Automatisches Layout von Diagrammen

Michael Jünger (Universität zu Köln)
Petra Mutzel (Technische Universität Wien)

9. Juni 2001

Zusammenfassung

Wir geben eine Einführung in das Gebiet des automatischen Zeichnens von Diagrammen und zeigen auf, wie mathematische und algorithmische Methoden des Operations Research dazu beitragen, qualitativ hochwertige Layouts praktische effizient zu erzeugen.

1. Einführung

Abbildung 1 zeigt ein Diagramm eines Datenmodells, das einem Buch über Datenbanken (Rauh und Stickel 1996) entnommen wurde.

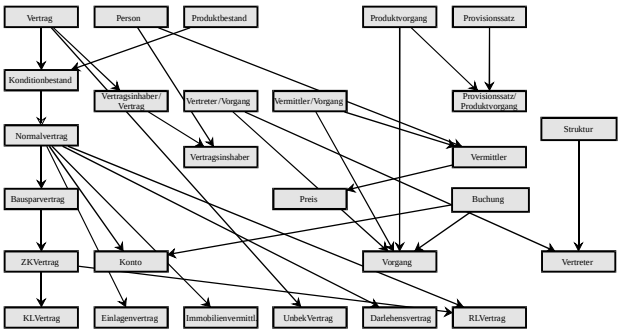


Abb. 1. Datenmodell: Originalzeichnung

Der Zweck solcher Diagramme ist, dem Betrachter den Überblick über die Beziehungen („Relations“) zwischen den Einheiten („Entities“) zu erleichtern. Die Zeichnung entstand durch Plazieren der rechteckigen Kästchen, die den Einheiten entsprechen, auf einem regelmäßigen Gitter und das Visualisieren der Beziehungen durch geradlinige Pfeile. Die abstrakte Struktur solcher Diagramme ist ein Graph mit *Knoten* und gerichteten oder ungerichteten *Kanten*, die Knotenpaare verbinden. Die Erstellung solcher Zeichnungen ist derzeit in der Regel ein zeitaufwändiger Prozess: Mit Hilfe einer Graphiksoftware werden die Positionen der Knoten und der Verlauf der Kanten von Hand bestimmt. Erscheint das Ergebnis nicht übersichtlich genug, werden Knoten verschoben und Kanten neu verlegt, bis keine Verbesserung mehr möglich oder in vertretbarer Zeit erreichbar erscheint. Die Zeichnung in Abb. 1 wirkt dennoch recht unübersichtlich. Es gibt zahlreiche Kantenüberkreuzungen, spitze Winkel zwischen Kantenpaaren, und manche Kanten laufen hinter unbeteiligten Knoten vorbei.

Wahrnehmungspsychologische Studien (Purchase et al., 1996) haben ergeben, dass wenige Kreuzungen mit Abstand das wichtigste Merkmal verständlicher Graphenzeichnungen sind, gefolgt von wenigen Knicken, kurzen Kantenverläufen, Vermeidung von spitzen Winkeln und Darstellung von Symmetrien.

Das wissenschaftliche Gebiet des automatischen Zeichnens von Graphen (Automatic Graph Drawing) beschäftigt sich mit Design, Analyse, Implementierung und experimenteller Evaluierung von Algorithmen zum automatischen Layout von Diagrammen. Zur Einstimmung wollen wir für das in Abb. 1 gezeigte Diagramm mit verschiedenen Verfahren (in sekundenschnelle) erstellte Zeichnungen zeigen.

In einer konzeptionell sehr einfachen Methode, die in zahlreichen Softwarepaketen implementiert ist, werden die Knoten als sich gegenseitig abstoßende Partikel und die Kanten als Federn modelliert. In diesem physikalischen System wird ein Kräftegleichgewichtszustand numerisch approximiert. Das Ergebnis ist eine Zeichnung wie die in Abb. 2.

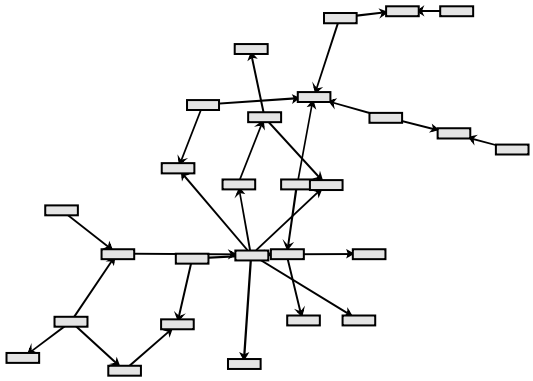


Abb. 2. Datenmodell: Kräfteverfahren

Die Struktur des Datenmodells ist zwar wesentlich besser erfassbar, allerdings ist die Auflösung unbefriedigend so wie die Tatsache, dass im allgemeinen offensichtlich unnötige Kreuzungen entstehen. Außerdem ist der Stil der Zeichnung im Datenbankbereich nicht besonders populär.

Ist die in Abb. 1 klar sichtbare Hierarchie (als Pfeile zeigen von oben nach unten) wichtig, so kann ein in Sugiyama et al. (1981) publiziertes und sehr populäres Verfahren, das ebenfalls in vielen Softwarepaketen implementiert ist, eine Zeichnung wie die in Abb. 3 automatisch erstellen.

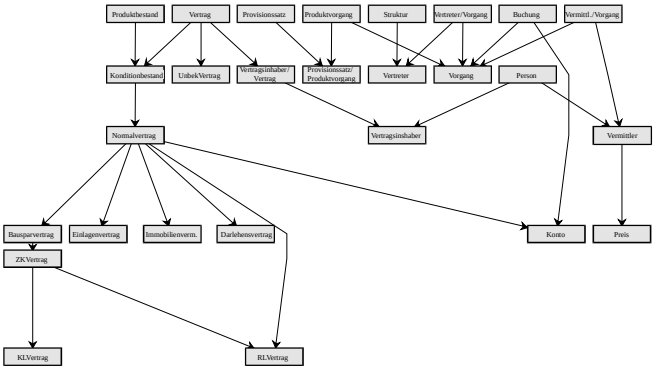


Abb. 3. Datenmodell: Hierarchisches Verfahren

Statt der in der Originalzeichnung vorhandenen 24 Kreuzungen gibt es nur noch 4 Kreuzungen, gleichzeitig ist das Problem der spitzen Winkel und der „Knotenüberkreuzungen“ beseitigt.

Ist die Hierarchie unwichtig, und die Minimierung von Kreuzungen das wichtigste Kriterium, so kann man mit einem sogenannten Planarisierungsverfahren eine Zeichnung wie die in Abb. 4 gezeigte automatisch erzeugen.

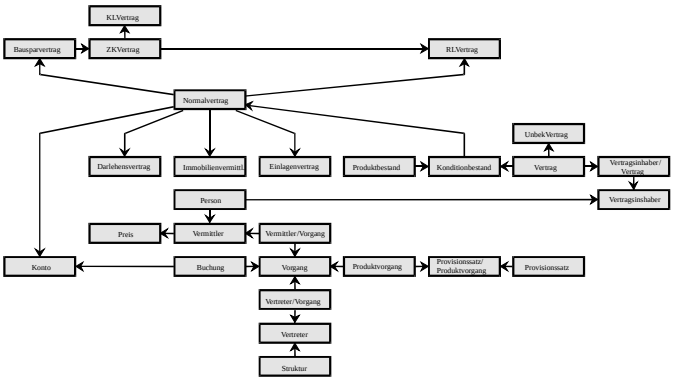


Abb. 4. Datenmodell: Planarisierungsverfahren

Diese Zeichnung enthält überhaupt keine Kreuzungen mehr und weist einen „quasi-orthogonalen“ Stil auf, in dem Kantensegmente außer in der Umgebung von Knoten mit vielen adjazenten Kanten horizontal oder vertikal gezeichnet werden, so dass unnötig spitze Winkel automatisch vermieden werden.

In diesem Artikel wollen wir einen Einblick in die zahlreichen Anwendungen des seit Anfang der neunziger Jahre aufblühenden Gebiets des automatischen Zeichnens von Diagrammen geben und aufzeigen, wie mathematische und algorithmische Methoden des Operations Research dazu beitragen, qualitativ hochwertige Layouts praktisch effizient zu erzeugen.

2. Anwendungen

2.1 Datenmodelle

Wir haben bereits in Abschnitt 1 ein Lehrbuchbeispiel eines Datenbankmodells diskutiert. Abb. 5 zeigt die Zeichnung eines anonymisierten Datenmodells aus der Versicherungswirtschaft, wie es an der Bürowand der für die Pflege der Datenbank verantwortlichen Mitarbeiter (im Format 1x2 m) hängt und ständig konsultiert wird.

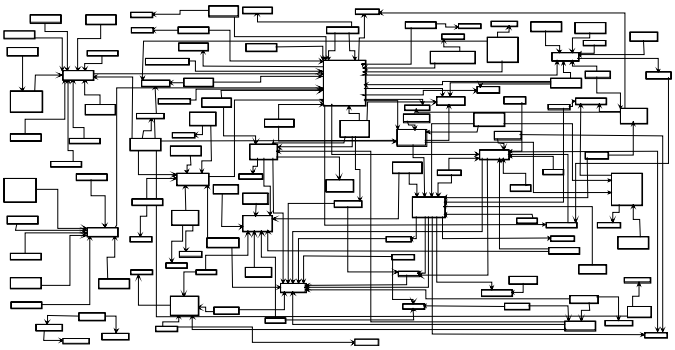


Abb. 5. Datenmodell einer Versicherung: Originalzeichnung

Die mit Hilfe einer kommerziellen Software erstellte Zeichnung stellt 120 Einheiten und 161 Beziehungen dar, und enthält Anomalien in Form von Linien, die durch Einheiten hindurchgezeichnet werden. Es handelt sich um eine orthogonale nicht-hierarchische Zeichnung mit 122 Kreuzungen.

Dieses Beispiel zeigt, dass automatisches Layout von Diagrammen auch ein wertvolles Analysewerkzeug ist. Es ist nämlich möglich, eine strenge Hierarchie zu erkennen, wenn ein automatisches Layout wie in Abb. 6 erzeugt wird.

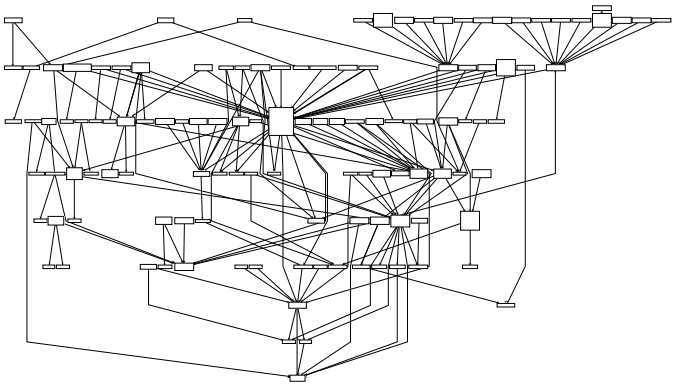


Abb. 6. Datenmodell einer Versicherung: Hierarchisches Verfahren

Ist jedoch eine übersichtlichere Zeichnung in ähnlichem Stil wie im Original erwünscht, so ergibt die Anwendung eines Planarisierungsverfahrens eine wie die in Abb. 7 gezeigte Zeichnung mit nur noch wenigen Kreuzungen.

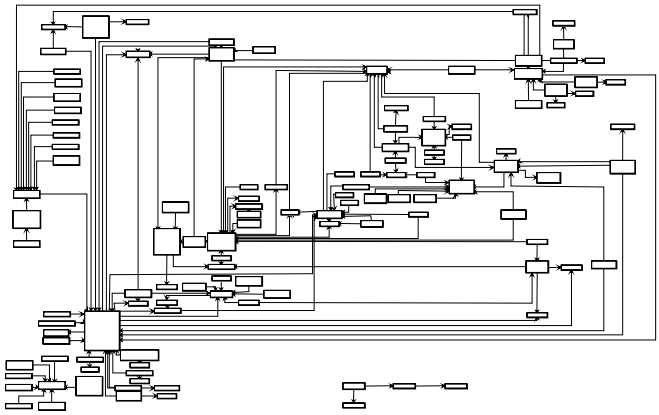
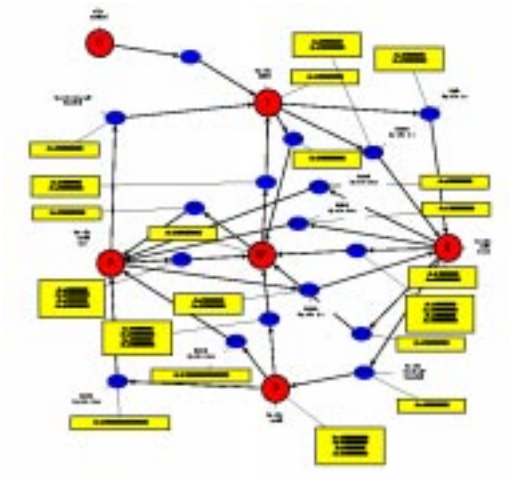


Abb. 7. Datenmodell einer Versicherung: Planarisierungsverfahren

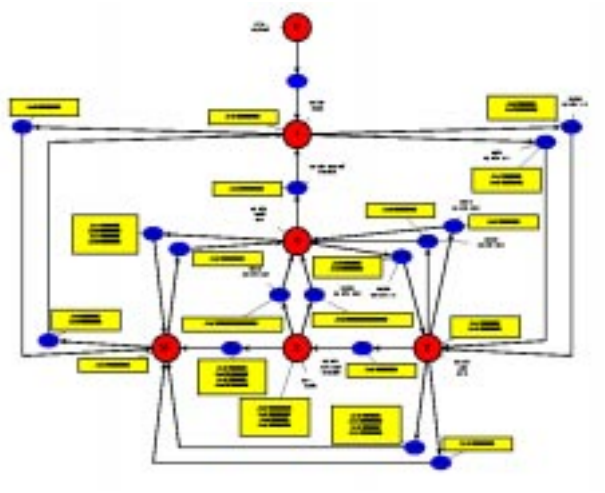
Neben dem Layout von Datenbankmodellen finden sich für das automatische Zeichnen von Diagrammen zahlreiche weitere Anwendungsgebiete, von denen wir eine kleine Auswahl präsentieren möchten.

2.2. Technische Zustandsdiagramme

Abb. 7b zeigt ein automatisches Layout einer in Abb. 7a gezeigten Ingenieurszeichnung. Die nummerierten Kreise stellen Zustände einer Maschine dar, die Ovale beschreiben Zustandsübergänge, die Rechtecke sind (anonymisierte) Etiketten, die technische Erklärungen enthalten.



(a)



(b)

Abb. 7. Technisches Zustandsdiagramm (a) Handzeichnung (b) automatisches Layout

2.3 Design von elektronischen Schaltkreisen

Im Rahmen eines neuartigen Gesamtsystems einer kalifornischen Firma, das die Entwicklung elektronischer Schaltkreise vom logischen Design bis zur Hardware-Realisierung begleitet, sind gewisse Flussdiagramme auf dem Computer graphisch darzustellen. Abb. 8 zeigt ein typisches durch ein hierarchisches Verfahren erzeugtes Diagramm.

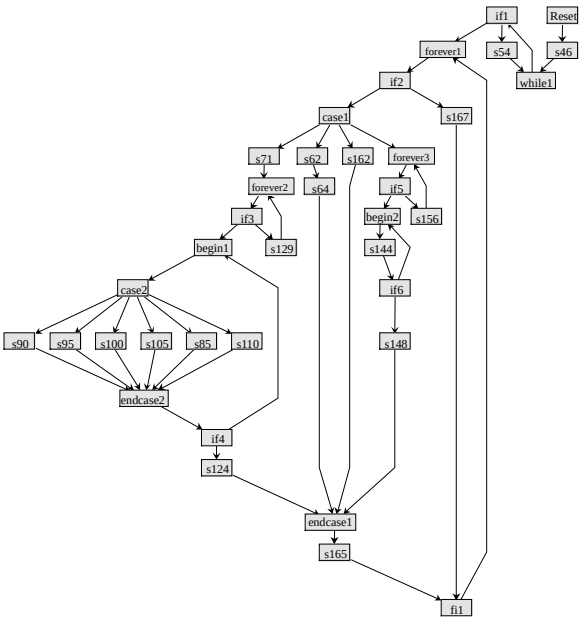


Abb. 8. Flussdiagramm im Rahmen des Chiplayouts

2.4 Geschäftsprozessmodelle

Geschäftsprozessmodellierung („business process modelling“) hat zum Ziel, die Struktur von Geschäftsprozessen auf verschiedenen Detaillierungsstufen zu organisieren und

transparent zu machen. Diagramme, die beispielsweise die Interaktionen von Produktionsprozessen oder die Struktur und Abhängigkeiten von Geschäftsprozessen darstellen, unterstützen das Management in der Analyse und der Optimierung von Geschäftsprozessen. Abb. 9 zeigt eine verschachtelte Struktur, die höhere Anforderungen an die Layoutsoftware stellt als die bisher gezeigten Beispiele. Die Methodenentwicklung für das automatische Layout solcher verschachtelter Strukturen ist Gegenstand aktueller Forschung.

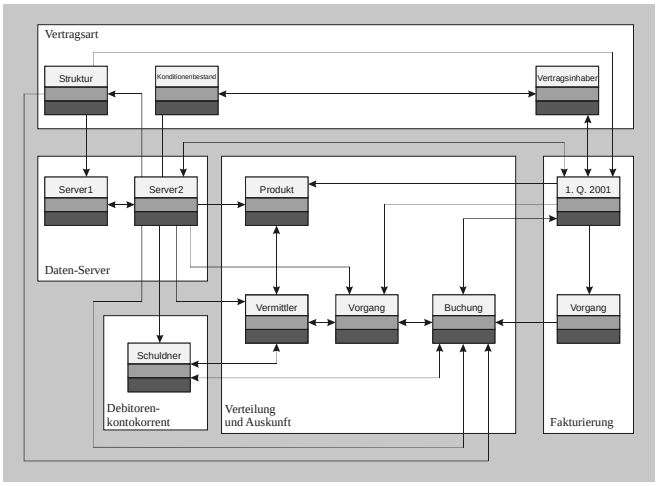


Abb. 9. Geschäftsprozessmodell

2.5 Layout von UML-Diagrammen

UML (Unified Modelling Language) hat sich in der letzten Zeit zu einem Graphikstandard entwickelt. Eine besondere Herausforderung liegt im Zeichnen von UML-Klassendiagrammen, wie sie im Bereich des Softwareengineering und –reengineering auftreten. Im Gegensatz zu den oben verwendeten Zeichenstilen handelt es sich hier um eine Mischform von hierarchischen und nicht-hierarchischen Elementen.
Die derzeit mangelhafte Darstellung von UML-Diagrammen ist eine Schwäche derzeitiger Softwareengineering und –reengineering-Tools. Die Entwicklung geeigneter Methoden und deren Umsetzung in Software ist ebenfalls Gegenstand aktueller Forschung. Abb. 10 zeigt ein UML-Klassendiagramm, das von einer prototypischen Software erzeugt wurde.

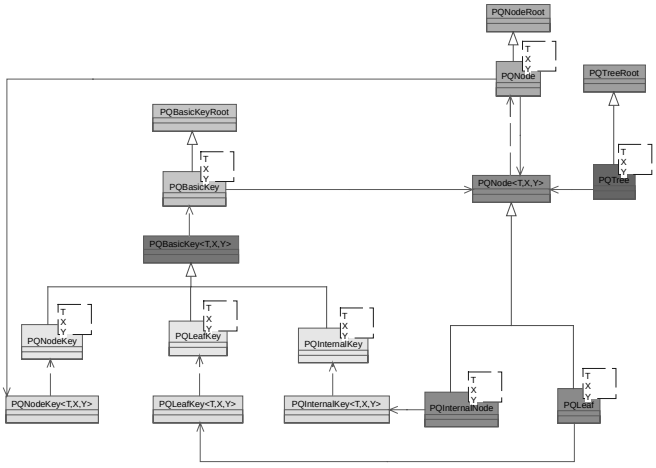


Abb. 10. UML-Klassendiagramm

3. Operations Research Methoden

Wir wollen nun die Bedeutung von mathematischen und algorithmischen Methoden des Operations Research für das automatische Layout von Diagrammen skizzieren. Sie ergänzen in idealer Weise die in der Regel von Informatikern entwickelten Grundbestandteile guter Methoden zum automatischen Layout von Diagrammen.

3.1. Kräfteverfahren

Die Approximation energieminimaler Systemzustände im Kräftemodell entspricht der approximativen Bestimmung eines lokalen Minimums einer nicht-linearen Hamiltonfunktion. Hier kommen Methoden der *Nichtlinearen Optimierung* zum Einsatz. Eine besondere Herausforderung ist die Anwendbarkeit dieser Methodik auf Graphen mit mehreren hundert Knoten und Kanten, hier wurden erst in den letzten beiden Jahren Durchbrüche erzielt (Marks 2000).

3.2. Hierarchische Verfahren

Das am häufigsten verwendete Verfahren für hierarchische Diagramme wurde von Sugiyama et al. (1981) vorgeschlagen. Hier wird zunächst jedem Knoten eine Schicht zugeteilt. Danach werden die Knoten innerhalb der Schichten so umgeordnet, dass die Anzahl der Kreuzungen möglichst klein wird. Basierend auf diesen Schritten werden die waagrechten und senkrechten Positionen der Objekte sowie die Verbindungslinien festgelegt.

Kritisch bei diesem Verfahren ist der Kreuzungsminimierungsschritt. Sugiyama, Tagawa und Toda schlugen vor, die Kreuzungsminimierung statt über alle Schichten gleichzeitig jeweils nur für zwei benachbarte Schichten zu betrachten. Hierbei wird z.B. die Ordnung der Knoten der obersten Schicht fixiert, während man versucht, die zweite Schicht so umzuordnen, dass die Anzahl der Kreuzungen zwischen den beiden Schichten minimiert wird. Danach fixiert man die zweite Schicht und ordnet die dritte, usw. Selbst dieses

eingeschränkte Optimierungsproblem gehört noch zu der Klasse der NP- schwierigen Probleme.

Überraschenderweise konnte unsere Forschungsgruppe trotzdem einen Algorithmus entwickeln, der dieses eingeschränkte Problem für fast alle praktischen Instanzen innerhalb weniger Sekunden optimal lösen kann (Jünger und Mutzel 1997).

Wir betrachten das 2-Schichten-Kreuzungsminimierungsproblem, bei dem die obere Schicht fixiert ist. Unsere Aufgabe ist es, eine Knotenpermutation für die untere Schicht zu bestimmen, so dass die Anzahl der Kantenüberkreuzungen minimal wird. Für ein Knotenpaar *(i,j)* der unteren Schicht bezeichne $c_{i,j}$ die Anzahl der Kreuzungen zwischen den mit *i* und *j* inzidenten Kanten, falls *i* links von *j* platziert wird. Unsere Aufgabe kann also folgendermassen formuliert werden: Finde eine Permutation π von {*1,2,...n*}, die

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{\pi(i),\pi(j)}$$

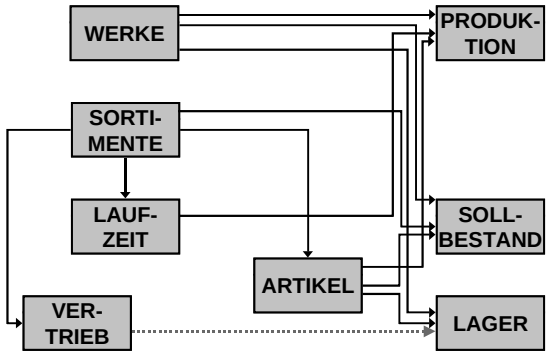
minimiert. Dieses Problem ist bekannt als das *Lineare Ordnungsproblem*, das in den meisten Fällen praktisch effizient mit Hilfe von Branch-and-Cut Algorithmen gelöst werden kann (Grötschel et al. 1984). Weiter unten werden wir ein Branch-and-Cut Verfahren für ein anderes Problem kurz skizzieren.

In der Literatur wurde lange Zeit behauptet, dass das iterative Verfahren das hierarchische Kreuzungsminimierungsproblem sehr gut annähern würde. Erst unsere Arbeiten konnten diese Behauptung widerlegen: Die mit Hilfe des iterativen Verfahrens berechnete Kreuzungsanzahl war in vielen Fällen wesentlich mehr als fünfmal höher als die minimale Kreuzungszahl. Diese können wir bisher jedoch nur für eine sehr eingeschränkte Klasse von Diagrammen berechnen. Unsere Arbeiten haben jedoch gezeigt, dass es sich lohnt, weiterhin Forschungen auf dem Gebiet der hierarchischen Kreuzungsminimierung zu betreiben.

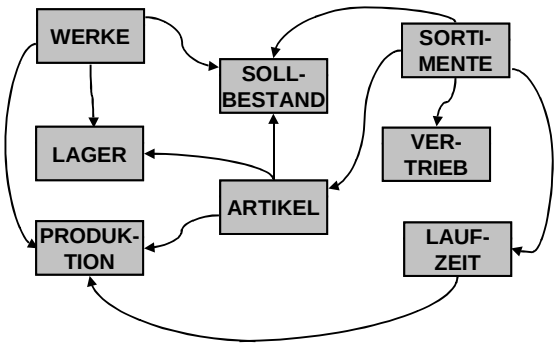
Inzwischen ist es Healy und Kuusik (1999) gelungen, mit Hilfe ganzzahliger Optimierungsansätze praxisrelevante Instanzen des Kreuzungsminimierungsproblems über alle Schichten beweisbar optimal zu lösen.

3.3. Planarisierungsverfahren

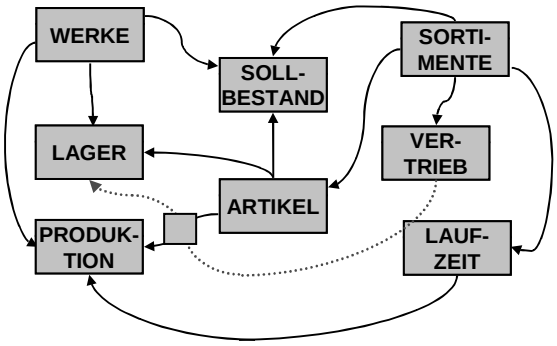
Das in Nardelli et al. (1984) vorgestellte Planarisierungsverfahren besteht aus drei Phasen: Topologie-Form-Metrik. Deshalb wird es in der Literatur auch Topology-Shape-Metrics Verfahren genannt. Wir demonstrieren es anhand des in Abb. 11a gezeigten Beispiels einer Handzeichnung eines kleinen Datenmodells, das mit Hilfe eines Planarisierungsverfahrens automatisch gezeichnet werden soll. Zunächst wird eine möglichst kleine Menge von Kanten entfernt, so dass der reduzierte Graph planar wird, das heißt auf der Ebene überkreuzungsfrei zeichenbar. Unser Beispielgraph ist nicht planar, aber es reicht, die zwischen „Vertrieb“ und „Lager“ verlaufende Kante zu streichen, um einen planaren Graphen zu erhalten, für den effizient (in Zeit proportional zur Knotenanzahl) eine Layoutskizze (Abb. 11b) berechnet werden kann. Die vorher entfernte Kante wird geschickt wieder eingefügt und die dabei entstehende Kreuzung wird durch einen künstlichen Knoten ersetzt (Abb. 11c). Der planar eingebettete „Hilfsgraph“ legt die *Topologie* der Zeichnung fest. Dann wird die *Form* einer orthogonalen Zeichnung festgelegt, wobei die Anzahl der Kantenknicke minimiert wird. Schließlich entsteht die endgültige Zeichnung durch Festlegung der Längen der Kantensegmente und durch anschließende Entfernung des künstlichen Knotens (Abb. 11d).



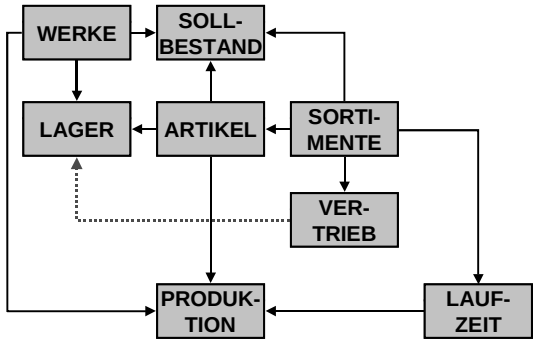
(a) Entfernen der markierten Kante ergibt einen planaren Graphen



(b) Eine planare Layout-Skizze



(c) Wiedereinfügen der entfernten Kante



(d) Orthogonales Zeichnen mittels planarer Methode

Abb. 11. Planarisierungsmethode

Nun ist die Bestimmung einer Kantenmenge minimaler Kardinalität, deren Entfernung zu einem planaren Graphen führt, NP-schwierig.

Selbst für Graphen praxisrelevanter Größe ist es aber oft möglich, Optimallösungen mit Hilfe eines in Jünger und Mutzel (1996) vorgestellten Branch-and-Cut Verfahrens zu finden. Wir skizzieren die Idee: Nach einem klassischen Resultat von Kuratowski ist ein Graph genau dann planar, wenn er weder eine Unterteilung des vollständigen Graphen auf 5 Knoten K_5 noch eine Unterteilung des vollständigen bipartiten Graphen $K_{3,3}$ enthält. Bei einer Unterteilung eines Graphen ist es erlaubt, beliebige zusätzliche Knoten auf den Kanten des ursprünglichen Graphen anzubringen. Die Unterteilungen von K_5 und $K_{3,3}$ heissen auch Kuratowski-Graphen.

Wir ordnen nun jeder Teilmenge *F* der Kantenmenge *E* eines gegebenen Graphen *G*=(*V*,*E*) einen |*E*|-dimensionalen charakteristischen Vektor $\chi^F \in \{0,1\}^E$ zu, dessen Komponenten mit den Kanten $e \in E$ indiziert sind und setzen

$$\chi_e^F = \begin{cases} 1, & falls\, e \in F \\ 0, & falls\, e \notin F. \end{cases}$$

Dann sind die charakteristischen Vektoren aller planarer Subgraphen von *G* genau die ganzzahligen Lösungen $x \in \Re^E$ des folgenden Ungleichungssystems:

$$0 \le x_e \le 1 \quad \forall e \in E \qquad (1)$$

$$\sum_{e \in K} x_e \le 1 \quad \forall K \subseteq E, \qquad (2)$$

wobei *K* die Kantenmenge eines Kuratowski Subgraphen von *G* definiert. Fügen wir die Zielfunktion

$$\max \sum_{e \in E} x_e$$

und die Ganzzahligkeitsbedingungen

$$x_e \text{ ganzzahlig für alle } e \in E \qquad (3)$$

hinzu, so haben wir unser Problem als ganzzahliges lineares Optimierungsproblem formuliert, denn die Ungleichungen schliessen die verbotenen Kuratowski-Subgraphen aus. Lässt man die Ganzzahligkeitsbedingungen weg, so entsteht ein lineares Optimierungsproblem, dessen optimaler Zielfunktionswert eine obere Schranke für die Anzahl der Kanten in einem planaren Subgraphen von *G* ist. Durch Hinzufügen weiterer Ungleichungen kann man diese Schranke verschärfen. In Jünger und Mutzel (1996) wird dargestellt, wie man solche Relaxierungen mit Hilfe eines Schnittebenenverfahrens („Cut“) lösen kann und wie die Lösungen solcher Relaxierungen als Schranken in einem Enumerationsverfahren („Branch“) zur Optimallösung des Planarisierungsproblems ausgenutzt werden können. Dieses Branch-and-Cut-Verfahren ist so angelegt, dass auch gute zulässige Lösungen (charakteristische Vektoren planarer Subgraphen grosser Kardinalität) gefunden werden, so dass zu jeder Zeit eine Lösung mit einer aus der oberen Schranke resultierenden Gütegarantie angegeben werden kann. So entsteht als Nebenprodukt eine Heuristik, die Lösungen mit sehr hoher Qualität im Vergleich zu einfacheren Heuristiken produziert.

Das Wiedereinfügen der entfernten Kanten sollte so gestaltet sein, dass möglichst wenige Kreuzungen entstehen. Wenn, wie in unserem Beispiel, nur eine Kante entfernt werden musste, und wir die *Einbettung* (die planare Zeichenskizze) des planaren Untergraphens fixieren, dann reduziert sich das Kreuzungsminimierungsproblem auf ein *Kürzestes Wegeproblem* im dualen Graphen. Der allgemeine Fall ist sehr schwierig und bis heute nur unzureichend verstanden. Selbst wenn man das Problem für eine gegebene feste Einbettung lösen könnte, wäre praktisch nicht viel erreicht, denn für eine andere Einbettung kann das Minimum erheblich kleiner sein. Überraschenderweise kann man das Kreuzungsminimierungsproblem beim Einfügen einer Kante über die Menge aller Einbettungen mit Hilfe der SPQR-Baum-Datenstruktur in linearer Zeit lösen (Gutwenger et al. 2001).

Ist die Topologie einmal festgelegt, so wird im nächsten Schritt die Form der Zeichnung festgelegt. Hierbei werden

sehr oft orthogonale oder quasi-orthogonale Zeichenmethoden verwendet, die auf Netzwerkflussmethoden beruhen. Für eine feste gegebene planare Einbettung hat Tamassia (1987) ein sehr elegantes Verfahren angegeben, das auf der Lösung eines *Minimum Kosten Flussproblems* beruht. Will man Knickminimierung über alle Einbettungen betreiben, kommen wieder Ansätze der *Ganzzahligen Optimierung* in Frage.

Im letzten Schritt geht es nun darum, die Längen der Kantensegmente festzulegen. Dabei soll die Zeichenfläche sowie die Gesamtkantenlänge möglichst minimiert werden. Dieses Problem ist dem bereits aus dem VLSI bekannten Kompaktierungsproblem sehr ähnlich. Vor kurzem konnte hier - durch eine neue kombinatorische Formulierung des Problems - mit Hilfe von Methoden der Ganzzahligen Optimierung Durchbrüche erzielt werden (Klau und Mutzel 2000).

Details zu allen hier beschriebenen Optimierungsproblemen finden sich in Mutzel (2001).

4. Sonstige Fragestellungen

Natürlich gibt es ausser den hier vorgestellten Fragestellungen im Bereich des Graphenzeichnens noch viele andere. Unter anderem werden zur Zeit folgende Fragestellungen von verschiedenen Gruppen untersucht:

- Symmetrien: das Finden von Symmetrien in Graphen sowie deren visuelle Darstellung
- Gruppierungen: das Bestimmen von geeigneten Gruppierungen bei sehr großen Graphen sowie die Darstellung solcher Gruppierungen
- Nebenbedingungen: Benutzer haben oft eigene Vorstellung über die Platzierung einzelner Knoten, Knotengruppen oder Kanten
- Inkrementelle Zeichenmethoden: Meist entstehen Zeichnungen Schritt für Schritt. Inkrementelle Zeichenmethoden versuchen die „Mental Map“ des Benutzers weitgehend zu erhalten.
- Beschriftung: Integration von Knoten- sowie Kantenbeschriftung
- Dreidimensionale Zeichnungen: Wie kann man die dritte Dimension erfolgbringend in der visuellen Darstellung nutzen?

Einen guten Überblick über die aktuelle Forschung auf dem Gebiet erhält man in den jährlich erscheinenden Proceedings des Internationalen Graph Drawing Symposiums (z.B. Marks 2000 oder Jünger et al. 2001).

5. Software

Obwohl die übersichtliche Darstellung komplexer Zusammenhänge zu den wichtigsten Aufgaben benutzerfreundlicher Software gehört, wird im kommerziellen Bereich die Komplexität der Aufgabe oft unterschätzt. Entsprechend werden in manchen marktführenden Softwarepaketen inadäquate Methoden eingesetzt. Jedoch gibt es einige kleiner Firmen, die gute Software zum Graphenzeichnen anbieten. Im akademischen Bereich ist eine erstaunliche Vielfalt von guter Graphenzeichensoftware vorhanden. Anlässlich der internationalen Tagung GRAPH DRAWING 2001 vom 23.-26. September 2001 in Wien (http://www.ads.tuwien.ac.at/gd2001) wird es eine Softwaremesse geben, von der wir hoffen, dass sie den Transfer der neuesten wissenschaftlichen Ergebnisse im Bereich des automatischen Layouts von Diagrammen beschleunigen wird.

Danksagungen

Unsere eigene Arbeit im automatischen Layout von Diagrammen wurde von der Deutschen Forschungsgemeinschaft, dem Bundesministerium für Bildung und Forschung, der Europäischen Union und der Siemens AG gefördert. Außerdem fördert die DFG-Ideenwerkstatt unser Graphenzeichenprojekt im Rahmen seiner Transferbemühungen. Im Oktober 2000 hat die Stiftung caesar ein Projekt „Graph Drawing“ eingerichtet, das zum Ziel hat, moderne Software zum automatischen Zeichnen von Diagrammen mit Hilfe eines Spin-offs an den Markt zu bringen. Schwerpunkt der derzeitigen Entwicklung ist das automatische Layout von UML-Klassendiagrammen wie in Abb. 10. Allen Sponsoren gilt unser Dank für die Förderung.

Literatur

M. Grötschel, M. Jünger, G. Reinelt (1984): A cutting plane algorithm for the linear ordering problem, Operations Research 32, 1195-1220

C. Gutwenger, P. Mutzel, R. Weiskircher (2001): Inserting an edge into a planar graph, Proc. Twelfth, ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), ACM Press, 246-255

P. Healy und A. Kuusik (1999): The Vertex-Exchange Graph: a new concept for multi-level crossing minimization, Graph Drawing 1999, Lecture Notes in Computer Science 1731, Springer, 205-216

M. Jünger, S. Leipert und P. Mutzel (2001): Graph Drawing 2001, Lecture Notes in Computer Science, Springer, to appear

M. Jünger und P. Mutzel (1996): Maximum planar subgraphs and nice embeddings: practical layout tools, Algorithmica 16 (1):33-59

M. Jünger und P. Mutzel (1997): 2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms, Journal of Graph Algorithms and Applications 1 (1), 1--25

J. Marks (2000): Graph Drawing 2000, Lecture Notes in Computer Science 1984, Springer

G. Klau und P. Mutzel (1999): Optimal compaction of orthogonal grid drawings, Integer Programming and Combinatorial Optimization (IPCO ’99), Lecture Notes in Science 1610, Springer

P. Mutzel (2001): Optimization Problems in Graph Drawing, in Pardalos, P.M. und M.C.G. Resende (eds.), Handbook of Applied Optimization, Oxford University Press, erscheint 2001

C. Batini, M. Talamo, R. Tamassia (1984): Computer Aided Layout of Entity Relationship Diagrams, Journal of Systems and Software 4, 163-173

H. Purchase, R. Cohen, M. James (1996): Validating Graph Drawing Aesthetics, Graph Drawing 1995, Lecture Notes in Computer Science 1027, 435-446

O. Rauh, E. Stickel (1996): Fallstudien zum Datenbankentwurf, Th. Gabler, Wiesbaden, 1997

K. Sugiyama, S. Tagawa, M. Toda (1981): Methods for visual understanding of hierarchical systems, IEEE Transact. Syst. Man. Cybern, SMC-11 (2):109-125

R. Tamassia (1987): On embedding a graph in the grid with the minimum number of bends. SIAM J. Comput. 16 (3):421-444