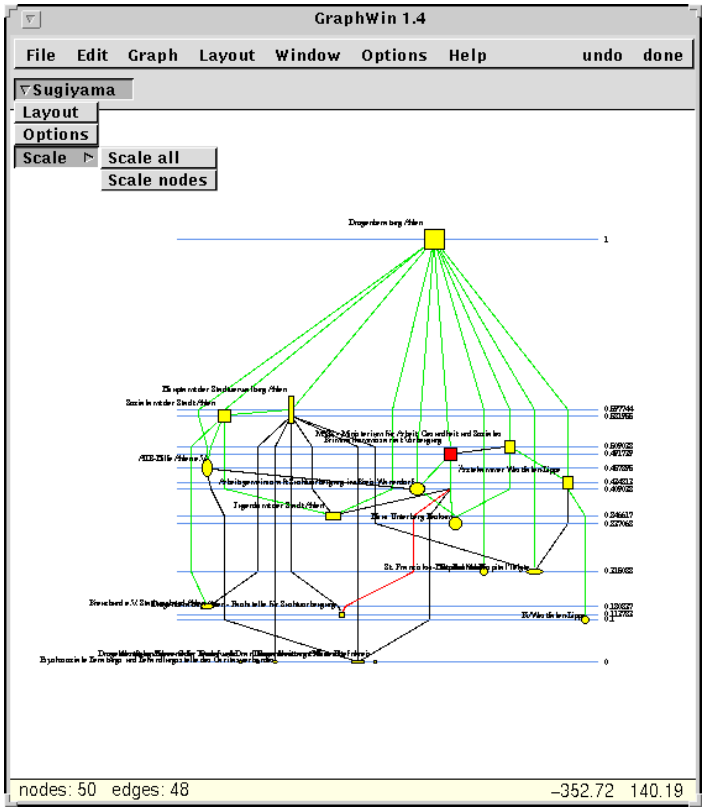


Abbildung A.1: Beispiel mit geöffnetem Menu Sugiyama

einer eigens für dieses Programm implementierten Version des *GlobalSifting* verwendet. Wird *GlobalSifting nach* der Erstellung einer ersten Knotenreihenfolge für die einzelnen Level durch die *BarycenterHeuristic* angewandt, so ergibt sich eine zum Teil erhebliche Verminderung der Kreuzungen. Zur genauen Analyse betrachte die in Tabelle 5.3 gemachten Untersuchungen. Genauere Beschreibungen zur Kreuzungsminimierung finden sich in einem späteren Teil dieses Kapitels.
Ist auch die *Kreuzungsminimierung* abgeschlossen, kann mit der genauen Positionierung der Knoten und Kanten begonnen werden. Dazu wird im ersten Schritt die Klasse `MyHierarchicalLayout` benutzt. Nach dem Einfügen der Level, die durch blaue Kanten repräsentiert werden, muß geprüft werden, ob Level geclustert wurden oder nicht. Ist dies der Fall, wird im Anschluß an die Verwendung des `MyHierarchicalLayout` die Clusterung der Level rückgängig gemacht, was sich im Einfügen neuer Dummyknoten wiederspiegelt. Die Vorgehensweise dabei ist die folgende:
Durchkreuzt eine Kante ein verschmolzenes Level, ohne einen Endknoten in-

nerhalb dieser Clusterung zu haben, wird der bereits durch die oben erwähnte `GraphCopy` gesetzte Dummyknoten gelöscht und durch zwei neue ersetzt. Die x-Koordinate entspricht dabei der x-Koordinate des gelöschten Knotens, die y-Koordinate ist durch das jeweils oberste und unterste Level der Clusterung bestimmt.
Befindet sich ein Endknoten einer Kante, deren zweiter Endknoten höher liegt, innerhalb oder auf dem untersten Level eines verschmolzenen Layers, so wird ein zusätzlicher Dummyknoten auf dem obersten Level eingefügt. Analoges gilt für den Fall, daß der zweite Knoten tiefer liegt und der erste innerhalb oder auf dem obersten Level.
Die x-Koordinate des neu gesetzten Knotens entspricht zu Beginn der horizontalen Koordinate des entsprechenden Endknotens. Da jedoch mehrere Kanten die Hinzunahme eines solchen Dummyknotens erfordern könnten, müssen die Knoten nach rechts oder links verschoben werden, um eine Überlagerung zu verhindern. Welcher Knoten auf welche Seite verschoben wird, wird anhand der Lage des jeweils folgenden Dummy- oder Endknoten bestimmt. Je nach Position des Nachfolgers (oder - falls die Kanten in entgegengesetzter Richtung verlaufen - des Vorgängers) wird einer der Dummyknoten um eine Konstante nach rechts oder links versetzt. Ist auch diese Position schon durch einen Dummyknoten belegt, wird wiederum für diese zwei entschieden, welcher Knoten versetzt wird. Die Entscheidung, welcher Knoten auf welche Seite verschoben wird, wird anhand der Steigung des nächsten Teilstückes der Kante bestimmt, d.h. die Steigung des Abschnittes zwischen dem eben gesetzten Dummy und dem auf der Kante folgenden Nachfolger (bzw. Vorgänger).
Um das endgültige Layout zu erhalten, werden Pfeile an den Kanten, falls sie existieren, gelöscht und die Richtung durch Farben kenntlich gemacht. Führt eine Kante von oben nach unten, erhält sie die Farbe *rot*, führt sie von unten nach oben die Farbe *schwarz* und falls zu einer Kante eine Rückwärtskante existiert, wird eine der beiden Kanten gelöscht und die andere *grün* gefärbt. Kanten zwischen Knoten desselben Levels werden als Kreiskanten, also `circle-edges`, dargestellt. Dazu wird zusätzlich ein *bendpoint* eingefügt in der Mitte der beiden Knoten, mit einer y-Koordinate, die eine konstante Entfernung vom gemeinsamen Level hat und je nach Bedarf ober- oder unterhalb gesetzt wird. Durch den Verlauf des Kreisbogens wird die Richtung der Kante anzeigt. Verläuft eine Kante von links nach rechts, so wird der Bogen oberhalb des Levels gezeichnet, ansonsten unterhalb. Schleifen (*loops*), also `circle-edges` mit demselben Anfangs- und Endknoten werden immer oberhalb der Level positioniert.

A.2.2 Bestimmung der Hierarchie

Die Bestimmung der Hierarchie, also die *Layerzuweisung* kann auf unterschiedliche Arten geschehen. Im Vordergrund steht die Verwendung der Klasse `PrestigeRanking`, somit die Benutzung von Statuswerten. Weitere Möglichkeiten bestehen beispielsweise, wie bereits oben erwähnt, in der Verwendung einer Tiefen- oder Breitensuche. Da diese Möglichkeiten in der **AGD** zur Verfügung und nicht im Mittelpunkt dieser Arbeit stehen, wird auf eine ausführlichere Beschreibung verzichtet. Bei Verwendung einer Clusterung kommt eine weitere Klasse ins Spiel, die Klasse `LevelMelting`.

Die Berechnung der Statuswerte Aufgabe der Klasse `PrestigeRanking` ist es, die Knoten eines Graphen den Leveln bezüglich ihrer Statuswerte zuzuordnen. Dazu wird dem Konstruktor der Klasse der *Graph* und ein *node_array* übergeben. In diesem *node_array* wird zu jedem Knoten des *Graphen* das zugehörige Level als ganze Zahl gespeichert. Der Index des untersten Level ist 0, der Index des obersten ist *Anzahl der Level - 1*. Die Berechnung und Initialisierung erfolgt durch Aufruf der Funktion `call()`.
Zu Beginn wird die Adjazenzmatrix A bestimmt. Für diese Matrix werden alle Eigenwerte und Eigenvektoren mit Hilfe des **L**inear**A**lgebra**PACK**age `LAPACK` berechnet. Der größte,reelle, berechnete Eigenwert wird zur Bestimmung des Faktors $1/a$ verwendet, dessen Herleitung im Kapitel 5, Abschnitt 5.1 beschrieben wird. Die Eigenwerte und -vektoren werden in der Klasse `my_lapack` durch Aufruf einer externen, zu `LAPACK` gehörenden Funktion, berechnet. Wird diese Rechnung korrekt ausgeführt, kann der größte, reelle Eigenwert bestimmt und ausgegeben werden. Der Aufruf der externen `LAPACK`-funktion `Lapack_LU_linear_solve(Diff, x)` liefert eine Lösung zum linearen Gleichungssystem

$$Diff * p = x,$$

welche die benötigten Statuswerte im Vektor p speichert (siehe oben). Hierbei ist *Diff* die durch $[1/a * I - A']$ beschriebene Matrix und x der Vektor,der aus den Knoteneingangsgraden besteht. Anhand der hier berechneten Statuswerte werden die Knoten in ihre jeweiligen Level eingeteilt, d.h. das zuvor dem Konstruktor übergebene *node_array* wird initialisiert. Dabei muß beachtet werden, daß die Statuswerte normiert werden, der kleinste Statuswert ist immer 0, der größte immer 1. Die Reihenfolge der Layer in einer `Hierarchy` verläuft normalerweise von oben nach unten. In dem hier gewählten Verfahren muß diese umgekehrt werden, um ein dem Problem angepasstes Layout zu erhalten. Damit im Hauptprogramm auf die Statuswerte zugegriffen werden

kann, müssen die Werte "von Hand" auf das zum Hauptprogramm gehörende *node_array* `status` übertragen werden.

Die Clusterung der Layer Die Klasse `LevelMelting` kann für eine gegebene Hierarchie eine *Clusterung* vornehmen. Bedingungen, die an diese Clusterung gestellt werden, müssen dafür in der Funktion `melt_level()` implementiert werden. In dem hier umgesetzten Fall werden zwei Layer jeweils dann verschmolzen, wenn die vertikale Distanz kleiner als eine selbst festzusetzende Schranke `epsilon` ist. Der Wert kann mittels `get_bound()` abgefragt und mittels `set_bound()` auf einen Defaultwert (hier: 0.05) oder durch `set_bound(value)` auf den Wert "value" gesetzt werden. Initialisiert werden muß ein Objekt der Klasse `LevelMelting` mit dem im Hauptprogramm verwendeten `graph` *G*, `GraphWin` *gw*, einem `node_array` *level*, in dem zu jedem Knoten das zugehörige Layer nach der Verschmelzung gespeichert wird, und einem `node_array` *original_rank*, in dem zu jedem Knoten die ursprünglich berechneten Statuswerte enthalten sind.
Der Aufruf der Clusterung erfolgt mittels der Funktion `call()`.
In den Abbildungen `map<int,bool> uppermost_bound` und `map<int,bool> lowest_bound` werden zu jeder Layernummer die obere und untere Grenze gespeichert. Dies bedeutet, wurde das Layer i verschmolzen, so enthält `uppermost_bound[i]` den Statuswert des Layers, das beim Verschmelzen den höchsten Statuswert hatte, `lowest_bound[i]` den Statuswert des Layers, das den niedrigsten Statuswert hatte. Wurde das Layer nicht verschmolzen, so sind `uppermost_bound[i]` und `lowest_bound[i]` identisch und entsprechen dem Statuswert des Layers. Durch Vergleich von `uppermost_bound[i]` und `lowest_bound[i]` kann somit sehr schnell entschieden werden, ob ein Level verschmolzen wurde oder nicht. Daher sollten diese *maps* auf das Hauptprogramm übertragen werden.

A.2.3 Minimierung der Kreuzungen

Die Minimierung der Kreuzungen verläuft in der Implementation in zwei Schritten. Im ersten Schritt werden die Kreuzungen mittels einer in der **AGD** verfügbaren Heuristik primär geordnet. Auf diese bestehende Ordnung wird dann das *GlobalSifting* angewandt, welches auf keinen Fall eine Erhöhung der Kreuzungszahl zur Folge hat, da jeweils die beste Position im Layer gesucht wird und im Notfall die Ausgangsposition beibehalten werden kann.
Die Berechnung der primären Kreuzungsminimierung mit Hilfe der *BarycenterHeuristic* erfolgt innerhalb der Klasse `SugiyamaLayout` durch das Festsetzen der Heuristik mittels der Funktion `set_cross_min(BarycenterHeuristic())` und dem folgenden Aufruf von `reduce_crossings(GC,H)`. Es muß also eine

Kopie des Graphen durch ein Objekt *GC* der Klasse `GraphCopy` und eine mit dem Knotenarray, das die Nummern der Level für jeden Knoten enthält, initialisierten Hierarchie *H* übergeben werden. Zur Verfügung stehen neben der *BarycenterHeuristic* unter anderem die *SplitHeuristic*, die *MedianHeuristic* oder die *GreedyInsertHeuristic*, die in Kapitel 5, Abschnitt 5.2 angesprochen wurden.
Die Umsetzung des *GlobalSifting* ist in der Klasse `MyCrossing` implementiert.

Verwendung von GlobalSifting Der Unterschied zwischen *GlobalSifting* und den oben erwähnten Heuristiken zur Kreuzungsminimierung ist in erster Linie die Tatsache, daß es sich bei *GlobalSifting* nicht um eine *2-Schichten-Kreuzungsminimierung* handelt, sondern daß die Knoten *global*, d.h. nicht durch Betrachtung von je zwei Schichten geordnet werden. Der Vorteil dieser Arbeitsweise ist in der Verminderung der Kreuzungszahl deutlich zu erkennen. Voraussetzung für diese guten Ergebnisse ist eine gute Ausgangssituation, also eine schon recht gut geordnete Hierarchie. Nur dann können auch gute Ergebnisse erzielt werden.
Der Aufruf der Kreuzungsminimierung erfolgt, anders als beispielsweise in den Klassen, die eine *2-Schichten-Kreuzungsminimierung* implementieren, durch `call(GC,H)`, also die Übergabe einer `GraphCopy` *GC* und einer `Hierarchy` *H*. Die `call(GC,L)`-Aufrufe in den anderen Klassen erhalten jeweils eine `GraphCopy` *GC* und ein einzelnes `Layer` *L*, wobei die Klassen aber mittels `init(GC,H)` mit der `GraphCopy` und der `Hierarchy` initialisiert werden müssen.
Der genaue Aufbau des Algorithmus für *GlobalSifting* ist in Kapitel 5, Abschnitt 5.2 beschrieben.
Die Anzahl der *Fehlschläge* wird in der Variablen `failcount` gespeichert. Überschreitet diese Anzahl eine zuvor festgelegte Konstante `MAXFAILS`, wird der Sortiervorgang abgebrochen. Aufgrund der in Testreihen erzielten, guten Ergebnisse (s.[MMS99]) wurde diese Schranke auf den Wert 0 gesetzt. Die Anzahl der Kreuzungen zwischen Layer i und Layer $i+1$ wird jeweils mit Hilfe der für eine `Hierarchy` gegebenen Funktion `calc_crossings(i)` berechnet. Da die Länge einer Kante durch das Einfügen der Dummyknoten immer 1 ist, genügt es, die Kreuzungen zwischen dem Layer, auf dem der betrachtete Knoten liegt, und den beiden jeweils direkt benachbarten Layern zu betrachten. Eine Ausnahme ist die Situation, daß der Knoten auf dem obersten oder untersten Layer plaziert ist. Dann beschränkt sich die Betrachtung auf die Kreuzungen zwischen diesem und dem einen, direkt benachbarten Layer.
Da die im Konstruktor übergebene Hierarchie eine Referenz ist, werden automatisch alle Änderungen übertragen und nach außen sichtbar. Die neu

entstandene Hierarchie ist somit sofort im Hauptprogramm verfügbar und kann zur genauen Positionierung benutzt werden.

A.2.4 Positionierung

Die Positionierung der Knoten und Kanten erfolgt in mehreren Schritten. Zunächst wurden die Knoten mittels der *Kreuzungsminimierung* auf ihren jeweiligen Layern geordnet. Wurde bei der Bestimmung der Hierarchie *Clusterung* benutzt, so kann sich das Aussehen und die Anzahl der Layer mit und ohne Verschmelzung von Layern erheblich unterscheiden. Daher müssen besonders bei der endgültigen Positionierung Unterschiede gemacht werden, die beispielweise in der Verwendung der Klasse `MyHierarchicalLayout` deutlich werden. Es ist außerdem möglich, daß neue Dummyknoten eingefügt werden, um ein übersichtlicheres Layout zu erhalten.

Erster Teil der Koordinatenbestimmung Die Klasse `FastHierarchyLayout` ist die Ausgangsbasis für die hier verwendete Klasse `MyHierarchicalLayout`. Der Unterschied liegt in der Bestimmung der y-Koordinate für die Knoten (und damit die Layer), ebenso wie Bestimmung des minimalen Abstandes zwischen zwei Knoten auf einem Layer. Daher sei im folgenden nur die Modifikation beschrieben, der beibehaltene Teil wird nicht weiter erläutert und kann in der Beschreibung der Ursprungsklasse `FastHierarchyLayout` nachgelesen werden, z.B. in [BJL99].
Die Modifikation mußte aus mehreren Gründen gemacht werden. Zum einen aufgrund der nicht äquidistanten Setzung der Layer, zum anderen durch die mögliche *Clusterung* einiger Layer. Durch die *Clusterung* läßt sich einem verschmolzenen Layer keine eindeutige y-Koordinate zuweisen. Um dieses Problem zu beheben wird jedem geclusterten Layer als y-Koordinate der Durchschnitt der y-Koordinaten der in diesem einen Layer zusammengefaßten Layer zugewiesen. Wird die Verschmelzung im Hauptprogramm nicht rückgängig gemacht, werden die Layer also an dieser, somit bestimmten vertikalen Position gezeichnet. `FastHierarchyLayout` zeichnet die Layer in gleichem Abstand, d.h. eine minimale Layerdistanz muß gewährleistet sein. Dies ist in `MyHierarchicalLayout` nicht der Fall, da die Layer entsprechend ihrer Statuswerte plaziert werden. Wird die Clusterung rückgängig gemacht, was in der hier gewählten Implementation der Fall ist, müssen neue Dummyknoten eingefügt und positioniert werden.

Aufhebung der Clusterung Wurden Layer verschmolzen, wird dies nach der primären Positionierung der Knoten und Kanten durch

`MyHierarchicalLayout` rückgängig gemacht. Da zu jedem Knoten im `node_array` *org_rank* die ursprünglichen Statuswerte gespeichert sind, werden die Knoten dementsprechend in ihrer Höhe positioniert. Analoges gilt für das Zeichnen der Layer, deren y-Koordinate durch einen auf ihnen liegenden Knoten gegeben ist. Die x-Koordinate jedes Knotens ist durch den in `MyHierarchicalLayout` berechneten x-Wert genau bestimmt und wird nicht mehr verändert.
Das Einfügen und Positionieren neuer Dummyknoten ist im Abschnitt A.2.1 beschrieben.

A.3 Benutzerführung

Die Beschreibung des Umgangs mit den hier verwendeten Klassen ist im Abschnitt über die jeweilige Klasse beschrieben. Daher wird nur kurz die Benutzung von GraphWin und im Anschluß der Gebrauch des im Hauptprogramm entworfenen Menues beschrieben.
Für die Benutzung der implementierten Version des Sugiyamaalgorithmus ist es nicht wichtig, ob ein Graph von Hand erzeugt wurde, also durch den Gebrauch der Maus, das kreieren eines neuen Graphen durch den Menupunkt *Graph* → *create* (s. Abbildung A.2) oder ob er geladen wird. Es können also auch bereits bestehende Graphen benutzt werden, die im GraphWin-Format abgespeichert sind.

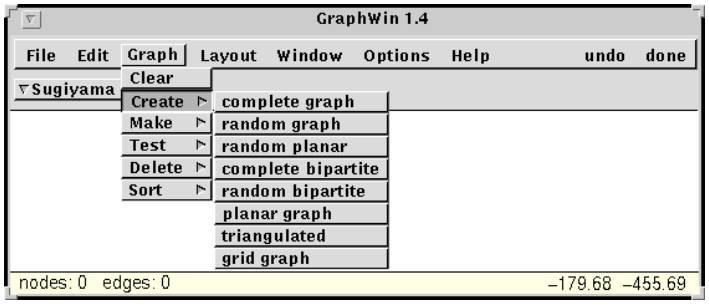


Abbildung A.2: Möglichkeiten zur Grapherzeugung mittels Menupunkt **create**

Ferner ist bereits eine Version des Algorithmus von Sugiyama implementiert, die unter *Layout* → *AGD − Algorithms* → *Sugiyama* (s. Abbildung A.3) zu finden ist.

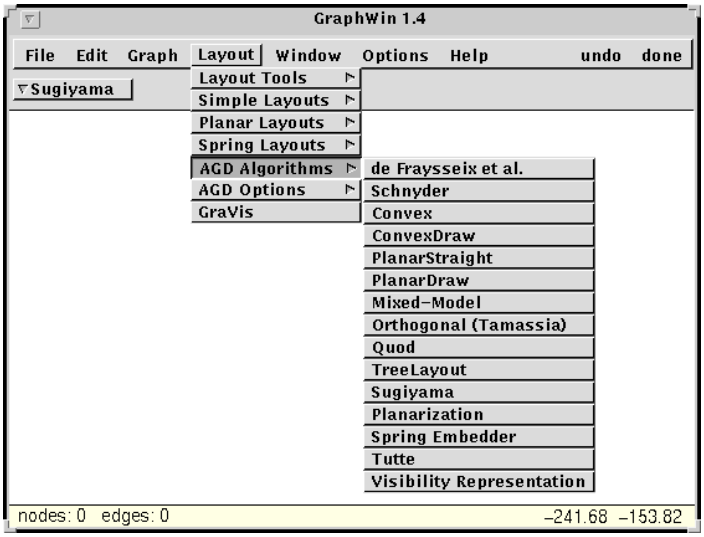


Abbildung A.3: Bereits implementierte AGD-Algorithmen, u.a. Algorithmus von Sugiyama

Unter dem Menupunkt *Window* gibt es unter anderem Möglichkeiten, den aktuell angezeigten Graphen zu vergrößeren oder zu verkleinern. Mittels *zoom to graph* wird eine Darstellung erreicht, die das komplette GraphWinfenster füllt. Probleme kann es geben, falls die Koordinaten des Graphen zu groß sind. Dann muß durch *zoom to area* ein Ausschnitt gewählt werden, der dann das komplette Fenster füllt. Diese Problem beruhen auf der Implementation der Klasse `GraphWin`.

A.3.1 Spezielle Optionen des Hauptprogrammes

Das Programm `sugi_menu` bietet einige Erweiterungen zum ursprünglichen `GraphWin`. Das Menu *Sugiyama*, sichtbar im Kopfbereich des GraphWin-Fensters, enthält drei Unterpunkte, *Layout, Options* und *Scale*, deren Bedeutung nun genauer erklärt wird.

Layout Die Benutzung von *Layout* startet die Berechnung eines Layouts unter den in *Options* voreingestellten Bedingungen. Werden die Wahlmöglichkeiten in *Options* nicht genutzt, wird die *Default*-Einstellung zur Layoutberechnung zugrunde gelegt. Dies bedeutet, die Hierarchie wird anhand der Statuswerte augebaut, die Hauptrichtung der Kanten ist von unten nach oben, der Faktor $1/a$ wird automatisch berechnet und eine Clusterung wird

- falls möglich - durchgeführt. Näheres zu den möglichen Voreinstellungen findet sich im folgenden Paragraphen.

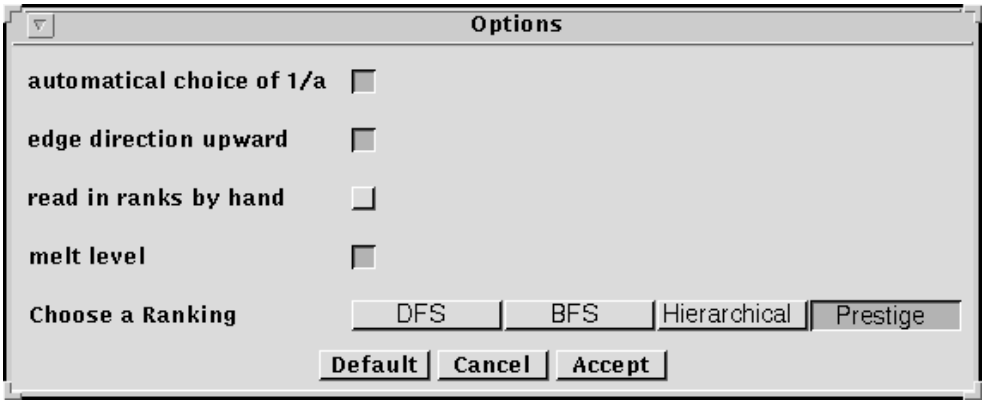


Abbildung A.4: Das Menu Options

Options Der Menupunkt *Options* bietet die Möglichkeit, die Berechnung des Layouts zu beeinflußen. Die Wahl des Faktors $1/a$ kann automatisch erfolgen oder durch Eingabe von Hand. Wichtig hierbei ist, daß je nach Größe des Faktors die Zuweisung der Level für die Knoten variieren kann. Ein gutes Beispiel dafür ist der Graph 5.6. Ändert man den Wert $1/a$ von 2 auf 1, ist der Akteur mit dem höchsten Status nicht mehr Akteur F, sondern Akteur A. Die Eingabe des Faktors von Hand erfolgt nach Aufruf des *Layout* in einem separaten Fenster.
Mit der Option *edge direction upward* kann der Aufbau der Hierarchie beeinflußt werden, d.h. die Hauptrichtung der Kanten. Verlaufen die Kanten im originalen Sugiyamaalgorithmus - soweit möglich - von oben nach unten, ist in diesem Layout das Gegenteil der Fall. Die wichtigsten Akteure sitzen oben im Layout, die Hauptrichtung der Kanten ist somit *bottom-up*, also von unten nach oben.
Sollte es erforderlich sein, die Ränge der Knoten von Hand über Standard-Input einzugeben, kann dies mittels der Option *read in ranks by hand* angegeben werden. Der Benutzer wird dann nach dem Starten des *Layout* jeweils nach der Nummer des Knotens und dem zugehörigen Rang gefragt.
Durch die Option *Choose a Ranking* kann die Art der Bestimmung für eine Hierarchie angegeben werden. Die Möglichkeiten bestehen in der Verwendung einer *Tiefensuche*, einer *Breitensuche*, der von der **AGD** empfohlenen Bestimmung der Hierarchie für ein `SugiyamaLayout` oder der speziell für dieses Problem implementierten Bestimmung der *Prestige*-Werte. Der Button

melt_level bestimmt, ob eine Clusterung durchgeführt werden soll oder nicht. Die Defaulteinstellung empfiehlt eine Clusterung.

Scale *Scale* bietet zwei Möglichkeiten der Skalierung, Skalierung des *gesamten Graphen*, also der Positionen, Knoten und Kanten und Skalierung der *Knoten*. Da die berechneten Werte für den *Status* ursprünglich sehr hoch sein können und nach der Normierung im Bereich zwischen 0 und 500 liegen, kann es sein, daß die Knoten durch diese Normierung so klein gemacht werden, daß sie nicht mehr oder nur sehr schlecht zu erkennen sind. Dies kann mittels einer Veränderung der Breite und Höhe verbessert werden. Werden Breite und Höhe nicht im selben Maße verändert, kann unter Umständen eine Eigenschaft verloren gehen. Spiegelt sich im Verhältnis Höhe zu Breite das Verhältnis zwischen ein- und auslaufenden Kanten wieder, gilt dies nach einseitiger Veränderung der Daten nicht mehr.

Anhang B

Graphische Beispiele

Die hier dargestellten Beispiele stammen alle aus dem Gebiet der sozialen Netzwerke. Auf die Beschriftung der Akteure wurde verzichtet, da sich diese Arbeit mit der Positionierung der Akteure, nicht mit den Akteuren selbst beschäftigt und da somit eine bessere Lesbarkeit erreicht wird.
Die Abbildungen B.1, B.2 und B.3 zeigen drei verschiedene Möglichkeiten der Visualisierung ein und desselben Graphen. Abbildung B.1 zeigt die Akteure in einem Kreisdiagramm, für das Layout von Abbildung B.2 wurde ein 2D-Springembedder benutzt und Abbildung B.3 zeigt schließlich eine Visualisierung mittels der in der Arbeit beschriebenen Methode zur Darstellung von Status unter Zuhilfenahme des Algorithmus von Sugiyama. Benutzt werden hierbei die *Statuswerte* zur Bestimmung der vertikalen Koordinaten, eine *Clusterung* zum Verschmelzen der Layer, die Kombination von *Barycenter*-Heuristik und *Global Sifting*, das Einfügen neuer *Dummyknoten* aufgrund der Clusterung und das Setzen der horizontalen Koordinaten, beschrieben in den Klassen `MyHierarchicalLayout` und `FastHierarchyLayout`.
Im Anschluß daran folgt eine Reihe von weiteren Bildern, die alle auf dieselbe Weise wie Abbildung B.3 erzeugt wurden.
Einige der hier dargestellten Graphen haben außerdem die Eigenschaft, durch das Verhältnis von Höhe und Breite eines Knotens das Verhältnis von ein- und auslaufenden Kanten wiederzuspiegeln (siehe auch Kapitel 4, Abschnitt 4.3).

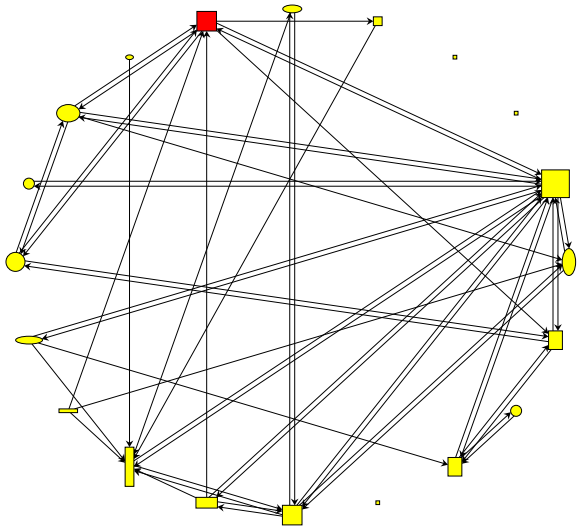


Abbildung B.1: Layout mittels eines Kreisdiagrammes

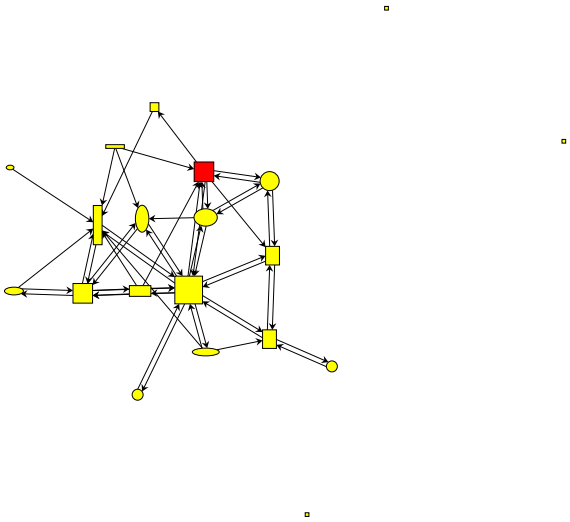


Abbildung B.2: Layout mittels eines 2D-Springembedders

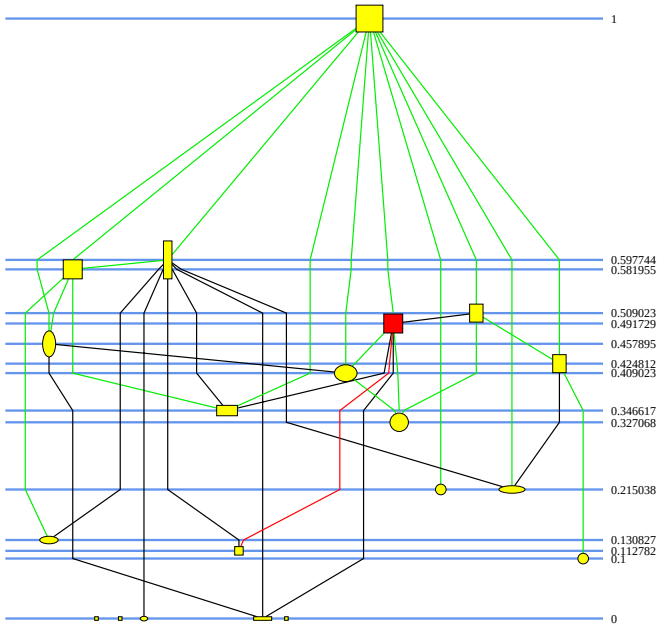


Abbildung B.3: Layout mittels Sugiyama-Algorithmus

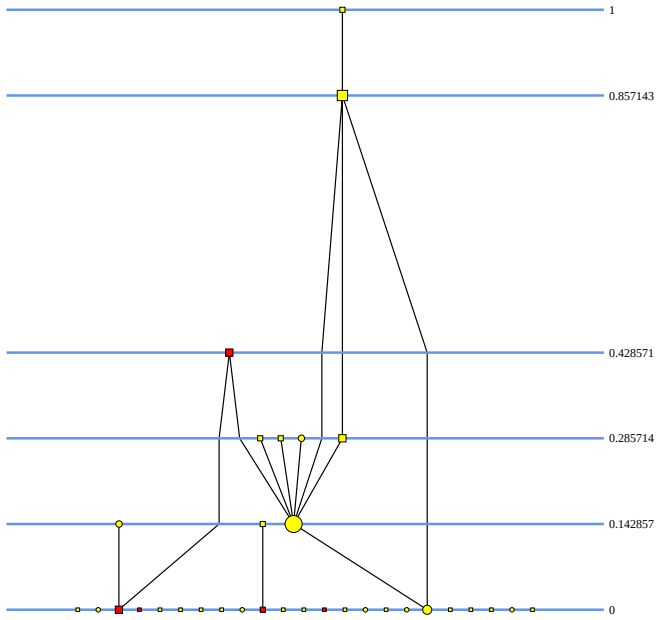


Abbildung B.4:

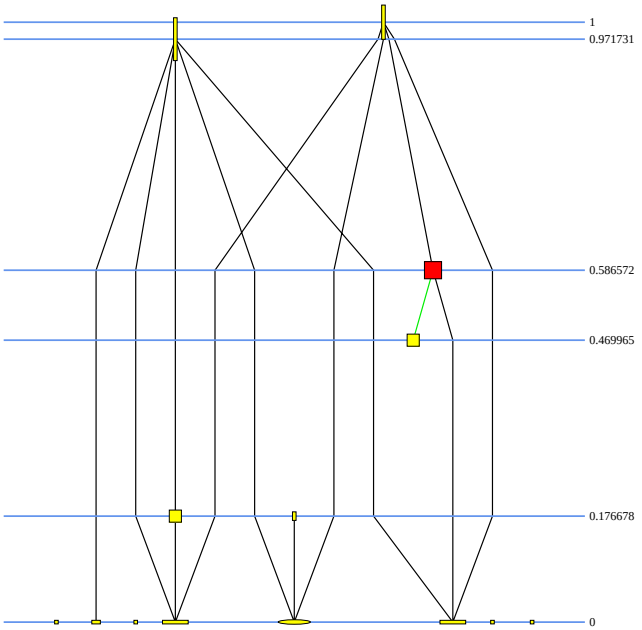


Abbildung B.5:

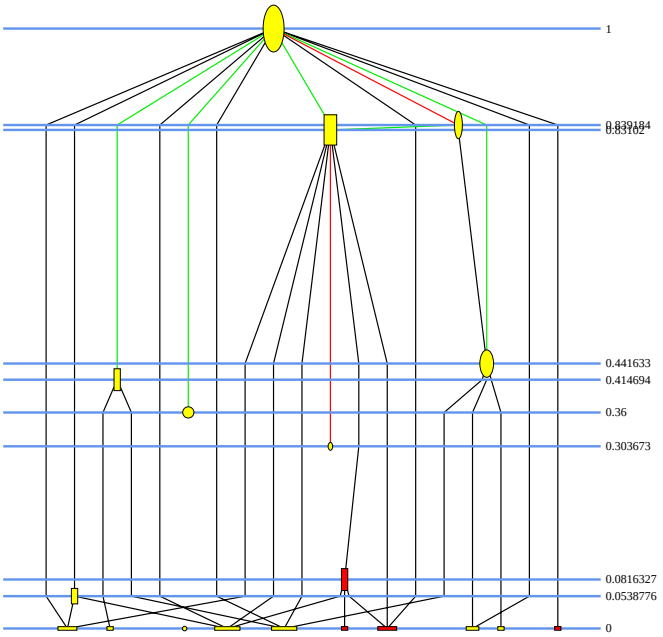


Abbildung B.6:

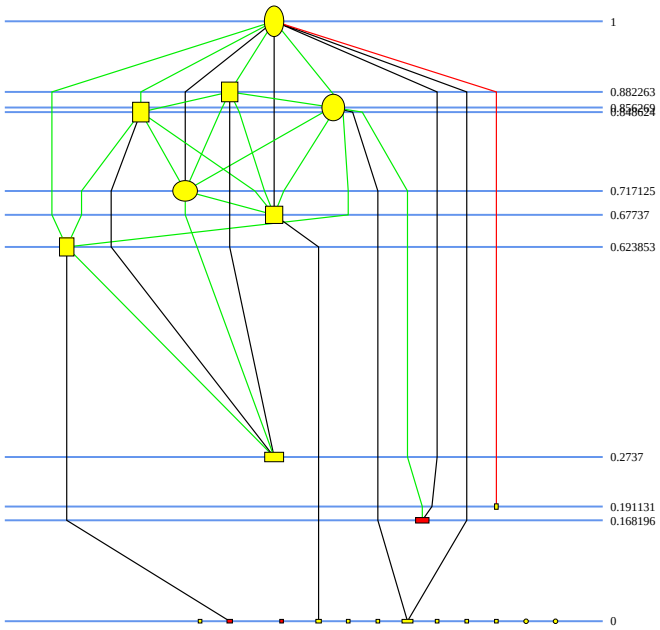


Abbildung B.7:

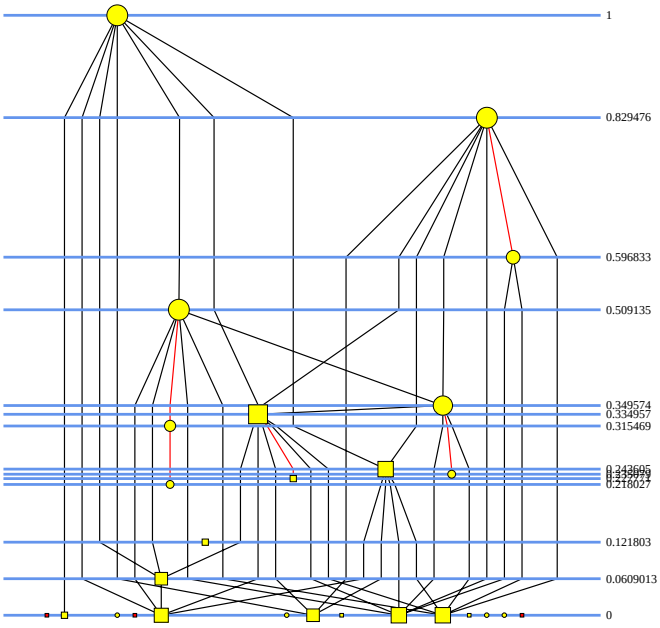


Abbildung B.8:

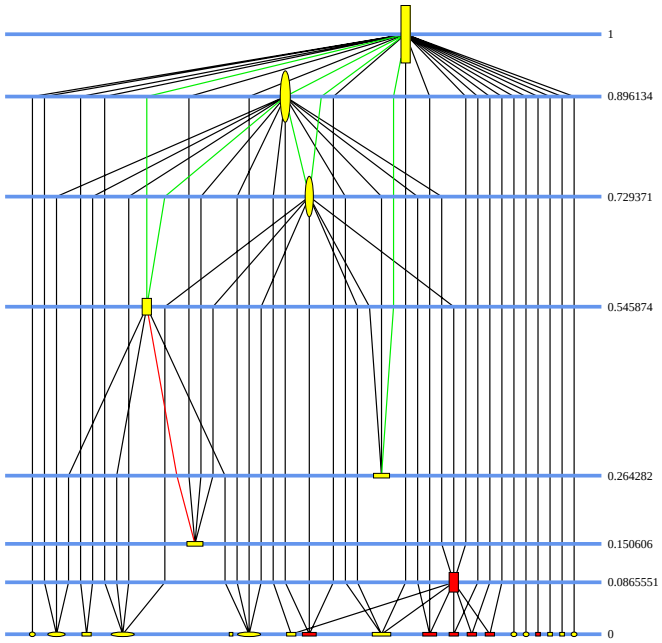


Abbildung B.9:

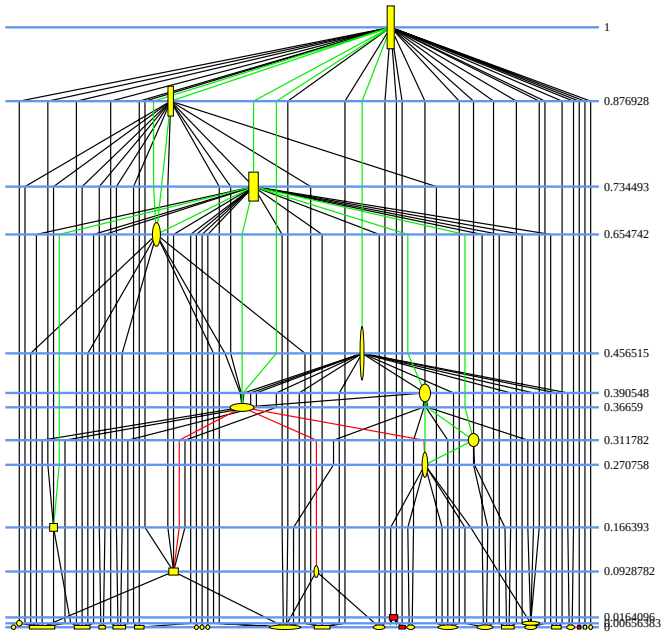


Abbildung B.10:

Literaturverzeichnis

[AHU83] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms*. Addison–Wesley, 1983.

[Ant71] J.M. Anthonisse. *The Rush in a Graph*. Mathematische Centrum, 1971.

[Bar69] J.A. Barnes. Graph theory and social networks: A technical comment on connectedness and connectivity. *Sociology*, 3:215–232, 1969.

[Bav48] A. Bavelas. A mathematical model for group structure. *Human Organizations*, 7:16–30, 1948.

[Ber83] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.

[BJL99] Christoph Buchheim, Michael Jünger, and Sebastian Leipert. A fast layout algorithm for k-level graphs. Technical report, Institut für Informatik, 1999.

[BKR+99] Ulrik Brandes, Patrick Kenis, Jörg Raab, , and Dorothea Wagner. Explorations into the visualization of policy networks. *Journal of Theoretical Politics*, 11(1):75–106, 1999.

[Bla77] P.M. Blau. *Inequality and Herterogeneity*. Free Press, 1977.

[Bot57] E. Bott. *Family and Social Network*. Tavistock, 1957.

[Bra99] Ulrik Brandes. *Layout of Graph Visualizations*. PhD thesis, Universität Konstanz, 1999.

[Cat95] T. Catarci. The assignment heuristic for crossing reduction. *IEEE Transactions on Systems, Man and Cybernetics*, 25(3):515–521, 1995.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. The MIT Press and McGraw-Hill Book Company, 1990.

[DETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[Dre95] S. Dresbach. A new heuristic layout algorithm for directed acyclic graphs. *Operations Research Proceedings*, pages 121–126, 1995.

[EK86] Peter Eades and D. Kelly. Heuristics for reducing crossings in 2-layered networks. *Ars Combin.*, 21:89–98, 1986.

[ES90] Peter Eades and Kozo Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 13:424–437, 1990.

[FK46] E. Forsyth and Leo Katz. A matrix approach to the analysis of sociometric data: Preliminary report. *Sociometry*, 9:340–347, 1946.

[Fre79] Linton C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1:215–239, 1979.

[Fre80] Linton C. Freeman. The gatekeeper, pair-dependency, and structural centrality. *Quality and Quantity*, 14:585–592, 1980.

[Fri96] Arne Frick. Upper bounds on the number of hidden nodes in sugiyama's algorithm. [Nor96], pages 169–183.

[Gal85] J. Galaskiewicz. *Social Organization of an Urban Grants Economy*. Academic Press, 1985.

[GJ83] Michael R. Garey and David S. Johnson. Crossing number is $\mathcal{NP}$–complete. *SIAM*, 4(3):312–316, 1983.

[GJ91] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$–Completeness*. W.H. Freeman & Co, 1991.

[GKNV] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and K.P. Vo. A technique for drawing directed graphs.

[GKNV93] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and K.P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, 1993.

[Gou87] R.V. Gould. Measures of betwwenness in non-symmetric networks. *Social Networks*, 9:277–282, 1987.

[GW89] J. Galaskiewicz and Stanley Wasserman. Mimetic and normative processes within an interorganizational field: An empirical test. *Administrative Science Quarterly*, 34:454–480, 1989.

[Har59] Frank Harary. Status and contrastatus. *Sociometry*, 22:23–43, 1959.

[HNC65] Frank Harary, R. Z. Norman, and D. Cartwright. *Structural Models: An Introduction to the Theory of Directed Graphs*. John Wiley and Sons, 1965.

[JM97] Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal on Graph Algorithms and Applications*, 1(1), 1997.

[Jor69] C. Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 70:185–190, 1869.

[Kar72] R. Karp. Reducibility among combinatorical problems. *Plenum Press*, pages 85–103, 1972.

[Kat53] Leo Katz. A new status index derived from sociometrix analysis. *Psychometrika*, 18:39–43, 1953.

[KB83] David Knoke and Ronald S. Burt. Prominence. *Applied Network Analysis*, pages 195–222, 1983.

[Ken78] D. Kent. *The Rise of the Medici: Faction in Florence, 1426 - 1434*. Oxford University Press, 1978.

[Klo81] Alden S. Klovdahl. A note on images of networks. *Social Networks*, 3:197–214, 1981.

[LG66] Edward O. Laumann and L. Guttman. The relative associational contiguity of occupations in an urban setting. *American Sociological Review*, 31:169, 1966.

[Lin92] X. Lin. *Analysis of Algorithms for Drawing Graphs*. PhD thesis, University of Queensland, 1992.

[LK87] Edward O. Laumann and David Knoke. *The Organizational State: Social Choice in National Policy Domains*. University of Wisconsin Press, 1987.

[LP76] Edward O. Laumann and F. Pappi. *Networks of Collective Action: A Perspective on Community Influence Systems*. Academic Press, 1976.

[LS38] George A. Lundberg and M. Steele. Social attraction patterns in a village. *Sociometry*, 1:327–419, 1938.

[LS77] S. Lam and R. Sethi. Worst case analysis of two scheduling problems. *SIAM Journal on Computing*, 6:518–536, 1977.

[Mäk90] Erkki Mäkinen. Experiments on drawing 2-level hierarchical graphs. *Internat. J. Comput. Math.*, 36:175–181, 1990.

[MMS] C. Matuszewski, P. Molitor, and R. Schönfeld. Using sifting for k-layer straightline crossing minimization.

[MNF90] S. Masuda, K. Nakajiama, , and T. Fujisawa. Crossing minimization in linear embeddings of graphs. 39(1):124–127, 1990.

[Mor53] Jakob L. Moreno. *Who Shall Survive: Foundations of Sociometry, Group Psychotherapy, and Sociodrama*. Beacon House, 1953.

[Mut96] Petra Mutzel. An alternative method to crossing minimization on hierarchical graphs. In North [Nor96], pages 318–333.

[Nor40] Mary L. Northway. A method for depicting social relationships obtained by sociometric testing. *Sociometry*, 3:144–150, 1940.

[Nor96] Stephen C. North, editor. *Proceedings of the 4th International Symposium on Graph Drawing (GD '96)*, volume 1190 of *Lecture Notes in Computer Science*. Springer, 1996.

[PA93] John F. Padgett and Christopher K. Ansell. Robust action and the rise of the Medici. 1400–1434. *American Journal of Sociology*, 98(6):1259–1319, 1993.

[Pit79] F.R. Pitts. The medieval river trade network of russia revisited. *Social Networks*, 1:285–292, 1979.

[RD39] Fritz J. Roethlisberger and William J. Dickson. *Management and the Worker*. Harvard University Press, 1939.

[Rud93] Richard Rudell. Dynamic variable ordering for ordered binary decision diagrams. *Proc. International Conf. on Computer Aided Design (ICCAD)*, pages 42–47, November 1993.

[Sha54] M.E. Shaw. Group structure and the behavior of individuals in small groups. *Journal of Psychology*, 38:139–149, 1954.

[Shi53] A. Shimbel. Structural parameters of communication networks. *Bulletin of Mathematical Biophysics*, 15:501–507, 1953.

[Sni81a] T.A.B. Snijders. The degree variance: An index of graph heterogeneity. *Social Networks*, 3:163–174, 1981.

[Sni81b] T.A.B. Snijders. Maximum value and null moments of the degree variance. Technical Report TW-report 229, Department of Mathematics, University of Groningen, 1981.

[STT81] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, February 1981.

[Syl82] J.J. Sylvester. On the geometrical forms called trees. *Johns Hopkins University Circle*, 1:201–202, 1882.

[SZ89] K. Stevenson and M. Zelen. Rethinking centrality: Methods and applications. *Social Networks*, 11:1–37, 1989.

[Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[WF94] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

Index

2-Layer-Crossing-Minimization, 70

Adjacent-Exchange-Algorithmus, 73
Adjazenzmatrix, 43
affiliation network, 9
AGD, 98, 99
Akteur, 8
Assignment Heuristik, 82

Barycenter, 74
Beziehungen, 9

Clusterung, 67, 104, 106
Coffman-Graham-Layering, 60
Cycle Removal, 55

Denominator, 15
Dummyknoten, 56
Dyade, 10

Global Sifting, 86, 105
Graph, 4
graph, 99
GraphCopy, 99
GraphWin, 99
Greedy-Insertion-Algorithmus, 82
Gruppe, 10

Hawthorne-Studie, 40
Hierarchie, 56

Integer-Programmierung, 62

Koordinatenzuweisung, 91, 106
Kreuzungsmatrix, 71
Kreuzungsminimierung, 70, 104
    k-Schicht-, 85
    optimale, 83
Kreuzungszahl, 71

Lady Bountiful Diagramm, 41, 42
LAPACK, 98, 100
Layer-by-layer-sweep, 70
Layerzuweisung, 55, 103
LEDA, 98
Longest-Path-Layering, 58

Median, 75
    Average-, 80
    Semi-, 80
Multi-Prozessor-Scheduling, 59

Numerator, 15

One Mode Network, 9
One-Sided-Crossing-Minimization, 70

Planarisation, 84
Prestige, 14
    Akteur-Degree-, 34
    Akteur-Proximity-, 35
    Akteurprestige, 14
    Gruppen-Degree-, 35
    Gruppen-Proximity-, 36
    Gruppenprestige, 15
    Status, 37

Relation, 11

Schichtung, 56
Sifting, 81
Sink, 56

Source, 56
Soziales Netzwerk, 11
Soziogramm, 7
Soziometrie, 7
Split-Algorithmus, 73
Spring-Embedder, 64
Status, 37
Stochastische Heuristik, 82
Sugiyama-Algorithmus, 54

Triade, 10
Two Mode Network, 9

Untergruppe, 10

Visualisierung, 40
    Algorithmus, 51
    Animation, 49
    Design, 47
    Layoutspezifikation, 49
    Qualitat, 45
    Renderingspezifikation, 49
    Substanz, 46

Zentralitat, 14
    Akteur-Betweenness-, 28
    Akteur-Closeness-, 22
    Akteur-Degree-, 17
    Akteur-Information-, 32
    Akteurzentralitat, 14
    Allgemeiner Gruppenindex, 15
    Betweenness-, 27
    Closeness-, 21
    Degree-, 16
    Gruppen-Closeness-, 24
    Gruppen-Degree-, 18
    Gruppen-Information-, 33
    Gruppenzentralitat, 15
    Information-, 32