Zeichnen von Diagrammen — Theorie und Praxis —

Petra Mutzel*
Max-Planck-Institut für Informatik
Saarbrücken

Zusammenfassung

Ausgehend von praxisnahen Anwendungen, wie das Zeichnen von Zustandsdiagrammen für Steuerungen in der Automatisierungstechnik (Siemens AG, München), die Darstellung der Abhängigkeiten in Datenbanken für Anwendungsentwickler (EDV-Abteilung der Molkerei Alois Müller GmbH&Co, Aretsried), sowie die Visualisierung der Zusammenhänge zwischen Objekten für polizeiliche Anwendungen (Fa. Genesys GmbH, München), führen wir in das Gebiet des automatischen Zeichnens von Diagrammen ein und zeigen auf, inwieweit unsere theoretisch wissenschaftlichen Forschungen dazu führen können, übersichtliche Diagramme zu erstellen.

1 Einführung

Diagramme wie das in Abb. 1 gezeigte sind immer öfter in Zeitungen, Zeitschriften, Dokumentationen oder wissenschaftlichen Arbeiten zu finden. Die meisten dieser Diagramme werden per Hand oder mit Hilfe von Zeichenprogrammen wie CorelDraw erstellt. Solange diese Diagramme klein sind oder man diese nur selten erstellen muß (z.B. für Präsentationen) ist dies kein Problem. Sobald die Diagramme größer werden oder diese Aufgabe regelmäßig anfällt, ist ein Softwaretool hilfreich, das Layouts von Diagrammen automatisch erzeugt.

Die Eingabe ist entweder eine grobe Skizze des Diagramms oder eine abstrakte Datenmenge, wie z.B. Objekte und deren Beziehungen untereinander. Aufgabe der Software ist es nun, ein "schönes" Layout zu erzeugen. "Schön" kann hier je nach Anwendung verschiedenes bedeuten, wie "gut lesbar", "verständlich", oder auch "ästhetisch schön". Zumeist sollte das Layout die Struktur der Eingabedaten wiedergeben.

Das Forschungsgebiet "Automatisiertes Zeichnen von Graphen und Diagrammen" widmet sich diesen Aufgaben. Historisch aus dem CAD (Computer Aided Design) entstanden, hat sich das Gebiet des Automatisierten Zeichnens von Diagrammen und Graphen inzwischen zu einem selbständigen und lebendigen Forschungsgebiet entwickelt.

Wir stellen im folgenden drei unterschiedliche Zeichenverfahren vor: Planarisierungsverfahren, kräfte-basierte Methoden sowie hierarchische Zeichenverfahren. Planarisierungsverfahren eignen sich sehr gut für Zustandsdiagramme, Endliche Automaten, Datenmodelle

*MPI Informatik, Im Stadtwald, D-66123 Saarbrücken, mutzel@mpi-sb.mpg.de

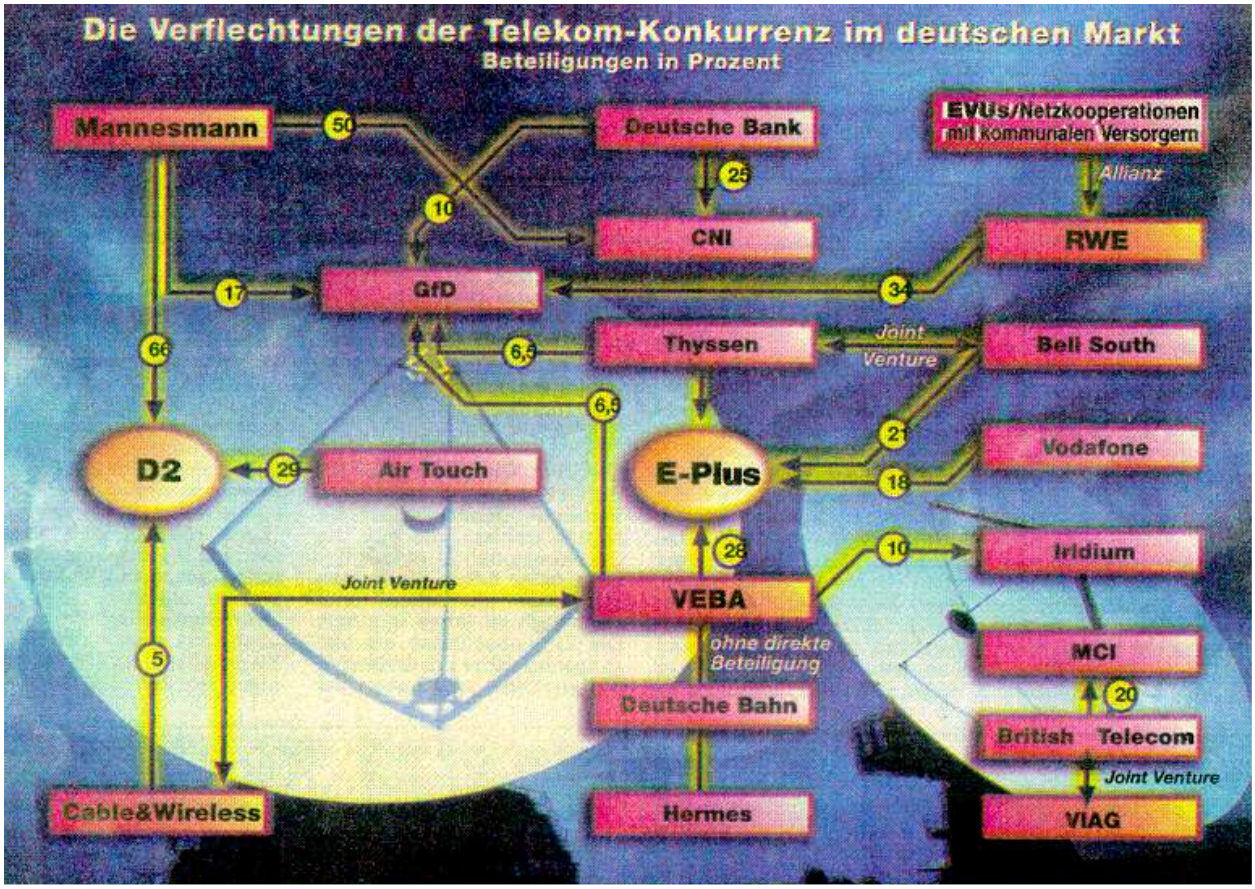


Abbildung 1: Ein Diagramm aus der Zeitung.

in Datenbanken, Entity-Relationship Diagramme oder PERT-Diagramme während sich die hierarchischen Verfahren eher für Organisationsdiagramme, Ablaufdiagramme oder hierarchische Strukturen eignen. Kräfte-basierte Verfahren sind zwar grundsätzlich für alle Arten von Diagrammen anwendbar; zu empfehlen sind sie jedoch nur für solche Anwendungen, die eine *dünne* Netzwerkstruktur besitzen.

Während der Aufwand — sowohl theoretisch als auch praktisch — für die Implementierung von kräfte-basierten Verfahren sehr gering ist (*wenige Wochen*), ist dieser für ein funktionierendes Planarisierungsverfahren sehr hoch (*einige Jahre*). Der Aufwand für hierarchische Verfahren liegt dazwischen (*einige Monate*). Dementsprechend sieht auch die Verteilung der Softwaretools zum automatisierten Zeichnen aus: Es gibt weltweit sehr viele Implementierungen von kräfte-basierten Verfahren, einige für hierarchische Verfahren, jedoch nur sehr wenige für Planarisierungsverfahren. Eines davon wurde im Rahmen des DFG-Schwerpunktprogramms "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen" in Kooperation von Gruppen am MPI Saarbrücken, der Universität Halle sowie der Universität zu Köln unter der Leitung von Petra Mutzel, Stefan Näher und Michael Jünger entwickelt und ist frei für Forschungseinrichtungen und Universitäten innerhalb der Softwarebibliothek AGD erhältlich [1]. Ab Januar 1999 ist AGD auch für kommerziell über Algorithmic Solutions GmbH erhältlich [2].

Ich möchte in diesem Vortrag versuchen aufzuzeigen, inwieweit in diesem Forschungs-

gebiet die Theorie die Praxis beeinflußt und umgekehrt. Hier gibt es je nach Anwendung große Unterschiede. Wir untersuchen diese Beziehungen anhand von unterschiedlichen Anwendungsbeispielen.

2 Planarisierungsverfahren

Wir werden anhand zweier Beispiele aus der Praxis die Planarisierungsmethode näher untersuchen.

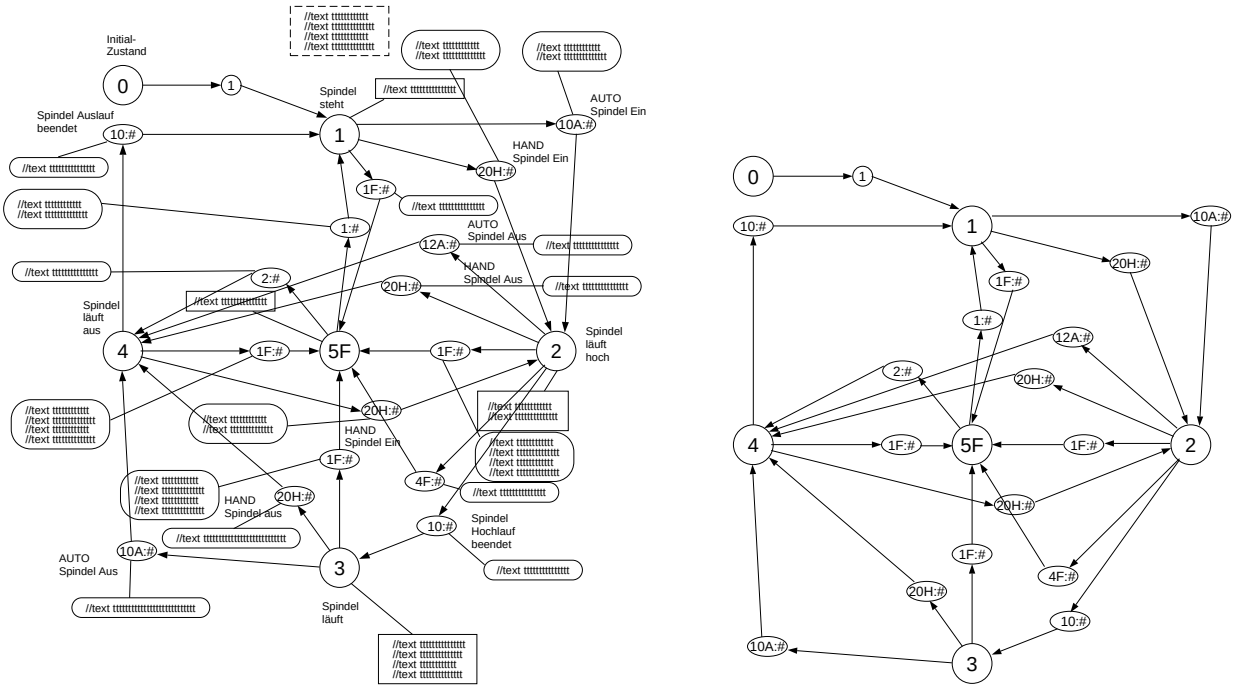


Abbildung 2: Ein per Hand gezeichnetes Zustandsdiagramm.

Das in Abb. 2 (a) gezeigte Diagramm zeigt ein von Hand gezeichnetes Zustandsdiagramm einer Steuerung eines Motors, das uns von der Siemens AG zur Verfügung gestellt wurde. Die großen runden Kreise zeigen die jeweils möglichen Zustände und die Linien zwischen diesen zeigen die Zustandsübergänge an. Die rechteckigen Kästchen geben die Beschriftung an, die hier verfremdet wurden. Um ein gutes automatisches Layout dieses Diagramms zu erhalten, entfernen wir zunächst die Beschriftungsetiketten von dem eigentlichen Diagramm. Das Ergebnis ist in Abb. 2 (b) gezeigt.

Das aktuelle Layout wirkt durch die vielen Linienkreuzungen sehr unübersichtlich. Eine Frage, die sich aus der Praxis stellt lautet: “Kann dieses Diagramm so gezeichnet werden, daß keine Linienüberkreuzungen vorhanden sind?” — Die Theorie hat sich mit dieser Frage, die auch *Planaritätstest* genannt wird, bereits seit langer Zeit befaßt. Im Jahr 1917 erschien in dem berühmten Rätselbuch von Dudeney das folgende Rätsel, das bis heute weitverbreitet ist:

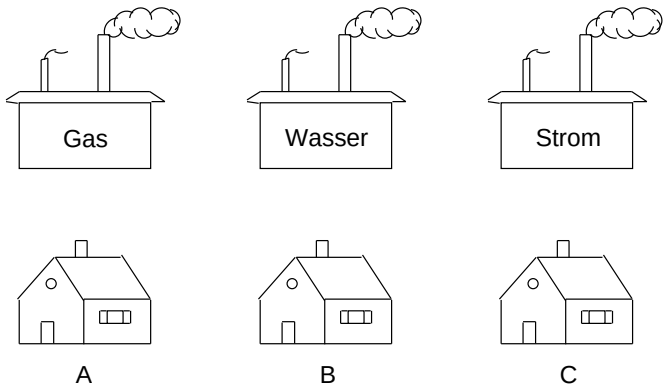
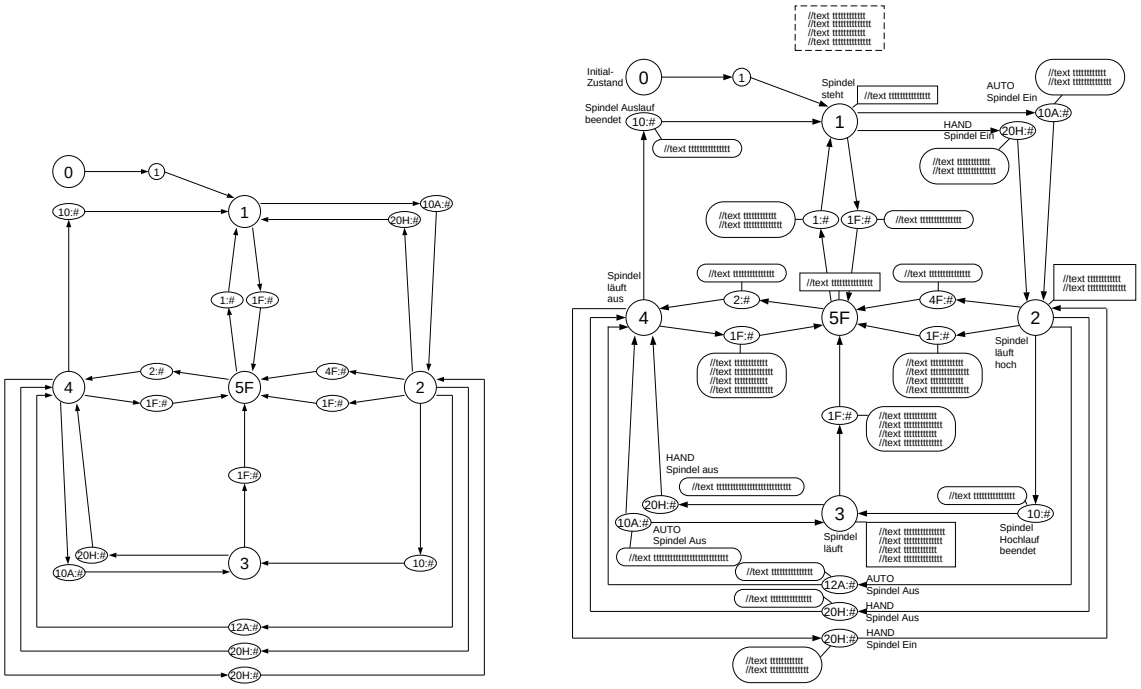


Abbildung 3: Das Rätsel Gas–Wasser–Strom.

Jedes der Häuser A, B und C soll Wasser, Gas und Strom bekommen (s. Abb. 3). Können die Leitungen so gelegt werden, daß sie sich nicht überkreuzen? Nach einigen Versuchen wird man wohl zu der Vermutung gelangen, daß dies nicht möglich ist. Doch was passiert, wenn man zuläßt, daß die Häuser jeweils an beliebige Orte plaziert werden können? Nun gibt es ungleich mehr Möglichkeiten, so daß man leicht in Versuchung gerät zu glauben, daß es nun möglich sei. Nur erscheint es sehr schwierig, alle möglichen Fälle auszuprobieren. Glücklicherweise hat der bekannte Informatiker Robert Tarjan vor einigen Jahren (1974) ein automatisches Verfahren entwickelt, das in der Lage ist, einen solchen Planaritätstest sehr effizient (innerhalb von wenigen Sekunden für viele Tausende von Objekten) zu berechnen [10]. Dieses Verfahren würde unsere Frage für unsere sechs Objekte innerhalb eines Bruchteils einer Sekunde mit "NEIN" beantworten. Wir können uns also bequem zurücklehnen, das Rätsel ist gelöst.

Doch zurück zu unserem Zustandsdiagramm. Hier lautet die Antwort des Planaritätstests: "JA". Doch bringt uns diese Antwort alleine auch nicht viel weiter. Wir wissen nun zwar, daß sich unsere Mühe lohnen könnte, jedoch ist dies für uns nicht allzuviel hilfreich. Eigentlich bräuchte man eine ebene Skizze des Diagramms oder Information darüber, wie denn nun die Linien in einer kreuzungsfreien Zeichnung angeordnet werden müßten. Dies nennt man *Einbettung*. In dem Artikel von Tarjan steht eine Andeutung, daß eine Einbettung wohl grundsätzlich mit einem ähnlichen Verfahren erzeugt werden könnte. Dies genügte den Theoretikern für viele Jahre. Das Problem schien gelöst. Erst nach einem Anstoß aus der Praxis — als ich für meine Diplomarbeit im Jahr 1990 ein Einbettungsverfahren implementieren mußte — stieß ich auf diese Lücke. Schließlich entwickelten Kurt Mehlhorn und ich das erste Einbettungsverfahren, das auf dem Planaritätstest von Robert Tarjan aufbaut [17]. Bis heute ist die in Saarbrücken erhältliche Software weltweit die einzige korrekte Implementierung dieses Verfahren. (Es gibt jedoch inzwischen andere effiziente Einbettungsverfahren; korrekte Implementierungen sind jedoch immer noch sehr selten erhältlich.)

Mit Hilfe des Einbettungsverfahrens können wir nun eine geordnete kreuzungsfreie Skizze unseres Diagramms erstellen. Um ein Layout zu erzeugen, können wir einen der vielen *planaren Zeichenalgorithmen* anwenden. Planare Zeichenalgorithmen haben die spezielle Eigenschaft, daß sie eine kreuzungsfreie Eingabe (wie sie der Einbettungsalgorithmus erstellt) benötigen. Mit Hilfe der Theorie planarer Graphen sind sie oft in der Lage "optimierte"



(a) (b)

Abbildung 4: Ein automatisch gezeichnetes Zustandsdiagramm.

Zeichnungen zu erstellen. Hier wird z.B. die Anzahl der Knicke, die Fläche oder die Verbindungslinienlänge minimiert (s. [22, 7, 9] oder [15]).

Das erzeugte Layout für unser Diagramm ist in Abb. 4 (a) zu sehen. Dieses ist wesentlich übersichtlicher als das in Abb. 2 (b) gezeigte Layout. Nun können wir die Beschriftungen mit einem relativ einfachen Verfahren hinzufügen. Das Ergebnis ist in Abb. 4 zu sehen.

Wir hatten jedoch sehr viel Glück mit unserer Anwendung. Nicht immer können wir davon ausgehen, daß der Planaritätstest für das zu zeichnende Diagramm positiv ausgeht. Um auch nach einer negativen Antwort die planaren Verfahren anwenden zu können, greifen wir zu einem Trick: Kurzzeitig entfernen wir aus unserem Diagramm wenige Linien, so daß das Restdiagramm überkreuzungsfrei zeichenbar ist. Erst nach der Anwendung des planaren Zeichenalgorithmus werden diese Verbindungen wieder in das Diagramm hinzugefügt. Eine andere Methode besteht darin, zunächst eine kreuzungsfreie Einbettung zu erstellen und in diese die entfernten Verbindungslinien wieder kreuzungsminimal einzufügen. Kurzfristig werden die dabei entstehenden Kreuzungen durch neue künstliche Objekte ersetzt, damit ein planares Zeichenverfahren angewendet werden kann.

Das Planarisierungsverfahren ist in Abb. 5 am Beispiel eines Datenmodells einer Datenbank graphisch erläutert. Ähnliche Diagramme wie in Abb. 5 (a) (Handzeichnung) tauchen z.B. in der Abteilung Anwendungsentwicklung–EDV bei der Molkerei Alois Müller GmbH&Co auf. Nach Entfernen der gestrichelt gezeichneten Verbindung gibt der Planaritätstest eine positive Antwort. Das Einbettungsverfahren liefert die in Abb. 5 (b) gezeigte Skizze. Das Einfügen der entfernten Verbindung ergibt eine Kreuzung, die kurzzeitig

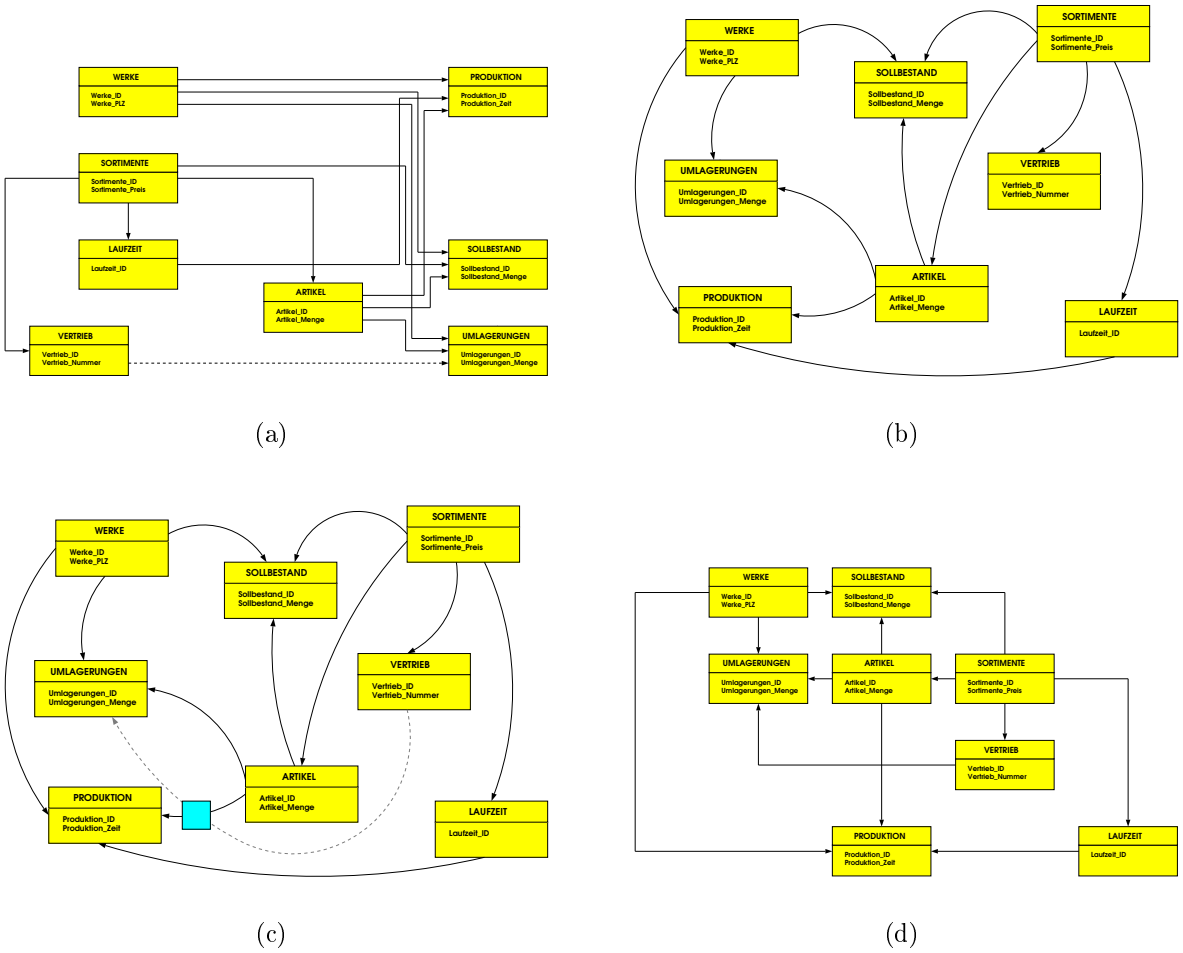


Abbildung 5: Der Planarisierungsprozeß.

durch einen künstlichen Knoten ersetzt wird (s. Abb. 5 (c)). Ein planares Layoutverfahren (z.B. [15]) liefert die in Abb. 5 (d) gezeigte Zeichnung.

Bei diesem Planarisierungsprozeß tauchen einige schwierige Probleme auf: (1) Zum einen möchte man möglichst wenige Linien entfernen; (2) zum anderen möchte man diese möglichst kreuzungsminimal wieder einfügen. Beide Probleme gehören zu einer Klasse von Problemen, die in Informatikerkreisen als extrem schwierig gelten. Für Problem (1), das sogenannte *Planarisierungsproblem*, gab es bis vor wenigen Jahren keinen exakten Algorithmus. Als ich im Rahmen meiner Dissertation begann, mich mit diesem Thema zu beschäftigen, stellte sich heraus, daß auch einige Näherungsmethoden, die in der Literatur vorgeschlagen wurden, nicht korrekt waren (s.[12, 11]). Im Rahmen meiner Dissertation entwickelte ich einen exakten Algorithmus zur Lösung des Planarisierungsproblems, der — entgegen aller Vorhersagen der Theorie — die meisten der praktischen Probleminstanzen bis zu 60 Objekten innerhalb kurzer Rechenzeit (wenige Sekunden bis Minuten) optimal lösen kann (s. [14]). Das folgende Beispiel zeigt, daß sich unsere Anstrengungen tatsächlich lohnen.

Abbildung 6 zeigt zwei verschiedene Zeichnungen einer Wettbewerbsinstanz innerhalb des Graphzeichenwettbewerbs im Jahr 1997. Für Abb. 6 (a) benutzten wir eine der besten

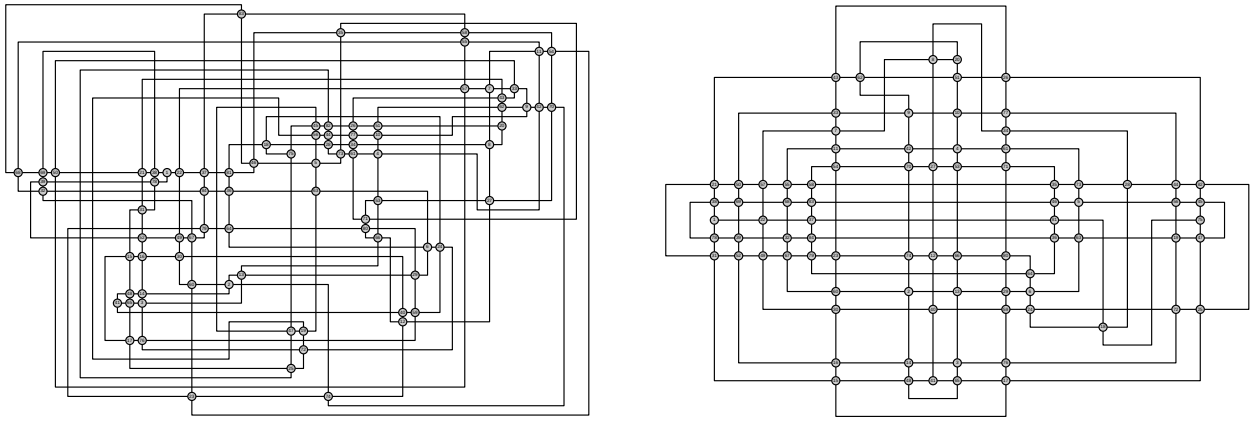


Abbildung 6: Näherungsweise gegen exakte Planarisierung.

in der Praxis verwendeten Methoden zur Planarisierung (vgl. [18, 16]). Hier wurden 39 Verbindungslinien entfernt; das Wiedereinfügen führte zu 180 Kreuzungen. In Abb. 6 (b) verwendeten wir unsere entwickelte exakte Software. Hier wurden nur 19 Verbindungslinien entfernt, was zu einer Zeichnung mit nur 51 Kreuzungen führt. Dadurch verringert sich auch der Platzbedarf der Zeichnung immens, was dazu führt, daß sich die Auflösung verbessert.

Die durch die Praxis angestoßene Theorie konnte hier tatsächlich helfen, die praktischen Ergebnisse zu verbessern. Seit kurzem haben wir damit begonnen, auch Problem (2) zu studieren. Bisher gibt es bereits erste Resultate, die darauf hinweisen, daß auch hier Verbesserungen möglich sind [20].

3 Kräfte-basierte Verfahren

Das in Abb. 7 gezeigte Diagramm zeigt ein graphisch aufbereitetes Vernehmungsprotokoll eines Überfalls (*Messerstecherei*). In besonderen Fällen darf das Bundeskriminalamt sämtliche vorhandenen Datenbanken zusammenschalten, um den Objekten (z.B. Opfer, Zeugen, Arbeitsstätten, etc.) weitere gespeicherte Daten zuzuordnen. Auf diese Weise können eventuelle Zusammenhänge zwischen den Objekten aufgedeckt werden. Die Datenbankabfrage wird mittels einer Software der Fa. Genesys durchgeführt. Die aktuelle Ausgabe der Datenbankabfrage ist jedoch nicht sehr übersichtlich (s. Abb. 8).

Da hier eine besondere *dünne* Netzwerkstruktur vorliegt (d.h., es gibt für viele Objekte nur eine Verbindung), eignen sich hier die kräfte-basierten Verfahren sehr gut. Solche Verfahren modellieren die Objekte und die Verbindungen eines Diagramms als physikalisches System: die Objekte entsprechen Massenkugeln, die sich gegenseitig abstoßen. Die Verbindungen hingegen werden als Federn modelliert, die bestrebt sind, möglichst eine ideale Länge anzunehmen. Dieses physikalische System befindet sich im sogenannten Gleichgewichtszustand (energieminimaler Zustand), wenn die Kräfte gleichmäßig verteilt sind. Aufgabe eines kräfte-basierten Verfahrens ist es, die Kräfte adäquat zu verteilen und einen Zustand des Systems mit möglichst kleiner Energie zu berechnen. Abbildung 9 zeigt das Ergebnis der Datenbankabfrage nach der Anwendung eines kräfte-basierten Verfahrens. Nun ist ein direk-

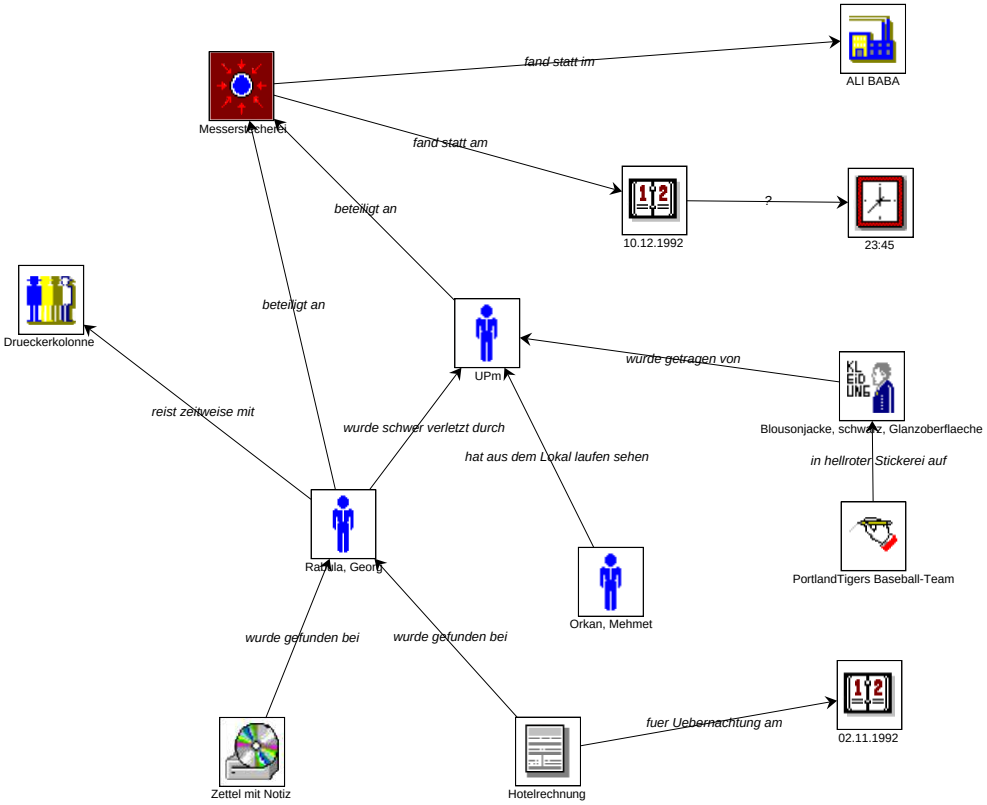


Abbildung 7: Graphisch aufbereitetes Vernehmungsprotokoll.

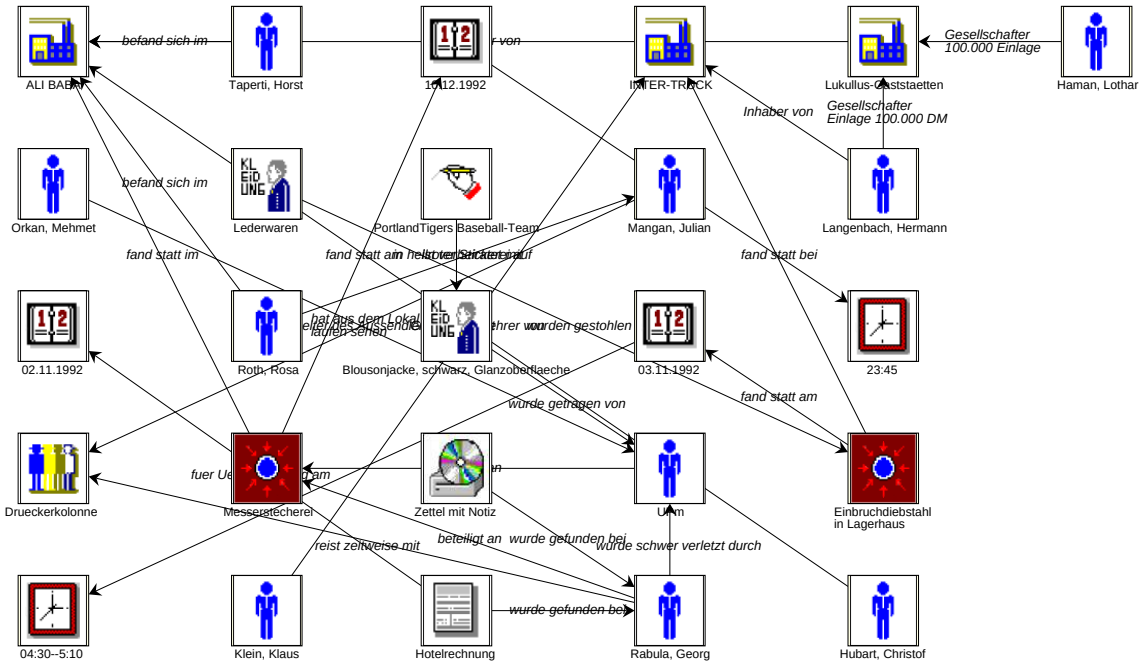


Abbildung 8: Ergebnis der Datenbankabfrage.

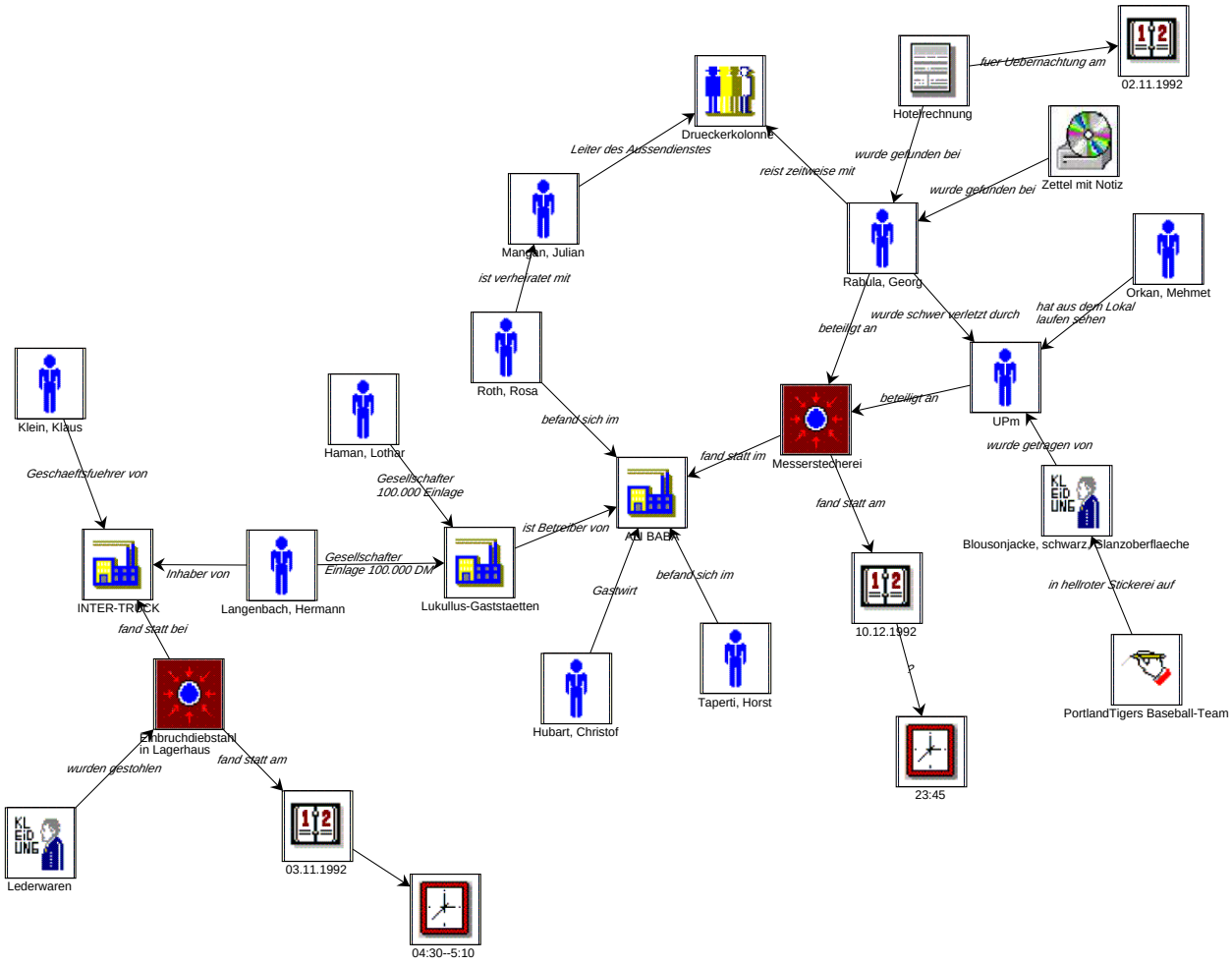


Abbildung 9: Die automatisch generierte Zeichnung des Ergebnisses der Datenbankabfrage.

ter Zusammenhang zwischen der *Messerstecherei* und einem *Einbruchdiebstahl im Lagerhaus* zu erkennen. Können Sie noch andere Zusammenhänge erkennen?

Ich möchte nicht näher auf kräfte-basierte Verfahren eingehen, weil unsere Forschungsschwerpunkte nicht hier liegen. Es gibt jedoch eine Vielzahl an Literatur und Software zu diesen Verfahren (s. [3, 5, 8]).

4 Hierarchische Verfahren

Abbildung 10 zeigt ein automatisch erstelltes Diagramm zur Modellierung des Begutachtungsprozesses bei einer wissenschaftlichen Zeitschrift. Activity- und Statecharts treten in technischen Anwendungen, in Planungssystemen oder auch in Workflow-Management Systemen auf. Hierbei soll die Zeichnung die Hierarchie der Anwendung wiedergeben. Das am häufigsten verwendete Verfahren für hierarchische Diagramme wurde 1986 von Sugiyama, Tagawa und Toda vorgeschlagen [21]. Dabei wird zunächst jedem Objekt eine Schicht zugeteilt. Danach werden die Objekte 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 extrem 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 [13].

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

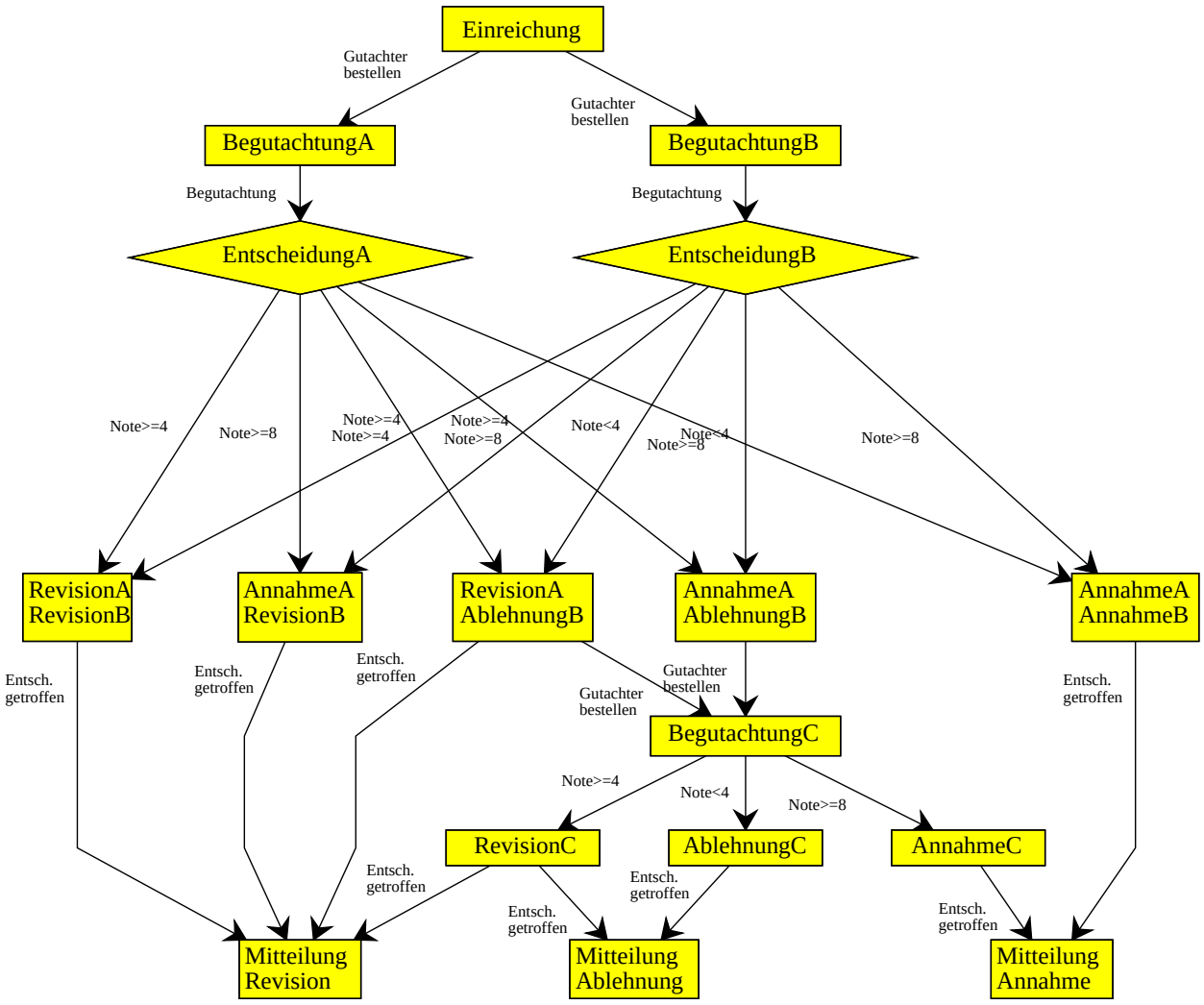


Abbildung 10: Begutachtungsprozeß bei einer wissenschaftlichen Zeitschrift.

Kreuzungszahl [13]. 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.

Zum Abschluß möchte ich noch eine weitere interessante Anwendung vorstellen. Seit kurzem arbeiten wir an einem EU-Projekt in Kooperation mit dem *Statistical Bureau of the Netherlands*, dem *Office for National Statistics* in Großbritannien sowie zwei weiteren Statistikinstituten in Spanien und Finnland, in dem es um den Entwurf, die Interviews und die Auswertung von nationalen Umfragen geht. Die meisten nationalen statistischen Institute entwerfen heute Ihre Meinungsumfragen am Computer. Immer öfter werden die Interviews selbst direkt mit einem Notebook geführt. Um so wichtiger ist es für die Designer, die Interviewer, die Auswerter sowie die Auftraggeber einer Umfrage, eine gute Dokumentation zu erhalten. Diese Dokumentation enthält unter anderem auch Diagramme, wie das in Abb. 11 gezeigte. Das Diagramm zeigt einen Ausschnitt der Übersicht des offiziellen *Adult Dental Health Survey 1998* für Großbritannien. Im allgemeinen können diese Diagramme durchaus sehr groß werden. Unsere Aufgabe innerhalb des Projekts wird sein, die graphische Komponente der Dokumentation zu erstellen.

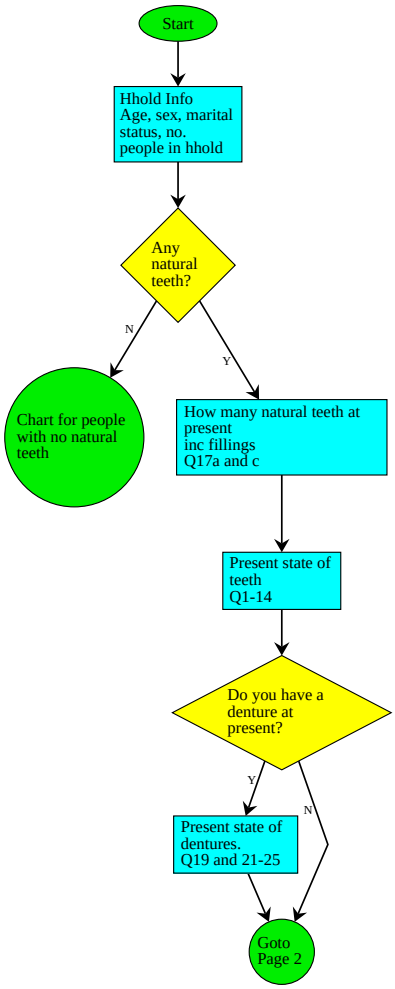


Abbildung 11: Ausschnitt der Übersicht des *Adult Dental Health Survey 1998* für Großbritannien.

5 Zusammenfassung

Wir haben verschiedene Anwendungen für das automatische Layout von Diagrammen kennengelernt. Ich hoffe, daß ich Sie ein wenig neugierig auf dieses Forschungsgebiet und dessen Anwendungen gemacht habe. Vielleicht müssen Sie ja auch ab und zu Diagramme zeichnen, bei denen wir Ihnen behilflich sein können. Weitere einführende Literatur finden Sie z.B. in [4, 19] oder [6]. Besuchen Sie auch unsere Web-Seiten auf `http://www.mpi-sb.mpg.de/AGD/`.

An dieser Stelle möchte ich mich bei unseren Kooperationspartnern und Interessenten für die Genehmigung bedanken, Ihre Anwendungen hier präsentieren zu dürfen: die Forschungsabteilung der Fa. Siemens AG (München), die EDV-Abteilung der Molkerei Alois Müller GmbH&Co (Aretsried), die Fa. Genesys GmbH (München) sowie das *Office for National Statistics* in London.

Literatur

[1] AGD. *AGD User Manual*. Max-Planck-Institut Saarbrücken, Universität Halle, Universität Köln, 1998. Available via "http://www.mpi-sb.mpg.de/AGD/". Partially supported by the DFG-cluster "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen".

[2] Algorithmic Solutions GmbH, *Spin-off* des Max-Planck-Instituts für Informatik in Saarbrücken, `http://www.algorithmic-solutions.de`.

[3] F.J. Brandenburg, M. Himsolt, and C. Rohrer. An experimental comparison of force-directed and randomized graph drawing algorithms. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 76–87. Springer-Verlag, 1996.

[4] F.J. Brandenburg, M. Jünger, and P. Mutzel. Algorithmen zum automatischen Zeichnen von Graphen. *Informatik Spektrum*, 20(4):199–207, 1997.

[5] P. Eades. A heuristic for graph drawing. *Congr. Numer.*, 42:149–160, 1984.

[6] P. Eades and P. Mutzel. *Graph Drawing Algorithms*, CRC Handbook of Algorithms and Theory of Computation, Chapter 9, M. Atallah (Ed.). CRC Press, 1999. To appear.

[7] U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 254–266. Springer-Verlag, 1996.

[8] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 388–403. Springer-Verlag, 1995.

[9] C. Gutwenger and P. Mutzel. Planar polyline drawings with good angular resolution. In S. Whitesides, editor, *Graph Drawing (Proc. GD '98)*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[10] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[11] M. Jünger, S. Leipert, and P. Mutzel. Pitfalls of using PQ-trees in automatic graph drawing. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 193–204. Springer-Verlag, 1997.

[12] M. Jünger, S. Leipert, and P. Mutzel. A note on computing a maximal planar subgraph using PQ-trees. *IEEE Transactions on Computer-Aided Design*, 17(7), 1998.

[13] M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications (JGAA) (http://www.cs.brown.edu/publications/jgaa/)*, 1(1):1–25, 1996.

[14] M. Jünger and P. Mutzel. Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica, Special Issue on Graph Drawing*, 16(1):33–59, 1996.

[15] G. W. Klau and P. Mutzel. Quasi–orthogonal drawing of planar graphs. Technical Report MPI-I-98-1-013, Max–Planck–Institut für Informatik, Saarbrücken, 1998.

[16] S. Leipert. Berechnung maximal planarer Untergraphen mit Hilfe von PQ-Bäumen. Master's thesis, Institut für Informatik, Universität zu Köln, 1995.

[17] K. Mehlhorn and P. Mutzel. On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica*, 16(2):233–242, 1996.

[18] P. Mutzel. *The maximum planar subgraph problem*. PhD thesis, Institut für Informatik, Universität zu Köln, 1994.

[19] P. Mutzel. *Automatisiertes Zeichnen von Diagrammen*, pages 425–431. Jahrbuch 1995 der Max-Planck-Gesellschaft, Vandenhoeck & Ruprecht Verlag, Göttingen, 1995.

[20] P. Mutzel and T. Ziegler. The constrained crossing minimization problem — a first approach. In P. Kall and H.-J. Luethi, editors, *Proceedings of the International Conference on Operations Research*, ETH-Zürich, Schweiz, 1998. Springer-Verlag. to appear.

[21] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Trans. Syst. Man Cybern.*, SMC-11(2):109–125, 1981.

[22] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.