Zuerst ersch. in: it - Information Technology 46 (2004), 3, pp. 129-134

Netzwerkvisualisierung

Network Visualization

Ulrik Brandes, Universität Konstanz, Dorothea Wagner, Universität Karlsruhe

Zusammenfassung Das Thema Netzwerkvisualisierung umfasst alle Aspekte der Darstellung relationaler Strukturen. Die automatische Erzeugung von Netzwerkvisualisierungen hat wichtige Anwendungen nicht nur innerhalb der Informatik, sondern generell in allen Bereichen, in denen graphische Datenanalyse oder visuelle Informationsvermittlung eine Rolle spielen. Als eigenständiges Forschungsgebiet der Informatik ist die Netzwerkvisualisierung – oft auch als *Graphenzeichnen* bezeichnet – seit etwa 10 Jahren etabliert. Dazu gehören neben Algorithmentheorie und Algorithmen Engineering insbesondere Fragestellungen aus Graphentheorie und Kombinatorik, Modellierungsfragen und die Entwicklung von Software-Systemen. Dieser Artikel gibt einen Überblick über die wichtigsten methodischen Errungenschaften des Gebiets. ►►► **Summary** Network visualization deals with all aspects of representing relational structures. The automatic generation of network visualizations is of relevance not only in computer science, but in virtually every area concerned with graphical data analysis or visual communication of information. For more than 10 years, network visualization – also known as *graph drawing* – is an established research field of its own. It encompasses design and analysis of algorithms and algorithm engineering, as well as modelling aspects, topics from graph theory and combinatorics, and the development of sofware tools. This article gives an overview of the main methodological contributions of the field.

KEYWORDS I.3 [Computer Graphics] Netzwerkvisualisierung, Graphenzeichnen, Graph Drawing

1 Einleitung

Ein Schlagwort der heutigen Zeit lautet „Vernetzung". Im Zuge der wachsenden Bedeutung von Kommunikation und Informationsverbreitung sowie der zunehmenden Wirtschaftsbeziehungen und Verflechtungen in Politik und Gesellschaft treten relationale Strukturen immer mehr in den Vordergrund unserer Wahrnehmung. Entsprechend ist die Visualisierung von Netzwerken als Mittel der Analyse und des Verständnisses dieser Strukturen ein Gebiet mit breiter Relevanz.

Standardbeispiele von Netzwerkdarstellungen, die den Bedarf an automatischen Verfahren zur Visualisierung deutlich werden lassen, sind UML-Diagramme, wie sie etwa in der Softwaretechnik benutzt werden, Diagramme biochemischer Reaktionsketten, Abbildungen der Internet-Topologie, Familienstammbäume oder Repräsentationen von Sozialstrukturen (siehe Bild 1).

Allein schon aufgrund der Größe derartiger Netzwerke ist die weitgehend automatische Erzeugung ihrer Visualisierungen unabdingbar. Sollen diese zudem zum Verständnis der Daten beitragen, oder gar den Zweck der objektiven Darstellung erfüllen, wird die systematische Erstellung zu einer komplexen Aufgabe aus einer Fülle von unterschiedlichen Aspekten.

Zunächst ist die Wahl der graphischen *Repräsentation*, d. h. der Darstellungsform für die Knoten und Kanten des die Netzwerkstruktur beschreibenden Graphen, abhängig von Herkunft und Eigenschaften der Netzwerkdaten. Die vermutlich gebräuchlichste Repräsentationsform bildet Knoten auf graphische Symbole und Kanten auf gerade, gebogene oder rechtwinklig verlaufende Linien, welche die zugehörigen Knoten verbinden, ab. Je nach Struktur des Graphen sind jedoch auch Repräsentationen, bei denen die Relationen nicht durch Linien, sondern zum Beispiel durch Umschließung oder Berührung der Knotensymbole ausgedrückt werden, möglich und geeignet. Nebenbedingungen an die Darstellung betreffen etwa die Richtung der Kanten, absolute oder relative Einschränkungen an die Knotenpositionen oder Einschränkungen an die Positionierung von Teilnetzen.

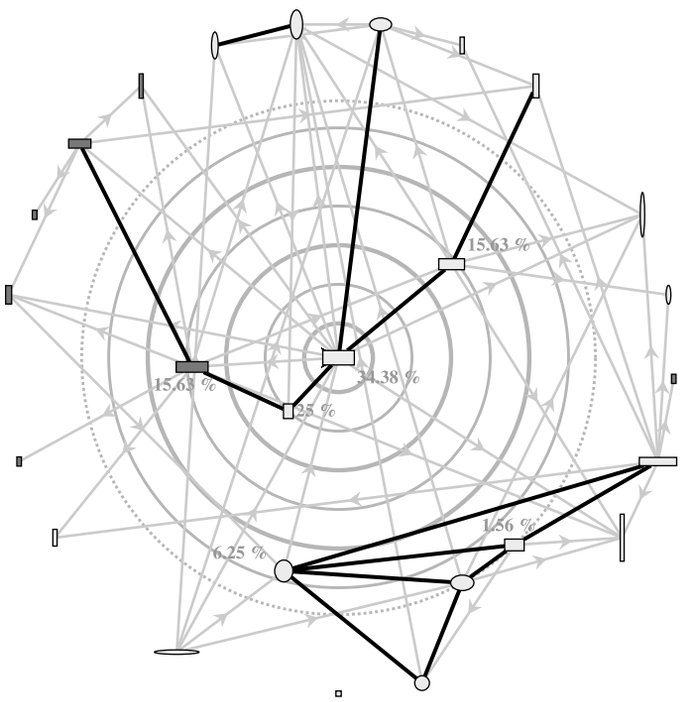

Bild 1 Zentrale Akteure einer Sozialstruktur.

Welche Darstellungen zulässig sind, ist durch die Repräsentationsform und zusätzliche *Nebenbedingungen* festgelegt. Typischerweise ist die Erstellung einer zulässigen Darstellung dann noch verbunden mit der Optimierung gewisser für die Lesbarkeit oder Ästhetik der Visualisierung relevanter *Kriterien*, etwa die Anzahl der Kantenkreuzungen, die Anzahl der Kantenknicke, die Varianz der Kantenlängen oder die Größe von Winkeln zwischen Kanten.

Repräsentationsform, Nebenbedingungen und Optimierungskriterien bestimmen den algorithmischen Kern der Visualisierungsaufgabe. Damit tritt bei der systematischen Darstellung eines Netzwerks die Erstellung des *Layouts*, d. h. der zwei- oder mehrdimensionalen Einbettung der Graphenstruktur in den Vordergrund. Im Allgemeinen ergeben sich sehr komplexe Optimierungsprobleme, die meist sogar $\mathcal{NP}$-schwer sind. Die Wahl von Farben, Formen, etc. ist dagegen zwar für die jeweilige Anwendung wichtig, für die algorithmische Lösung jedoch unerheblich.

In der Informatik beziehungsweise Mathematik ist die grundlegende Fragestellung „Wie zeichnet man einen Graphen?“ bereits Mitte des letzten Jahrhunderts untersucht worden.[1] Mit einer jährlichen internationalen Tagung zum Thema ist „Graphenzeichnen“ seit etwa 10 Jahren als eigenständiges Forschungsgebiet der Informatik etabliert.[2]

Die bedeutenden Errungenschaften dieses Gebiets bestehen in der Entwicklung methodischer Vorgehensweisen zur Netzwerkvisualisierung. Darauf basierend sind nicht nur eine Fülle von Algorithmen zu ausgewählten Aufgabestellungen entstanden, sondern auch eine Reihe von vornehmlich in Forschungsinstitutionen entwickelten Software-Bibliotheken.

Wir geben hier einen Überblick über die wichtigsten methodischen Ergebnisse des Graphenzeichnens. Dabei beschränken wir uns auf die algorithmische Kernaufgabe der Netzwerkvisualisierung, die Erstellung eines Layouts. Als weiterführende Literatur empfehlen wir die im Literaturverzeichnis angegebenen Bücher.

[1] „How to Draw a Graph“ überschrieb denn auch William T. Tutte einen Aufsatz von 1963, der heute zu den Klassikern der Graphentheorie gehört. Im gleichen Jahr veröffentlichte Donald E. Knuth übrigens „Computer Drawn Flowcharts“ in *Communications of the ACM*.

[2] Die Proceedings des *International Symposium on Graph Drawing* erscheinen in der Springer-Reihe LNCS.

2 Graphenlayout

Ein *Layout* eines Graphen besteht in der Festlegung topologischer und geometrischer Eigenschaften der graphischen Repräsentation, insbesondere also in der Festlegung von Positionen der Layoutelemente dieser Repräsentation. Zum Beispiel ist bei einer geradlinigen Repräsentation das gesamte Layout bereits durch die Knotenpositionen festgelegt. Die Eigenschaften können durch entsprechende Variablen, Nebenbedingungen an das Layout durch die Angabe von Wertebereichen und zulässigen Kombinationen, und Optimierungskriterien durch entsprechende Zielfunktionen ausgedrückt werden.

Im Folgenden skizzieren wir die wichtigsten Klassen von Methoden zur Erzeugung von Layouts unter idealisierten Bedingungen, die den methodischen Kern spezieller Algorithmen bilden. Knotengrößen, Kantendicken oder die Anbringung von Labeln an Knoten oder Kanten werden dabei zunächst ignoriert. Die Eignung dieser Methoden hängt vom jeweiligen Anwendungsszenario ab. Neben der Vorgehensweise und ihrer Anwendbarkeit selbst sind dabei vor allem die Beweisbarkeit der Qualität der Layouts und die Effizienz der Verfahren Unterscheidungsmerkmale.

2.1 Globale Optimierung

Als universellste Layoutaufgabe kann die Erzeugung eines Graphenlayouts angesehen werden, das für einen Graphen, über den keine zusätzlichen strukturellen Informationen vorliegen, eine globale Zielfunktion optimiert. Die prototypische Methode für diese Situation sind so genannte *Spring Embedder*. Deren Grundidee besteht in der Interpretation eines global optimalen

Layouts als ein System interagierender physikalischer Objekte, die sich im Gleichgewicht befinden; Erweiterungen verwenden zusätzliche Einflüsse wie etwa Magnetfelder und Gravitation. Da es ohnehin schon schwierig ist, die globale Optimalität eines Layouts zu formalisieren, ist die Zielfunktion oft aus verschiedenen lokalen Optimierungskriterien zusammengesetzt, oder – wie im Fall der Spring Embedder – nur implizit gegeben.

Typische lokale Kriterien, die bei globalen Methoden für geradlinige Layouts explizit oder implizit zugrunde gelegt werden, sind eine gleichmäßige Verteilung der Knoten, uniforme Kantenlängen und Mindestabstände zwischen nichtverbundenen Knoten. In der physikalischen Analogie entsprechen die Knoten Objekten, die sich gegenseitig abstoßen, und die Kanten Federn, für die gewisse Stärken und Ideallängen festgeschrieben sind. Das Layout für Bild 2 wurde auf diese Weise erstellt.

Das Modell kann auf eine Vielzahl von Optimierungskriterien erweitert werden, beispielsweise auf die Vermeidung von Überlappungen oder Kreuzungsminimierung, und es ist für wesentlich allgemeinere Situationen wie etwa gerichtete Graphen, dreidimensionale Layouts oder auch zur Platzierung von Labeln anwendbar.

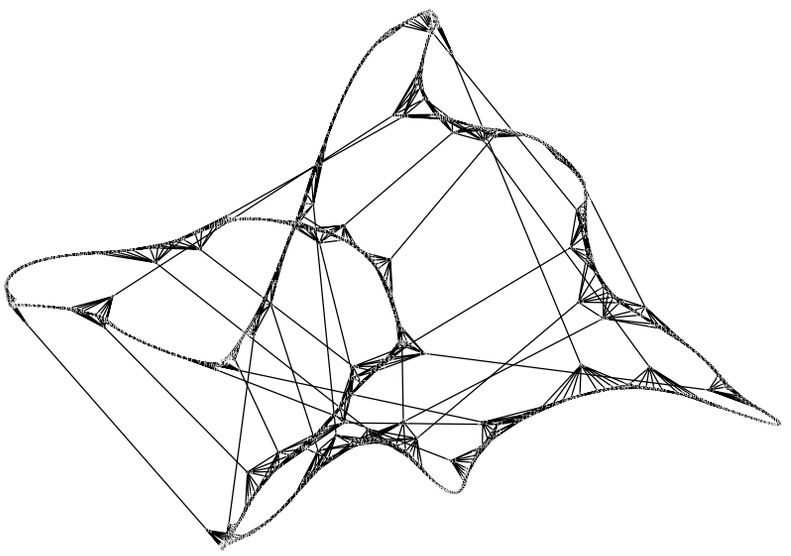

Bild 3 Zufallsgraph im „small world"-Modell.

Für die algorithmische Lösung können allgemeine Optimierungsverfahren wie Gradientenmethoden, Simulated Annealing oder evolutionäre Verfahren verwendet werden. Da es sich bei den betrachteten Zielfunktionen meistens um $\mathcal{NP}$-schwere Probleme handelt und es zudem schwierig ist, die Modelle und verwendeten Algorithmen analytisch zu untersuchen, fehlen zu den globalen Methoden theoretisch fundierte Qualitäts- und Laufzeitgarantien.

Interessanterweise kann das bereits 1963 von Tutte in dem Artikel „How to Draw a Graph" eingeführte Verfahren als ein Spezialfall des Spring Embedders interpretiert werden, bei dem jeder Knoten im Schwerpunkt seiner Nachbarn positioniert wird, indem den Federn die Ideallänge Null zugeordnet wird. Da dieses Modell durch gleiche Positionierung aller Knoten ein degeneriertes Optimum aufweist, werden die Positionen ausgewählter Knoten in Form einer Nebenbedingung festgelegt.

Die *Spektralmethode* ist eine Variante des Schwerpunktmodells, in der eine uniformere Nebenbedingung verwendet wird, um das unerwünschte Optimum auszuschliessen. Der Name rührt daher, dass die Lösung des Layoutproblems aus Eigenvektoren der einem Graphen zugeordneten Laplace'schen Matrix besteht. Spektrale Layouts geben Symmetrien sehr viel besser wieder als zum Beispiel Spring Embedder (siehe Bild 3), führen aber bei strukturell ungeeigneten Graphen zu einer sehr ungleichmäsigen Verteilung der Knoten.

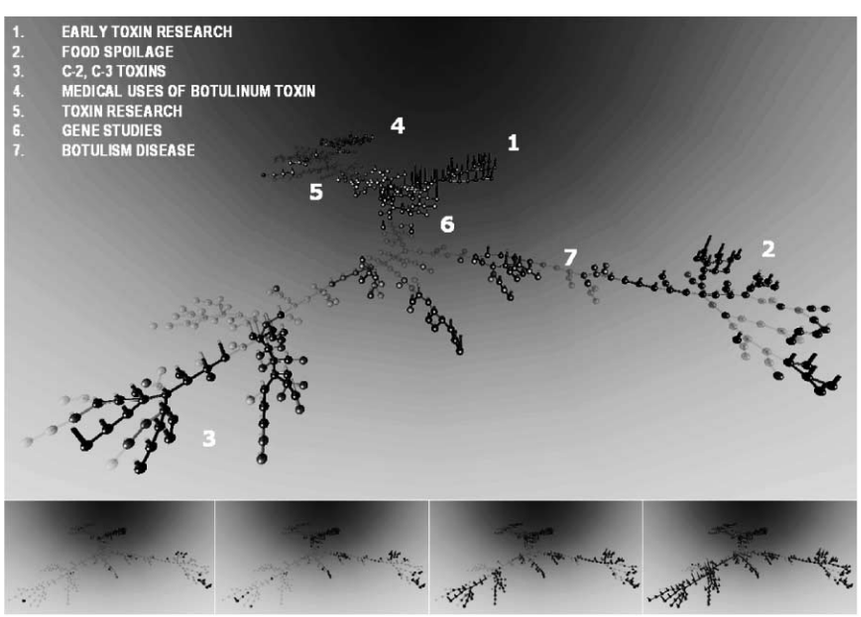

Bild 2 Entwicklung der Zitatstruktur eines Forschungsgebiets (mit freundlicher Genehmigung von Chaomei Chen).

2.2 Flussmethoden

Ein sehr mächtiges Instrument der Kombinatorischen Optimierung besteht in der Modellierung von Optimierungsaufgaben durch Flussprobleme. Grundlegende Idee der Anwendung von *Flussmethoden* beim Graphenzeichnen ist die Modellierung von Winkeln oder Knicken durch Flüsse in geeignet definierten Netzwerken. Beispielhaft erläutern wir dieses Konzept für die Knickminimierung in orthogonalen planaren Layouts.

Voraussetzung für das Flussmodell ist eine vorgegebene planare Einbettung des Graphen. Ziel ist eine Zeichnung des Graphen in ein orthogonales Gitter unter Beibehaltung der vorgegebenen Einbettung und mit minimaler Anzahl an Kantenknicken. Das zugehörige Flussnetzwerk besteht aus dem zur Einbettung gehörenden symmetrisch gerichteten Dualgraph, sowie gerichteten Kanten von jedem Graphknoten zu seinen angrenzenden Facettenknoten.

Wählt man Kapazitäten und andere Parameter des Flussnetzwerks geeignet, so kann ein ganzzahliger Fluss unmittelbar in eine orthogonale Einbettung überführt werden, indem jede Flusseinheit als 90°-Winkel an einem Knoten beziehungsweise 90°-Knick auf einer Kante interpretiert wird. Belegt man ferner die Flusseinheiten auf den Dualkanten mit Kosten, so induziert jeder Fluss mit minimalen Kosten eine orthogonale Einbettung mit minimaler Knickzahl.

Da ein Fluss mit minimalen Kosten in polynomialer Laufzeit berechnet werden kann, liefert die Methode einen effizienten Algorithmus, der jedoch nur für den skizzierten Spezialfall eines eingebetteten Graphen anwendbar ist. Das Potenzial der Methode liegt in der Erweiterbarkeit auf allgemeinere Varianten orthogonaler Layouts, etwa für nichtplanare Graphen oder die Kompaktifizierung (siehe auch Bild 4). Ähnliche Flussmodelle werden zum Beispiel für die Winkelauflösung in planaren geradlinigen Layouts und optimale aufwärtsgerichtete planare Zeichnungen von gerichteten Graphen verwendet.

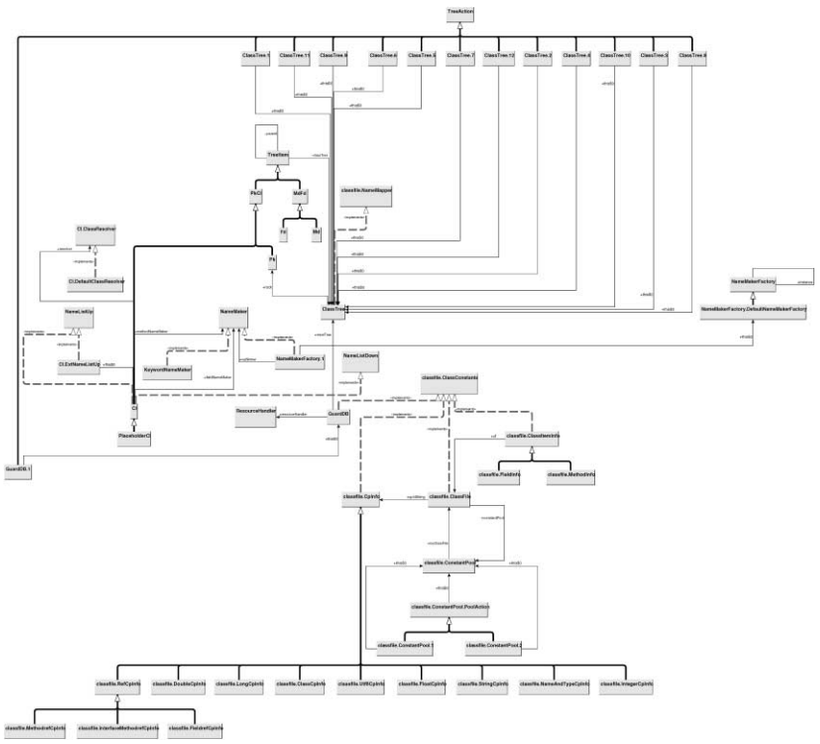

Bild 4 UML Klassendiagramm (mit freundlicher Genehmigung von Markus Eiglsperger)

2.3 Geschichtete Layouts

Geschichtete Layouts bestehen in einer Anordnung der Knoten des Graphen auf horizontale Linien und einem weitgehend vertikalen Verlauf der Kanten. Diese Darstellungsform ist vor allem für gerichtete Graphen und Graphen, deren Knoten in einer Hierarchie angeordnet sind, gebräuchlich.

Eine gleichmäßige Verteilung der Knoten, wenige Kantenkreuzungen und die Vermeidung sehr langer oder entgegen die Hierarchie gerichteter Kanten sind typische Optimierungskriterien für geschichtete Layouts. In diesem Sinne optimale Layouts zu erstellen ist schwierig, oft sogar unmöglich, da sich im Allgemeinen einige dieser Kriterien widersprechen.

Die Standardmethode geht schrittweise vor. Zunächst werden gerichtete Kreise entfernt, indem einige Kanten in ihrer Richtung umgedreht werden. Es wird dann eine Zuweisung der Knoten auf die Schichten entsprechend der durch die Kantenrichtungen induzierten Hierarchie der Knoten vorgenommen. Da durch eine solche Schichtzuweisung nicht garantiert werden kann, dass Kanten nur zwischen aufeinanderfolgenden Schichten verlaufen, werden in eine Kante für jede Schichtüberquerung Hilfsknoten eingefügt. Im nächsten Schritt wird durch Umordnungen von Knoten auf den Schichten eine Kreuzungsminimierung angestrebt. Der letzte Schritt besteht dann in der Zuweisung der x-Koordinaten der Knoten und der Festlegung eines überlappungsfreien Verlaufs der Kanten.

Diese Schritte erfordern wiederum die Lösung schwieriger Optimierungsprobleme, die teilweise schon in anderem Kontext betrachtet wurden und mit unterschiedlichen Techniken behandelt werden können. Beispielweise ist die Entfernung gerichteter Kreise äquivalent zu einem bekannten kombinatorischen Problem, dem linearen Anordnungsproblem, Optimierungskriterien für die Schichtzuwei-

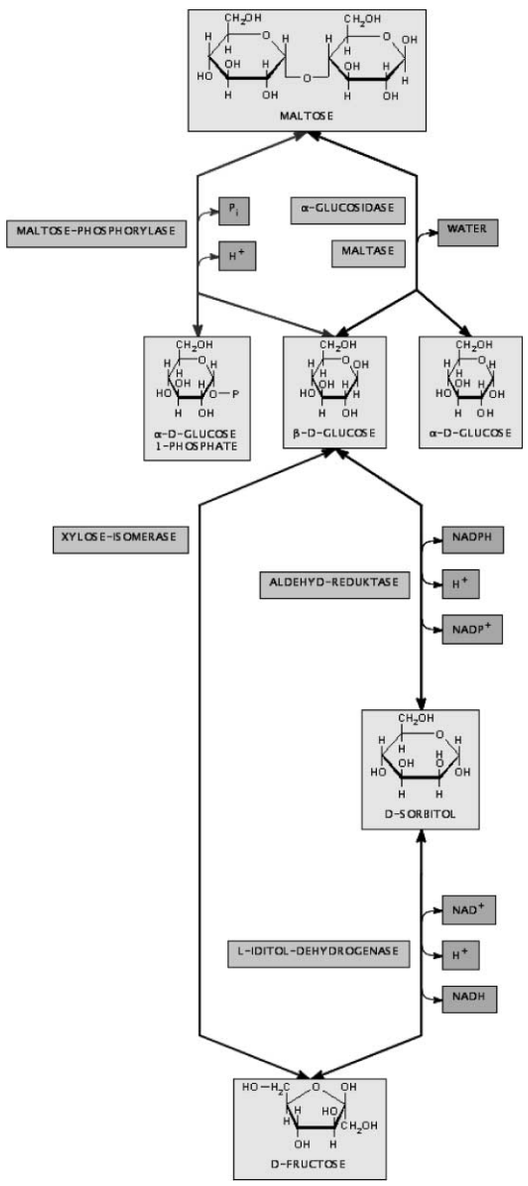

Bild 5 Biochemische Reaktionskette (mit freundlicher Genehmigung von Falk Schreiber)

sung wie die Minimierung der Layouthöhe ist eng verwandt zu Scheduling-Problemen und Verfahren zur Kreuzungsminimierung können beispielsweise von der Darstellung von Entscheidungsdiagrammen in der Logik-Synthese adaptiert werden. Bild 5 zeigt ein Beispiel eines mit dieser Methode gezeichneten Graphen.

2.4 Inkrementelle Methoden

Das inkrementelle Erstellen von Layouts durch schrittweises Hinzufügen von Teilgraphen oder sogar nur einzelner Knoten und Kanten ist insbesondere wegen seiner Anwendbarkeit auf Szenarien, in denen Interaktion ermöglicht werden soll, eine naheliegende Vorgehensweise. Gemeinsamkeit vieler inkrementeller Layoutalgorithmen für statische Graphen ist die Ausnutzung einer speziellen Knotenanordnung und die Benutzung einer geeigneten Repräsentation des bereits erzeugten Teillayouts.

Neben der globalen Optimierung lassen inkrementelle Verfahren den größten Spielraum für Repräsentationsform, Nebenbedingungen und Optimierungskriterien zu. Viele Verfahren für orthogonale Layouts, welche Gütegarantien für die Layoutfläche oder die Anzahl an Knicken ermöglichen, sind inkrementell. Eine ähnliche Beobachtung trifft für planare Layouts planarer Graphen und Layouts von Bäumen zu. Für weitere spezielle Graphenklassen, wie serien-parallele Graphen ist bereits aufgrund ihrer Definition die Anwendung eines inkrementelles Layoutverfahren naheliegend.

Grundlage beweisbarer Gütegarantien (hiermit sind in der Regel formale Kriterien wie Platzbedarf oder Anzahl der Kantenknicke gemeint) sind meistens die speziellen Eigenschaften der zugrundeliegenden Knotenanordnung. Die Berechnung der Knotenanordnung etwa mittels geeigneter Graphendurchlaufstrategien ist daher bereits für sich genommen ein interessanter Aspekt inkrementeller Verfahren.

3 Angewandte Netzwerkvisualisierung

Die beschriebenen prinzipiellen Ansätze müssen in der Regel für speziellere Situationen angepasst werden, beispielsweise durch geeignete Vor- und Nachbearbeitungsschritte. Dies kann insbesondere daran liegen, dass einige Algorithmen nur auf planare Graphen oder auf zweifach zusammenhängende Graphen anwendbar sind, oder dass orthogonale Layouts mit der beschriebenen Flussmethode zunächst nur für Graphen mit Maximalgrad vier erzeugt werden können.

Um benötigte Eigenschaften zu gewährleisten, werden vorab zum Beispiel Planarisierungsmethoden wie die Planarisierung durch Kantenwegnahme oder die Planarisierung durch Hinzufügen von Hilfsknoten, oder auch Ergänzen durch Kanteneinfügung oder Aufteilung von Knoten angewendet. Entsprechende Nachbearbeitungsschritte bestehen neben dem Wiederherstellen des Ausgangsgraphen durch Wiedereinfügen oder Entfernen von Kanten etwa aus der Kompaktifizierung von Layouts oder der Verbesserung von Kantenverläufen.

Auch Vor- und Nachbearbeitung führen wieder auf schwierige Optimierungsprobleme wie die Berechnung eines maximalen planaren Teilgraphen oder die Berechnung einer minimalen Ergänzung zu einem zweifach zusammenhängenden Graph.

Über das reine Graphenlayout hinaus bringen die vielfältigen Anwendungsbereiche von Netzwerkvisualisierungen natürlich noch eine Fülle weiterer Anforderungen mit sich, die, wie beispielsweise das Anbringen von Beschriftungen, noch unmittelbar mit den üblichen Methoden des Graphenzeichnen vereinbar sind, zunehmend jedoch zu anspruchsvolleren Problemstellungen führen.

Im Zuge der wachsenden Bedeutung der Visualisierung für die Datenanalyse werden Layoutmethoden für sehr große Graphen beziehungsweise für geclusterte Graphen benötigt. Hinzu kommen Interaktion durch Kollabieren und Expandieren von Teillayouts oder generell aufgrund von datenbezogenen Anforderungen. Damit verwandt ist die automatische Erstellung von Layouts für sich dynamisch verändernde Graphen sowie Graphenanimation.

4 Fazit

Eine wichtige Errungenschaft des Graphenzeichnens besteht in der Identifikation und Entwicklung mächtiger algorithmischer Schemata, die für immer neue anspruchsvolle Aufgaben instanziiert und weiterentwickelt werden können. In den letzten Jahren wurden die Forschungsaktivitäten bereits auf Themenbereiche wie Hierarchien, dynamische Graphen, Interaktion und sehr große Graphen ausgeweitet. Im Einzelfall sind dabei zwar meist neue, hochgradig schwierige Probleme zu lösen, die enstehenden Fragestellungen können jedoch häufig schon durch Modifikation und Anpassung der etablierten Methoden für die Anwendungszwecke zufriedenstellend behandelt werden.

Literatur

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

[2] Jünger, Michael and Petra Mutzel (eds.): *Graph Drawing Software*. Springer-Verlag, Mathematics and Visualization, 2003.

[3] Kamada, Tomihisa: *Visualizing Abstract Objects and Relations*. World Scientific, 1989.

[4] Kaufmann, Michael and Dorothea Wagner (eds.): *Drawing Graphs: Methods and Models*. Springer-Verlag, Lecture Notes in Computer Science, vol. 2025, 2001.

[5] Sugiyama, Kozo: *Graph Drawing and Applications for Software and Knowledge Engineers*. World Scientific, 2002.

1

2

1 Prof. Dr. Ulrik Brandes hat an der RWTH Aachen Informatik mit Nebenfach Mathematik studiert und 1994 mit dem Diplom abgeschlossen. Er hat 1999 an der Universität Konstanz über Netzwerkvisualisierung promoviert und sich 2002 ebenda für Informatik habilitiert. Von 2002–2003 war er C3-Professor für Algorithmik an der Universität Passau und ist seit Oktober 2003 C4-Professor für Praktische Informatik (Algorithmik) an der Universität Konstanz.
Adresse: Fachbereich Informatik & Informationswissenschaft, Universität Konstanz, Universitätsstrasse 10, 78464 Konstanz,
Tel.: +49-7531-8844-33,
Fax: +49-7531-8835-77,
E-Mail: Ulrik.Brandes@uni-konstanz.de

2 Prof. Dr. Dorothea Wagner hat an der RWTH Aachen studiert und 1983 mit dem Diplom in Mathematik abgeschlossen. In 1986 hat sie an der RWTH Aachen promoviert und sich 1992 an der TU Berlin habilitiert. Sie war 1994–2003 C4-Professorin für Praktische Informatik an der Universität Konstanz, und ist seit März 2003 C4-Professorin für Informatik an der Fakultät für Informatik der Universität Karlsruhe.
Adresse: Fakultät für Informatik, Universität Karlsruhe, Am Fasanengarten 5, 76131 Karlsruhe, Tel.: +49-721-6087-330,
Fax: +49-721-6084-211,
E-Mail: dwagner@ira.uka.de